TAG, Dynamic Programming, and the Perceptron for Efficent, Feature-Rich Parsing

Xavier Carreras, Michael Collins and Terry Koo

MIT CSAIL

Discriminative Models for Parsing

Structured Prediction methods like CRF or Perceptron train linear models defined on factored representations of structures:

$$\text{Parse}(\mathbf{x}) = \underset{y \in \mathcal{Y}(\mathbf{x})}{\text{argmax}} \sum_{r \in y} \mathbf{f}(\mathbf{x}, r) \cdot \mathbf{w}$$

Main Advantage:

- Flexibility of feature definitions in $\mathbf{f}(\mathbf{x}, r)$

Critical Difficulty:

- Training algorithms repeatedly parse the training sentences. Efficient parsing algorithms are crucial.

A Feature-rich Consituent Parsing Model

We present a TAG-style model to recover constituent trees.

It defines feature vectors looking at:

- CFG-based structure
- Dependency relations between lexical heads
- Second-order dependency relations
  with sibling and grandparent dependencies

These can be combined with surface features of the sentence.

Efficient Coarse-to-fine Inference

We use a coarse-to-fine parsing strategy on dependency graphs:

- We use general versions of the Eisner algorithm to parse with the full TAG parser
- Simple first-order dependency models restrict the space of the full model, making parsing feasible

We train a parser with discriminative methods at full-scale.

TAG + Dynamic Programming + Perceptron

We use the Averaged Perceptron to train the parameters of our TAG model:

- $\mathbf{w} = 0$, $\mathbf{w}_a = 0$
- For $t = 1 \ldots T$
  - For each training example $(\mathbf{x}, y)$
    1. $z = \text{Parse}(\mathbf{x}; \mathbf{w})$
    2. if $y \neq z$ then
       $$\mathbf{w} = \mathbf{w} + \mathbf{f}(\mathbf{x}, y) - \mathbf{f}(\mathbf{x}, z)$$
    3. $\mathbf{w}_a = \mathbf{w}_a + \mathbf{w}$
- return $\mathbf{w}_a$

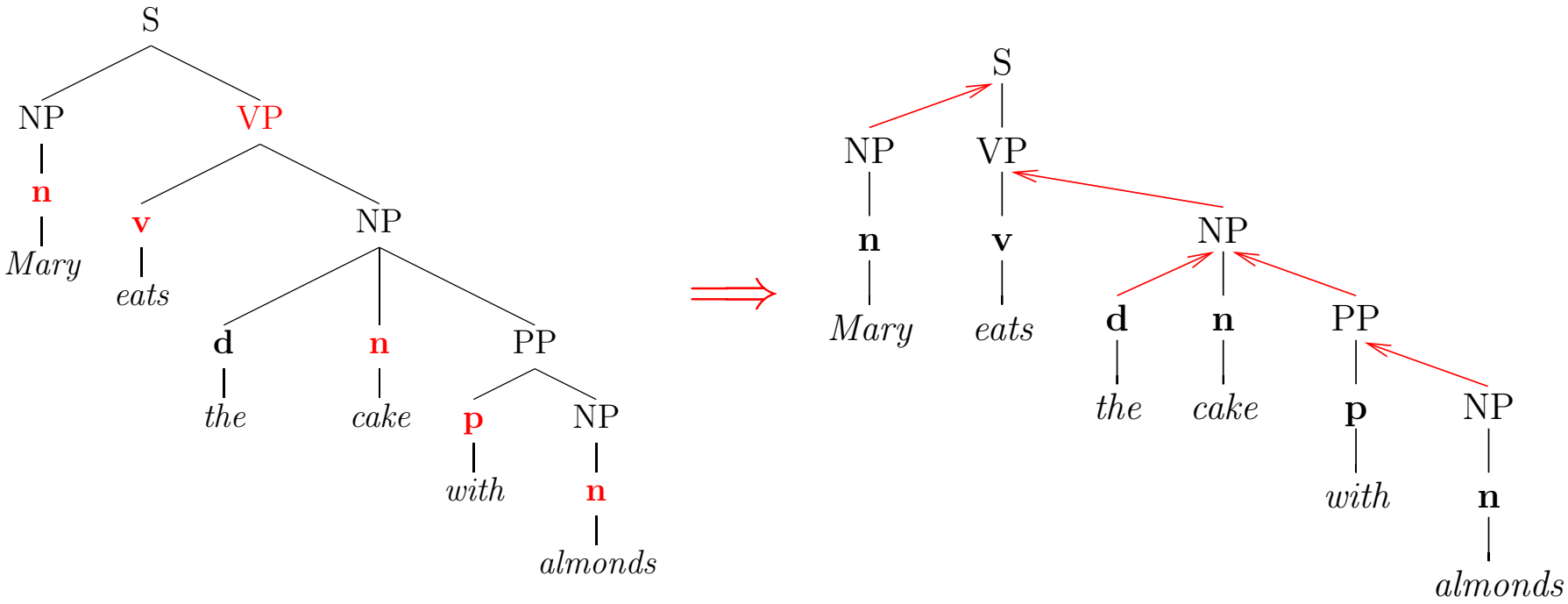
We obtain state-of-the-art results for English.

