Dependency parsing

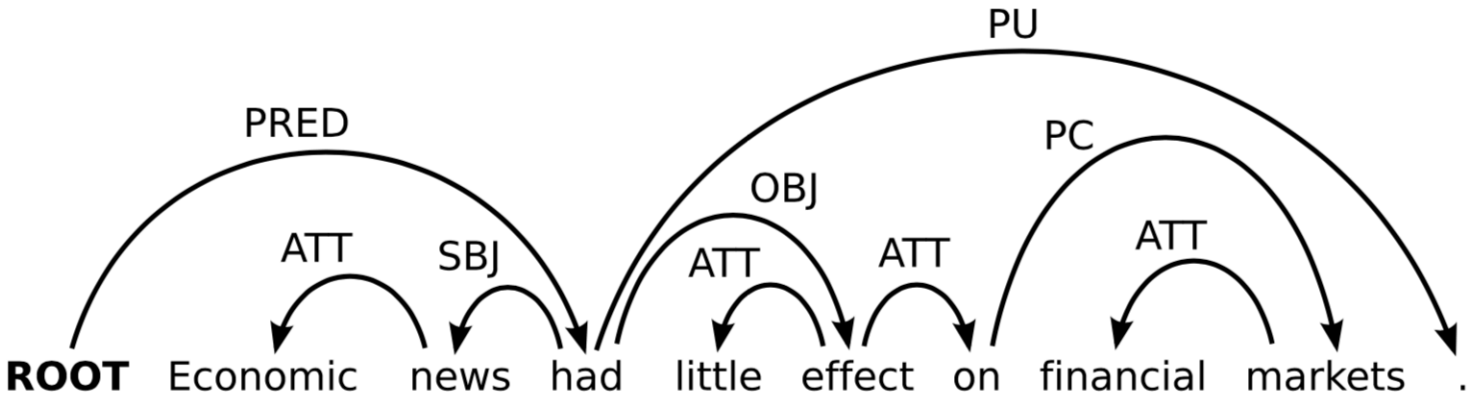
Benoît Sagot
Inria (ALMAnaCH), Paris (France)

MVA — Speech and Language Processing — Class #6 — 26th February, 2018

Credit and disclaimer: some of the following slides are taken from, illustrated or inspired by presentations and article figures by Nivre, Ballesteros, Rasooli and Tetreault

Introduction

- **Syntactic parsing of natural language**
  - Building the structure of natural language sentences
- **Dependency-based syntactic representations**
  - Long tradition in descriptive and theoretical linguistics
  - Have become popular in computational linguistics



Strategies for dependency parsing

- Graph-based parsing
- Transition-based parsing
- Other strategies

Graph-based parsing

- MSTParser (McDonald et al. 2005)
  - http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html
- Simplified version of the underlying idea:
  - Create all possible dependencies
  - Weigh them
  - Extract the optimal dependency tree
    - I.e. the tree that covers all words and minimises the overall weight of all retained dependencies

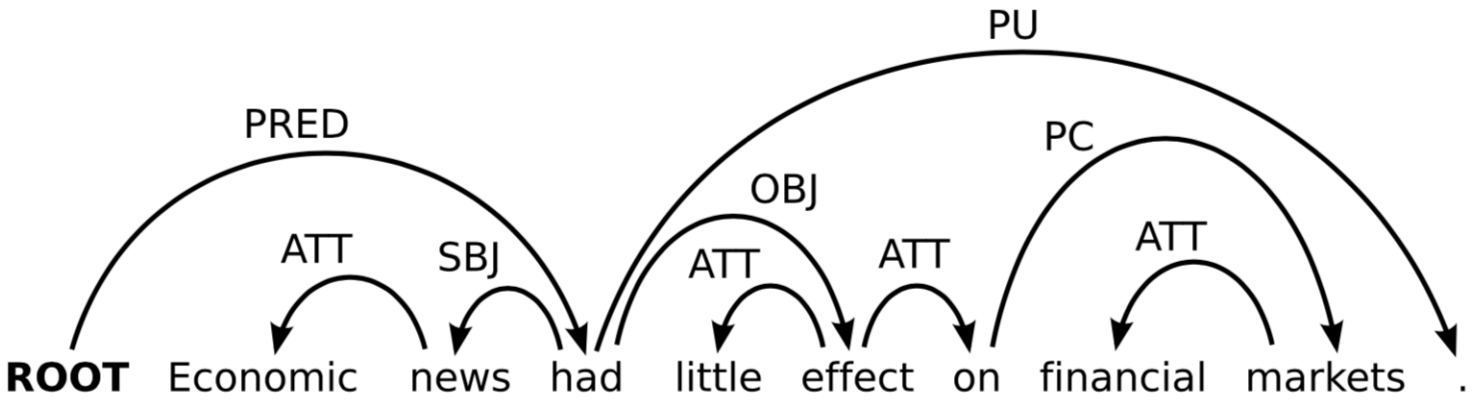
Arc-standard Transition-Based Parsing

Starting point

- **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
- Cf. Nivre (et al.)
  - http://www.maltparser.org

Formalising dependency trees

- A **dependency tree** is a **labelled directed tree** *T* with
  - a set *V* of nodes, labelled with wordforms (including the special "wordform" **ROOT**)
  - a set *A* of arcs, labelled 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



Parser configurations

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

  $([w_0]_S, [w_1, \dots, w_n]_Q, \{\})$
  (recall that w_0 = **ROOT**)
- **Termination**: $([w_0]_S, []_Q, A)$

Transitions for the “arc-standard algorithm”

- **Left-Arc**(*l*)

$$\frac{([\ldots, w_i, w_j]_S, Q, A)}{([\ldots, w_j]_S, Q, A \cup \{(w_j, l, w_i)\})} \quad [i \neq 0]$$

- **Right-Arc**(*l*)

$$\frac{([\ldots, w_i, w_j]_S, Q, A)}{([\ldots, w_i]_S, Q, A \cup \{(w_i, l, w_j)\})}$$

- **Shift**

$$\frac{([\ldots]_S, [w_i, \ldots]_Q, A)}{([\ldots, w_i]_S, [\ldots]_Q, A)}$$

Example Transition Sequence

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

action: Shift

ROOT Economic news had little effect on financial markets .

Example Transition Sequence

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

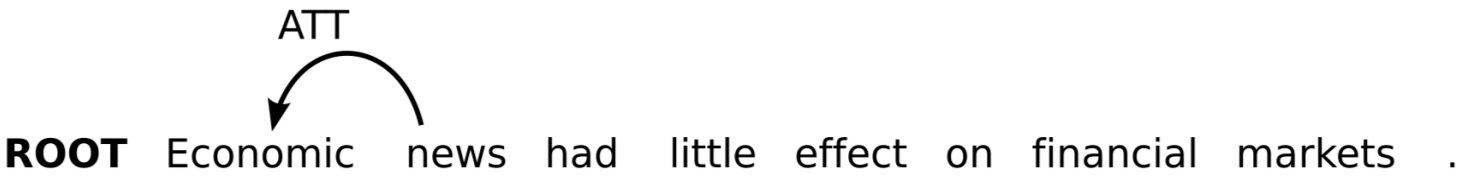
action: Shift

ROOT Economic news had little effect on financial markets .

Example Transition Sequence

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

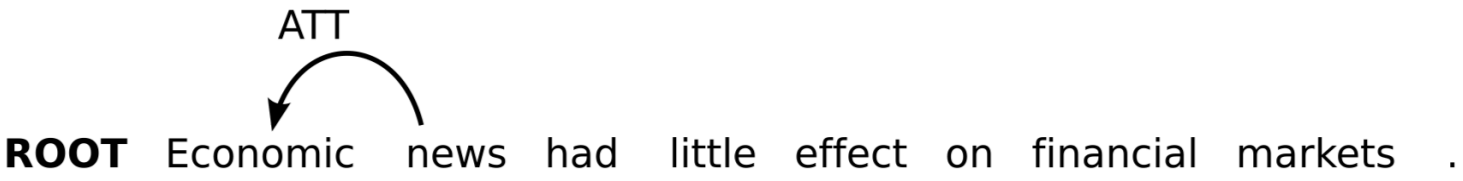
action: Left-Arc(ATT)



Example Transition Sequence

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

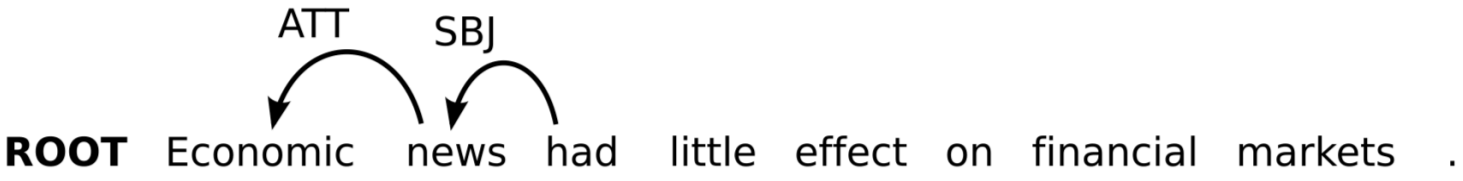
action: Shift



Example Transition Sequence

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

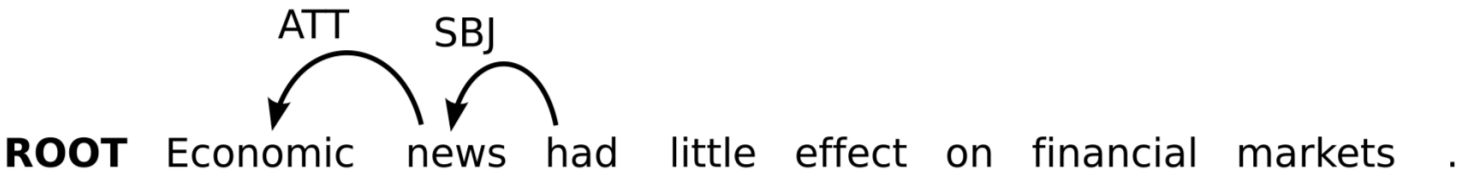
action: Left-Arc(SBJ)



Example Transition Sequence

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

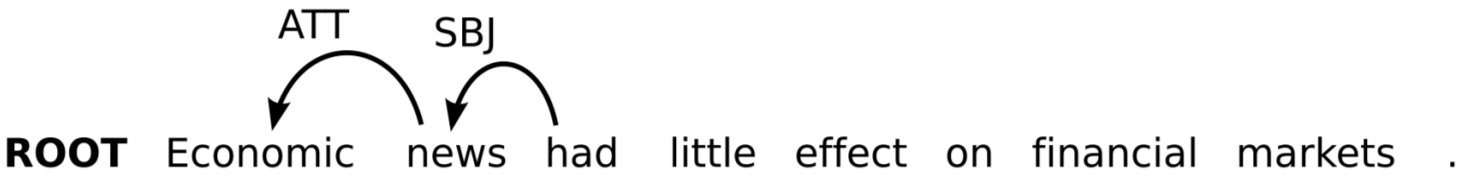
action: Shift



Example Transition Sequence

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

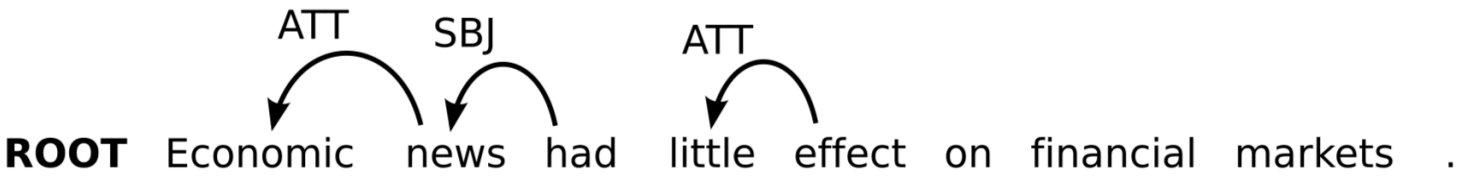
action: Shift



Example Transition Sequence

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

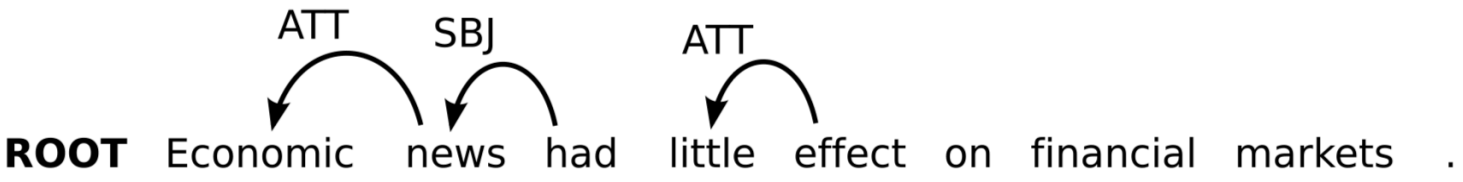
action: Left-Arc(ATT)



Example Transition Sequence

$[\text{ROOT, had, effect, on}]_S$ $[\text{financial, markets, .}]_Q$

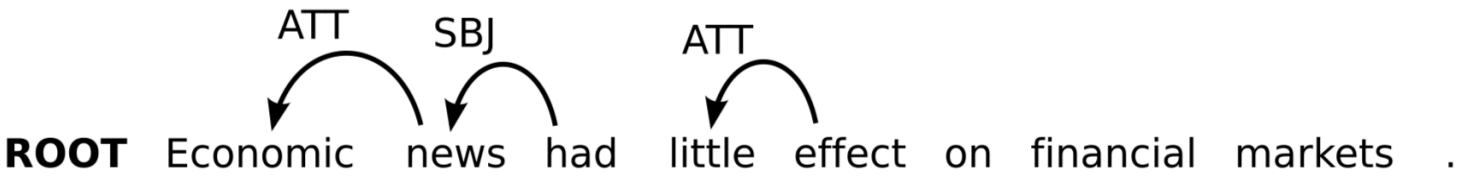
action: Shift



Example Transition Sequence

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}, \text{financial}]_S\ [\text{markets}, .]_Q$

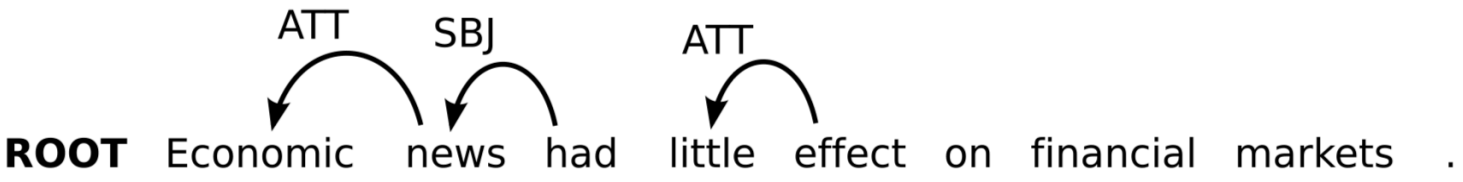
action: Shift



Example Transition Sequence

$[\text{ROOT, had, effect, on, financial, markets}]_S\ [.]_Q$

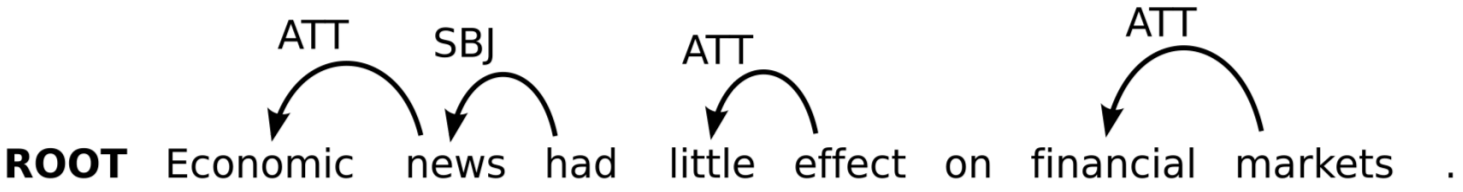
action: Shift



Example Transition Sequence

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

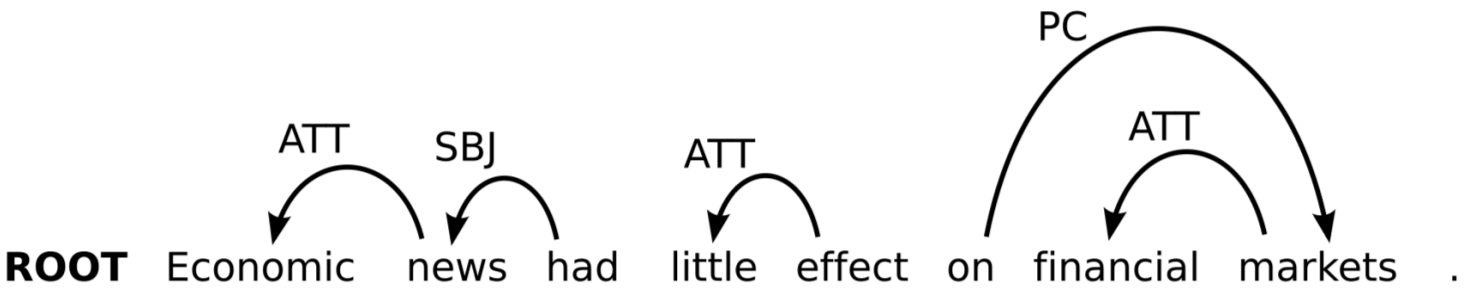
action: Left-Arc(ATT)



Example Transition Sequence

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

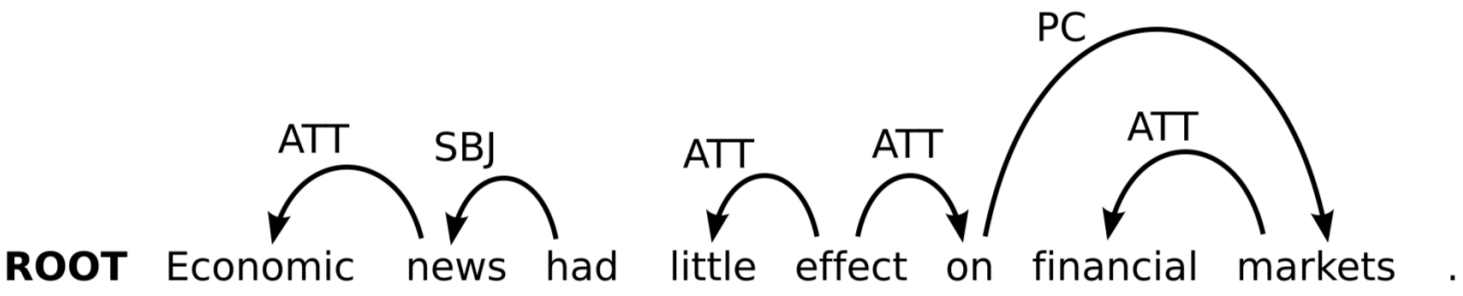
action: Right-Arc(PC)



Example Transition Sequence

$[\text{ROOT}, \text{had}, \text{effect}, \text{on}]_S\ [.]_Q$

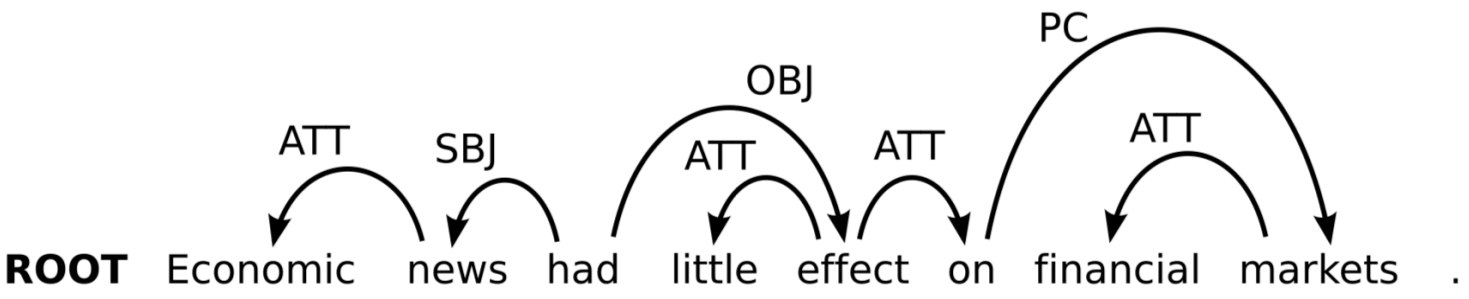
action: Right-Arc(ATT)



