Natural Language Processing (CSE 517): Dependency Structure

Noah Smith

University of Washington
nasmith@cs.washington.edu

February 24, 2016

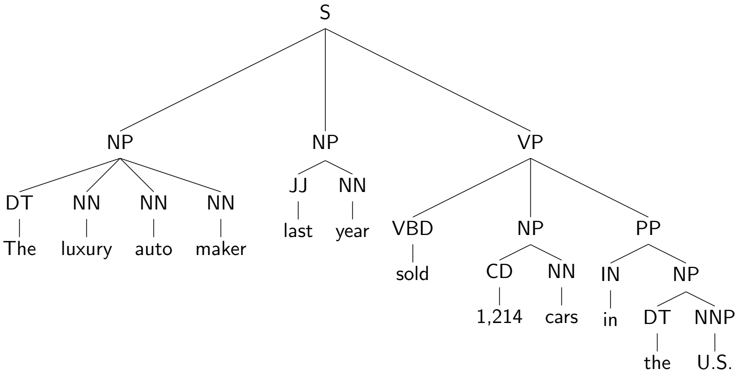
Why might you want to use a generative classifier, such as Naive Bayes, as opposed to a discriminative classifier, and vice versa?

How can one deal with out-of-vocabulary words at test time when one is applying an HMM for POS tagging or a PCFG for parsing?

What is marginal inference, and how can it be carried out on a factor graph?

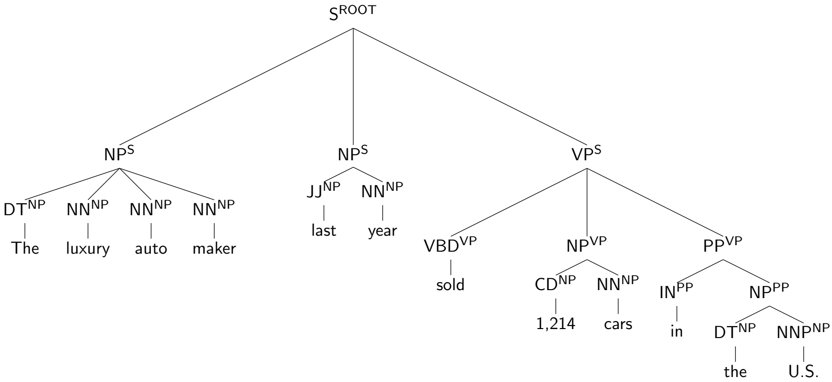
What are the advantages and disadvantages of using a context-free grammar in Chomsky normal form?

Starting Point: Phrase Structure



Parent Annotation

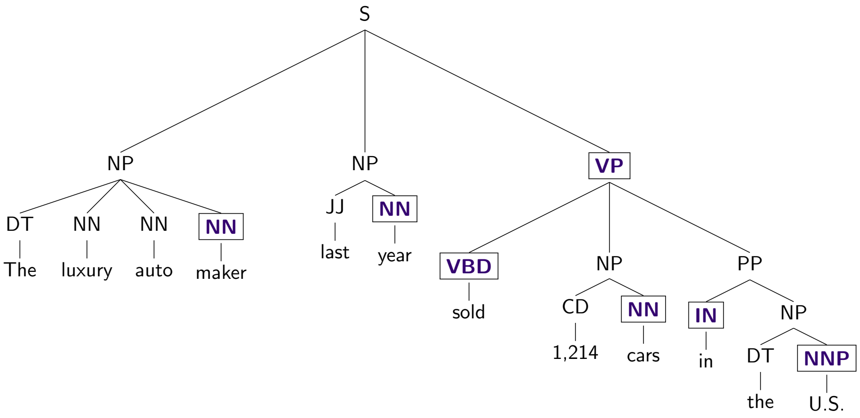
(Johnson, 1998)