Outline

Tree-Adjoining Grammar (TAG)

- In TAG formalisms [Joshi et al. 1975]:
  - The basic elements are trees
  - Trees can be combined to form bigger trees
- There are many variations of TAG
- Here we present a simple TAG-style grammar:
  - Allows rich features
  - Allows efficient inference

Decomposing trees into spines and adjunctions

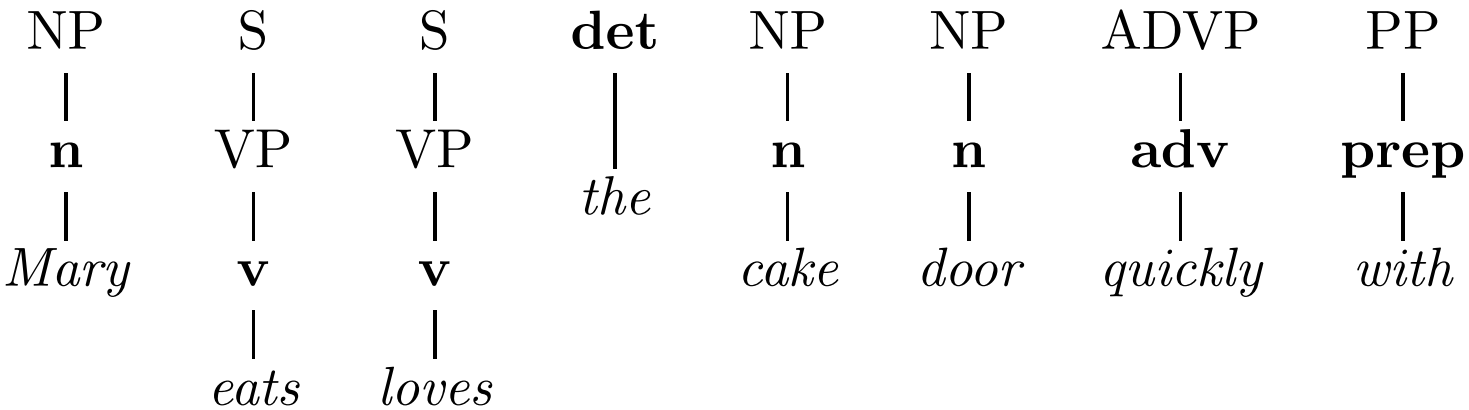


Syntactic constituents sit on top of their lexical heads.
The underlying structure looks like a dependency structure.

Spines

Spines are lexical units with a chain of unary projections.

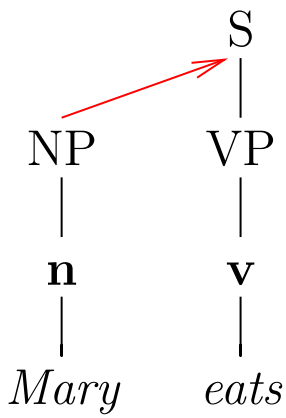
They are the elementary trees in our TAG.
(see also [Shen & Joshi 2005])



We build a dictionary of spines appearing in the WSJ.

Sister Adjunctions

Sister adjunctions are used to combine spines to form trees.