Example Transition Sequence

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

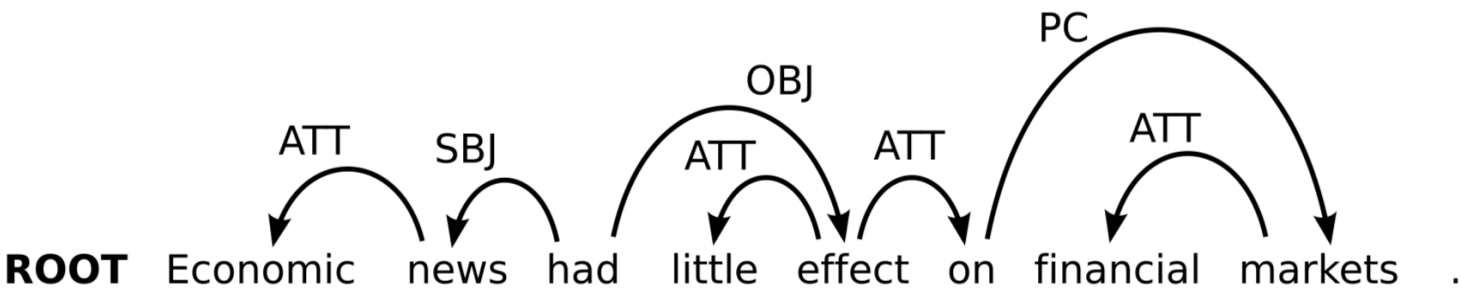
action: Right-Arc(OBJ)



Example Transition Sequence

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

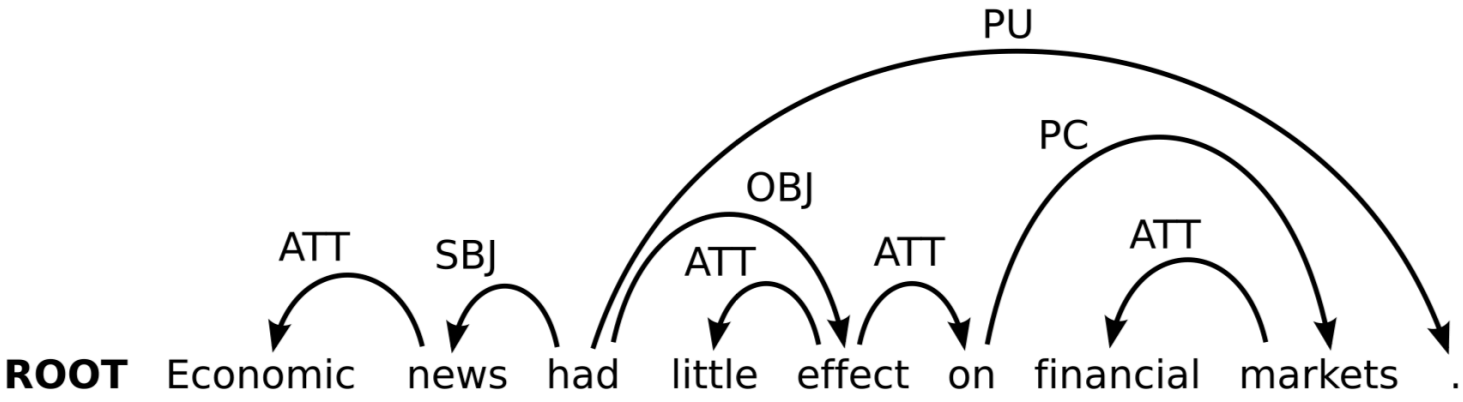
action: Shift



Example Transition Sequence

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

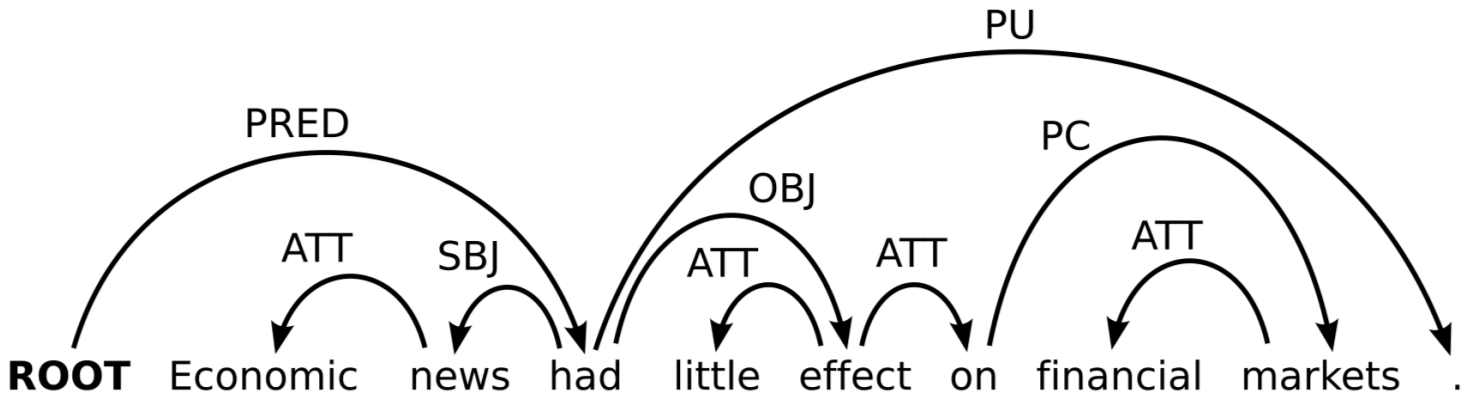
action: Right-Arc(PU)



Example Transition Sequence

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

action: Right-Arc(PRED)



Properties of the algorithm

- Every transition sequence outputs a projective dependency tree (**soundness**).
- Every projective dependency tree is output by some transition sequence (**completeness**).
- There are exactly 2*n* transitions in a sentence with *n* words.

Deterministic parsing

- If we have an **oracle** 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 $\quad t \leftarrow o(c)$
4 $\quad c \leftarrow t(c)$
5 **return** $T = (\{w_0, w_1, \ldots, w_n\}, A_c)$

Oracles as classifiers

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

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

- History-based feature representation **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 **w** learned from treebank data:
  - Reconstruct oracle transition sequence for each sentence
  - Construct training data set $D = \{(c, t) \mid o(c) = t\}$
  - Maximise accuracy of local predictions $o(c) = t$

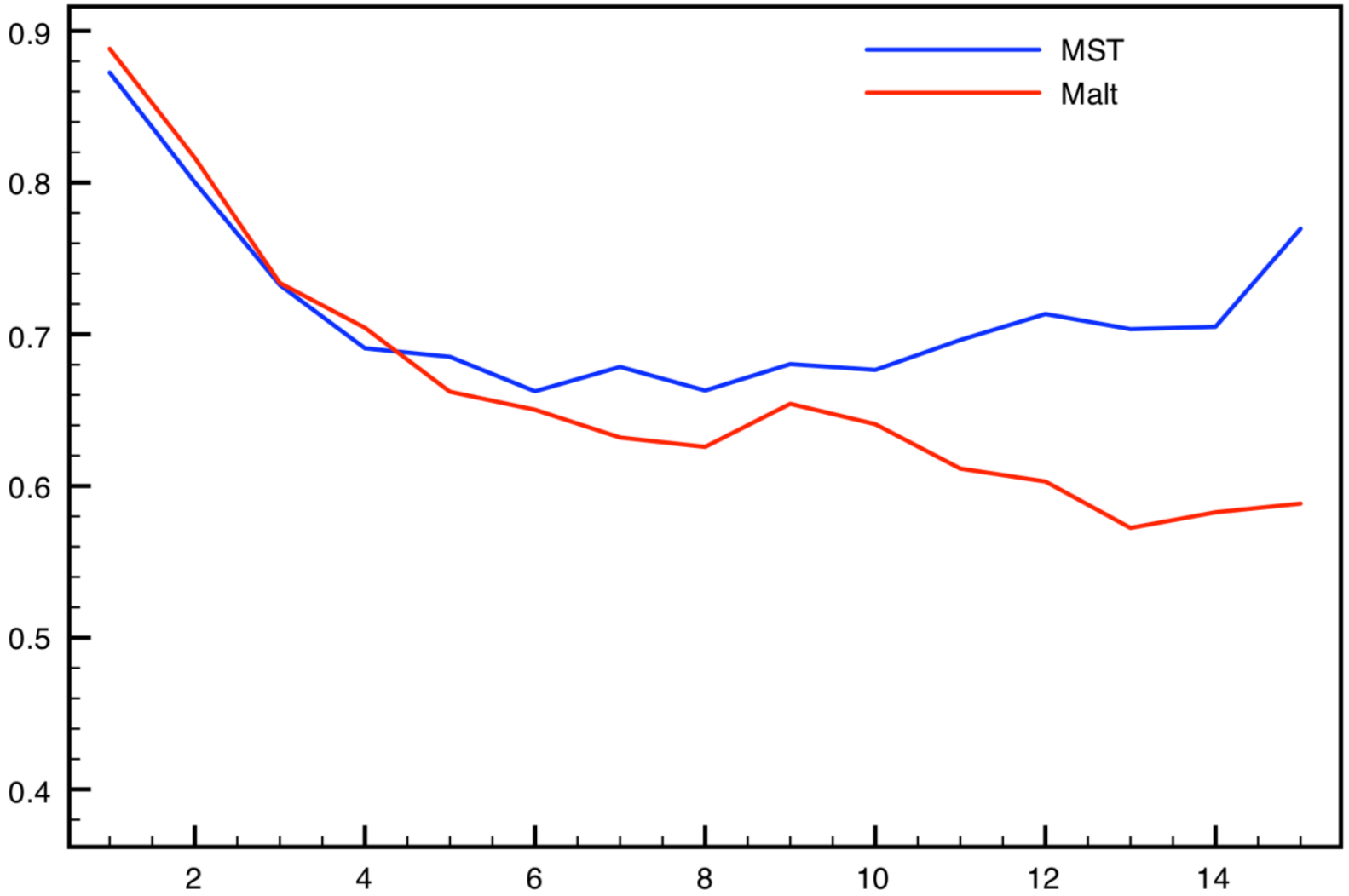
Deterministic classifier-based parsing

- 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 results: the CoNLL 2006 shared task

- CoNLL 2006 shared task (Buchholz and Marsi 2006):
  - **MaltParser** (Nivre et al. 2006) – transition-based, deterministic, local learning
  - **MSTParser** (McDonald et al. 2006) – graph-based, exact, global learning
  - Same average parsing accuracy over 13 languages
- Comparative **error analysis** (McDonald and Nivre2007):
  - 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
- Hypothesised **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, … , c_m) = $\sum_{i=1}^{m} \mathbf{w} \cdot \mathbf{f}(c_{j-1}, t_j)$
  - Simple combination of locally normalised classifier scores
  - Marginal gains in accuracy

Online question 1

What is the complexity of transition-based parsing extended with beam search?

1. O(n)
2. O(n log n)
3. O(n^2)

https://api.socrative.com/rc/KBcrE4

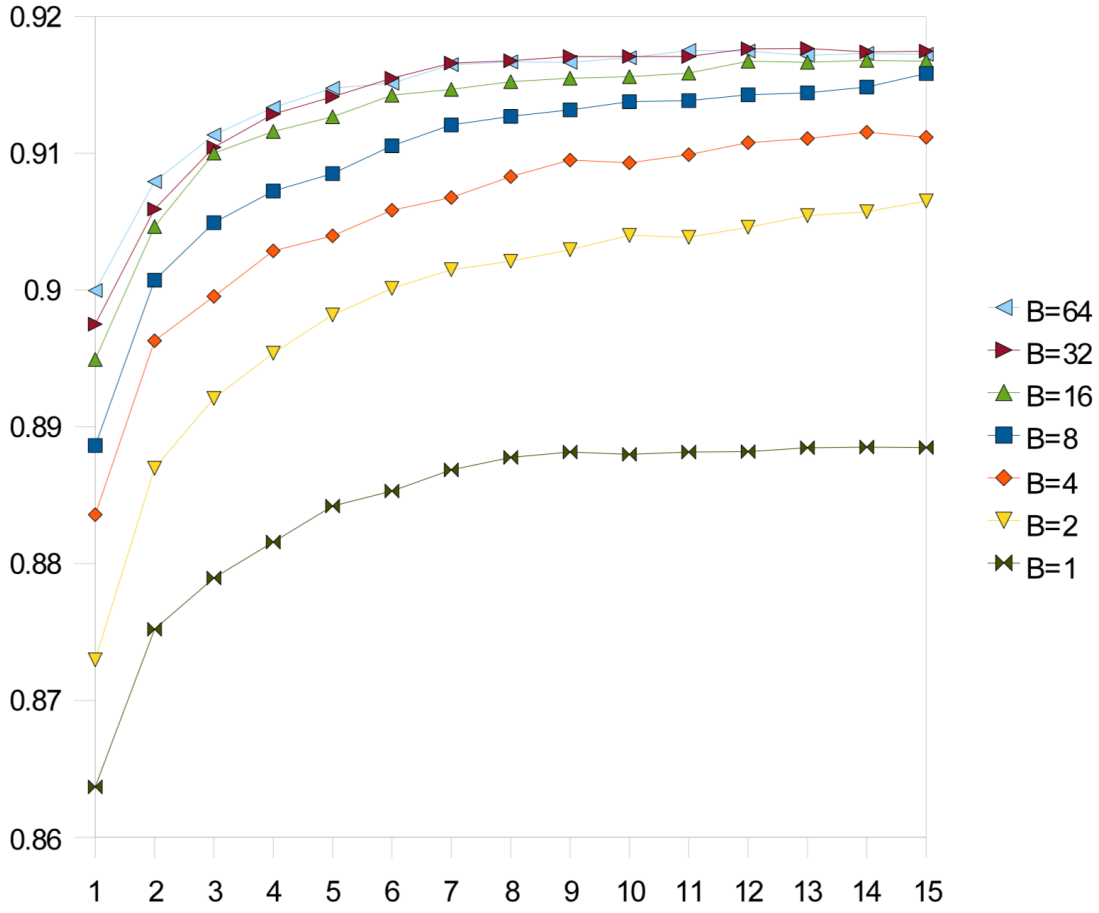
Structured prediction

- **Parsing as structured prediction** (Zhang and Clark 2008)**:**
  - Minimise 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

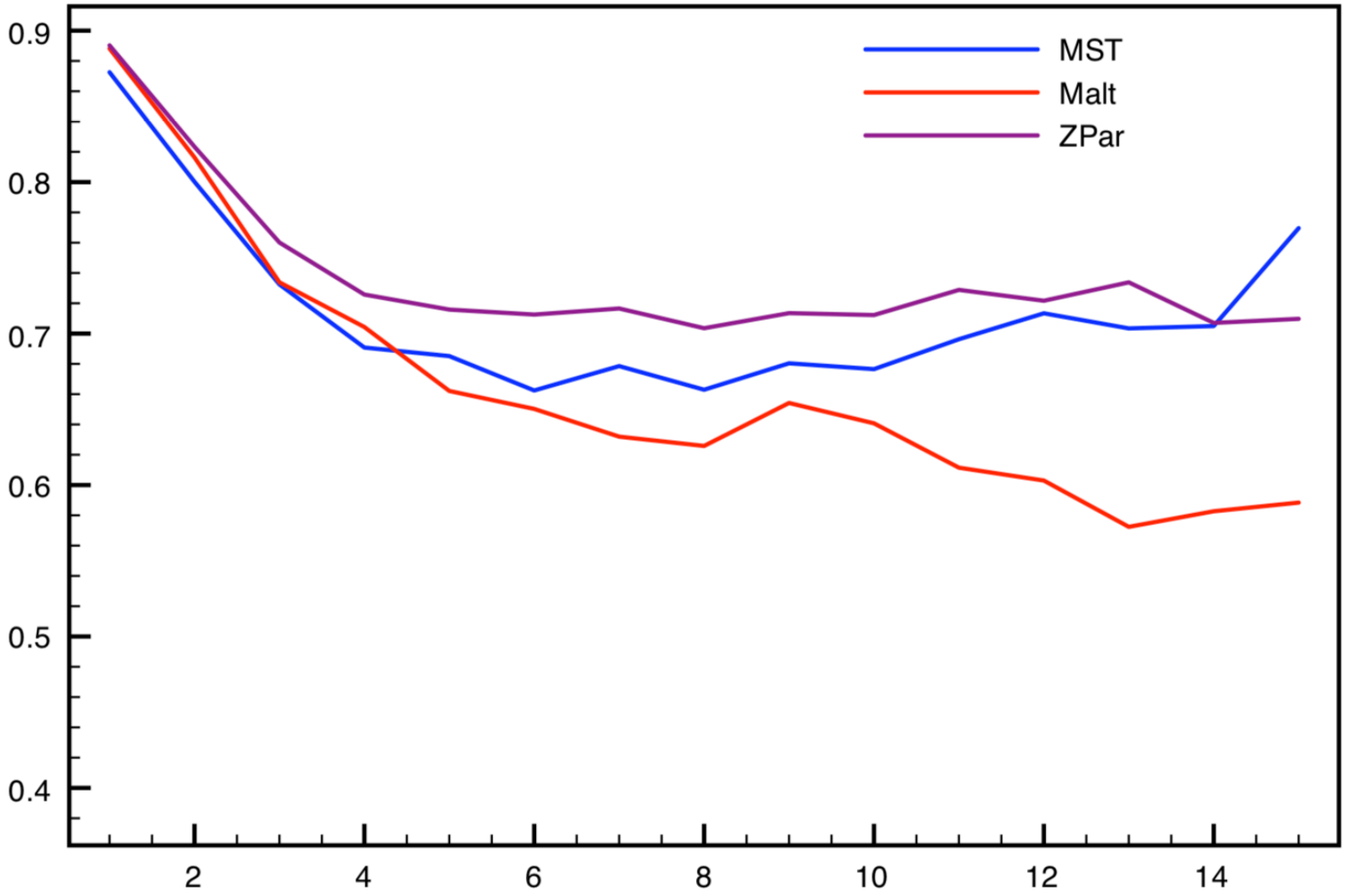


Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-Based and Transition-Based Dependency Parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 562–571.

The best of two worlds?