Increases the "vertical" Markov order:

$$p(\text{children} \mid \text{parent}, \text{grandparent})$$

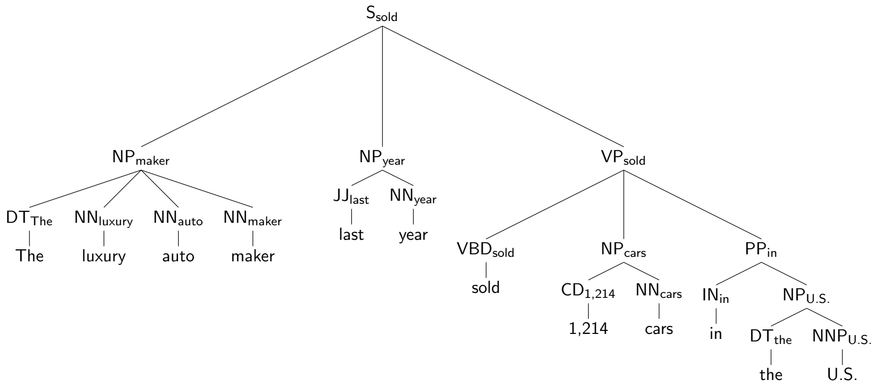
Headedness



Suggests "horizontal" markovization:

$$p(\text{children} \mid \text{parent}) = p(\text{head} \mid \text{parent}) \cdot \prod_i p(i\text{th sibling} \mid \text{head}, \text{parent})$$

Lexicalization



Each node shares a **lexical head** with its head child.

Transformations on Trees

Starting around 1998, many different ideas—both linguistic and statistical—about how to transform treebank trees.

All of these make the grammar larger—and therefore all frequencies became sparser—so a lot of research on *smoothing* the probability rules.

Parent annotation, headedness, markovization, and lexicalization; also category *refinement* by linguistic rules (Klein and Manning, 2003).

- These are reflected in some versions of the popular Stanford and Berkeley parsers.

Tree Decorations
(Klein and Manning, 2003)

- Mark nodes with only 1 child as UNARY
- Mark DTs (determiners), RBs (adverbs) when they are only children
- Annotate POS tags with their parents
- Split IN (prepositions; 6 ways), AUX, CC, %
- NPs: temporal, possessive, base
- VPs annotated with head tag (finite vs. others)
- DOMINATES-V
- RIGHT-RECURSIVE NP

Machine Learning and Parsing

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - K-best parsing: Huang and Chiang (2005)

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
    - K-best parsing: Huang and Chiang (2005)
- Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
    - These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).

Structured Perceptron

Collins (2002)

Perceptron algorithm for **parsing**:

- For $t \in \{1, \ldots, T\}$:
  - Pick i_t uniformly at random from $\{1, \ldots, n\}$.
  - $\hat{\boldsymbol{t}}_{i_t} \leftarrow \underset{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}_{i_t}}}{\operatorname{argmax}} \mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \boldsymbol{t})$
  - $\mathbf{w} \leftarrow \mathbf{w} - \alpha \left(\boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \hat{\boldsymbol{t}}_{i_t}) - \boldsymbol{\Phi}(\boldsymbol{x}_{i_t}, \boldsymbol{t}_{i_t}) \right)$

This can be viewed as stochastic subgradient descent on the *structured* hinge loss:

$$\sum_{i=1}^{n} \underbrace{\max_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}_{i_t}}} \mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t})}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t}_i)}_{\text{hope}}$$

Beyond Structured Perceptron (I)

Structured support vector machine (also known as **max margin parsing**; Taskar et al., 2004):

$$\sum_{i=1}^{n} \underbrace{\max_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}_{i_t}}} \mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t}) + \mathrm{cost}(\boldsymbol{t}_{i_t}, \boldsymbol{t})}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t}_i)}_{\text{hope}}$$

where $\mathrm{cost}(\boldsymbol{t}_i, \boldsymbol{t})$ is the number of local errors (either constituent errors or “rule” errors).

Beyond Structured Perceptron (II)

Log-loss, which gives parsing models analogous to **conditional random fields** (Miyao and Jun'ichi, 2002; Finkel et al., 2008):

$$\sum_{i=1}^{n} \underbrace{\log \sum_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}_i}} \exp \mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t})}_{\text{fear}} - \underbrace{\mathbf{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i, \boldsymbol{t}_i)}_{\text{hope}}$$

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - K-best parsing: Huang and Chiang (2005)
- Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
  - These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).
- Learn refinements on the constituents, as latent variables (Petrov et al., 2006).