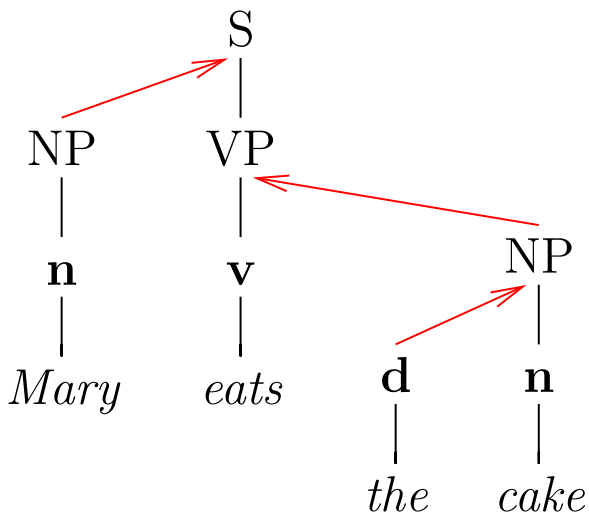


An adjunction operation attaches:

- A modifier spine
- To some position of a head spine

Sister Adjunctions

Sister adjunctions are used to combine spines to form trees.

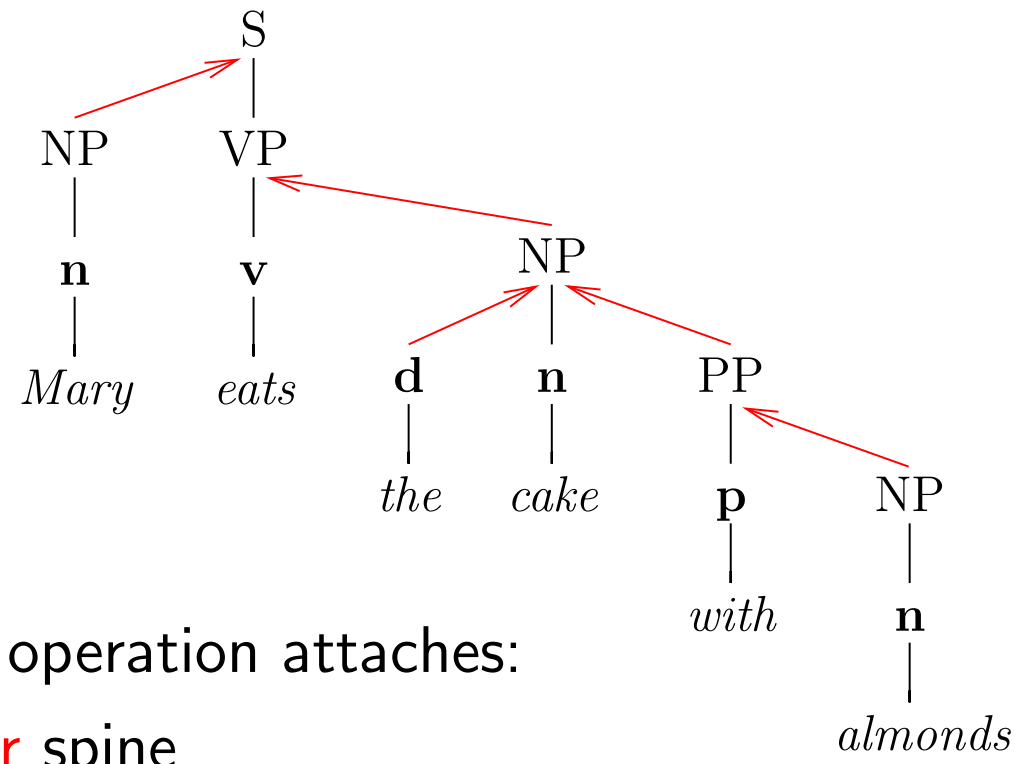


An adjunction operation attaches:

- A modifier spine
- To some position of a head spine

Sister Adjunctions

Sister adjunctions are used to combine spines to form trees.



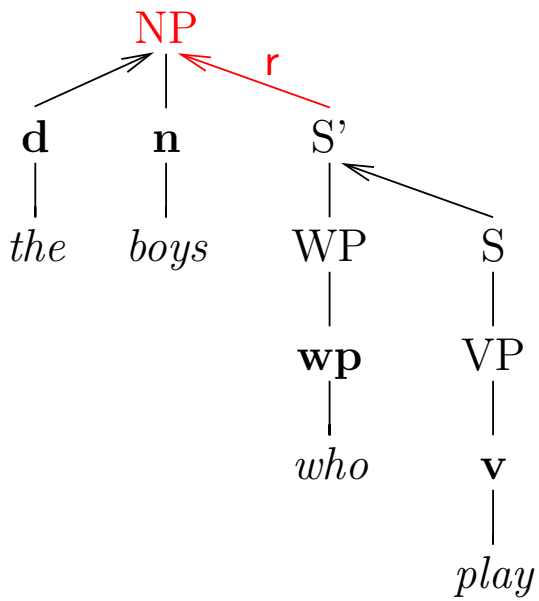
An adjunction operation attaches:

