COMP 786(Fall 2020)
Natural Language Processing

Lecture 4: Syntactic Parsing (Constituent, Dependency, etc.)

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

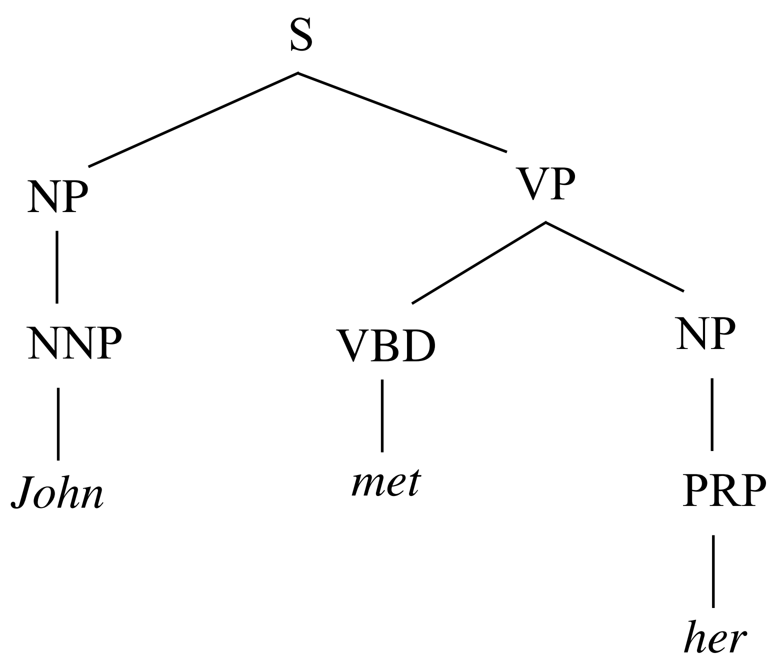
Mohit Bansal

(various slides adapted/borrowed from courses by Dan Klein, Richard Socher, Chris Manning, JurafskyMartin-SLP3, others)

Constituent Parsing

Syntactic Parsing -- Constituent

- Phrase-structure parsing or Bracketing



- Demos: https://parser.kitaev.io/

Syntactic Parsing -- Constituent

- Grammar and Lexicon

Grammar	**Lexicon**
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 12.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