- Like graph-based dependency parsing (MSTParser):
  - Global learning – minimise 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
- Example ZPar parser (Zhang and Clark 2011)
  - “Most heavily developed for English and Chinese”

Precision by dependency length, again



Even richer feature models

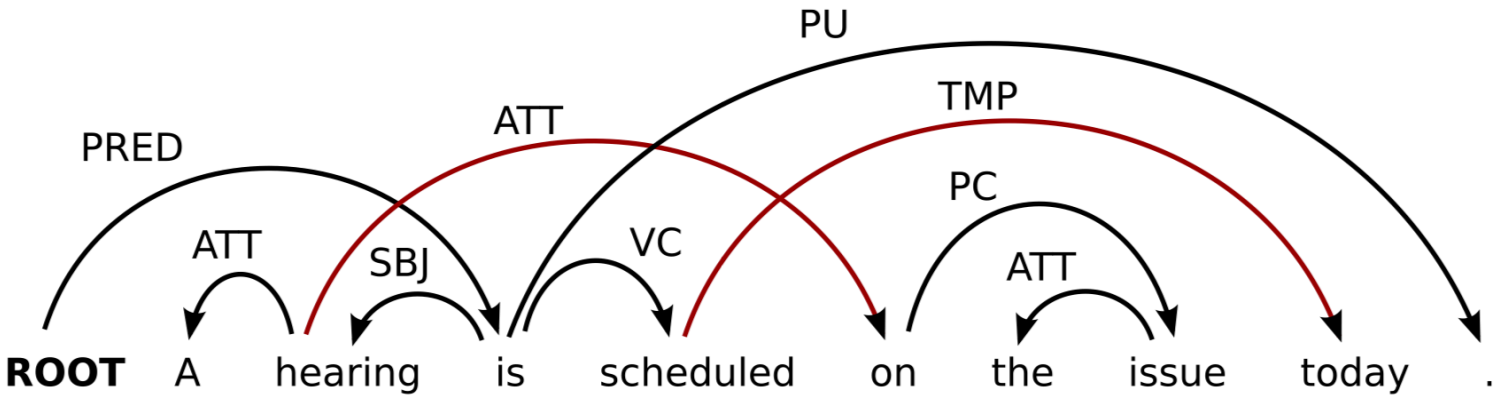
	ZPar	Malt
Baseline	92.18	89.37
+distance	+0.07	–0.14
+valency	+0.24	0.00
+unigrams	+0.40	–0.29
+third-order	+0.18	0.00
+label set	+0.07	+0.06
Extended	93.14	89.00

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)

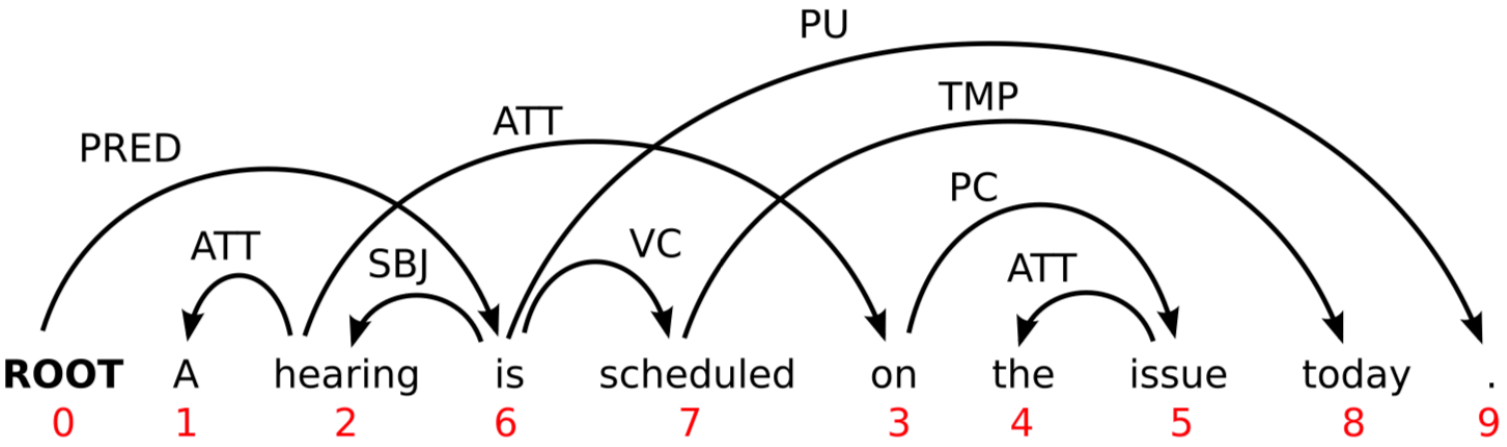


Non-projectivity in natural languages

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 %

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**

$$\frac{([\ldots, w_i, w_j]_S, [\ldots]_Q, A)}{([\ldots, w_j]_S, [w_i, \ldots]_Q, A)} \qquad [0 < i < j]$$

- 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

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

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

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

action: Shift

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

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

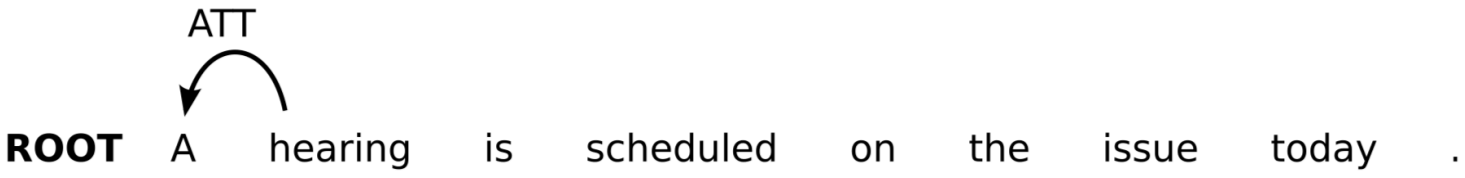
action: Shift

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

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

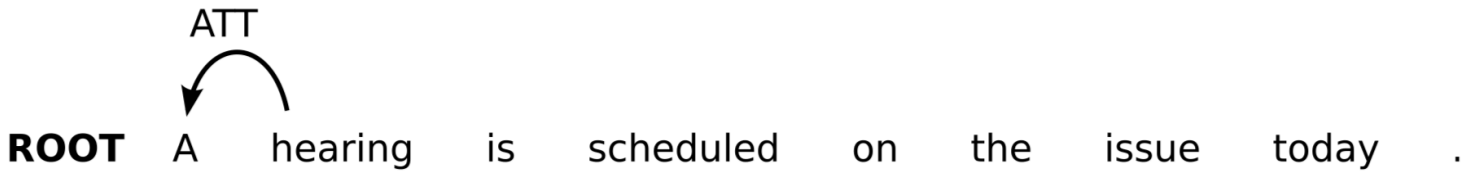
action: Left-Arc(ATT)



Example Transition Sequence

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

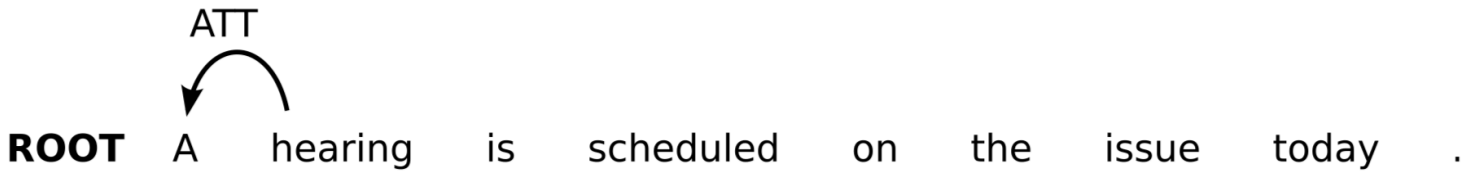
action: Shift



Example Transition Sequence

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

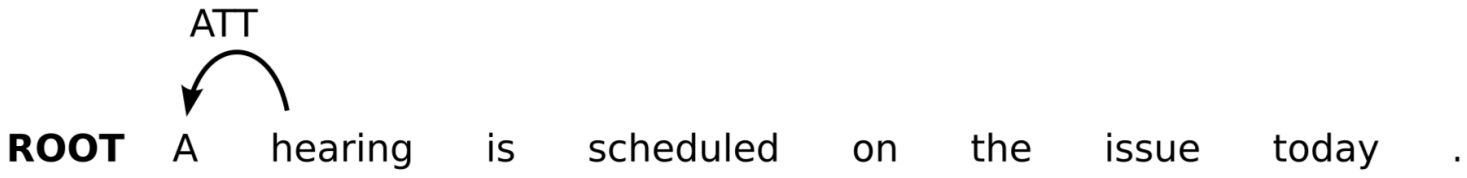
action: Shift



Example Transition Sequence

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

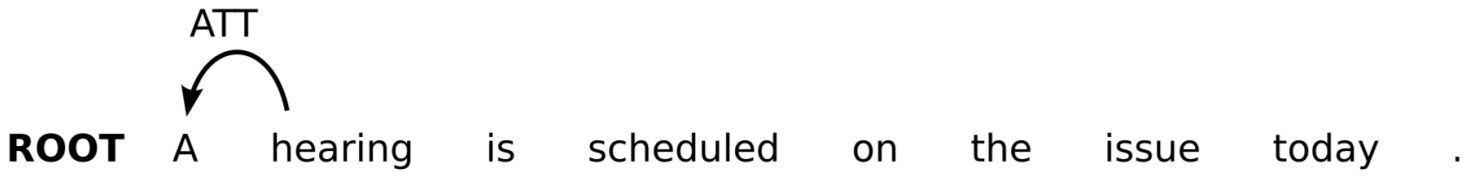
action: Shift



Example Transition Sequence

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

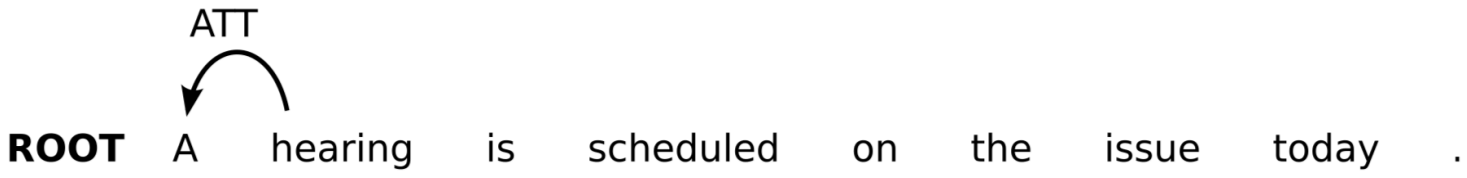
action: Shift



Example Transition Sequence

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

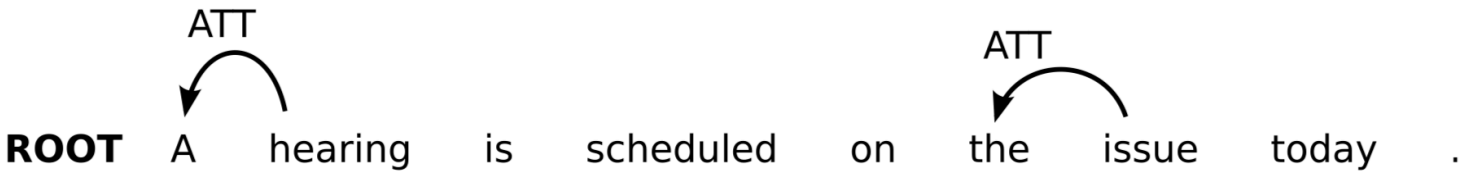
action: Shift



Example Transition Sequence

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

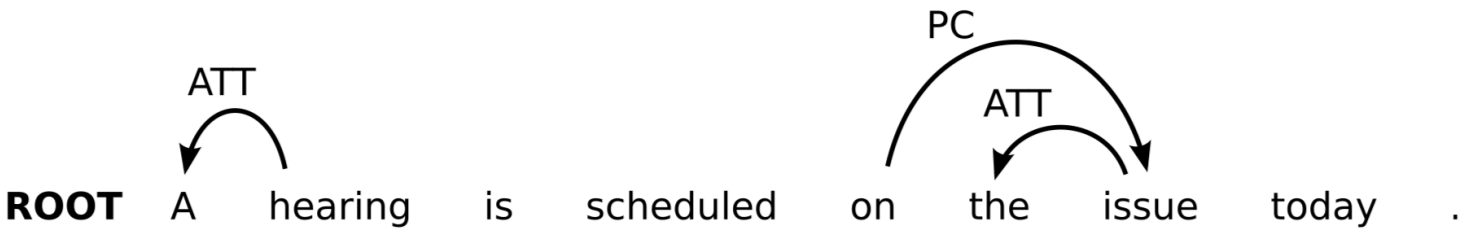
action: Left-Arc(ATT)



Example Transition Sequence

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

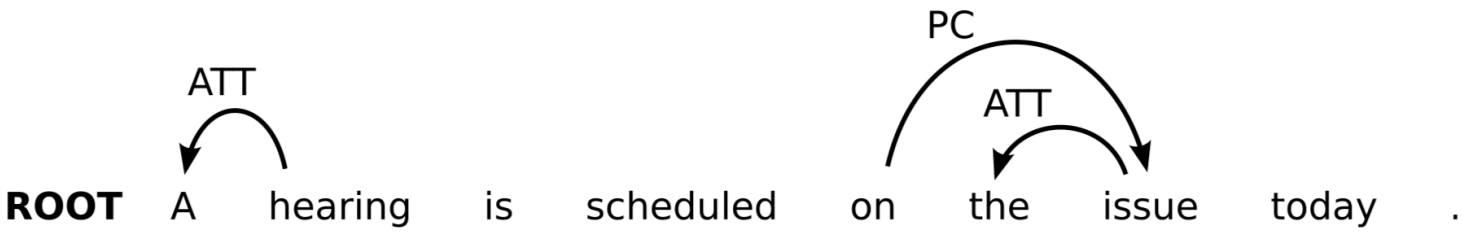
action: Right-Arc(PC)



Example Transition Sequence

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

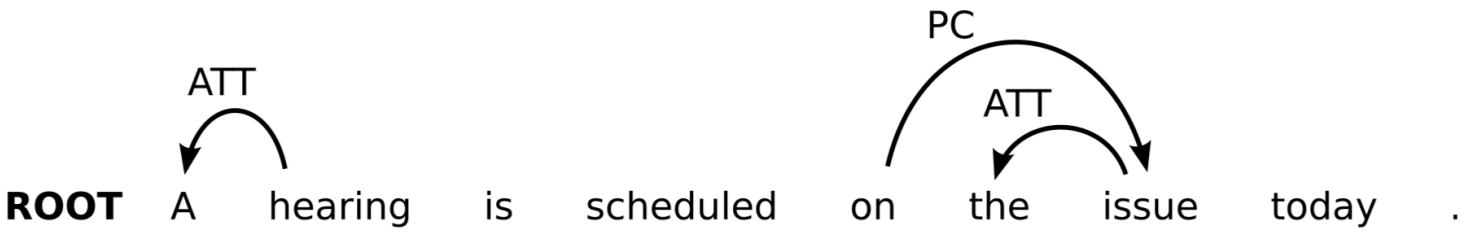
action: Swap



Example Transition Sequence

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

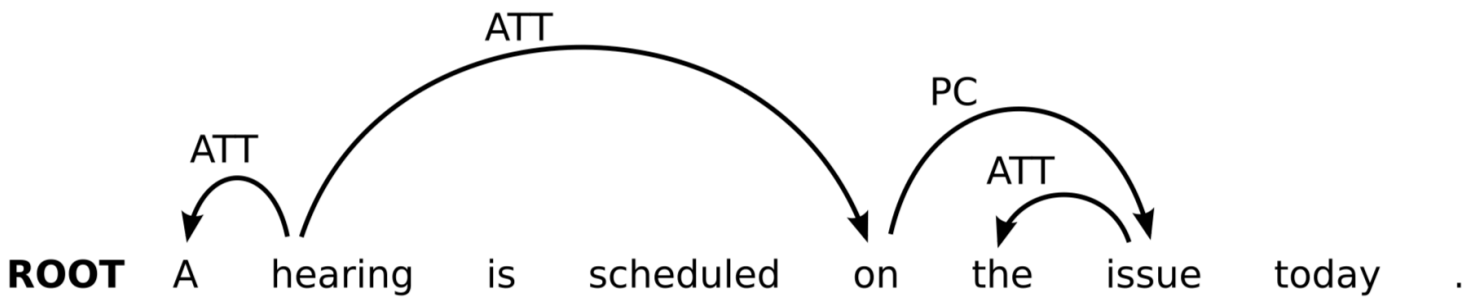
action: Swap



Example Transition Sequence

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

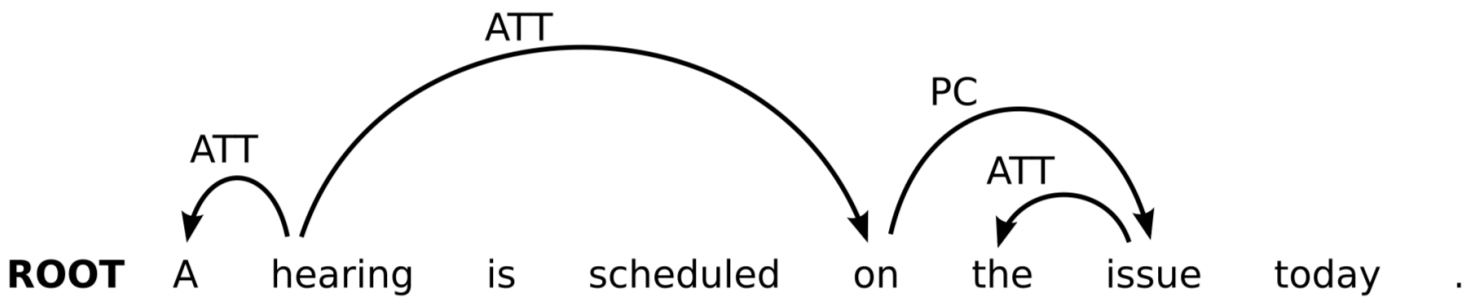
action: Right-Arc(ATT)



Example Transition Sequence

$[\text{ROOT, hearing, is}]_S\ [\text{scheduled, today, .}]_Q$

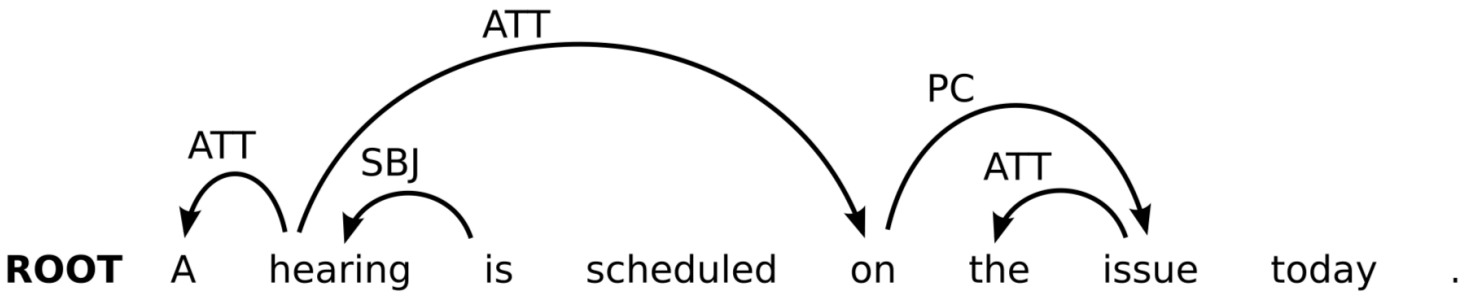
action: Shift



Example Transition Sequence

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

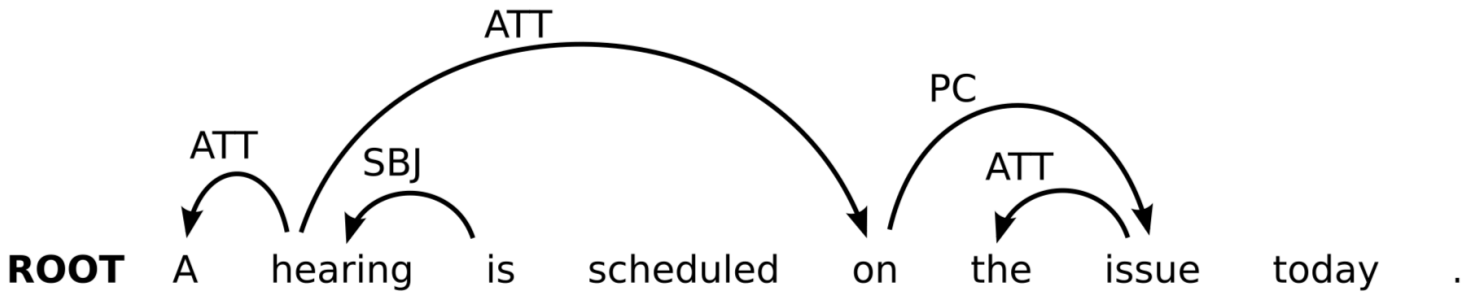
action: Left-Arc(SBJ)