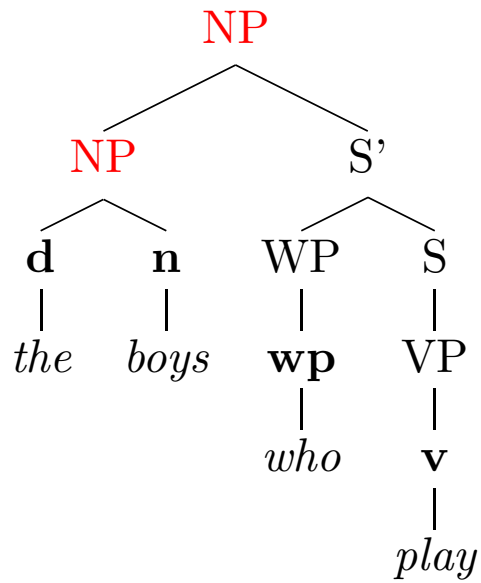
- A modifier spine
- To some position of a head spine

Regular Adjunctions

We also consider a regular adjunction operation.

It adds one level to the syntactic constituent it attaches to.



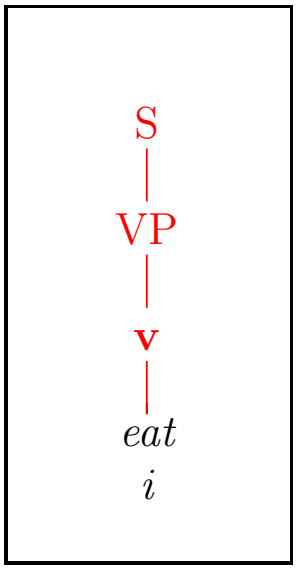


N.B.: This operation is simpler than adjunctions in classic TAG, resulting in more efficient parsing costs.

Derivations in our TAG

A tree is a set with two types of elements:

spines

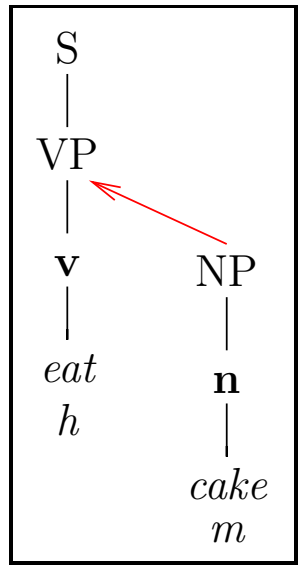


$\langle i, \sigma \rangle$

i : word position

σ : a spine

adjunctions



$\langle h, m, \sigma_h, \sigma_m, \texttt{POS}, \texttt{A} \rangle$

h m : head and modifier positions

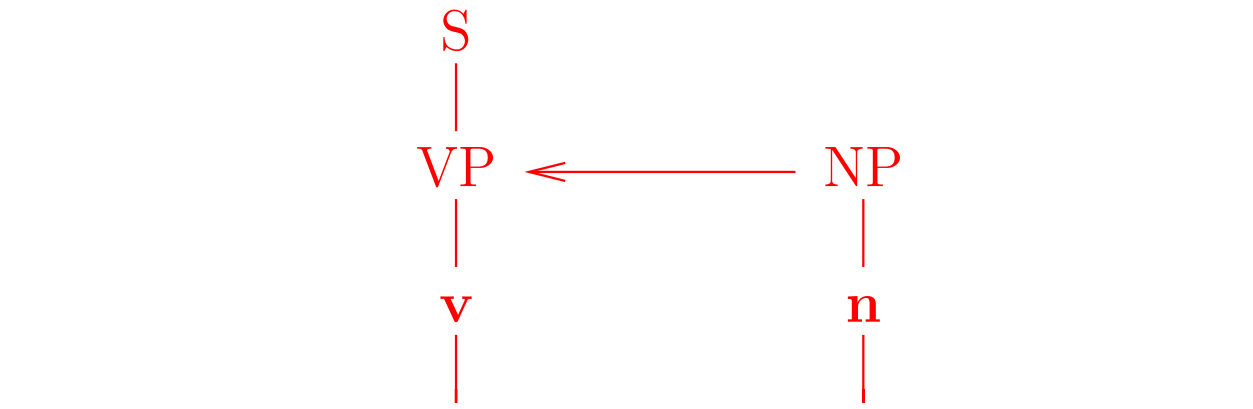
σ_h σ_m : spines of h and m

POS : the attachment position

A : sister or regular

A TAG-style Linear Model

$\mathbf{f_a}(\mathbf{x}, \langle h, m, \sigma_h, \sigma_m, \texttt{POS}, \texttt{A} \rangle)$