Machine Learning and Parsing

- Define arbitrary features on trees, based on linguistic knowledge; to parse, use a PCFG to generate a **k-best list** of parses, then train a log-linear model to rerank (Charniak and Johnson, 2005).
  - K-best parsing: Huang and Chiang (2005)
- Define rule-local features on trees (and any part of the input sentence); minimize hinge or log loss.
  - These exploit dynamic programming algorithms for training (CKY for arbitrary scores, and the sum-product version).
- Learn refinements on the constituents, as latent variables (Petrov et al., 2006).
- Neural, too:
  - Socher et al. (2013) define **compositional vector grammars** that associate each phrase with a vector, calculated as a function of its subphrases' vectors. Used essentially to rerank.
  - Dyer et al. (2016): **recurrent neural network grammars**, generative models like PCFGs that encode arbitrary previous derivation steps in a vector. Parsing requires some tricks.

Dependencies

Informally, you can think of **dependency** structures as a transformation of phrase-structures that

- maintains the word-to-word relationships induced by lexicalization,
- adds labels to them, and
- eliminates the phrase categories.

There are also linguistic theories built on dependencies (Tesnière, 1959; Mel'čuk, 1987), as well as treebanks corresponding to those.

- Free(r)-word order languages (e.g., Czech)

Dependency Tree: Definition

Let $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ be a sentence. Add a special ROOT symbol as "x_0."

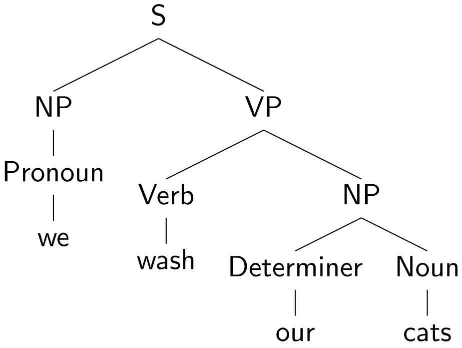
A dependency tree consists of a set of tuples $\langle p, c, \ell \rangle$, where

- $p \in \{0, \ldots, n\}$ is the index of a parent
- $c \in \{1, \ldots, n\}$ is the index of a child
- $\ell \in \mathcal{L}$ is a label

Different annotation schemes define different label sets $\mathcal{L}$, and different constraints on the set of tuples. Most commonly:

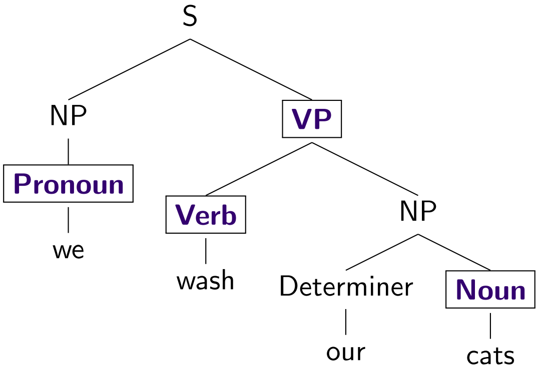
- The tuple is represented as a directed edge from x_p to x_c with label ℓ.
- The directed edges form an arborescence (directed tree) with x_0 as the root.

Example



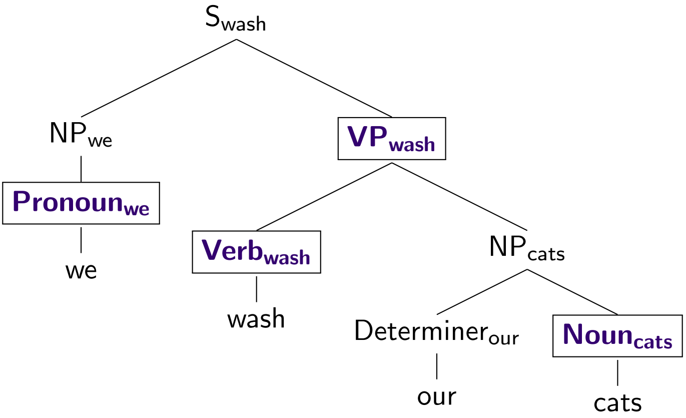
Phrase-structure tree.

Example



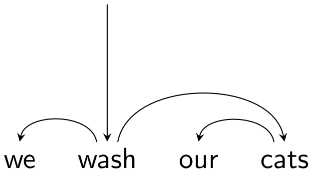
Phrase-structure tree with heads.

Example



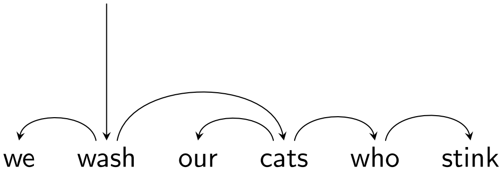
Phrase-structure tree with heads, lexicalized.