Example Transition Sequence

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

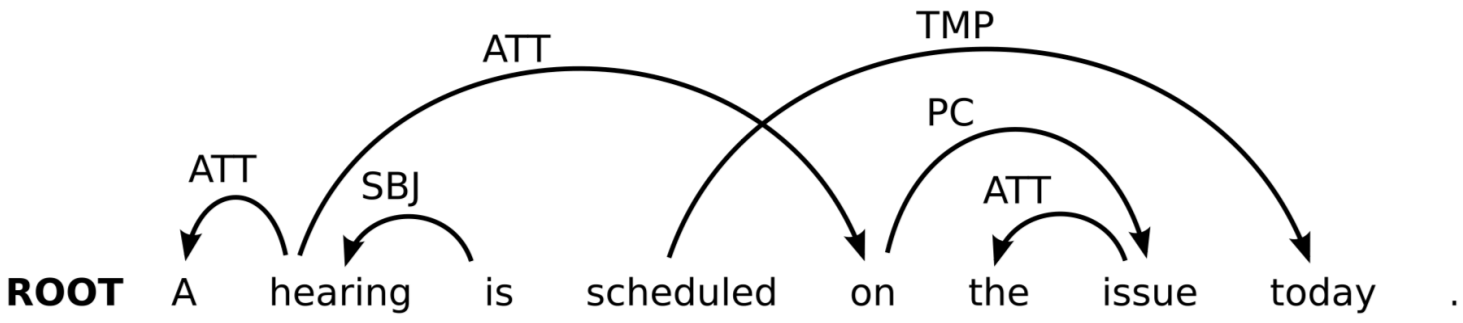
action: Shift



Example Transition Sequence

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

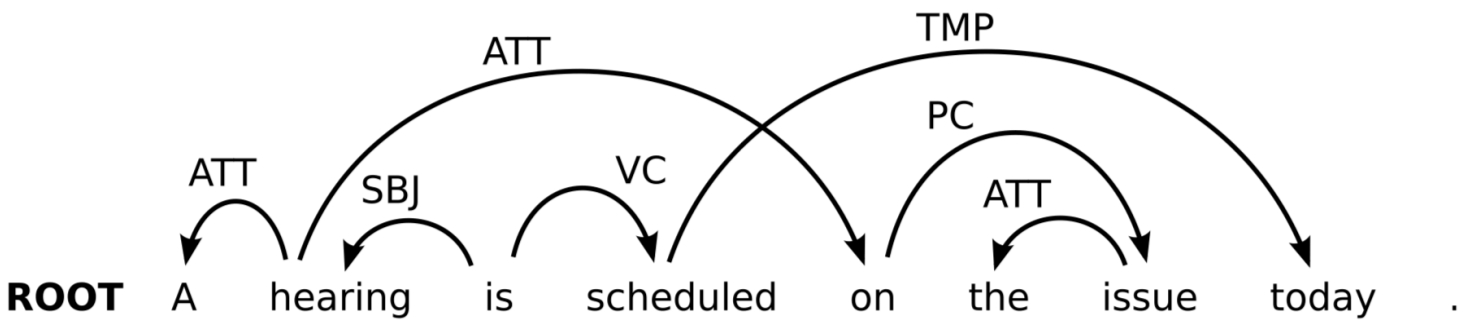
action: Right-Arc(TMP)



Example Transition Sequence

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

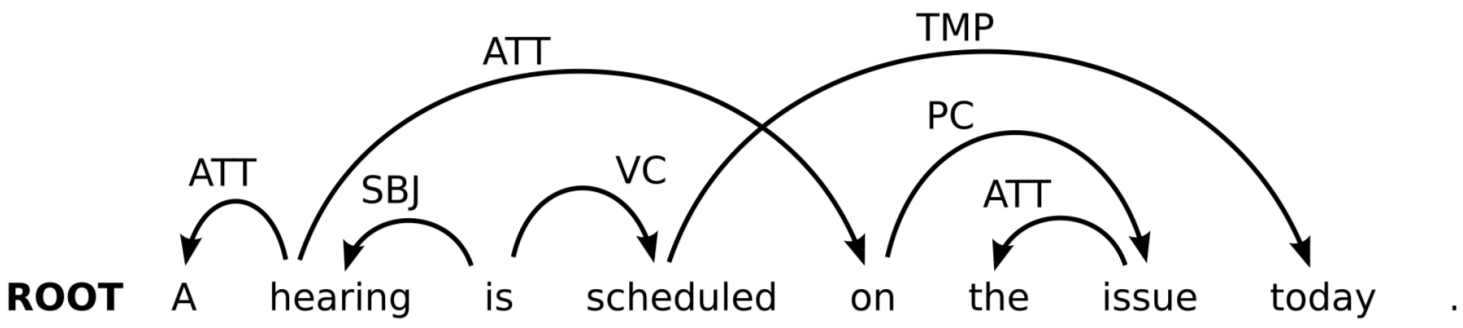
action: Right-Arc(VC)



Example Transition Sequence

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

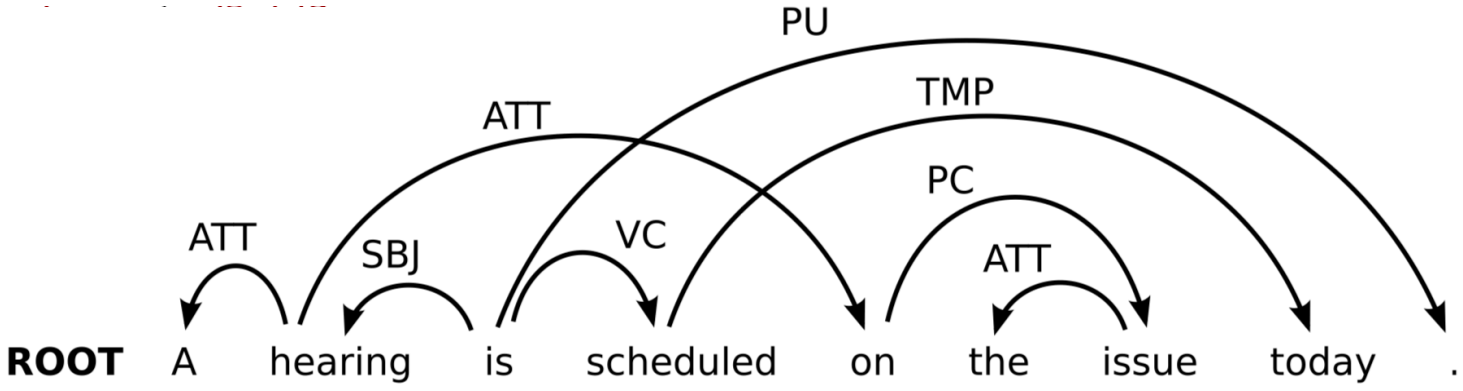
action: Shift



Example Transition Sequence

[ROOT, is, .]$_S$ []$_Q$

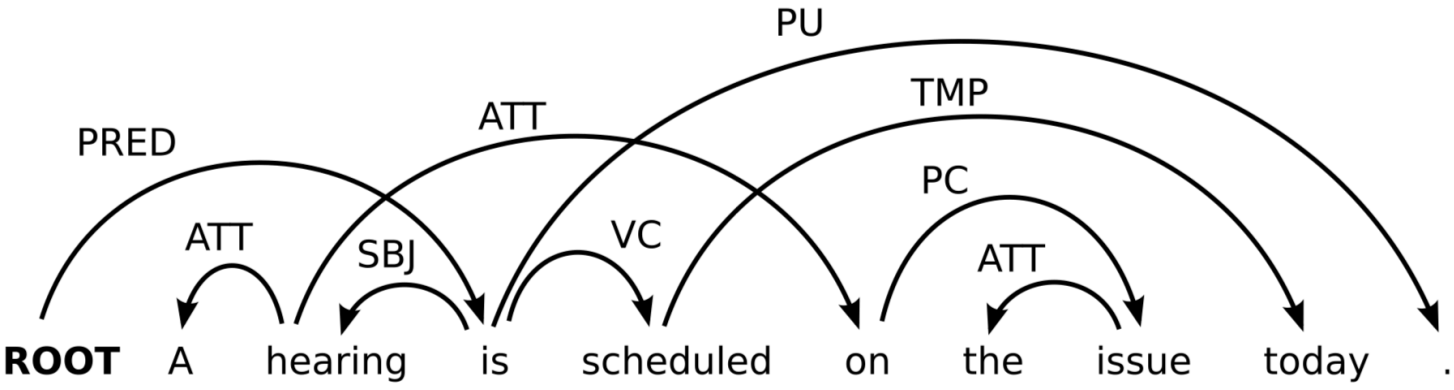
action: Right-Arc(PU)



Example Transition Sequence

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

action: Right-Arc(PRED)



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:
  - 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 PoS tagging and parsing

Part-of-Speech tagging for parsing

- 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
- Joint models for tagging and parsing:
  - Richly inflected languages (Lee et al. 2011)
  - Chinese (Li et al. 2011, Hatori et al. 2011)

Parsing with online reordering

- Redefine Shift transition (Hatori et al. 2011)
  - **ShiftAndTag(p)** where p is a PoS tag

$$\frac{([\ldots]_S, [w_i, \ldots]_Q, A)}{([\ldots, w_i/p]_S, [\ldots]_Q, A)}$$

- Transition-based parsing now with up to three interleaved processes:
  - Tag wordforms when they are shifted onto the stack
  - Sort words into projective order $<_p$ (if non-projectivity expected)
  - 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.

Online question 2

The arc-standard algorithm has several drawbacks. What is this its main drawback?

1. Segmenting the construction of the parse into individual Left-Arc()'s and Right-Arc()'s prevents a correct handling of compound words and idioms
2. The emission of Left-Arc()'s must be performed as soon as possible, which results in decisions being sometimes taken without the appropriate information
3. The emission of Right-Arc()'s is sometimes postponed, which results in decisions being sometimes taken without the appropriate information

https://api.socrative.com/rc/KBcrE4

Arc-eager Transition-Based Parsing

Limitations of the arc-standard algorithm

- The **arc-standard** system considered so far
  - builds a dependency tree strictly bottom-up
  - a dependency arc can only be added between two nodes if the dependent node has already found all its dependents.
  - As a consequence, it is often necessary to postpone the attachment of right dependents.
- This is a problem, as parsing decisions are easier to take when the governor and the governee of a dependency are immediately accessible

The problem of arc-standard on an example

[ROOT]$_S$ [La, température, a, un, très, gros, effet, sur, la, concentration]$_Q$

La température a un très gros effet sur la concentration.

The problem of arc-standard on an example

[ROOT, La]$_S$ [température, a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Shift

La température a un très gros effet sur la concentration.

The problem of arc-standard on an example

[ROOT, La, température]$_S$ [a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Shift

La température a un très gros effet sur la concentration.

The problem of arc-standard on an example

[ROOT, La, température]$_S$ [a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Left-Arc()

The problem of arc-standard on an example

[ROOT, température, a]$_S$ [un, très, gros, effet, sur, la, concentration]$_Q$

action: Shift

The problem of arc-standard on an example

[ROOT, température, a]$_S$ [un, très, gros, effet, sur, la, concentration]$_Q$

action: Left-Arc()

The problem of arc-standard on an example

[ROOT, a, un]$_S$ [très, gros, effet, sur, la, concentration]$_Q$

action: Shift

The problem of arc-standard on an example

[ROOT, a, un, très]$_S$ [gros, effet, sur, la, concentration]$_Q$

action: Shift

The problem of arc-standard on an example

[ROOT, a, un, très, gros]$_S$ [effet, sur, la, concentration]$_Q$

action: Shift

The problem of arc-standard on an example

[ROOT, a, un, très, gros]$_S$ [effet, sur, la, concentration]$_Q$

action: Right-Arc()

The problem of arc-standard on an example

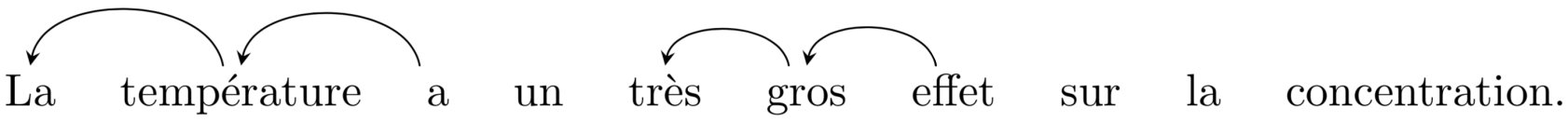
[ROOT, a, un, gros, effet]$_S$ [sur, la, concentration]$_Q$

action: Shift

The problem of arc-standard on an example

[ROOT, a, un, gros, effet]$_S$ [sur, la, concentration]$_Q$

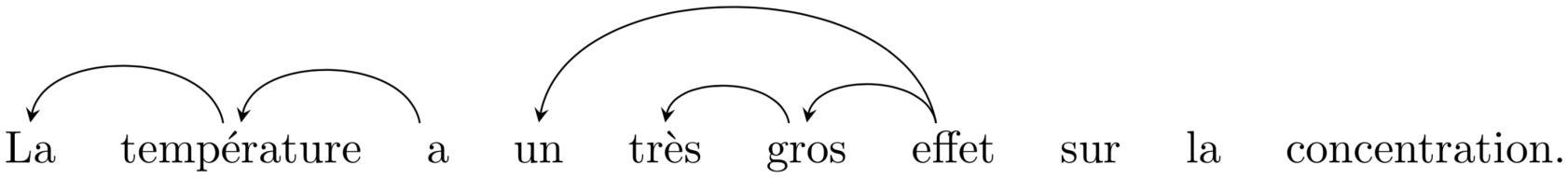
action: Left-Arc()



The problem of arc-standard on an example

[ROOT, a, un, effet]$_S$ [sur, la, concentration]$_Q$

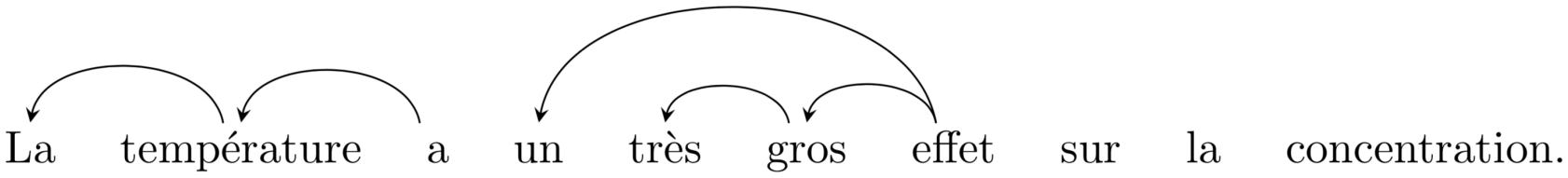
action: Left-Arc()



The problem of arc-standard on an example

$[\text{ROOT, a, effet, sur}]_S\ [\text{la, concentration}]_Q$

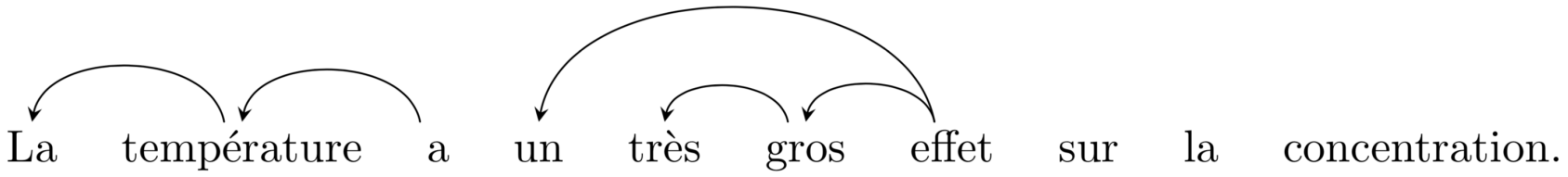
action: Shift



The problem of arc-standard on an example

[ROOT, a, effet, sur, la]$_S$ [concentration]$_Q$

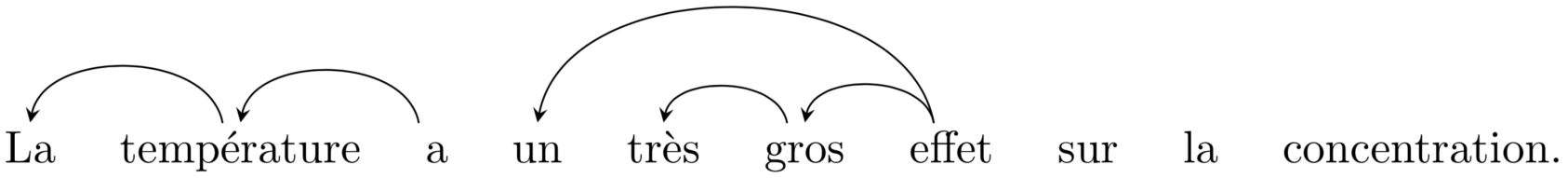
action: Shift



The problem of arc-standard on an example

[ROOT, a, effet, sur, la, concentration]$_S$ []$_Q$

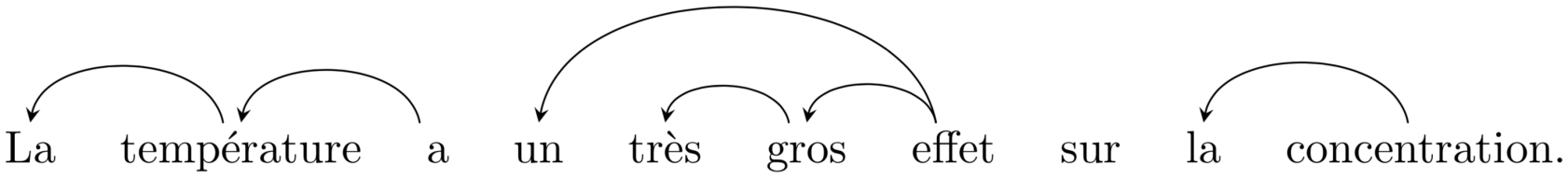
action: Shift



The problem of arc-standard on an example

[ROOT, a, effet, sur, la, concentration]$_S$ []$_Q$

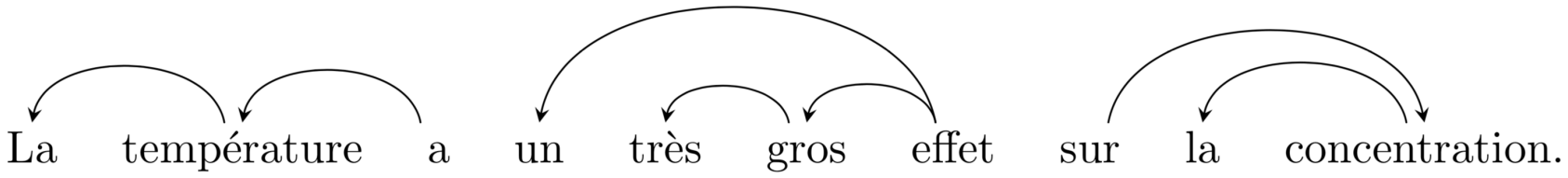
action: Left-Arc()



The problem of arc-standard on an example

[ROOT, a, effet, sur, concentration]$_S$ []$_Q$

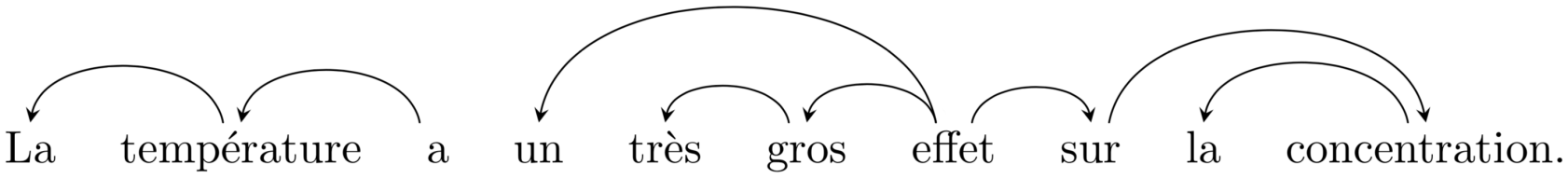
action: Right-Arc()



The problem of arc-standard on an example

[ROOT, a, effet, sur]$_S$ []$_Q$

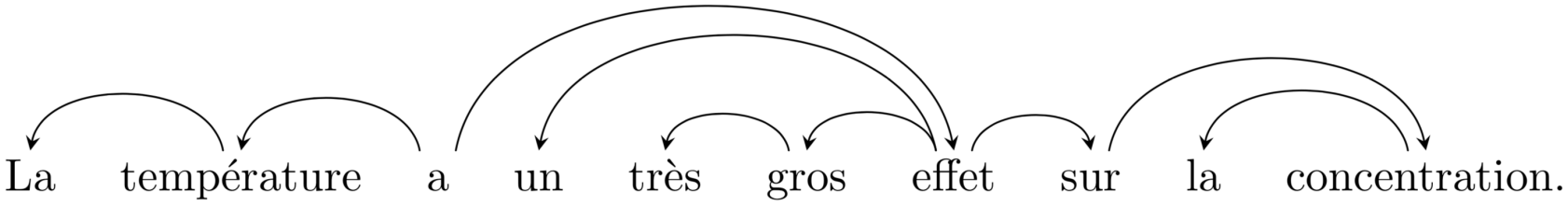
action: Right-Arc()