Syntactic Parsing -- Constituent

- Parsing Ambiguity

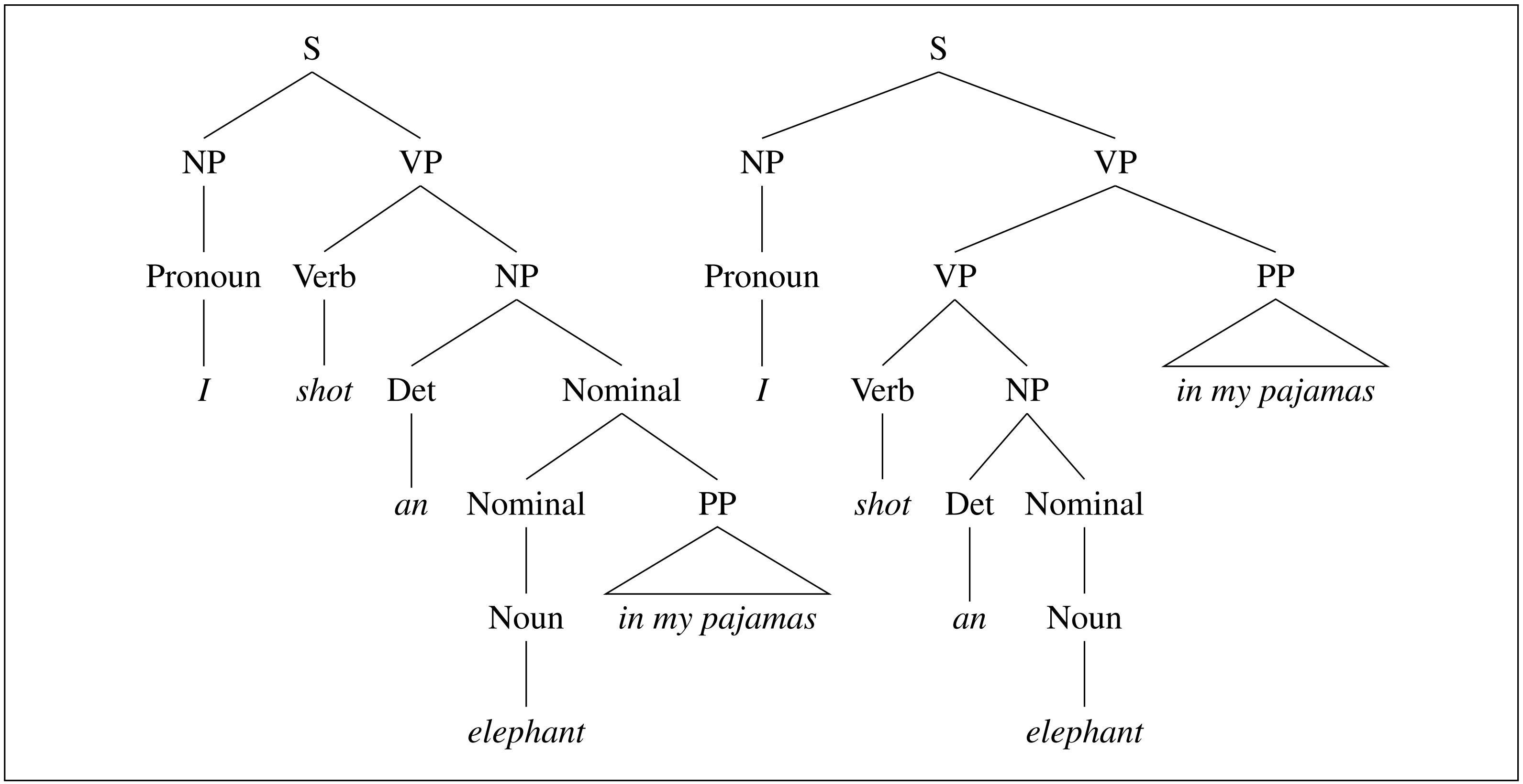


Figure 12.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

Probabilistic Context-free Grammars

- A context-free grammar is a tuple <*N, T, S, R*>

  N : the set of non-terminals

  Phrasal categories: S, NP, VP, ADJP, etc.

  Parts-of-speech (pre-terminals): NN, JJ, DT, VB

  T : the set of terminals (the words)

  S : the start symbol

  Often written as ROOT or TOP

  Not usually the sentence non-terminal S

  R : the set of rules

  Of the form $X \rightarrow Y_1\ Y_2 \ldots Y_k$, with $X, Y_i \in N$

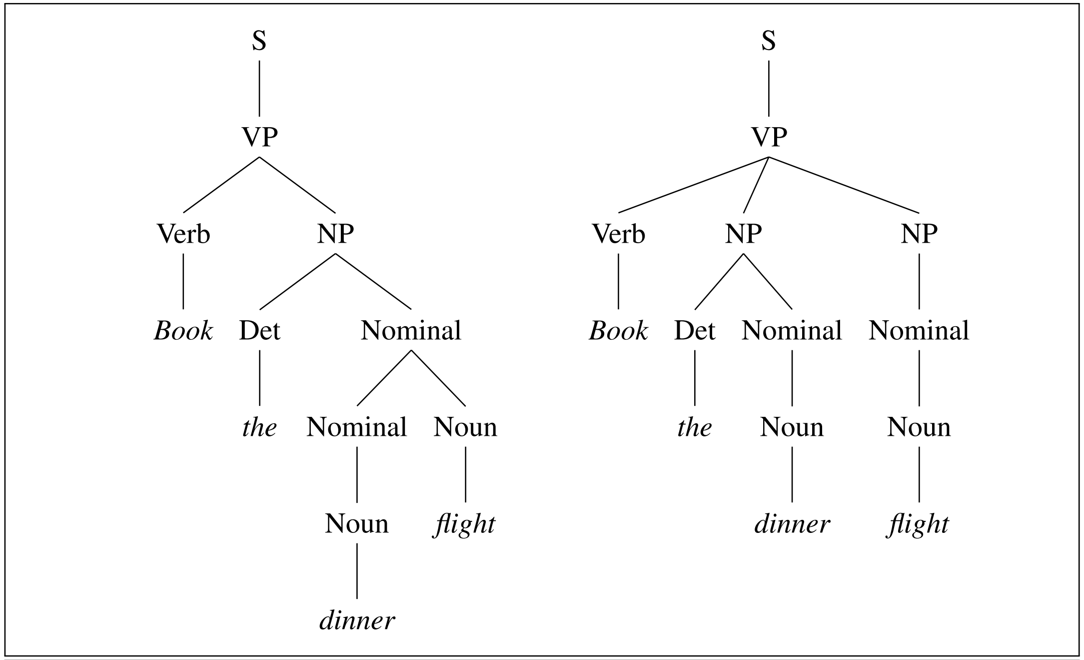
  Examples: $S \rightarrow NP\ VP$, $VP \rightarrow VP\ CC\ VP$

  Also called rewrites, productions, or local trees

Probabilistic Context-free Grammars

- A PCFG:
  - Adds a top-down production probability per rule $P(Y_1\ Y_2 \ldots Y_k \mid X)$
  - Allows us to find the ‘most probable parse’ for a sentence
  - The probability of a parse is just the product of the probabilities of the individual rules

Probabilistic Context-free Grammars



Rules			P	Rules			P
S	$\rightarrow$	VP	.05	S	$\rightarrow$	VP	.05
VP	$\rightarrow$	Verb NP	.20	VP	$\rightarrow$	Verb NP NP	.10
NP	$\rightarrow$	Det Nominal	.20	NP	$\rightarrow$	Det Nominal	.20
Nominal	$\rightarrow$	Nominal Noun	.20	NP	$\rightarrow$	Nominal	.15
Nominal	$\rightarrow$	Noun	.75	Nominal	$\rightarrow$	Noun	.75
				Nominal	$\rightarrow$	Noun	.75
Verb	$\rightarrow$	book	.30	Verb	$\rightarrow$	book	.30
Det	$\rightarrow$	the	.60	Det	$\rightarrow$	the	.60
Noun	$\rightarrow$	dinner	.10	Noun	$\rightarrow$	dinner	.10
Noun	$\rightarrow$	flight	.40	Noun	$\rightarrow$	flight	.40

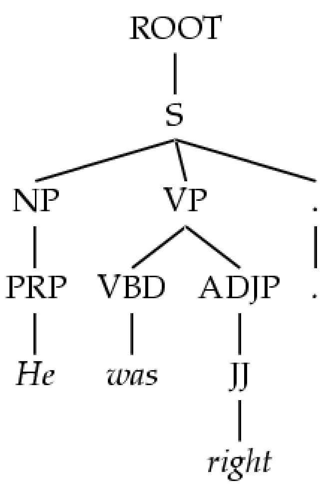
Figure 13.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' ".