Example

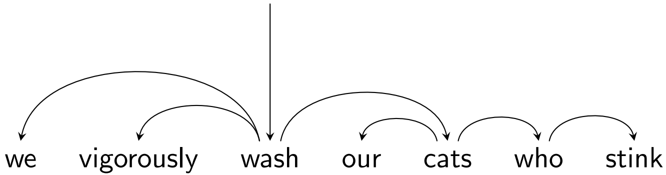


“Bare bones” dependency tree.

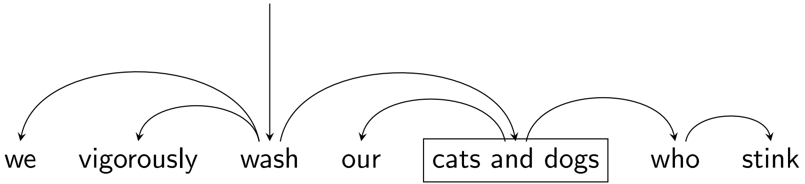
Example



Example

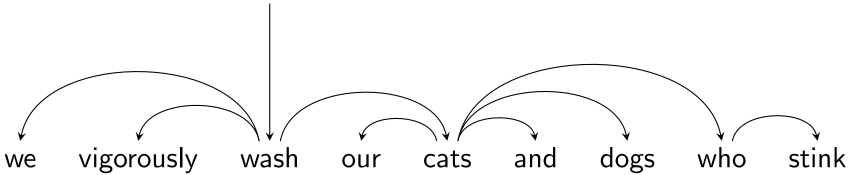


Example



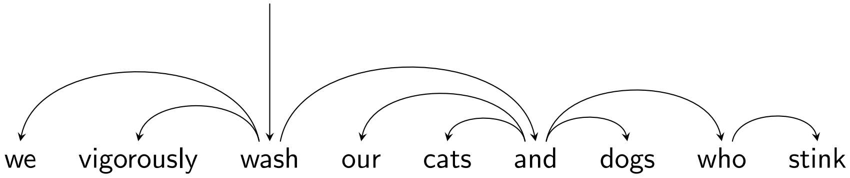
The bugbear of dependency syntax: coordination structures.

Example



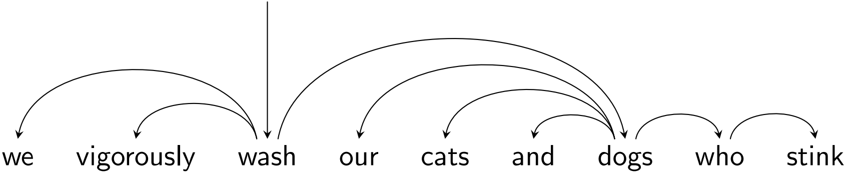
Make the first conjunct the head?

Example



Make the coordinating conjunction the head?

Example



Make the second conjunct the head?

Dependency Schemes

- Transform the treebank: define "head rules" that can select the head child of any node in a phrase-structure tree and label the dependencies.
- More powerful, less local rule sets, possibly collapsing some words into arc labels.
  - Stanford dependencies are a popular example (de Marneffe et al., 2006).
- Direct annotation.

Dependencies and Grammar

Context-free grammars can be used to encode dependency structures.

For every head word and constellation of dependent children:

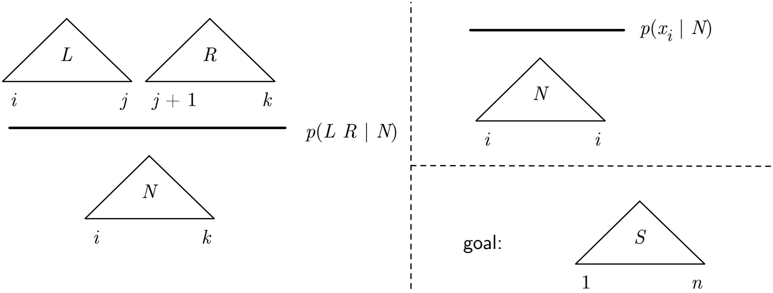
$$N_{\text{head}} \quad \rightarrow \quad N_{\text{leftmost-sibling}} \ldots N_{\text{head}} \ldots N_{\text{rightmost-sibling}}$$

And for every head word: $N_{\text{head}} \rightarrow \text{head}$

A **bilexical** dependency grammar binarizes the dependents, generating only one per rule, usually "outward" from the head.