The problem of arc-standard on an example

[ROOT, a, effet]$_S$ []$_Q$

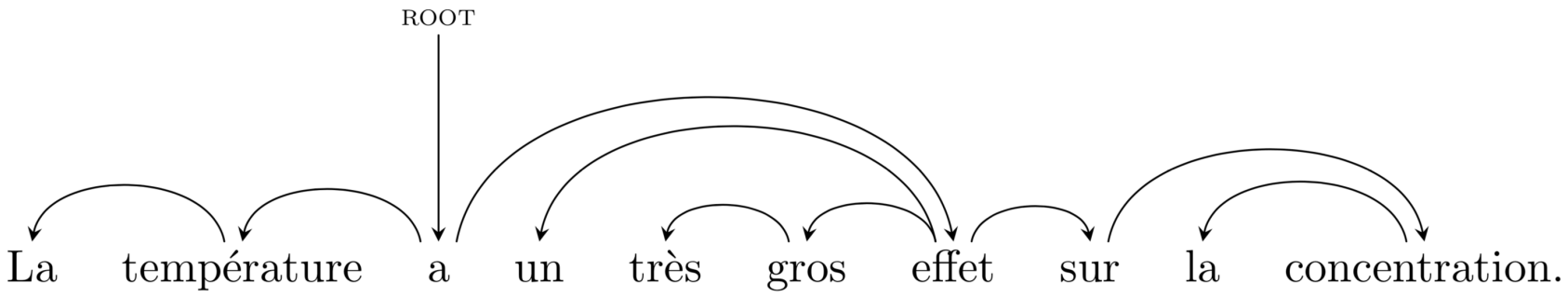
action: Right-Arc()



The problem of arc-standard on an example

[ROOT, a]$_S$ []$_Q$

action: Right-Arc()



The arc-eager system

- We will modify the basic set of actions in order to always add an arc at the earliest possible opportunity:
  - we will now build parts of the tree top-down instead of bottom-up
- Shift remains the same
- **Left-Arc is rewritten and subjected to a stricter condition** (allowed only if the dependent is not the root and has no incoming arcs)

$$\frac{([\dots, w_i]_S, [w_j, \dots]_Q, A)}{([\dots]_S, [w_j, \dots]_Q, A \cup \{(w_j, l, w_i)\})} \; [i \neq 0 \wedge \nexists(k, l') \mid (k, l', i) \in A]$$

The arc-eager system

- **Right-Arc is changed**: it does not discard w_i anymore:

$$\frac{([\dots, w_i]_S, [w_j, \dots]_Q, A)}{([\dots, w_i, w_j]_S, [\dots]_Q, A \cup \{(w_i, l, w_j)\})}$$

- We postpone the reduction of w_i to another, new action:
- **Reduction**, only possible if the top of the stack already has a head

$$\frac{([\dots, w_i]_S, Q, A)}{([\dots]_S, Q, A)} \quad \text{only if } \exists(k, l') \mid (k, l', i) \in A$$

Arc-eager on an example

[ROOT]$_S$ [La, température, a, un, très, gros, effet, sur, la, concentration]$_Q$

La température a un très gros effet sur la concentration.

Arc-eager on an example

[ROOT, La]$_S$ [température, a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Shift

La température a un très gros effet sur la concentration.

Arc-eager on an example

[ROOT, La]$_S$ [température, a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Left-Arc()

Arc-eager on an example

[ROOT, température]$_S$ [a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Shift

Arc-eager on an example

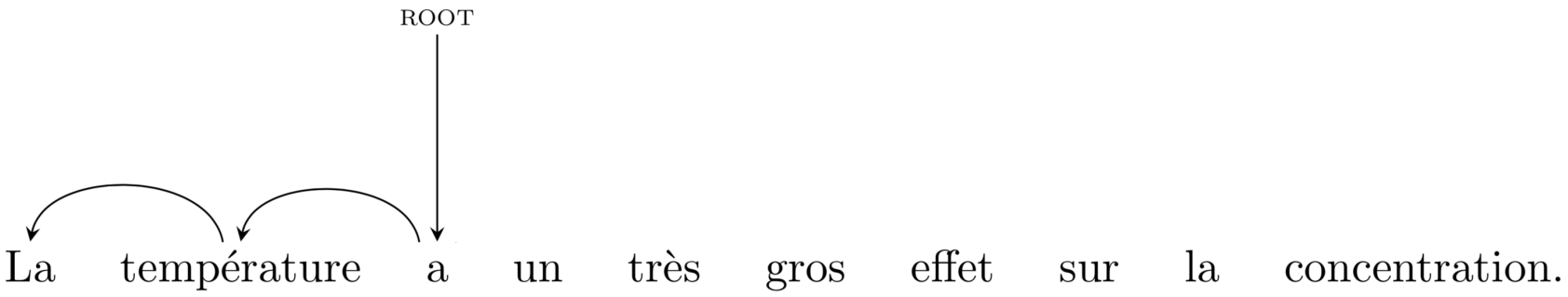
[ROOT, température]$_S$ [a, un, très, gros, effet, sur, la, concentration]$_Q$

action: Left-Arc()

Arc-eager on an example

[ROOT, a]$_S$ [un, très, gros, effet, sur, la, concentration]$_Q$

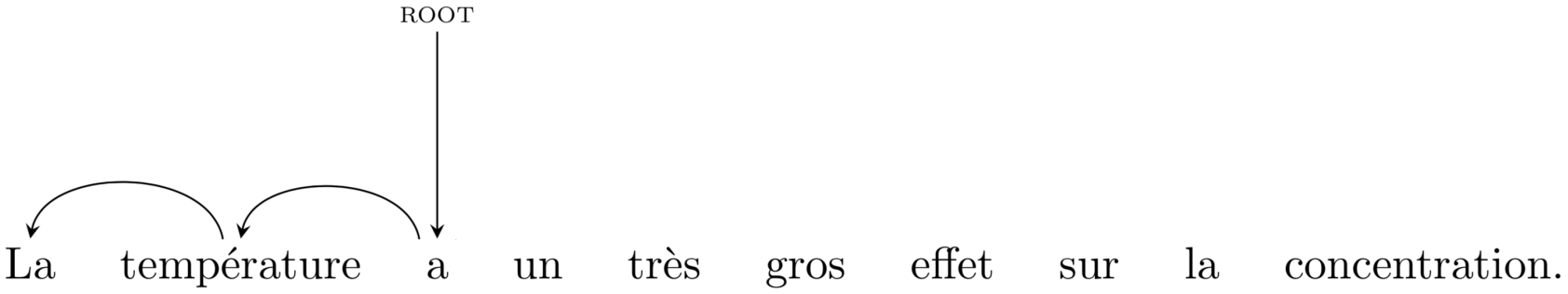
action: Right-Arc()



Arc-eager on an example

[ROOT, a, un]$_S$ [très, gros, effet, sur, la, concentration]$_Q$

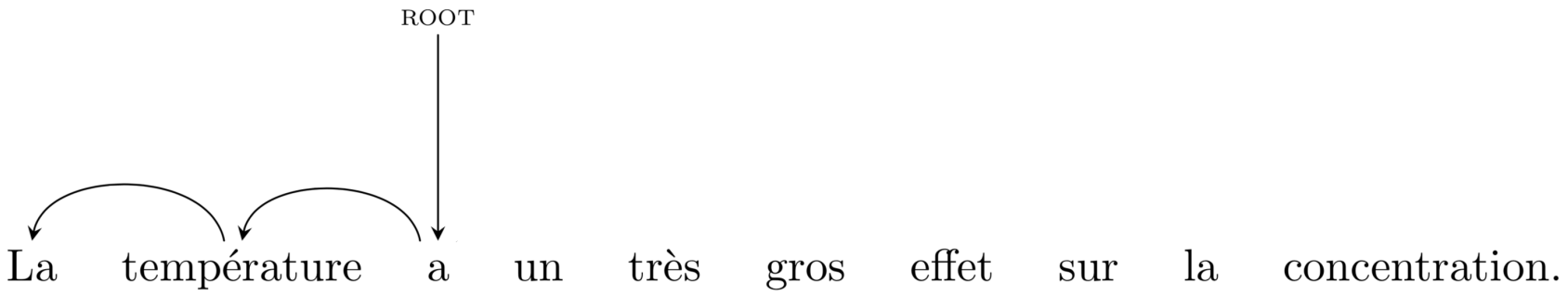
action: Shift



Arc-eager on an example

$[\text{ROOT}, \text{a}, \text{un}, \text{très}]_S$ $[\text{gros}, \text{effet}, \text{sur}, \text{la}, \text{concentration}]_Q$

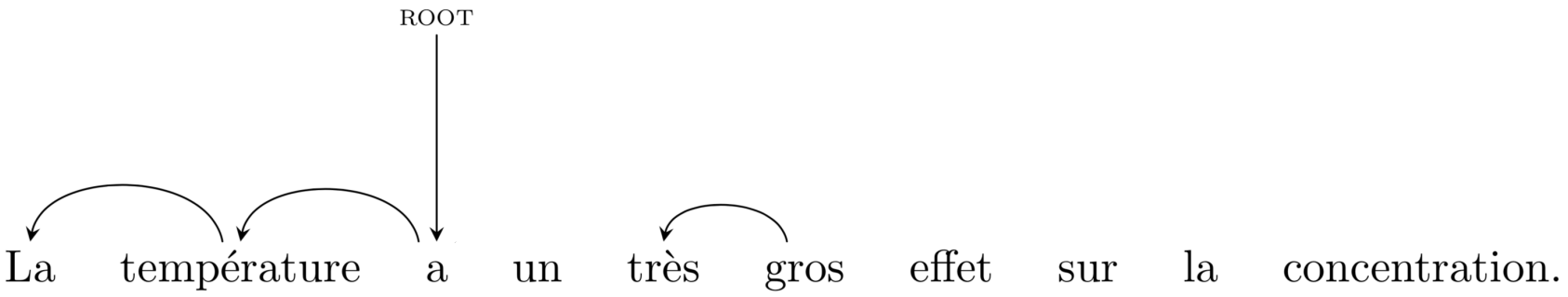
action: Shift



Arc-eager on an example

[ROOT, a, un, très]$_S$ [gros, effet, sur, la, concentration]$_Q$

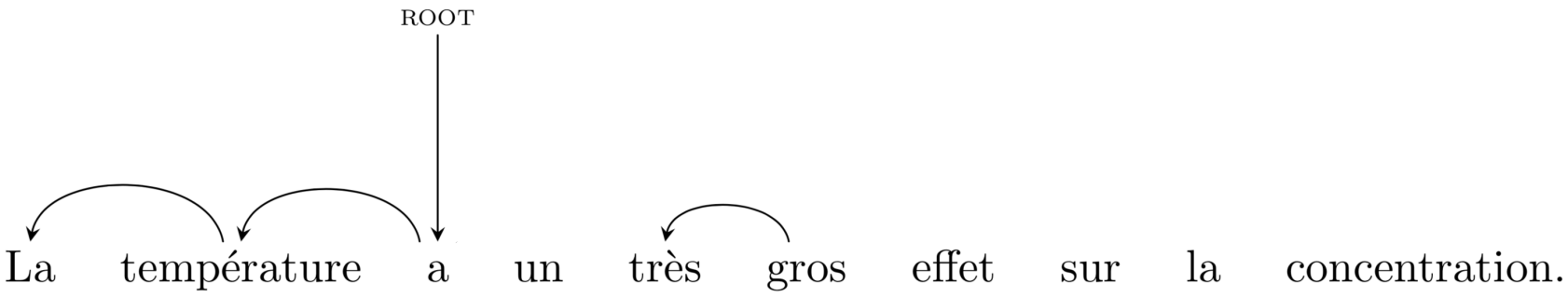
action: Left-Arc()



Arc-eager on an example

$[\text{ROOT, a, un, gros}]_S$ $[\text{effet, sur, la, concentration}]_Q$

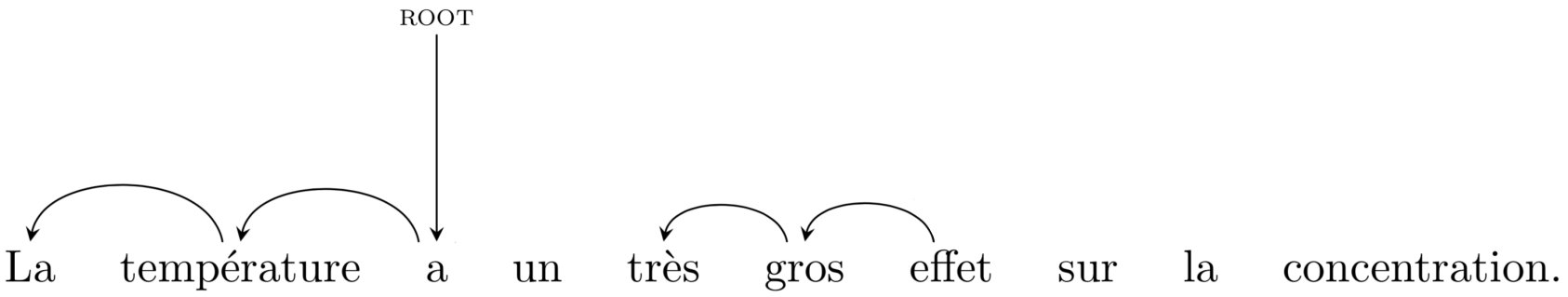
action: Shift



Arc-eager on an example

[ROOT, a, un, gros]$_S$ [effet, sur, la, concentration]$_Q$

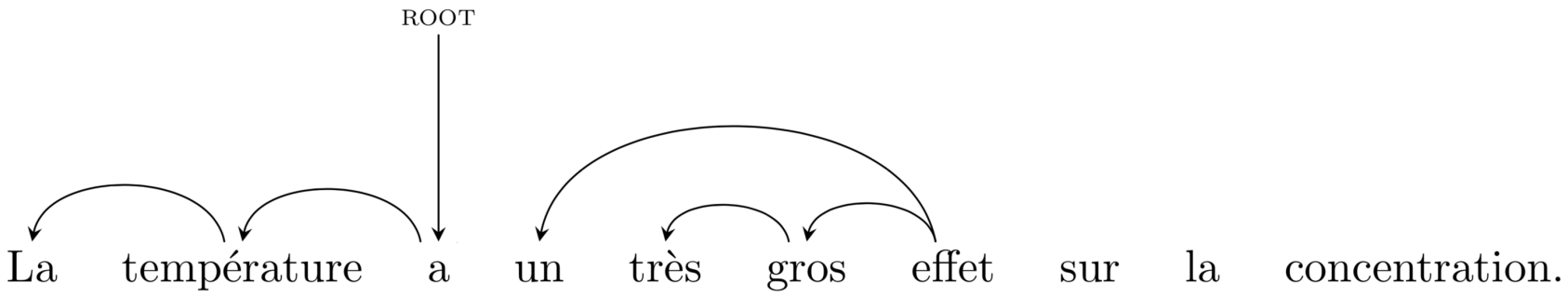
action: Left-Arc()



Arc-eager on an example

[ROOT, a, un]$_S$ [effet, sur, la, concentration]$_Q$

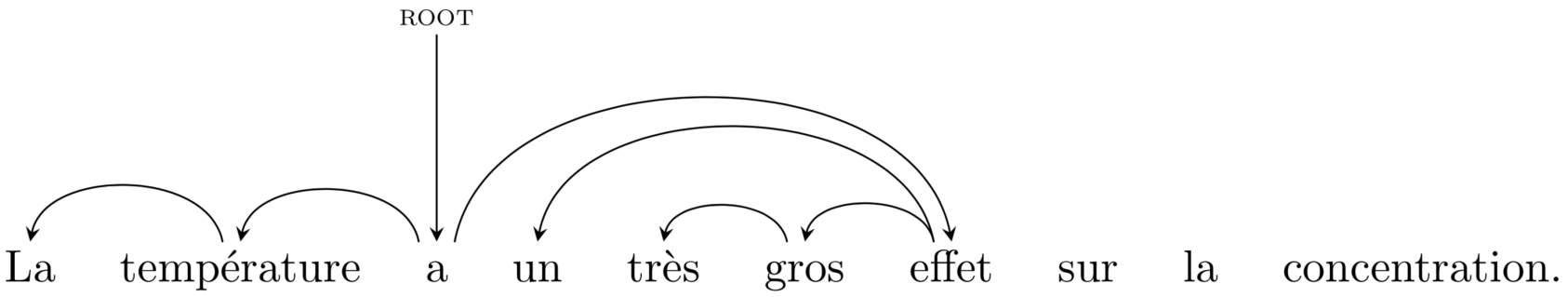
action: Left-Arc()



Arc-eager on an example

[ROOT, a, effet]$_S$ [sur, la, concentration]$_Q$

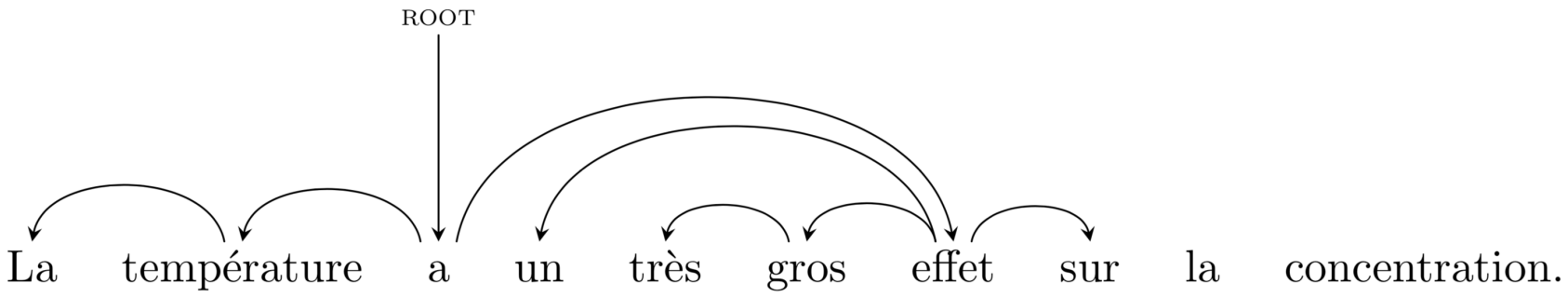
action: Right-Arc()



Arc-eager on an example

$[\text{ROOT, a, effet, sur}]_S$ $[\text{la, concentration}]_Q$

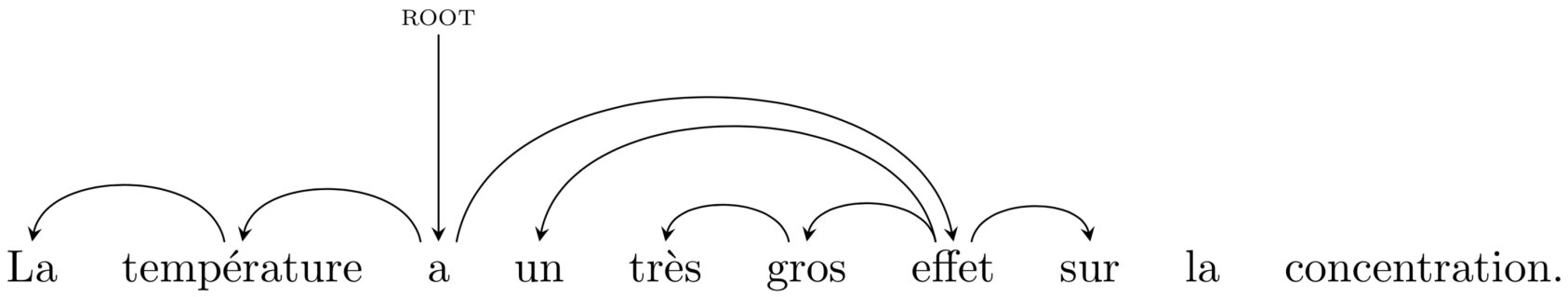
action: Right-Arc()



