Non-Projective Dependency Parsing

Based partly on the Kübler et al (2009) book

NASSLLI short course on Dependency Parsing
Summer 2010

Markus Dickinson & Sandra Kübler

Projectivity (review)

An arc $(w_i, r, w_j) \in A$ is *projective* iff $w_i \rightarrow^* w_k$ for all:

- $i < k < j$ when $i < j$
- $j < k < i$ when $j < i$

Covington's Algorithm (review)

How is it that Covington's algorithm allows for non-projectivity?

- Deterministic incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington(2001)]:

PARSE($x = (w_1, \ldots, w_n)$)
for $i = 1$ **up to** n
  for $j = i - 1$ **down to** 1
    LINK(w_i, w_j)

$$\text{LINK}(w_i, w_j) = \begin{cases} E \leftarrow E \cup (i,j) & \text{if } w_j \text{ is a dependent of } w_i \\ E \leftarrow E \cup (j,i) & \text{if } w_i \text{ is a dependent of } w_j \\ E \leftarrow E & \text{otherwise} \end{cases}$$

Nivre's Algorithm (review)

Why is Nivre's algorithm only projective?

- Four parsing actions:

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

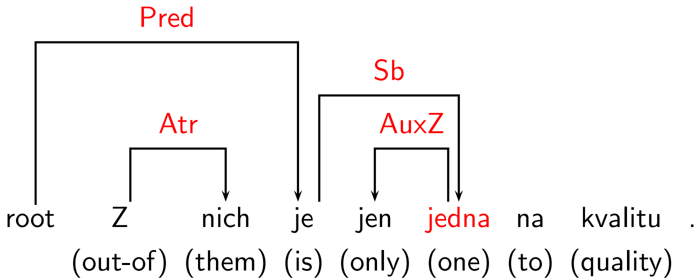
$$\text{Reduce} \quad \frac{[\ldots, w_i]_S \quad [\ldots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [\ldots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [w_j, \ldots]_Q \quad w_i \stackrel{r}{\leftarrow} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_j}{[\ldots, w_i, w_j]_S \quad [\ldots]_Q \quad w_i \stackrel{r}{\rightarrow} w_j}$$

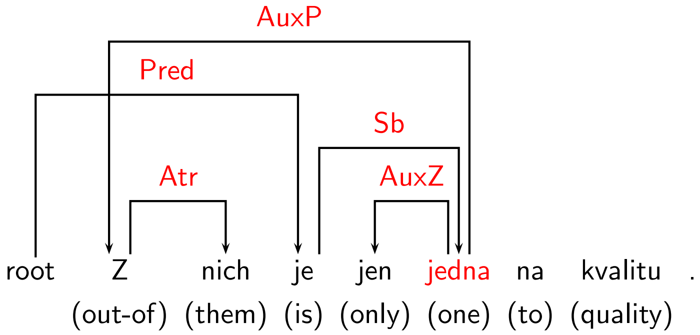
Limitations of some transition-based parsing

Why won't some transition-based parsers (e.g., Nivre's) handle non-projective structures?

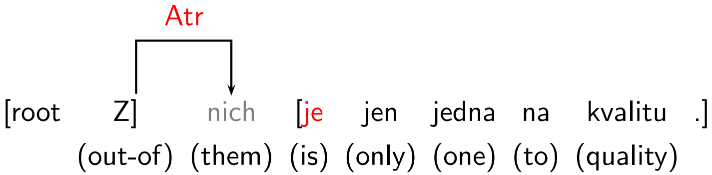


Limitations of some transition-based parsing

Why won't some transition-based parsers (e.g., Nivre's) handle non-projective structures?