Probabilistic Context-free Grammars

$$
\begin{aligned}
P(T_{left}) &= .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}} \\
P(T_{right}) &= .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}
\end{aligned}
$$

Treebank PCFG Probabilities

- Simple MLE approach: just count the frequency of each rule and normalize (similar to what we did for tagging)



ROOT → S	1
S → NP VP .	1
NP → PRP	1
VP → VBD ADJP	1
…..	

$$P(\alpha \to \beta | \alpha) = \frac{\text{Count}(\alpha \to \beta)}{\sum_\gamma \text{Count}(\alpha \to \gamma)} = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

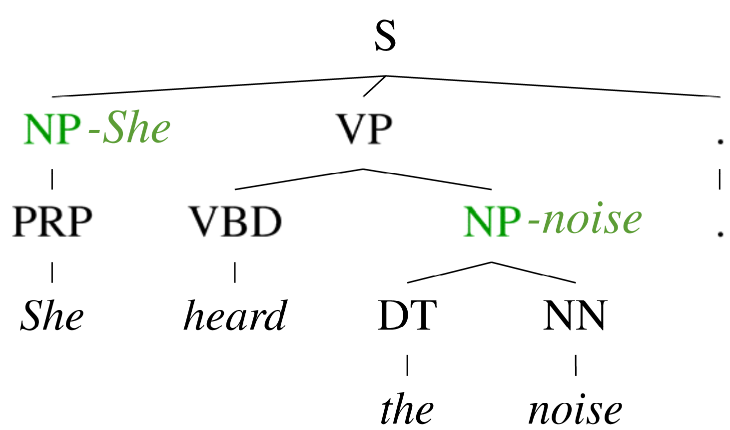
[Charniak, 1996]

Real Treebank Examples

- Long, complex sentences with several clauses, nested prepositions, etc.

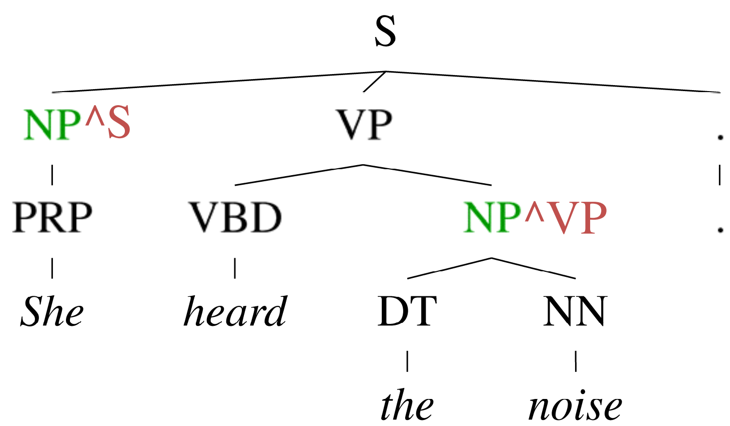
```
( (S (NP-SBJ The move)
     (VP followed
         (NP (NP a round)
             (PP of
                 (NP (NP similar increases)
                     (PP by
                         (NP other lenders))
                     (PP against
                         (NP Arizona real estate loans)))))
         ,
         (S-ADV (NP-SBJ *)
                (VP reflecting
                    (NP (NP a continuing decline)
                        (PP-LOC in
                                (NP that market))))))
     .))
```

Grammar Refinement



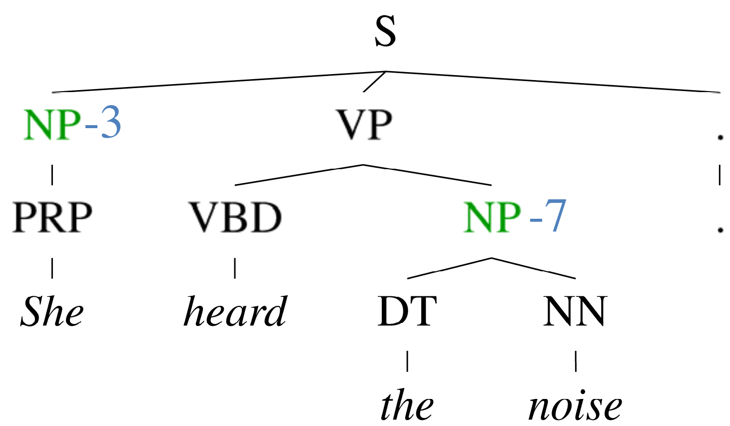
- Conditional independence assumptions often too strong! Not every NP expansion can fill every NP slot
- Better results by enriching the grammar e.g.,
  - Lexicalization [Collins, 1999; Charniak, 2000]

Grammar Refinement



- Conditional independence assumptions often too strong! Not every NP expansion can fill every NP slot
- Better results by enriching the grammar e.g.,
  - Lexicalization [Collins, 1999; Charniak, 2000]
  - Markovization, Manual Tag-splitting [Johnson, 1998; Klein & Manning, 2003]