Arc-eager on an example

[ROOT, a, effet, sur, la]$_S$ [concentration]$_Q$

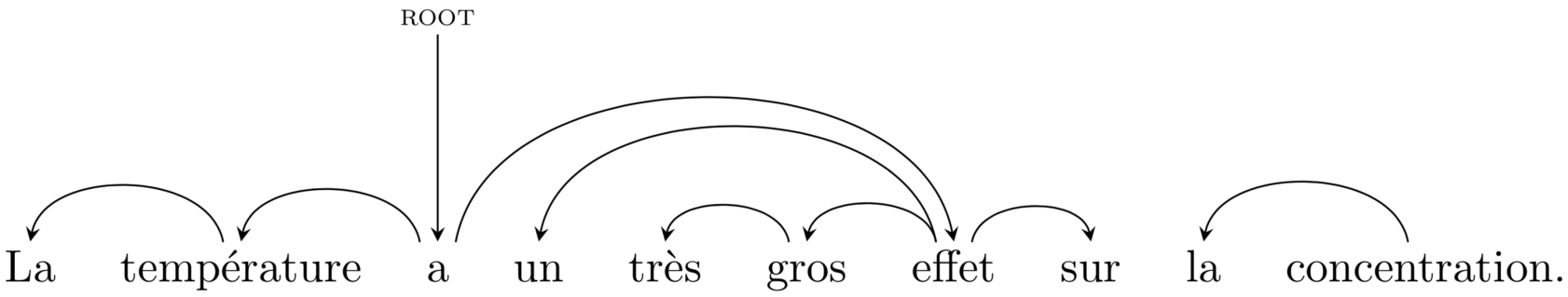
action: Shift()



Arc-eager on an example

[ROOT, a, effet, sur, la]$_S$ [concentration]$_Q$

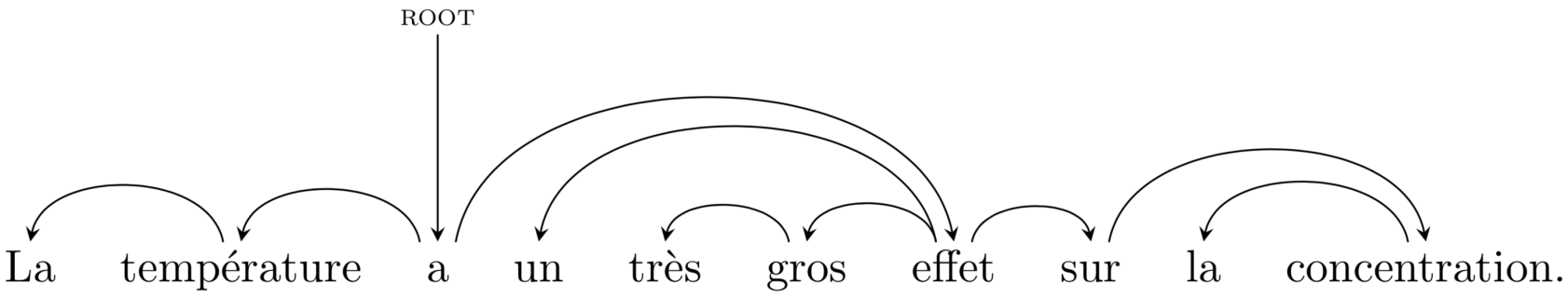
action: Left-Arc()



Arc-eager on an example

[ROOT, a, effet, sur, concentration]$_S$ []$_Q$

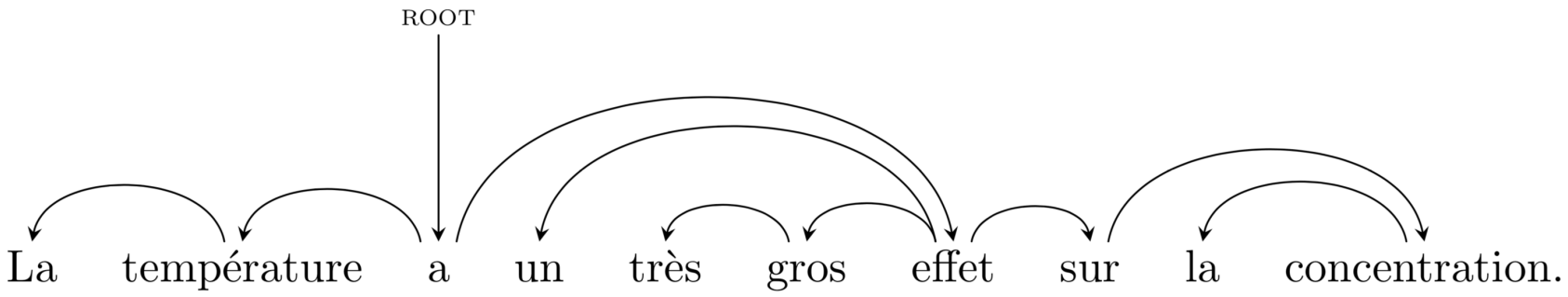
action: Right-Arc()



Arc-eager on an example

[ROOT, a, effet, sur, concentration]$_S$ []$_Q$

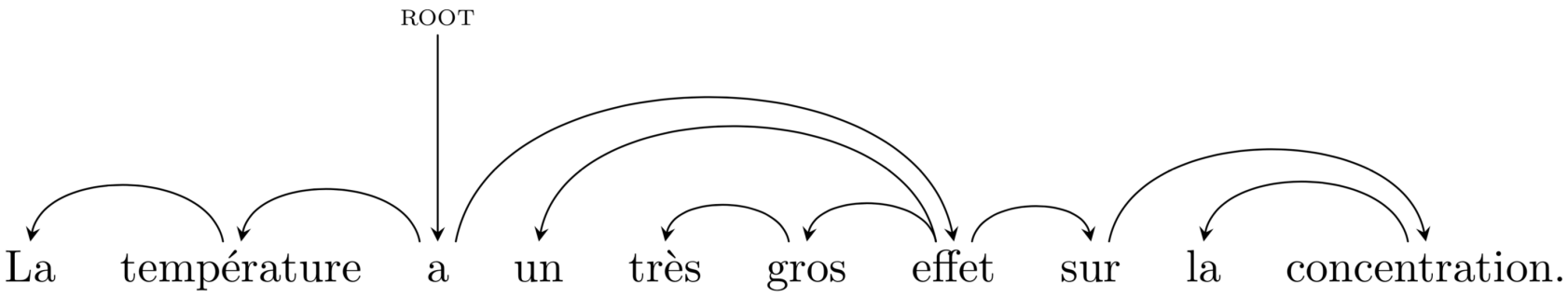
action: Reduce



Arc-eager on an example

$[\text{ROOT, a, effet, sur}]_S\ []_Q$

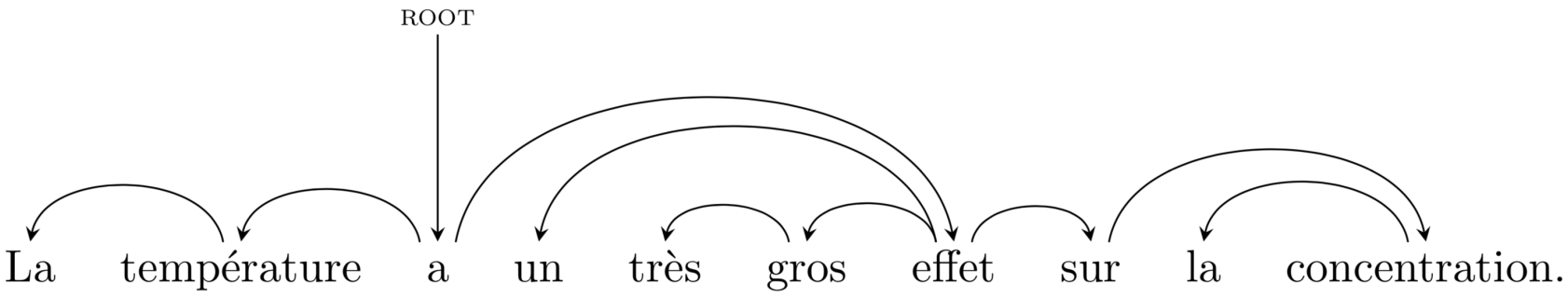
action: Reduce



Arc-eager on an example

$[\text{ROOT}, \text{a}, \text{effet}]_S\ []_Q$

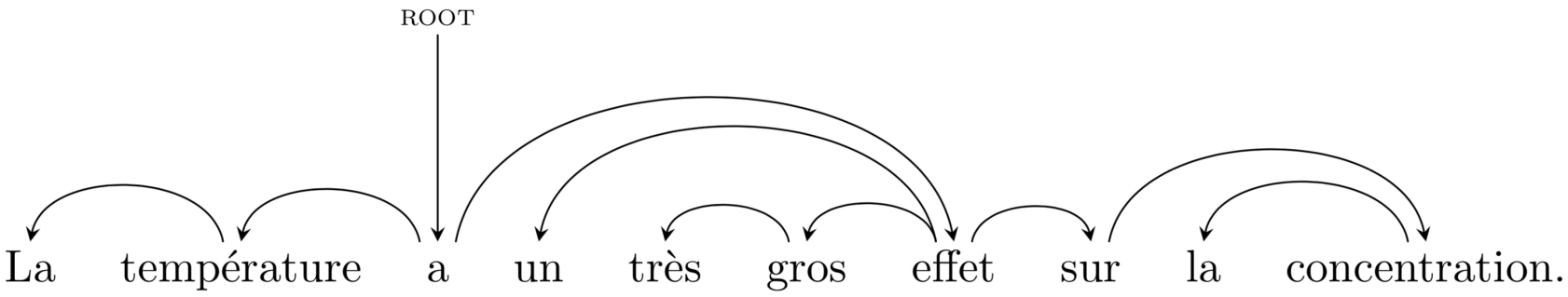
action: Reduce



Arc-eager on an example

[ROOT, a]$_S$ []$_Q$

action: Reduce



Drawbacks of the arc-eager algorithm

- The arc-eager system has a weaker soundness result than the arc-standard system
- It does not guarantee the output to be a dependency tree, only a sequence of (unconnected) trees (a forest).
- In the best case, this is a sequence of length 1, meaning that the tree is in fact a tree.
- In the worst case, this is a sequence of length n, meaning that each word is its own tree.
- The arc-eager parsers normally have a last step that attaches everything that remains in the stack to the root

Extending arc-eager: the case of disfluencies

Drawbacks of the arc-eager algorithm

- Two issues for spoken language processing:
  - ASR errors
  - Speech Disfluencies
    - ~10% of the words in conversational speech are disfluent
    - An extreme case of “noisy” input: http://www.youtube.com/watch?v=lj3iNxZ8Dww
  - Error propagation from these two errors can wreak havoc on downstream modules such as parsing and semantics

Drawbacks of the arc-eager algorithm

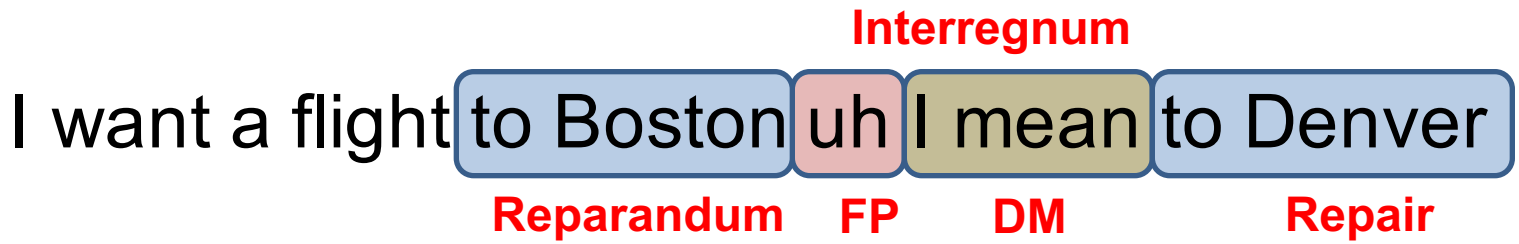
- Two is
  - ASR
  - Spee
    - ~ nt
    - A e.com/ w
  - Erro ak hav nd sem

Drawbacks of the arc-eager algorithm

- Two issues for spoken language processing:
  - ASR errors
  - Speech Disfluencies
    - ~10% of the words in conversational speech are disfluent
    - An extreme case of “noisy” input: http://www.youtube.com/watch?v=lj3iNxZ8Dww
  - Error propagation from these two errors can wreak havoc on downstream modules such as parsing and semantics

Disfluencies

- Three types
  - Filled pauses: e.g. uh, um
  - Discourse markers and parentheticals: e.g., I mean, you know
  - Reparandum (edited phrase)



Processing disfluent sentences

- Most approaches deal solely with disfluency detection as a pre-processing step before parsing
- Serialised method of disfluency detection and then parsing can be slow…
- Why not parse disfluent sentences at the same time as detecting disfluencies?
  - Advantage: speed-up processing, especially for dialogue systems
- **Joint dependency parsing and disfluency detection** with high accuracy and processing speed (Rasooli and Tetrault 2013)
  - Source: *I want a flight to Boston uh I mean to Denver*
  - Output: *I want a flight to Denver*

Rasooli and Tetrault' (2014) proposal

- Three new actions:
  - **IJ(*k*)**: remove the first *k* words from the queue and tag them as interjection
  - **DM(*k*)**: remove the first *k* words from the queue and tag them as discourse marker
  - **RP(*i*,*j*)**: remove *i* to *j* in the stack and tag them as reparandum

On an example

[ROOT]$_S$ [I, want, a, flight, to, Boston, uh, I, mean, to, Denver]$_Q$

I want a flight to Boston uh I mean to Denver

On an example

[ROOT, I]$_S$ [want, a, flight, to, Boston, uh, I, mean, to, Denver]$_Q$

action: Shift()

I want a flight to Boston uh I mean to Denver

On an example

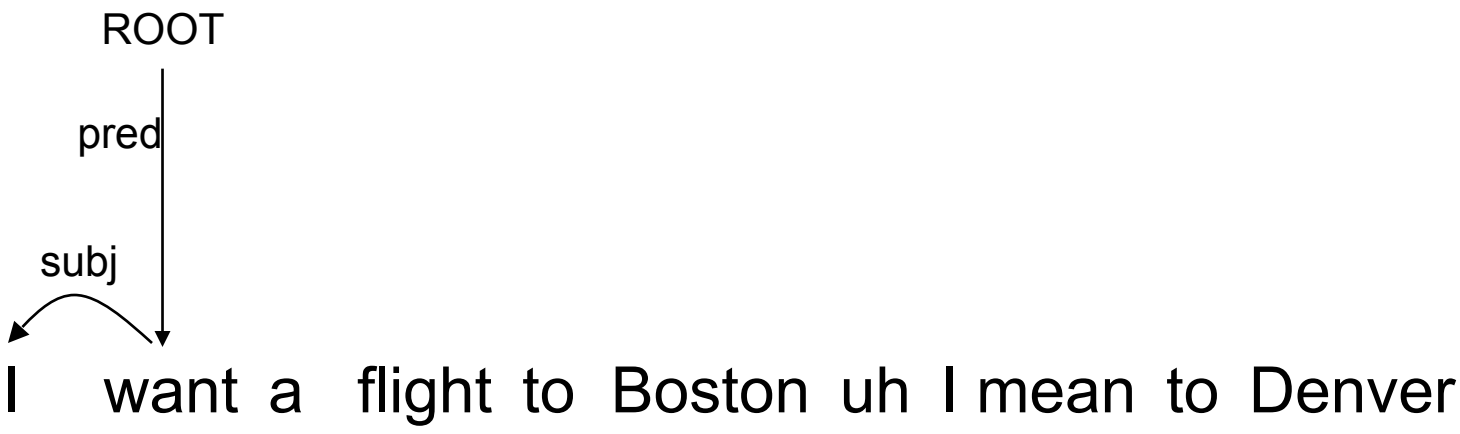
[ROOT, I]$_S$ [want, a, flight, to, Boston, uh, I, mean, to, Denver]$_Q$

action: Left-Arc(subj)

On an example

[ROOT, want]$_S$ [a, flight, to, Boston, uh, I, mean, to, Denver]$_Q$

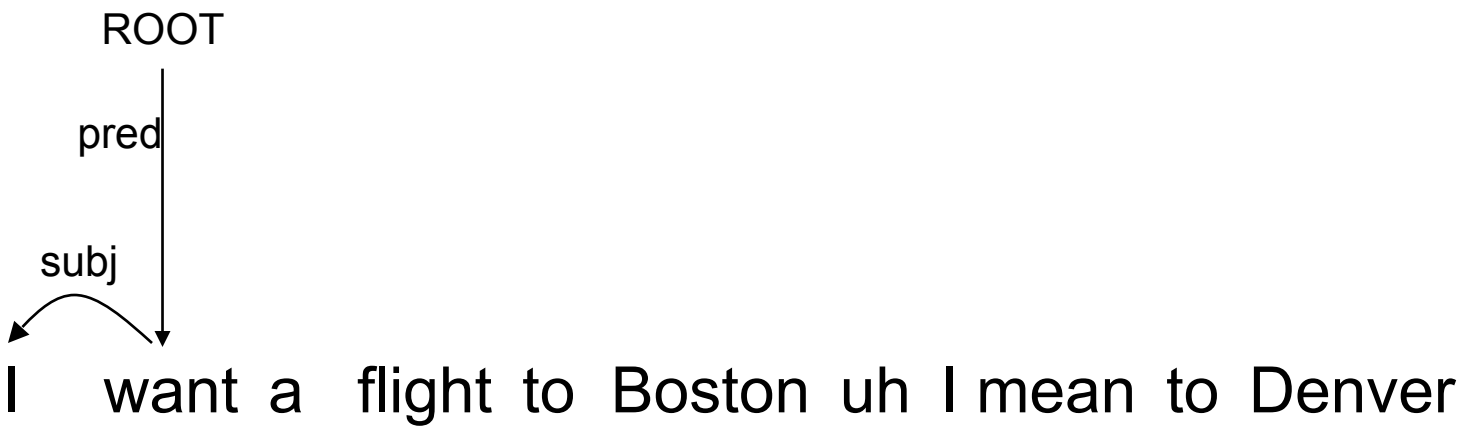
action: Right-Arc(PRED)



On an example

[ROOT, want, a]$_S$ [flight, to, Boston, uh, I, mean, to, Denver]$_Q$

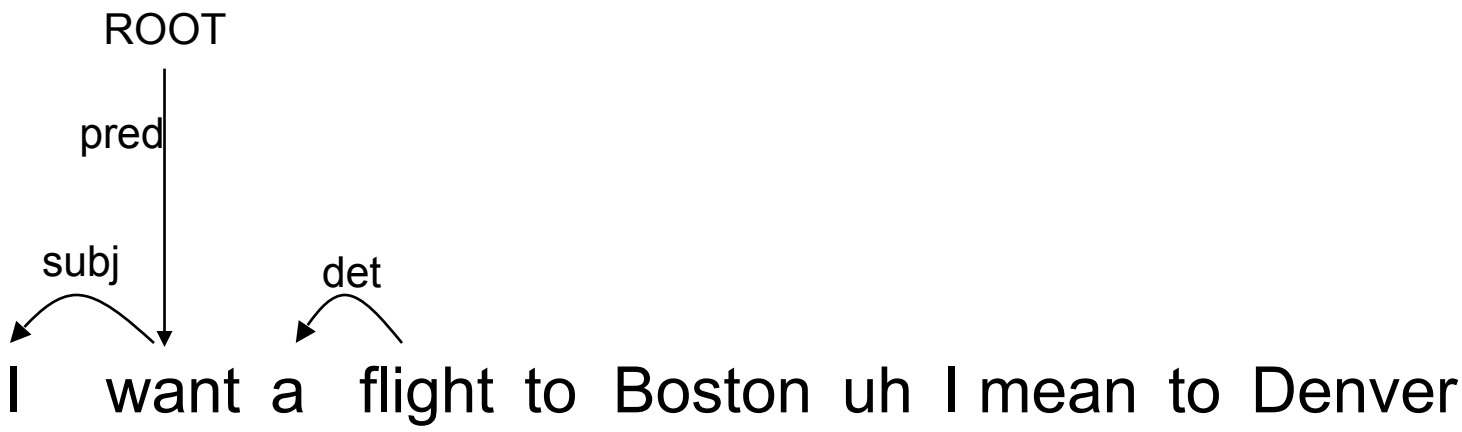
action: Shift()