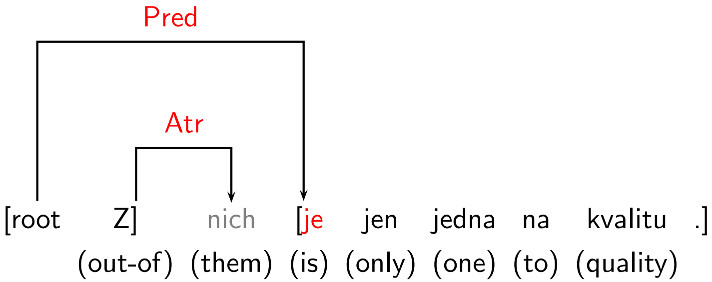
Allowable transitions



At this stage:

- *Z* is on top of the stack, and *je* is on top of the buffer

Allowable transitions



At this stage:

- *Z* is on top of the stack, and *je* is on top of the buffer
- We want an arc between root and *je*, leaving *Z* unattached

Non-Projective Dependency Parsing

- Many parsing algorithms are restricted to projective dependency graphs.
- Is this a problem?
- Statistics from CoNLL-X Shared Task [Buchholz and Marsi(2006)]
  - NPD = Non-projective dependencies
  - NPS = Non-projective sentences

Language	**%NPD**	**%NPS**
Dutch	5.4	36.4
German	2.3	27.8
Czech	1.9	23.2
Slovene	1.9	22.2
Portuguese	1.3	18.9
Danish	1.0	15.6

Two Main Approaches

- Algorithms for non-projective dependency parsing:
  - Constraint satisfaction methods [Tapanainen and Järvinen(1997), Duchier and Debusmann(2001), Foth et al.(2004)Foth, Daum and Menzel]
  - McDonald's spanning tree algorithm [McDonald et al.(2005)McDonald, Pereira, Ribarov and Hajič]
  - Covington's algorithm [Nivre(2006)]
- Post-processing of projective dependency graphs:
  - Pseudo-projective parsing [Nivre and Nilsson(2005)]
  - Corrective modeling [Hall and Novák(2005)]
  - Approximate non-projective parsing [McDonald and Pereira(2006)]

Non-Projective Parsing Algorithms

- Complexity considerations:
  - Projective (Proj)
  - Non-projective (NonP)

Problem/Algorithm	Proj	NonP
Complete grammar parsing [Gaifman(1965), Neuhaus and Bröker(1997)]	P	NP hard
Deterministic parsing [Nivre(2003), Covington(2001)]	$O(n)$	$O(n^2)$
First order spanning tree [McDonald et al.(2005)McDonald, Pereira, Ribarov and Hajič]	$O(n^3)$	$O(n^2)$
Nth order spanning tree ($N > 1$) [McDonald and Pereira(2006)]	P	NP hard

Changing the transition system

One way to allow for a limited amount of non-projectivity is to change the definitions of the transitions

Transition		Preconditio
NP-Left$_r$	$(\sigma \vert w_i \vert w_k, w_j \vert \beta, A) \Rightarrow (\sigma \vert w_k, w_j \vert \beta, A \cup \{(w_j, r, w_i)\})$	$i \neq 0$
NP-Right$_r$	$(\sigma \vert w_i \vert w_k, w_j \vert \beta, A) \Rightarrow (\sigma \vert w_i, w_k \vert \beta, A \cup \{(w_i, r, w_j)\})$	

These definitions apply to the second word on the stack

- Top word on the stack is treated as a context node
- Queue is now allowed to have items re-inserted on it

Changing the transition system

One way to allow for a limited amount of non-projectivity is to change the definitions of the transitions

Transition		Precondition
NP-Left$_r$	$(\sigma\|w_i\|w_k, w_j\|\beta, A) \Rightarrow (\sigma\|w_k, w_j\|\beta, A \cup \{(w_j, r, w_i)\})$	$i \neq 0$
NP-Right$_r$	$(\sigma\|w_i\|w_k, w_j\|\beta, A) \Rightarrow (\sigma\|w_i, w_k\|\beta, A \cup \{(w_i, r, w_j)\})$	

These definitions apply to the second word on the stack

- Top word on the stack is treated as a context node
- Queue is now allowed to have items re-inserted on it

Will this work for *A hearing is scheduled on the issue today?* (see handout based on [Nivre(2009)])