Grammar Refinement

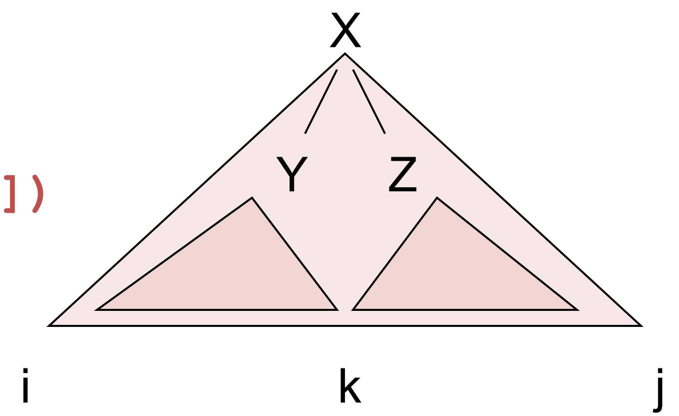


- Conditional independence assumptions often too strong! Not every NP expansion can fill every NP slot

- Better results by enriching the grammar e.g.,
  - Lexicalization [Collins, 1999; Charniak, 2000]
  - Markovization, Manual Tag-splitting [Johnson, 1998; Klein & Manning, 2003]
  - Latent Tag-splitting [Matsuzaki et al., 2005; Petrov et al., 2006]

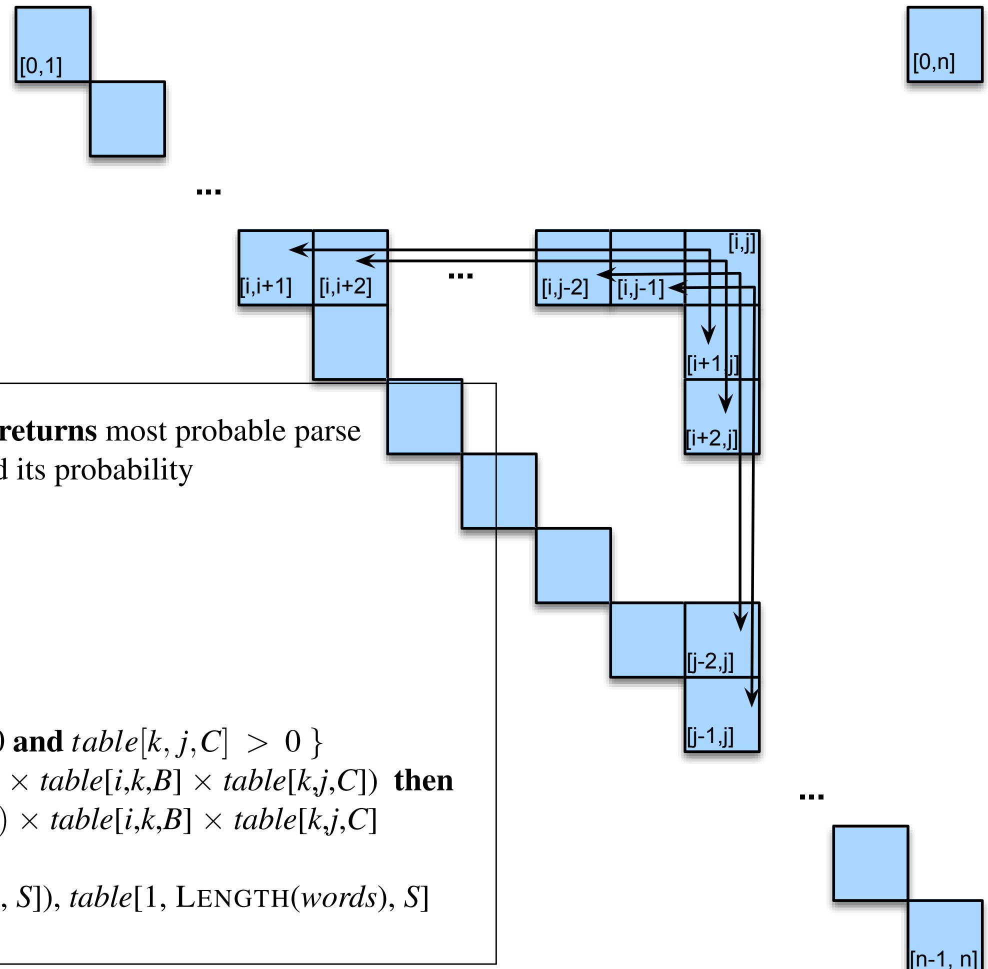
CKY (or CYK) Parsing Algorithm (Bottom-up)

```
bestScore(s)
 for (i : [0,n-1])
   for (X : tags[s[i]])
     score[X][i][i+1] = tagScore(X,s[i])
 for (diff : [2,n])
   for (i : [0,n-diff])
     j = i + diff
     for (X->YZ : rule)
       for (k : [i+1, j-1])
         score[X][i][j] = max{score[X][i][j], score(X->YZ)
                                               *score[Y][i][k]
                                               *score[Z][k][j]}
```



[Cocke, 1970; Kasami, 1965; Younger, 1967]