On an example

[ROOT, want, a]$_S$ [flight, to, Boston, uh, I, mean, to, Denver]$_Q$

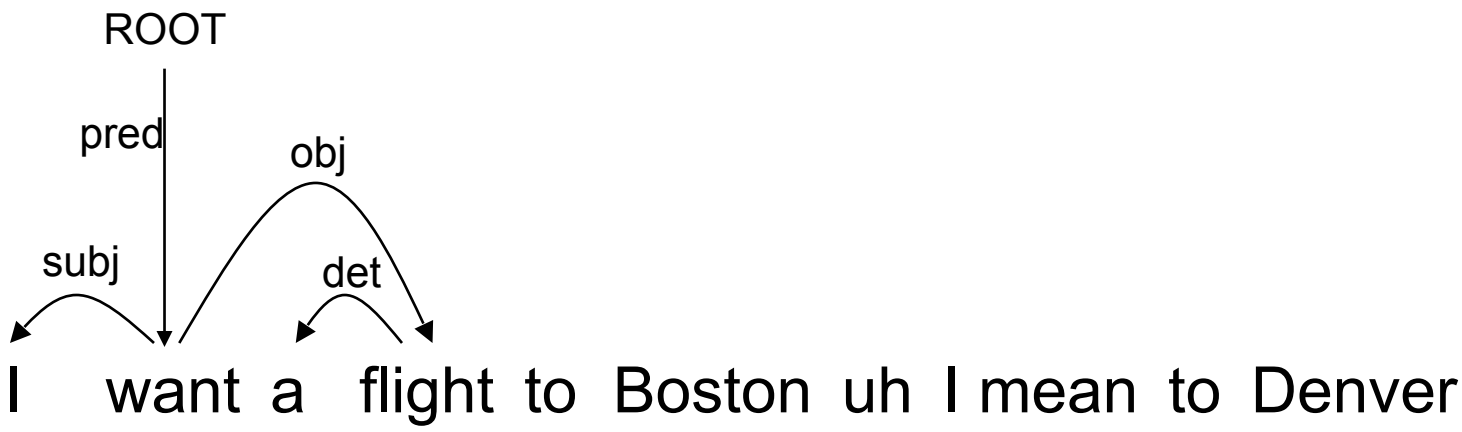
action: Left-Arc(DET)



On an example

$[\text{ROOT, want, flight}]_S$ [to, Boston, uh, I, mean, to, Denver]$_Q$

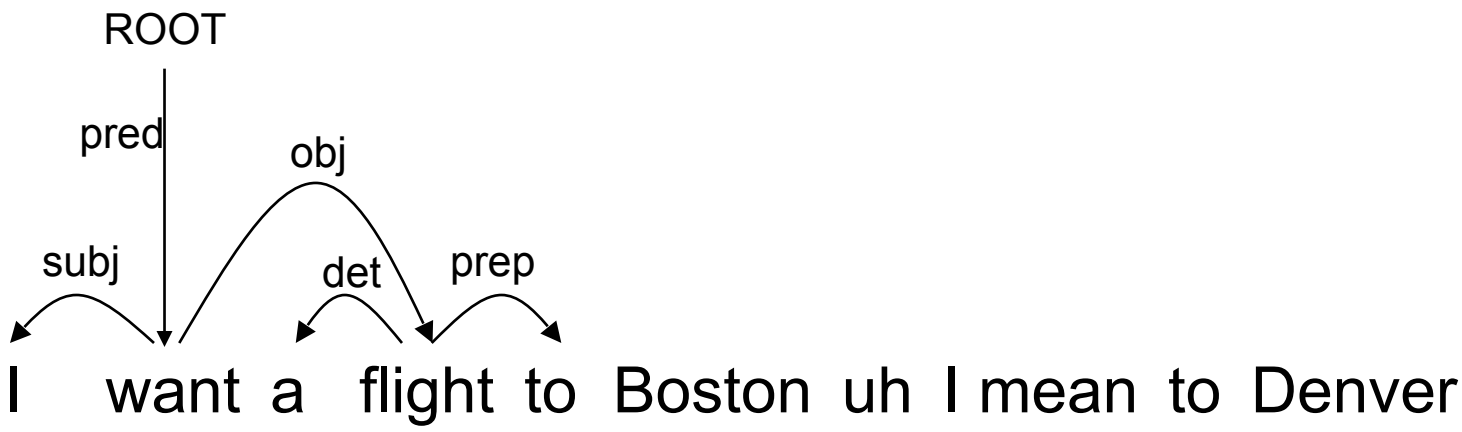
action: Right-Arc(OBJ)



On an example

[ROOT, want, flight, to]$_S$ [Boston, uh, I, mean, to, Denver]$_Q$

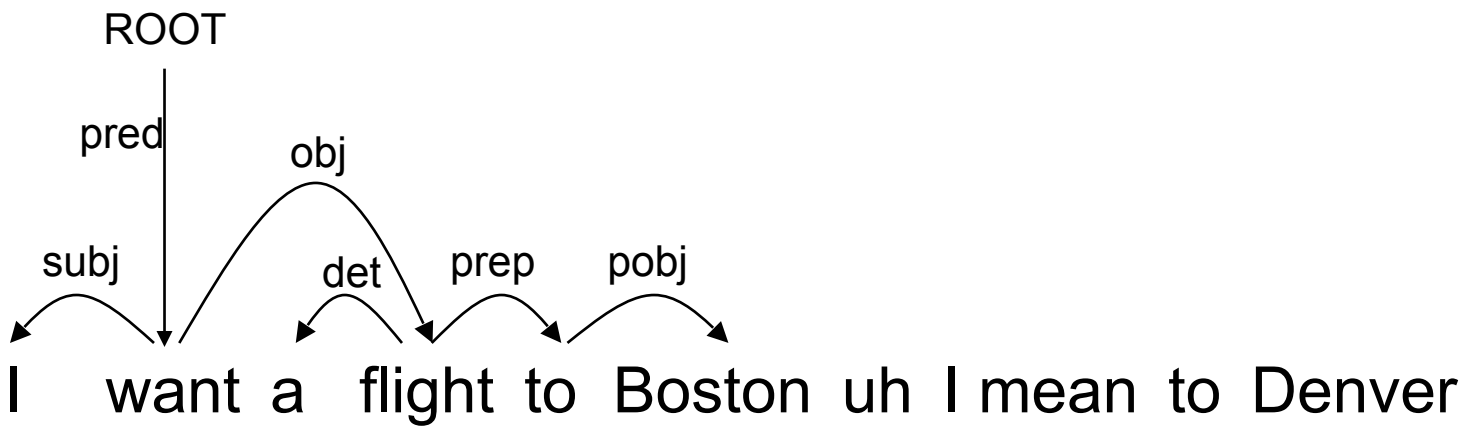
action: Right-Arc(PREP)



On an example

[ROOT, want, flight, to, Boston]$_S$ [uh, I, mean, to, Denver]$_Q$

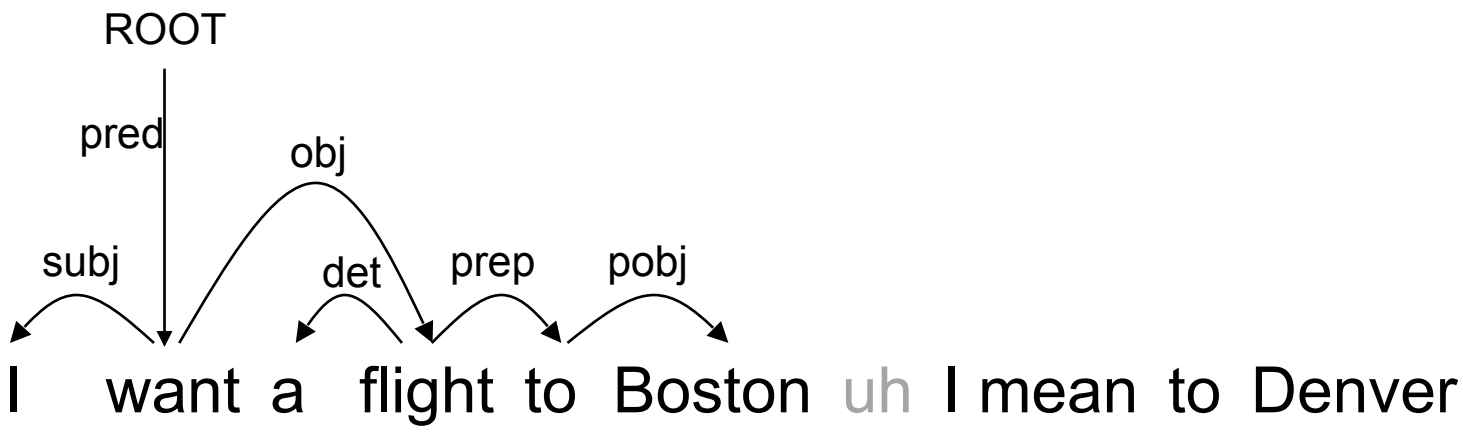
action: Right-Arc(POBJ)



On an example

[ROOT, want, flight, to, Boston]$_S$ [uh, I, mean, to, Denver]$_Q$

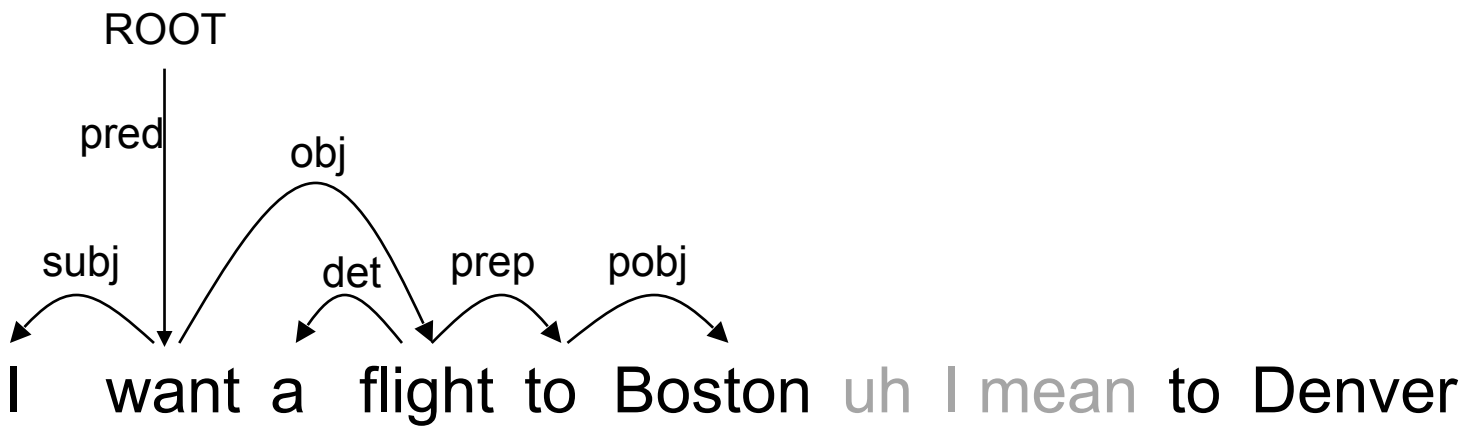
action: IJ



On an example

[ROOT, want, flight, to, Boston]$_S$ [I, mean, to, Denver]$_Q$

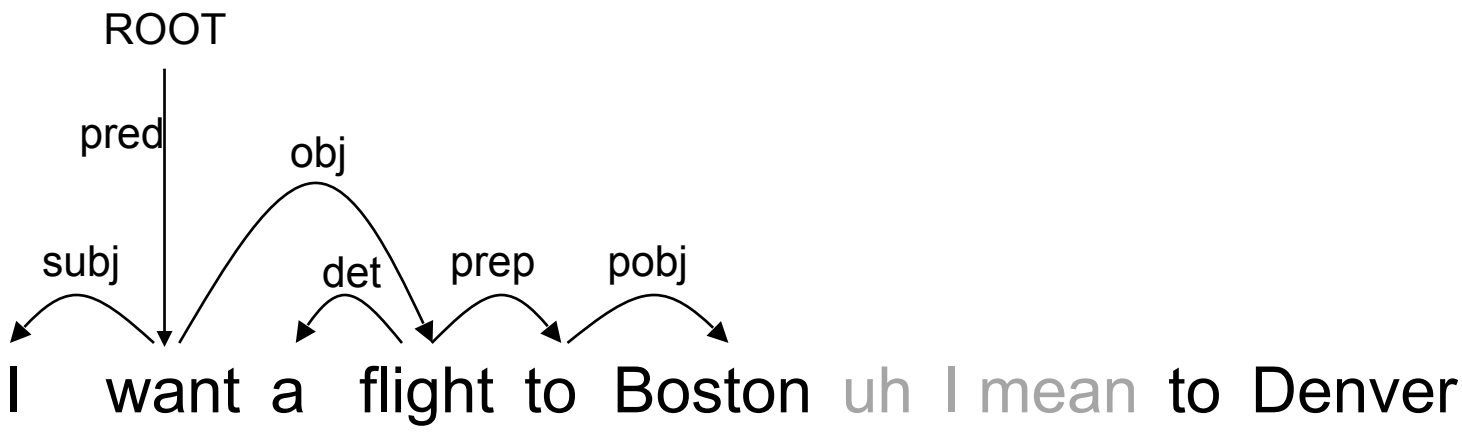
action: DM(2)



On an example

[ROOT, want, flight, to, Boston]$_S$ [to, Denver]$_Q$

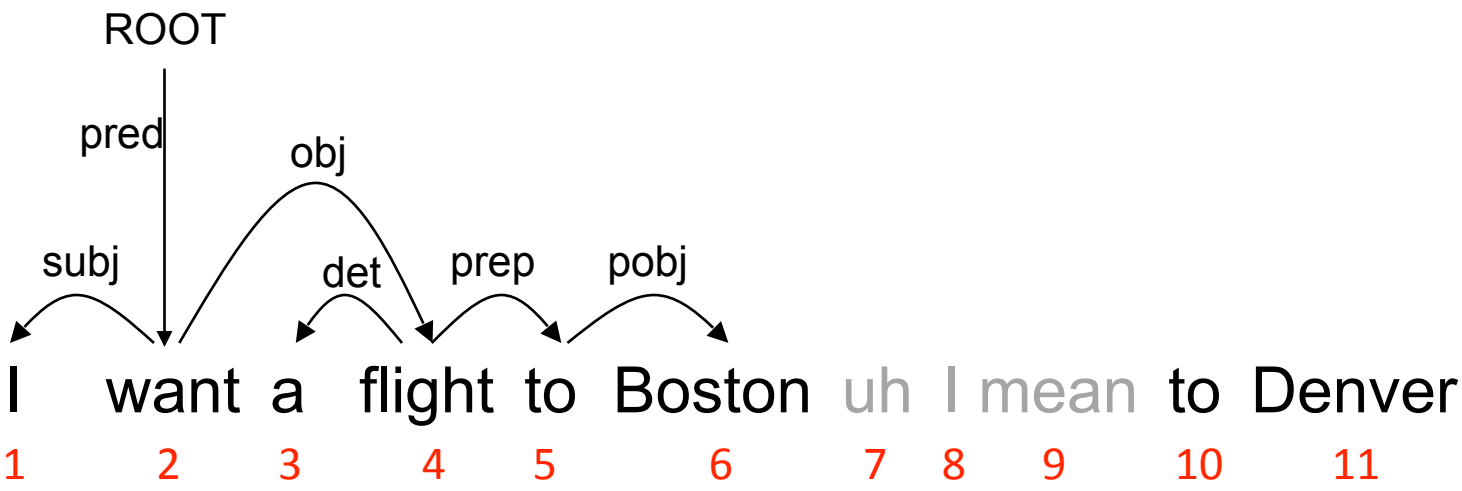
action: RP(5..6)



On an example

$[\text{ROOT, want, flight, to, Boston}]_S$ $[\text{to, Denver}]_Q$

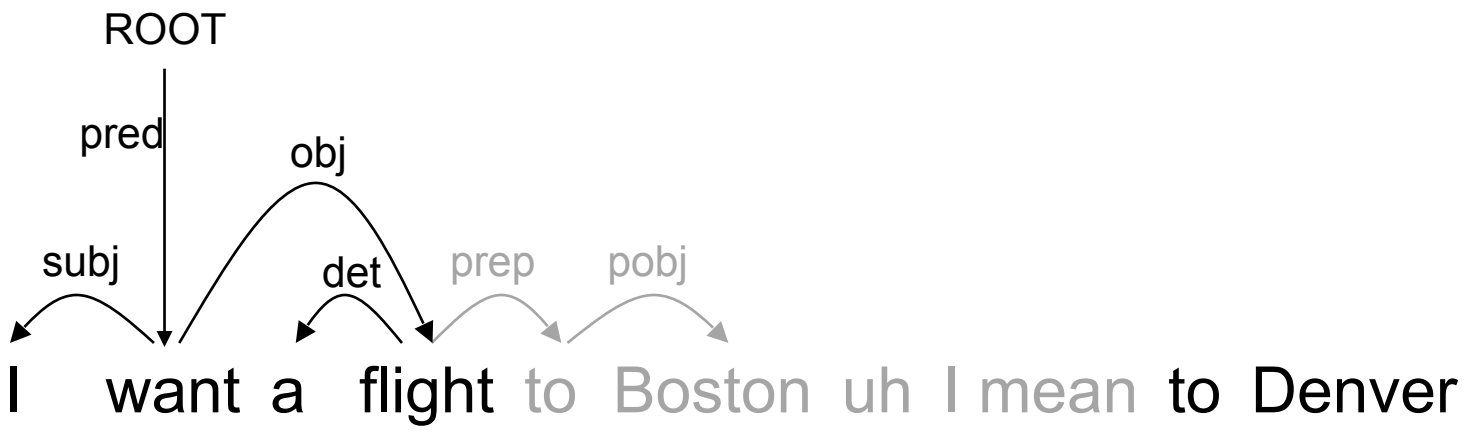
action: RP(5..6)



On an example

[ROOT, want, flight, to, Boston]$_S$ [to, Denver]$_Q$

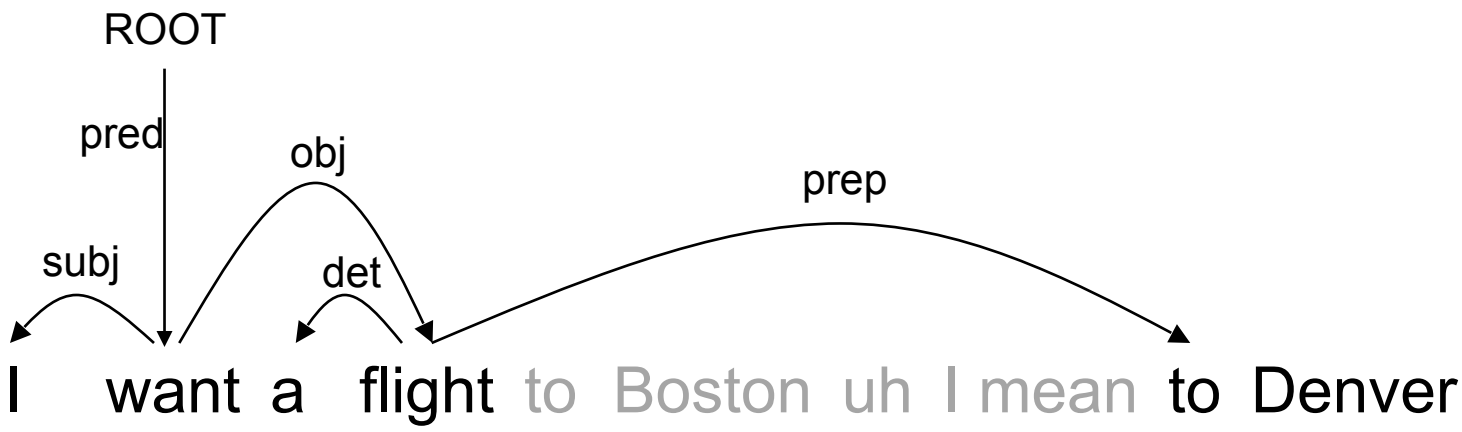
action: RP(5..6)