$$\text{Parser}(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}(\mathbf{x})} \sum_{\langle i,\sigma \rangle \in S(y)} \mathbf{f_s}(\mathbf{x}, \langle i, \sigma \rangle) \cdot \mathbf{w} + \sum_{\langle h,m,\ldots \rangle \in A(y)} \mathbf{f_a}(\mathbf{x}, \langle h, m, \ldots \rangle) \cdot \mathbf{w}$$

Outline

Parsing with the Eisner Algorithms

- Our TAG structures are a general form of dependency graph:
  - Dependencies are adjunctions between spines
  - Labels include the type and position of the adjunction

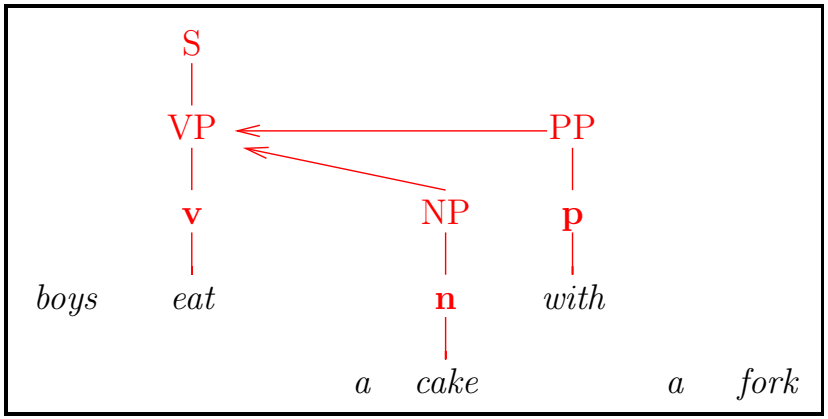
- Parsing can be done with the Eisner [1996,2000] algorithms
  - Applies to splittable dependency representations
    i.e., left and right modifiers are adjoined independently
  - Words in the dependency graph can have senses,
    like our spines
  - Parsing time is $O(n^3 G)$

- Can be extended to include second-order features.

Second-Order Features in our TAG

We incorporate recent extensions to the Eisner algorithm:

siblings

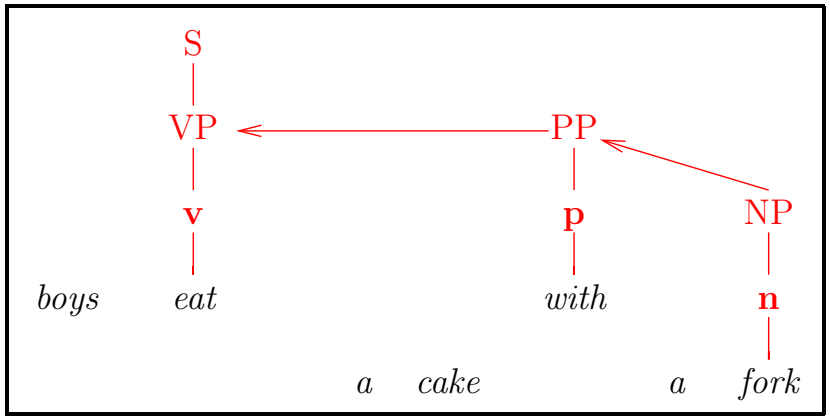


$O(n^3G)$

[Eisner 2000]

[McDonald & Pereira, 2006]

grandchildren



$O(n^4G)$

[Carreras, 2007]