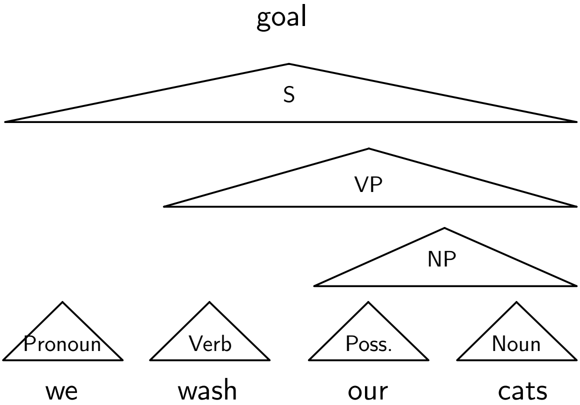
Such a grammar can produce only **projective** trees, which are (informally) trees in which the arcs don't cross.

Quick Reminder: CKY

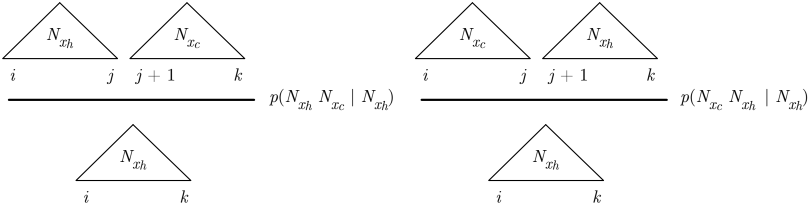


Each "triangle" item corresponds to a buildable phrase.

CKY Example



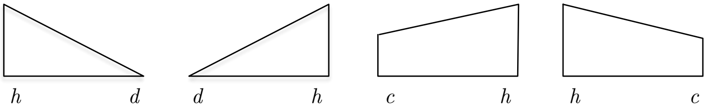
CKY for Bilexical Context-Free Grammars



Here we ignore the initial and goal rules.

Dependency Parsing with the Eisner Algorithm

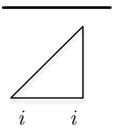
(Eisner, 1996)

Items: 

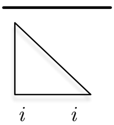
- Both triangles indicate that x_d is a descendant of x_h.
- Both trapezoids indicate that x_c can be attached as the child of x_h.
- In all cases, the words "in between" are descendants of x_h.

Dependency Parsing with the Eisner Algorithm
(Eisner, 1996)

Initialization:

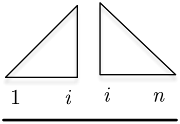


$p(x_i \mid N_{xi})^{1/2}$



$p(x_i \mid N_{xi})^{1/2}$

Goal:



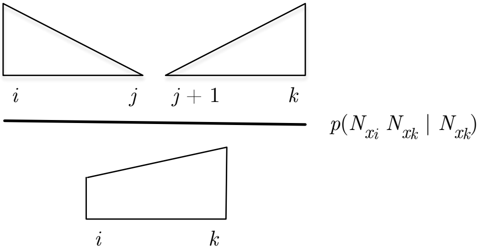
$p(N_{xi} \mid S)$

goal

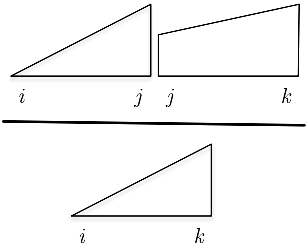
Dependency Parsing with the Eisner Algorithm

(Eisner, 1996)

Attaching a left dependent:



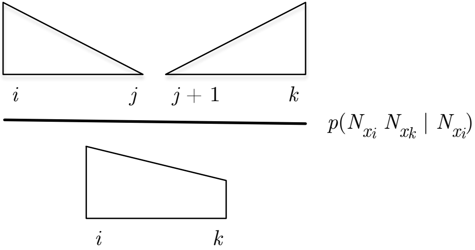
Complete a left child:



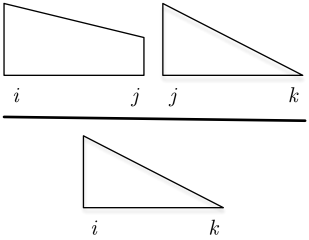
Dependency Parsing with the Eisner Algorithm