CKY Parsing Algorithm (Bottom-up)



function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

    for all $\{ A \mid A \rightarrow words[j] \in grammar \}$

      $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$

    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**

      for $k \leftarrow i+1$ **to** $j-1$ **do**

        for all $\{ A \mid A \rightarrow BC \in grammar,$

          and $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$

          if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**

            $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$

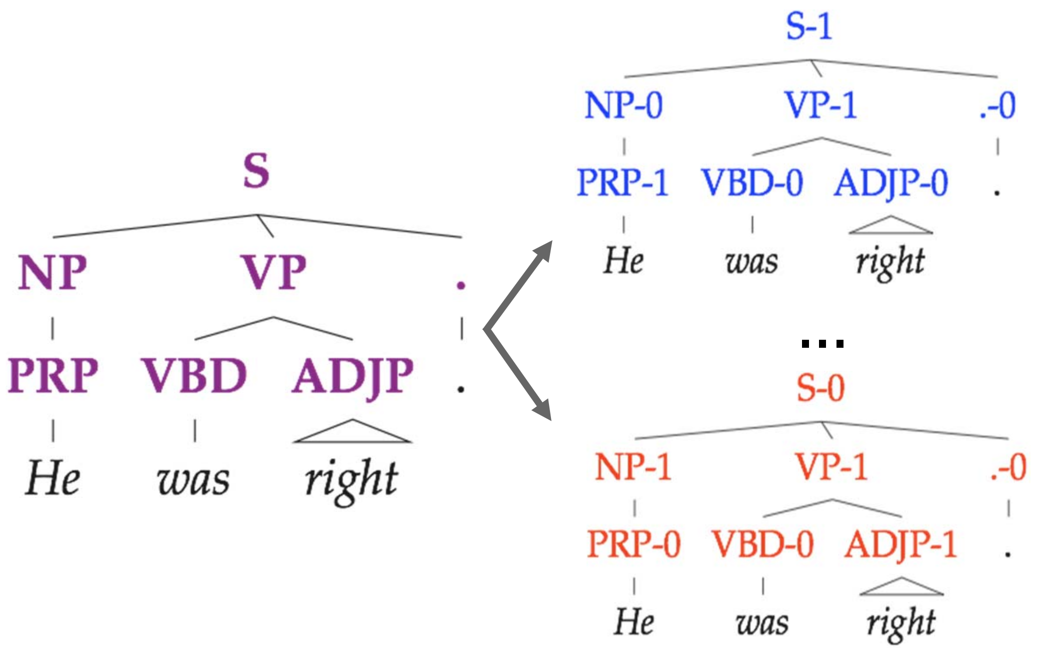
            $back[i,j,A] \leftarrow \{k, B, C\}$

    return BUILD_TREE(*back*[1, LENGTH(*words*), *S*]), *table*[1, LENGTH(*words*), *S*]

Figure 13.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

[Jurafsky-Martin-SLP3]

Latent Variable Grammars



Parse Tree T
Sentence w

Derivations $t : T$

Grammar G	
$S_0 \rightarrow NP_0\ VP_0$	?
$S_0 \rightarrow NP_1\ VP_0$	?
$S_0 \rightarrow NP_0\ VP_1$	?
$S_0 \rightarrow NP_1\ VP_1$	?
$S_1 \rightarrow NP_0\ VP_0$	?
...	
$S_1 \rightarrow NP_1\ VP_1$	?
...	
$NP_0 \rightarrow PRP_0$	?
$NP_0 \rightarrow PRP_1$	?
...	

Lexicon	
$PRP_0 \rightarrow$ She	?
$PRP_1 \rightarrow$ She	?
...	
$VBD_0 \rightarrow$ was	?
$VBD_1 \rightarrow$ was	?
$VBD_2 \rightarrow$ was	?
...	

Parameters θ

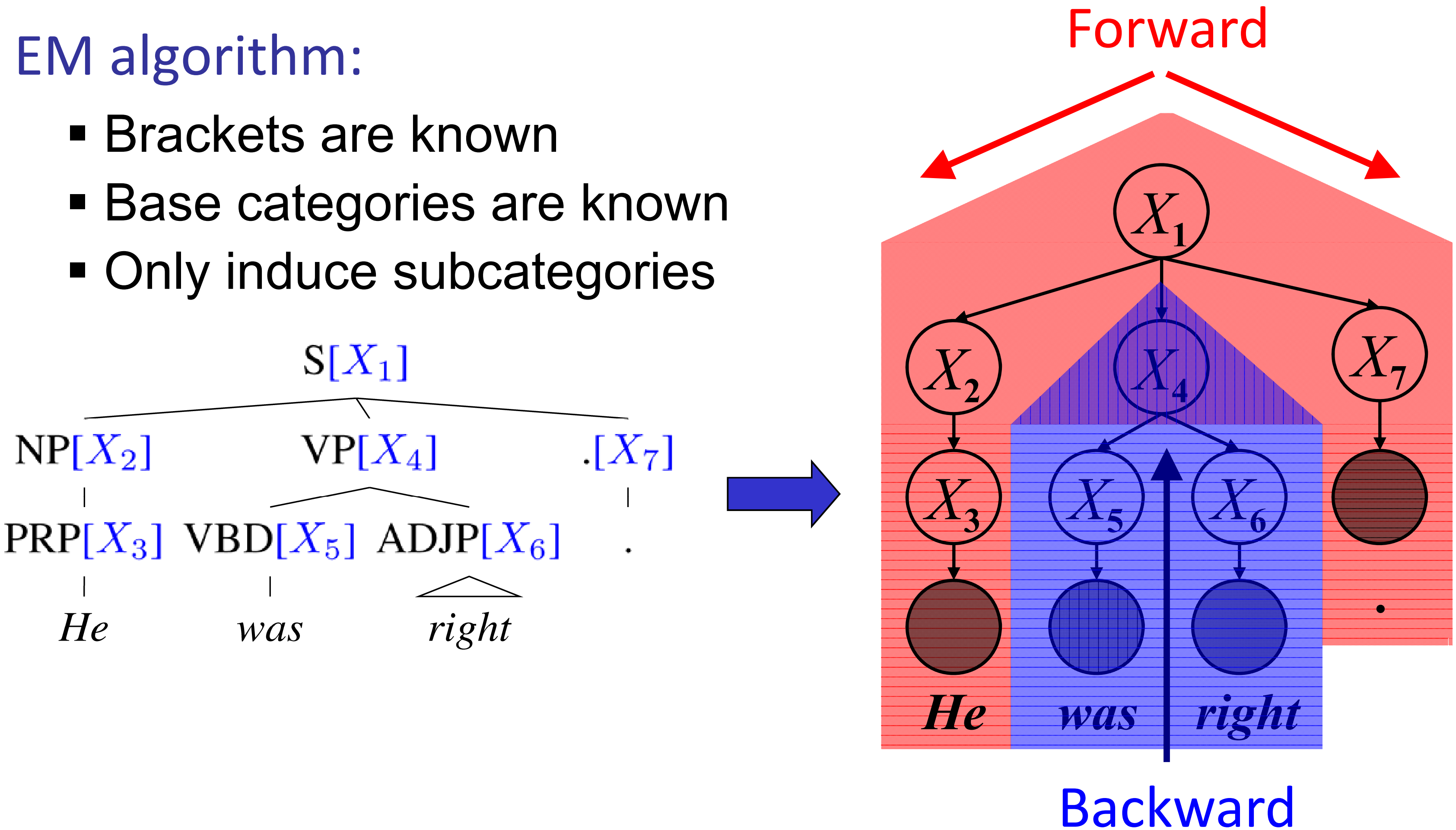
[Petrov et al., 2006]