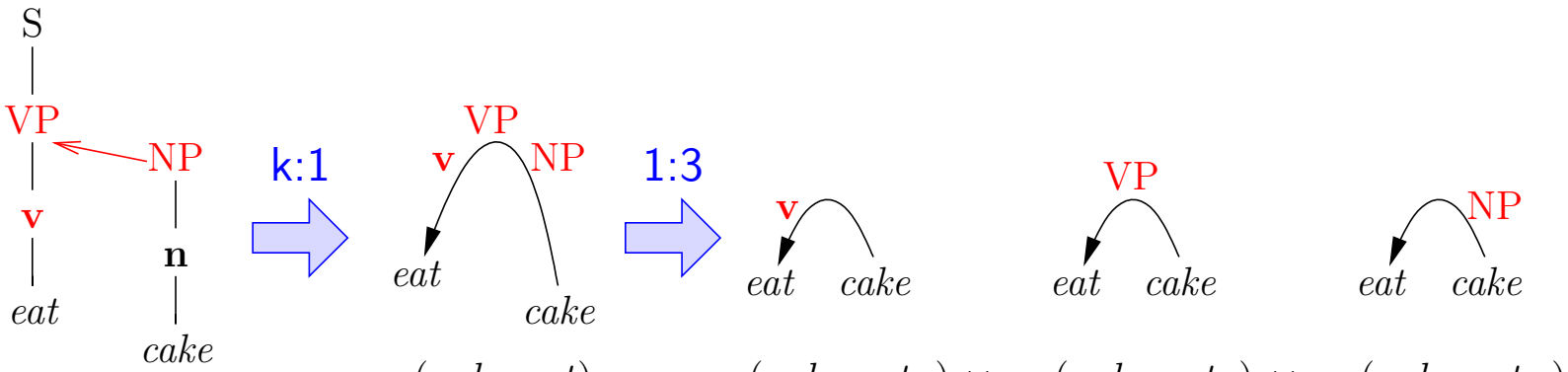
Exact Inference is Too Expensive

- Parsing time is at least $O(n^3 G)$.
  (it is $O(n^4 G)$ in our final model)

- The constant G is polynomial in the number of possible spines for any word, and the maximum height of any spine.
  This is prohibitive for real parsing tasks ($G > 5000$).

- *Solution:* Coarse-to-fine inference
  (e.g. [Charniak 97] [Charniak & Johnson 05] [Petrov & Klein 07])
  - Use simple dependency parsing models to restrict the space of possible structures of the full model.

A Coarse-to-fine Strategy for Fast Parsing



- First-order dependency models estimate conditional distributions of simple dependencies
- We build a beam of most likely dependencies:
  - Inside-Outside inference, in $O(n^3 H)$ with $H \sim 50$
  - We can discard 99.6 of dependencies and retain 98.5 of correct constituents
- The full model is constrained to the pruned space both at training and testing

A TAG-style Linear Model: Summary

A simple TAG-style model, based in spines and adjunctions:

- It allows a wide variety of features
- It's splittable, allowing efficient inference
  - $O(n^3G)$ for CFG-style, head-modifier and sibling features
  - $O(n^4G)$ for grandchildren dependency features
- The backbone dependency graph can be pruned with simple first-order dependency models

Other TAG formalisms have more expensive parsing algorithms [Chiang 2003] [Shen & Joshi 2005].

Outline

Parsing the WSJ Treebank

- Extraction of our TAG derivations from WSJ trees
  - Straighforward process using the head rules of [Collins 1999]
  - $\sim$300 spines, $\sim$20 spines/token
- Learning:
  - Train first-order models using EG [Collins et al. 2008]
    5 training passes, 5 hours per pass
  - Train TAG-style full model using Avg. Perceptron
    10 training passes, 12 hours per pass
- Parse test data and evaluate

Test results on WSJ data

Full Parsers	precision	recall	F_1
Charniak 2000	89.5	89.6	89.6
Petrov & Klein 2007	90.2	89.9	90.1
this work	**91.4**	**90.7**	**91.1**

Rerankers	precision	recall	F_1
Collins 2000	89.9	89.6	89.8
Charniak & Johnson 2005	·	·	91.4
Huang 2008	·	·	91.7

Evaluating Dependencies

- We look at the accuracy of recovering unlabeled dependencies
- We compare to state-of-the-art dependency parsing models using the same features and learner :

training structures	dependency accuracy
unlabeled dependencies (*)	92.0
labeled dependencies (*)	92.5
adjoined spines	**93.5**

(*) results from [Koo et al., ACL 2008]

constituent structure greatly helps parsing performance

Summary

A new efficient and expressive discriminative model for full consituent parsing:

- Represents phrase structure with a TAG-style grammar
- Has rich features combining phrase structure and lexical heads due to our spines being basic elements
- Parsing is efficient with the Eisner methods due to the splittable nature of our adjunctions

A very effective method to prune dependency-based graphs:

key to discriminative training at full scale

Thanks!