Changing the transition system (2)

Another approach is to define configurations with two stacks instead of one

Idea:

- A No-arc transition allows a node to be passed from the stack to an auxiliary stack without having been given a head
- Later, this node can return to the stack and be in a dependency relation
  - i.e., save the node for later analysis

See [Nivre(2008)] (sec. 5.1) for more details

Changing the transition system (3)

Another way to alter the transition system is to add a transition Swap

Transition		Precondition
Swap	$(\sigma \mid w_i \mid w_j, \beta, A) \Rightarrow (\sigma \mid w_j, w_i \mid \beta, A)$	$0 < i < j$

Consider: *A* **hearing** *is scheduled* **on** *the issue today .*

Transition	Stack	Buffer	Arc
...	[root$_0$, ..., scheduled$_4$, on$_5$]	[the$_6$, ..., .$_9$]	
Swap	[root$_0$, ..., is$_3$, on$_5$]	[scheduled$_4$, ..., .$_9$]	
Swap	[root$_0$, ..., hearing$_3$, on$_5$]	[is$_3$, ..., .$_9$]	
...	...	...	

See [Nivre(2009)] for more details

Post-Processing

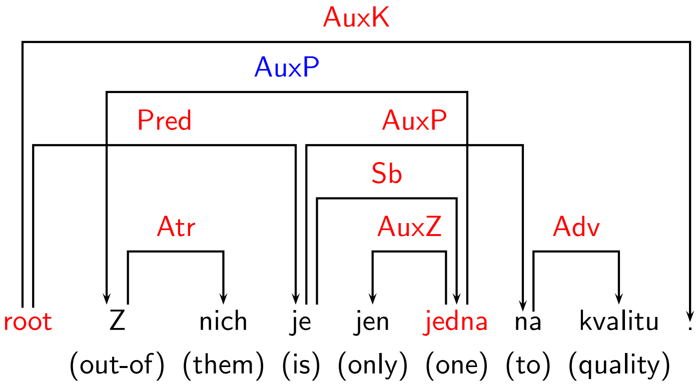
- Two-step approach:
  1. Derive the best projective approximation of the correct (possibly) non-projective dependency graph.
  2. Improve the approximation by replacing projective arcs by (possibly) non-projective arcs.
- Rationale:
  - Most "naturally occurring" dependency graphs are primarily projective, with only a few non-projective arcs.
- Approaches:
  - Pseudo-projective parsing [Nivre and Nilsson(2005)]
  - Corrective modeling [Hall and Novák(2005)]
  - Approximate non-projective parsing [McDonald and Pereira(2006)]

Pseudo-Projective Parsing

1. Projectivize dependency trees in the training set while encoding information about necessary transformations in augmented arc labels.
2. Train a projective parser on the transformed training set.
3. Parse new sentences using the projective parser.
4. Deprojectivize the output of the projective parser, using heuristic transformations guided by augmented arc labels

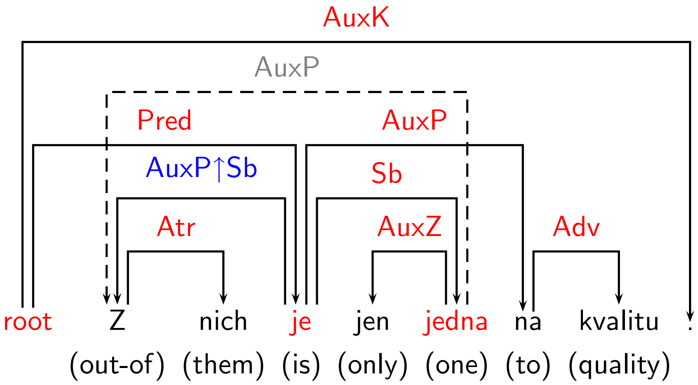
Pseudo-Projective Parsing

- Projectivize training data:
  - Projective head nearest permissible ancestor of real head
  - Arc label extended with dependency type of real head