On an example

$[\text{ROOT, want, flight, to, Boston}]_S\ [\text{to, Denver}]_Q$

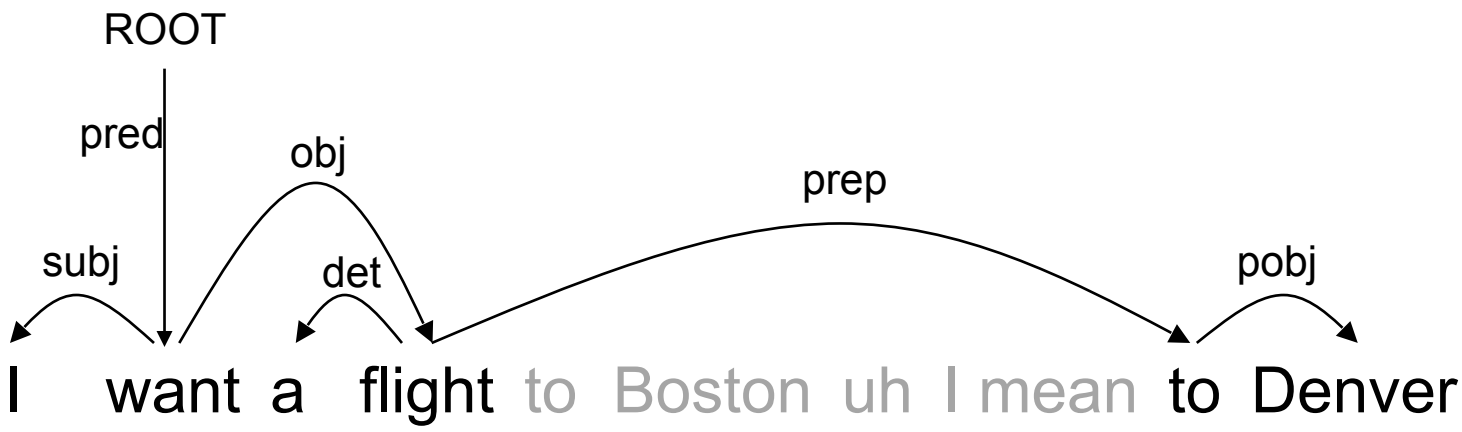
action: Right-Arc(PREP)



On an example

[ROOT, want, flight, to, Boston]$_S$ [Denver]$_Q$

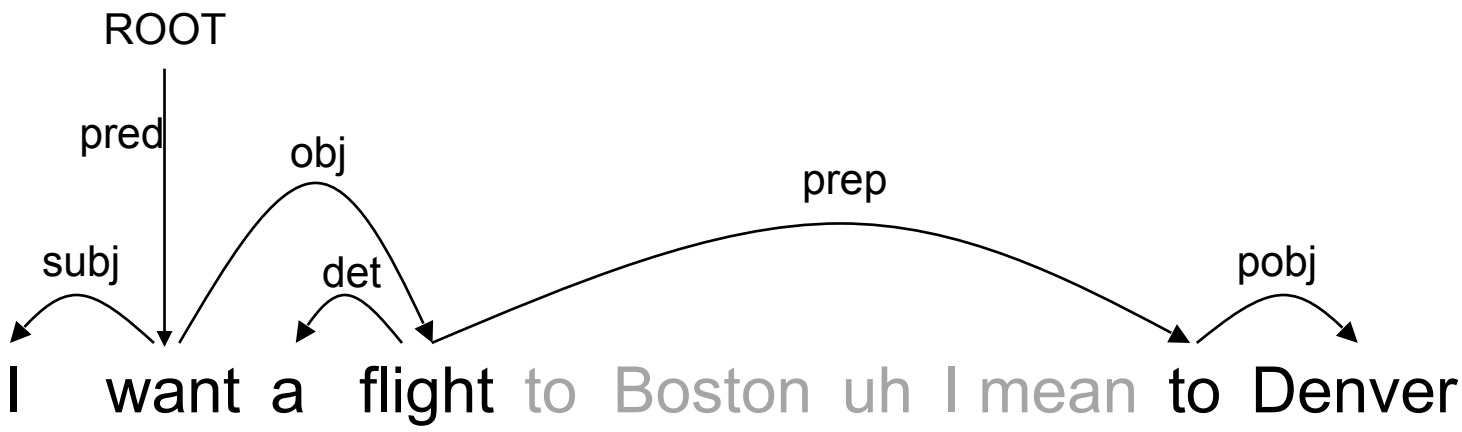
action: Right-Arc(POBJ)



On an example

[ROOT, want, flight, to, Boston]$_S$ []$_Q$

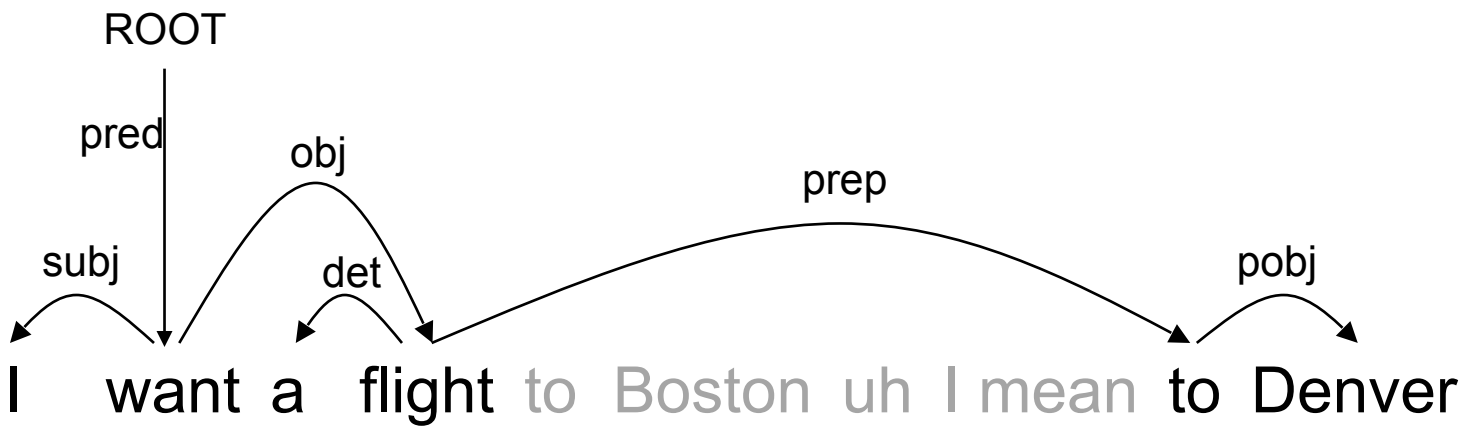
action: Reduce



On an example

[ROOT, want, flight, to]$_S$ []$_Q$

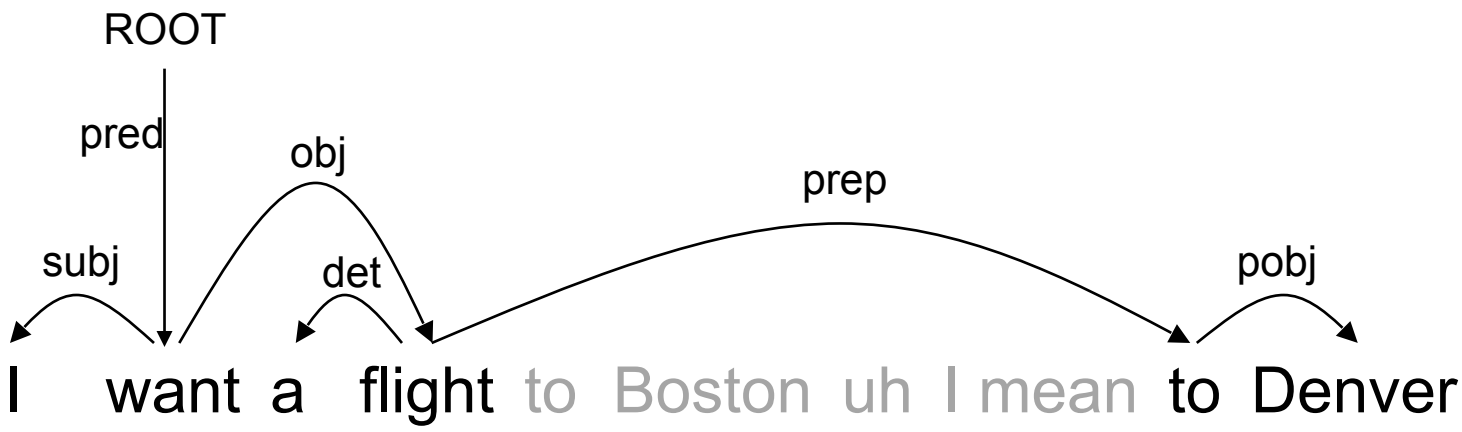
action: Reduce



On an example

$[\text{ROOT, want, flight}]_S\ []_Q$

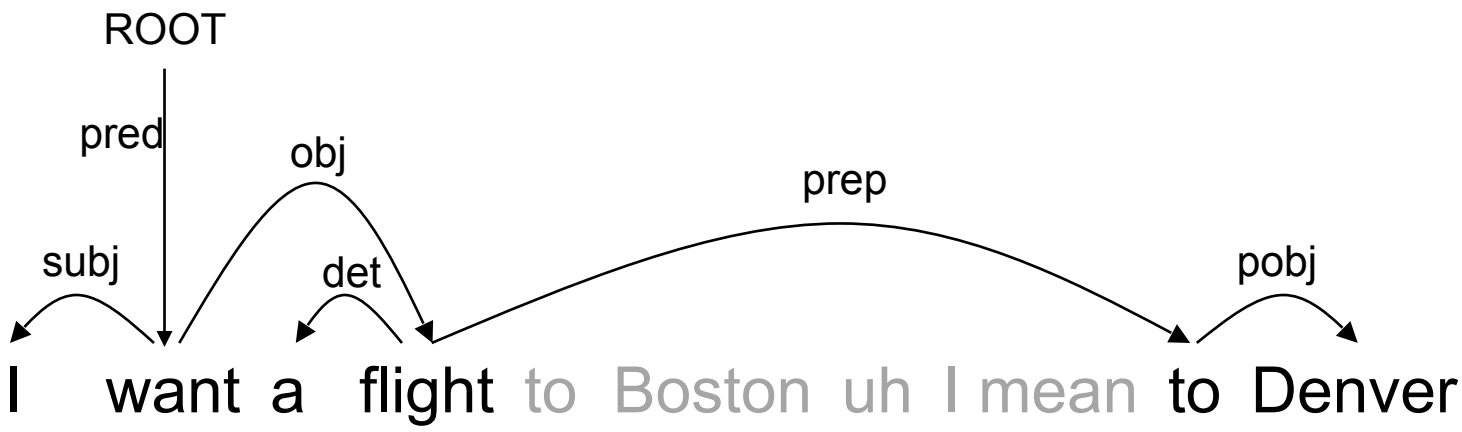
action: Reduce



On an example

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

action: Reduce



Evaluation

Model	Description	F-score
Miller and Schuler (2008)	Joint + PCFG Parsing	30.6
Lease and Johnson (2006)	Joint + PCFG Parsing	62.4
Kahn et al. (2005)	TAG + LM rerank	78.2
Qian and Lui, (2013)	IOB tagging	82.5
Rasooli and Tetrault (2013)	**Arc-Eager Parsing**	**81.4**
Rasooli and Tetrault (2014)	Arc-Eager Parsing	82.6

Neural Transition-Based Parsing

Example of a neural arc-standard algorithm

- Chen and Manning (2014)
- Replace the feature-based action selection module by a neural network

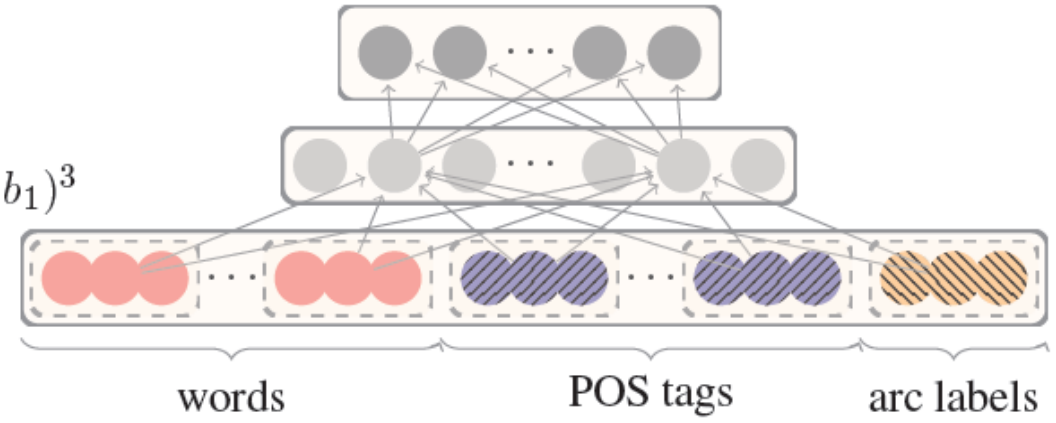
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

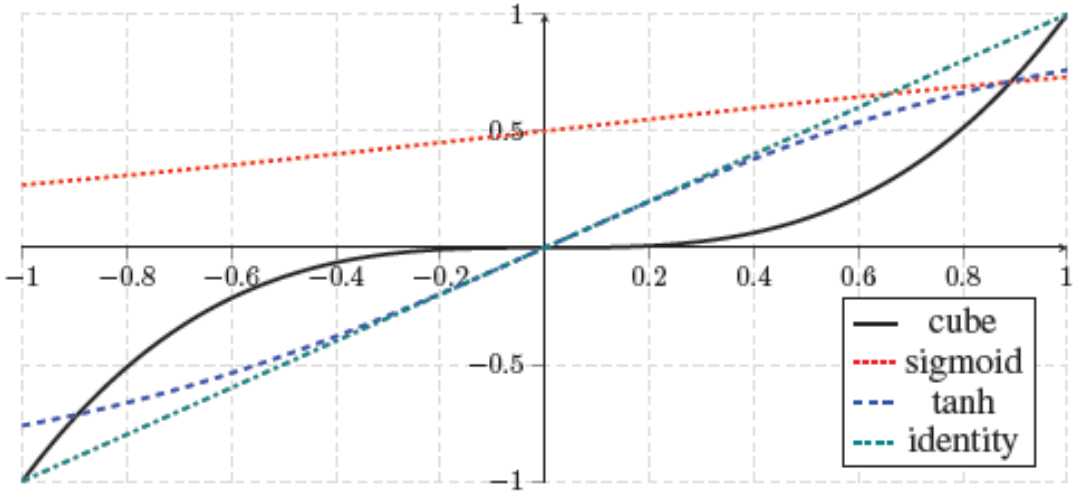
$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$



Example of a neural arc-standard algorithm

- Chen and Manning (2014)
- Replace the feature-based action selection module by a neural network
- Cube activation function



$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Example of a neural arc-standard algorithm

- Chen and Manning (2014)
- Replace the feature-based action selection module by a neural network
- Cube activation function
- POS tags and arc labels are discrete sets
  - Normally represented as one-hot vectors
  - Just like words, there should be similarities
    - NN (singular noun) should be similar to NNP (plural noun)
  - Dense embedding layer for POS tags and arc labels should capture relationships

Example of a neural arc-standard algorithm: Experimental Results

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	89.9	88.7	89.7	88.3	51
eager	90.3	89.2	89.9	88.6	63
Malt:sp	90.0	88.8	89.9	88.5	560
Malt:eager	90.1	88.9	90.1	88.7	535
MSTParser	92.1	90.8	**92.0**	90.5	12
Our parser	**92.2**	**91.0**	**92.0**	**90.7**	**1013**

Table 4: Accuracy and parsing speed on PTB + CoNLL dependencies.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	**92.0**	**89.7**	**91.8**	**89.6**	**654**

Table 5: Accuracy and parsing speed on PTB + Stanford dependencies.

Parser	Dev		Test		Speed
	UAS	LAS	UAS	LAS	(sent/s)
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	**84.0**	82.1	83.0	81.2	6
Our parser	**84.0**	**82.4**	**83.9**	**82.4**	**936**

Table 6: Accuracy and parsing speed on CTB.

Example of a neural arc-standard algorithm: Model comparison

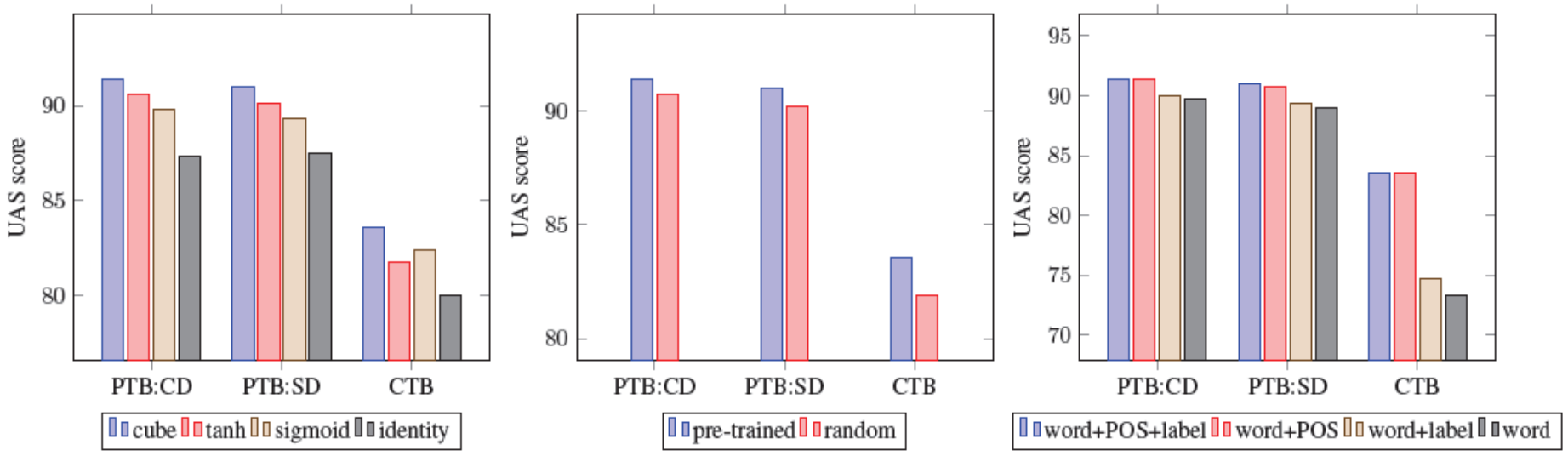


Figure 4: Effects of different parser components. Left: comparison of different activation functions. Middle: comparison of pre-trained word vectors and random initialization. Right: effects of POS and label embeddings.

Practical assignment 5

(will count 4 times less than other assignments)

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Google Translate translation:
The OL camp ordered 1500 eggs through translating through Google Translate. But it was wrong. 15,000 was delivered to the door. We wish good luck and hope that the Norwegian golden hopes are happy - very happy - in eggs

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation

Google Translate translation:
I'm not in the office right now.
Send any work to translate it.

Goal

- Understand the current state of the art in machine translation

Goal

- Understand the current state of the art in machine translation
- Use whichever (freely available) machine translation system to investigate the following questions:
  - Which syntactic constructions are correctly translated? Which ones are not? Can you guess why?
    - What about questions, long and/or complex sentences, sentences in the first person, idioms (*kick the bucket* vs. *casser sa pipe*), named entities (*London*)…
  - What is the influence of the language pair? Cf. English<->French vs. English<->Chinese or English<->(a rare language)
  - What is the influence of the context. E.g., how are the following discourse translated (English <-> French):
    - *A mouse appeared. It looked hungry.*
    - *Il aime bien le mouton. Surtout les côtelettes de mouton.*

Assigment deliverable

- A 1- to 2-page report discussing these questions, with examples

https://github.com/edupoux/MVA_2018_SL/tree/master/TD_%235


That’s all for today!