(Eisner, 1996)

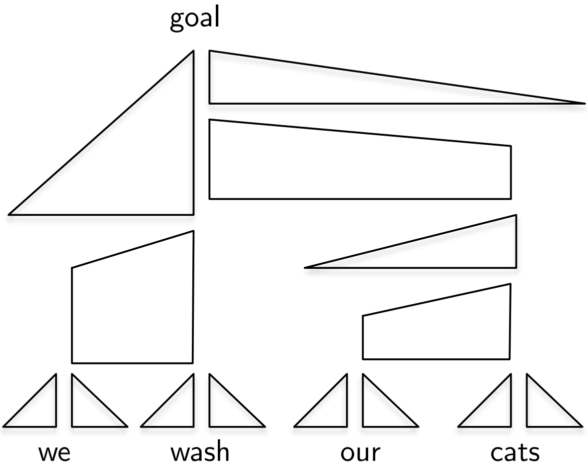
Attaching a right dependent:



Complete a right child:



Eisner Algorithm Example



Slight Generalization

The Eisner algorithm can be used to find the projective tree with the highest score whenever the score of the dependency tree has this form:

$$\prod_{\langle p,c,\ell\rangle \in \boldsymbol{t}} s(p,c,\ell;\boldsymbol{x}) = \exp \sum_{\langle p,c,\ell\rangle \in \boldsymbol{t}} \log s(p,c,\ell;\boldsymbol{x})$$

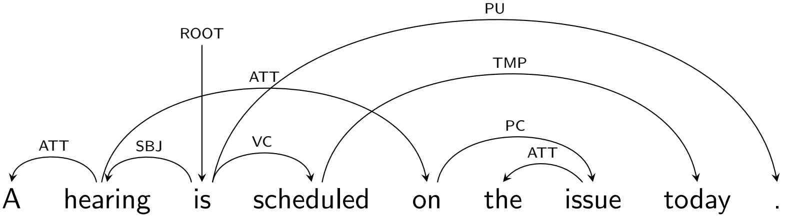
(Recall that a tree $\boldsymbol{t}$ consists of a set of parent/child/label tuples of the form $\langle p, c, \ell\rangle$; see slide 18.)

This property of a scoring function is called **arc factorization**; McDonald et al. (2005) called it "edge-based factorization."

Remarks on Dependency Parsing

- Naively using CKY with the bilexical grammar will have $O(n^5)$ runtime; Eisner gives us $O(n^3)$.
- Ask with CKY for phrase-structure, a narrow-to-wide ordering is reasonable, but an agenda may make parsing faster.
- As with phrase-structure parsing, you can get better accuracy with higher Markov order:
  - horizontal (among siblings)
  - vertical (grandparents)
- Transition-based approaches are popular among those who want speed.
- What about the projectivity assumption?
  - See the reading (McDonald et al., 2005)!

Nonprojective Example



Final Notes on Parsing

- Formalisms that are more powerful than context-free grammars include **tree adjoining grammars**, **combinatory categorial grammars**, and **unification-based grammars**.
  - Very attractive from a linguistic point of view
  - Large-scale annotation has been a challenge.
- What are parse trees good for?
  - Syntax is a scaffold for semantics (as we'll see next week), as well as information extraction, question answering, and sometimes machine translation.
  - Features in text categorization (e.g., sentiment)

Readings and Reminders

- McDonald et al. (2005)
- Assignment 4 is due March 2.
- Submit a suggestion for an exam question by Friday at 5pm.
- Your project is due March 9.

References I

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005.

Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, 2002.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*, 2006.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars, 2016. To appear.

Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*, 1996.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. Efficient, feature-based, conditional random field parsing. In *Proc. of ACL*, 2008.

Liang Huang and David Chiang. Better k-best parsing. In *Proc. of IWPT*, 2005.

Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–32, 1998.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proc. of ACL*, 2003.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*, 2005. URL `http://www.aclweb.org/anthology/H/H05/H05-1066`.

References II

Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice*. State University Press of New York, 1987.

Yusuke Miyao and Tsujii Jun'ichi. Maximum entropy estimation for feature forests. In *Proc. of HLT*, 2002.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*, 2006.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proc. of ACL*, 2013.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*. 2004.

L. Tesnière. *Éléments de Syntaxe Structurale*. Klincksieck, 1959.