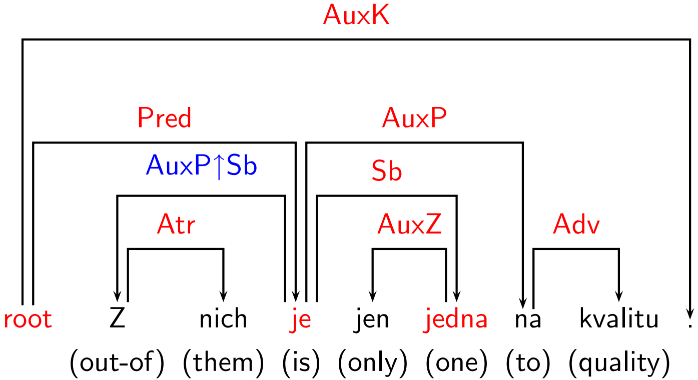
Pseudo-Projective Parsing

- Projectivize training data:
  - Projective head nearest permissible ancestor of real head
  - Arc label extended with dependency type of real head



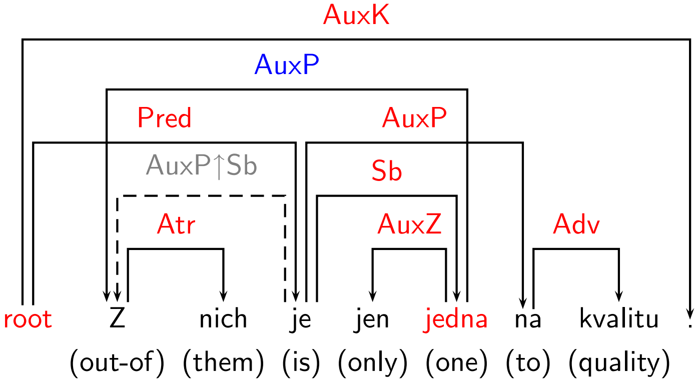
Pseudo-Projective Parsing

- Deprojectivize parser output:
  - Top-down, breadth-first search for real head
  - Search constrained by extended arc label



Pseudo-Projective Parsing

- Deprojectivize parser output:
  - Top-down, breadth-first search for real head
  - Search constrained by extended arc label



Pseudo-Projective Parsing

Advantages:

- Allows sentences with arbitrary non-projective structures to be parsed in linear time
  (if projectivization & deprojectivization are linear)
- The system is sound, as long as the base parser is

Disadvantages:

- Increase in the number of distinct dependency labels

Corrective Modeling

- Conditional probability model

$$P(h_i'|w_i, N(h_i))$$

  for correcting the head h_i of word w_i to h_i', restricted to the local neighboorhood $N(h_i)$ of h_i
- Model trained on parser output and gold standard parses (MaxEnt estimation)
- Post-processing:
  - For every word w_i, replace h_i by $\text{argmax}_{h_i'} P(h_i'|w_i, N(h_i))$.

Corrective Modeling

[Attardi and Ciaramita(2007)]:

- Revise a base parse tree, by applying tree revision rules
  - Revision rules are combinations of atomic revisions, e.g., u = move the dependency up one parent
- e.g., uu specifies moving up twice; -1 specifies moving to the left one token; etc.
- Can use global parse features to determine revisions

[Attardi and Dell'Orletta(2009)]:

- Use a second shift-reduce parser, run in the reverse direction and with additional features from the parse tree, to revise trees
- Use a voting method to select the best set of dependents from among 3 parsers

Motivating Graph-Based Parsing

In some sense, it would be best to start with a methodology that accounts for non-projectivity from the start

Graph-based parsing explicitly parameterizes over the substructures of a dependency tree

- Consider all possible arcs of a dependency forest at once
- Use graph-theoretic algorithms to reduce this to a tree
  - Such algorithms do not distinguish projective or non-projective arcs

Second-Order Non-Projective Parsing

- The score of a dependency tree y for input sentence x is

$$\sum_{(i,k,j)\in y} s(i,k,j)$$

  where k and j are adjacent, same-side children of i in y.
- The highest scoring projective dependency tree can be computed exactly in $O(n^3)$ time using Eisner's algorithm.
- The highest scoring non-projective dependency tree can be approximated with a greedy post-processing procedure:
  - While improving the global score of the dependency tree, replace an arc $h_i \rightarrow w_i$ by $h'_i \rightarrow w_i$, greedily selecting the substitution that gives the greatest improvement.