Learning Latent Splits (Inside-Outside)

- Forward-backward (last week) but for trees

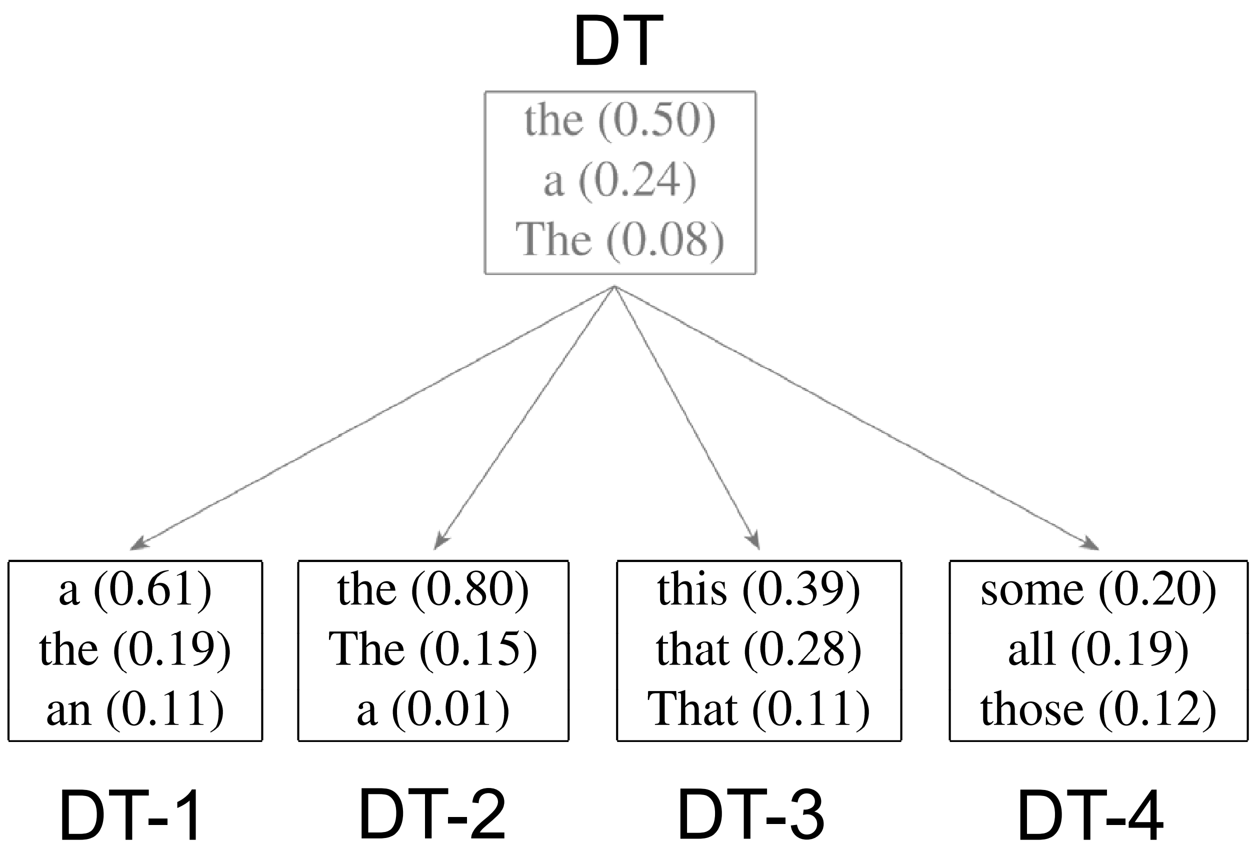
EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories



[Petrov et al., 2006]

DT Tag Splits Example



[Petrov et al., 2006]

Other Learned Splits

- Proper Nouns (NNP):

NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Personal pronouns (PRP):

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

[Petrov et al., 2006]

Other Learned Splits

- Relative adverbs (RBR):

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34

[Petrov et al., 2006]

Latent PCFG Results

		≤ 40 words F1	all F1
ENG	Charniak&Johnson ‘05 (generative)	90.1	89.6
	Split / Merge	**90.6**	**90.1**
GER	Dubey ‘05	76.3	-
	Split / Merge	**80.8**	**80.1**
CHN	Chiang et al. ‘02	80.0	76.6
	Split / Merge	**86.3**	**83.4**

[Petrov et al., 2006]

Evaluating Constituent Parsers

labeled recall: $= \dfrac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of correct constituents in reference parse of } s}$

labeled precision: $= \dfrac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of total constituents in hypothesis parse of } s}$

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R} \qquad\qquad F_1 = \frac{2PR}{P+R}$$

cross-brackets: the number of constituents for which the reference parse has a bracketing such as ((A B) C) but the hypothesis parse has a bracketing such as (A (B C)).

[Jurafsky-Martin-SLP3]

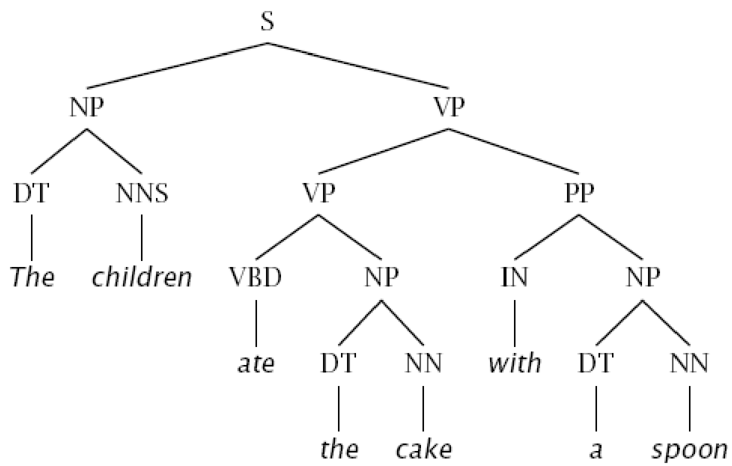
Other Results

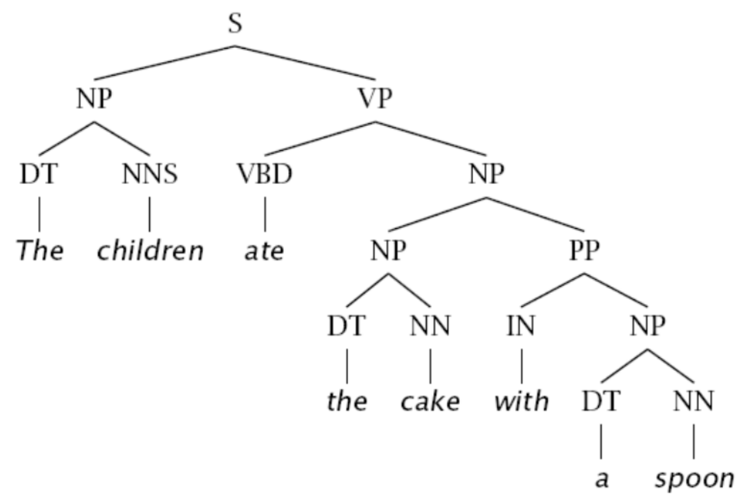
- Collins, 1999 → 88.6 F1 (generative lexical)
- Charniak and Johnson, 2005 → 89.7 / 91.3 F1 (generative lexical / reranking)
- Petrov et al., 2006 → 90.7 F1 (generative unlexical)
- McClosky et al., 2006 – 92.1 F1 (generative + reranking + self-training)

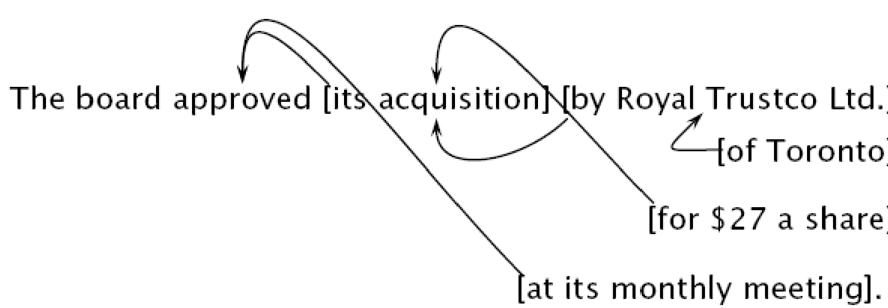
Syntactic Ambiguities

- I saw the old man with a telescope
- I shot an elephant in my pajamas
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

Real-Data PP Attachment Ambiguities







Attachment Ambiguity Types

- Prepositional phrases:
  They cooked the beans in the pot on the stove with handles.
- Particle vs. preposition:
  The puppy tore up the staircase.
- Complement structures
  The tourists objected to the guide that they couldn't hear.
  She knows you like the back of her hand.
- Gerund vs. participial adjective
  Visiting relatives can be boring.
  Changing schedules frequently confused passengers.