State of the Art – Czech

- Evaluation:
  - Prague Dependency Treebank (PDT)
  - Unlabeled accuracy per word (W) and per sentence (S)
    - Non-projective spanning tree parsing [McDonald et al.(2005)McDonald, Pereira, Ribarov and Hajič]
    - Corrective modeling on top of the Charniak parser [Hall and Novák(2005)]
    - Approximate non-projective parsing on top of a second-order projective spanning tree parser [McDonald and Pereira(2006)]
    - Pseudo-projective parsing on top of a deterministic classifier-based parser [Nilsson et al.(2006)Nilsson, Nivre and Hall]

Parser	W	S
McDonald and Pereira	85.2	35.9
Hall and Novák	85.1	—
Nilsson et al.	84.6	37.7
McDonald et al.	84.4	32.3
Charniak	84.4	–

- Attardi, Giuseppe and Massimiliano Ciaramita (2007). Tree Revision Learning for Dependency Parsing.
  In *Proceedings of NAACL-HLT-07*. Rochester, NY, pp. 388–395.
- Attardi, Giuseppe and Felice Dell'Orletta (2009). Reverse Revision and Linear Tree Combination for Dependency Parsing.
  In *Proceedings of NAACL-HLT-09, Short Papers*. Boulder, CO, pp. 261–264.
- Buchholz, Sabine and Erwin Marsi (2006). CoNLL-X Shared Task on Multilingual Dependency Parsing.
  In *Proceedings of the Tenth Conference on Computational Natural Language Learning*.
- Covington, Michael A. (2001). A Fundamental Algorithm for Dependency Parsing.
  In *Proceedings of the 39th Annual ACM Southeast Conference*. pp. 95–102.
- Duchier, Denys and Ralph Debusmann (2001). Topological Dependency Trees: A Constraint-based Account of Linear Precedence.
  In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*. pp. 180–187.
- Foth, Kilian, Michael Daum and Wolfgang Menzel (2004). A Broad-Coverage Parser for German Based on Defeasible Constraints.
  In *Proceedings of KONVENS 2004*. pp. 45–52.
- Gaifman, Haim (1965). Dependency systems and phrase-structure systems.

Information and Control 8, 304–337.

- Hall, Keith and Vaclav Novák (2005). Corrective modeling for non-projective dependency parsing.
  In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*. pp. 42–52.

- McDonald, Ryan 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)*. pp. 81–88.

- McDonald, Ryan, Fernando Pereira, Kiril Ribarov and Jan Hajič (2005). 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)*. pp. 523–530.

- Neuhaus, Peter and Norbert Bröker (1997). The Complexity of Recognition of Linguistically Adequate Dependency Grammars.
  In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. pp. 337–343.

- Nilsson, Jens, Joakim Nivre and Johan Hall (2006). Graph Transformations in Data-Driven Dependency Parsing.

In *Proceedings of COLING-ACL*.

- Nivre, Joakim (2003). An Efficient Algorithm for Projective Dependency Parsing. In Gertjan Van Noord (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. pp. 149–160.
- Nivre, Joakim (2006). Constraints on Non-Projective Dependency Graphs. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. pp. 73–80.
- Nivre, Joakim (2008). Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics* 34(4), 513–553.
- Nivre, Joakim (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*. Suntec, Singapore: Association for Computational Linguistics, pp. 351–359.
- Nivre, Joakim and Jens Nilsson (2005). Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. pp. 99–106.
- Tapanainen, Pasi and Timo Järvinen (1997). A non-projective dependency parser.

In *Proceedings of the 5th Conference on Applied Natural Language Processing*. pp. 64–71.