Attachment Ambiguity Types

- Modifier scope within NPs
  impractical design requirements
  plastic cup holder

- Multiple gap constructions
  The chicken is ready to eat.
  The contractors are rich enough to sue.

- Coordination scope:
  Small rats and mice can squeeze into holes or cracks in the wall.

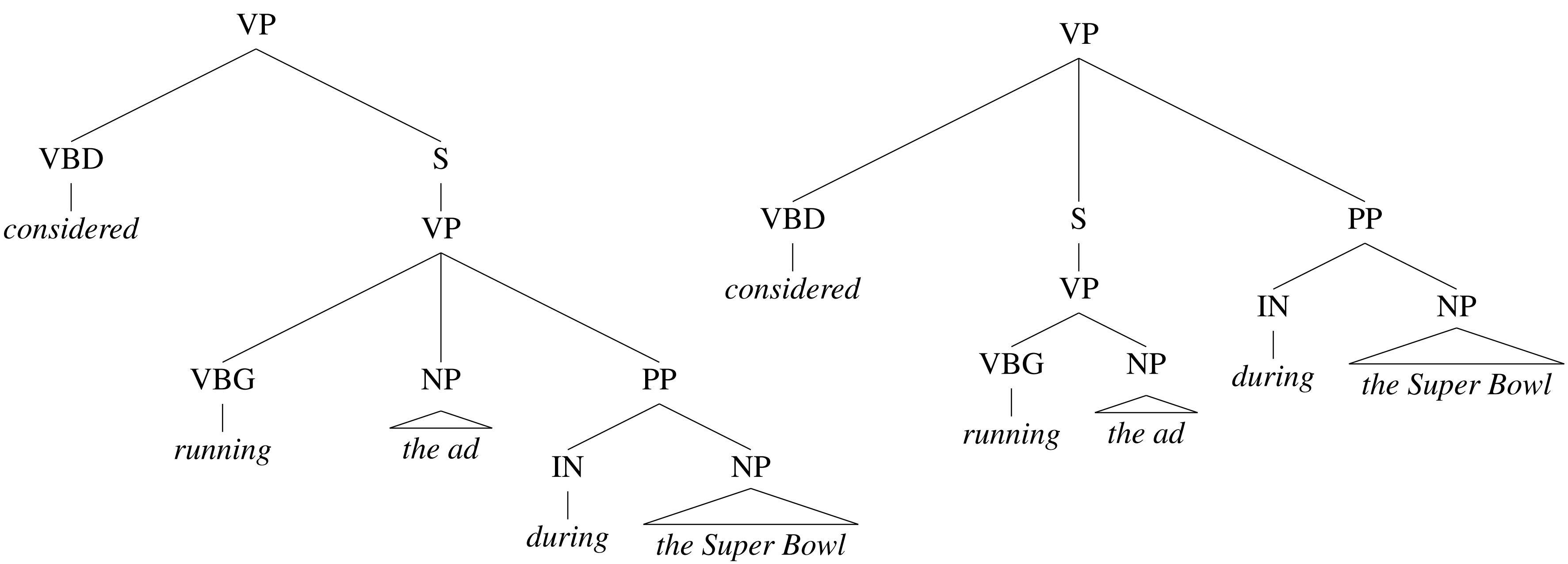
World Knowledge is Important

Web Features for Syntactic Parsing

Dependency:

They considered running the ad during the Super Bowl.

Constituent:



[Nakov and Hearst 2005; Pitler et al., 2010; Bansal and Klein, 2011]

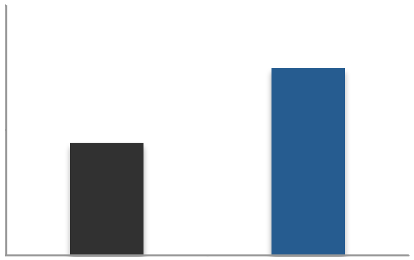
Web Features for Syntactic Parsing

They considered running the ad during the Super Bowl.

Web Ngrams

count(*running it during*) > count(*considered it during*)

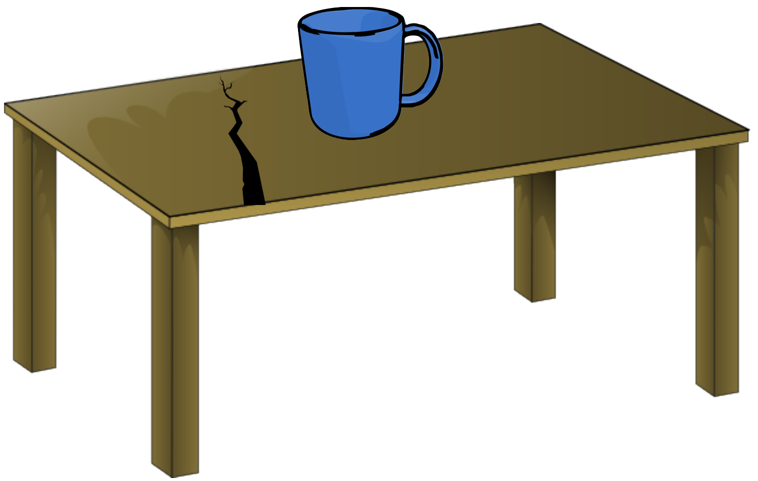
- 7-10% relative error reduction over 90-92% parsers



[Bansal and Klein, 2011]

Visual Recognition Cues

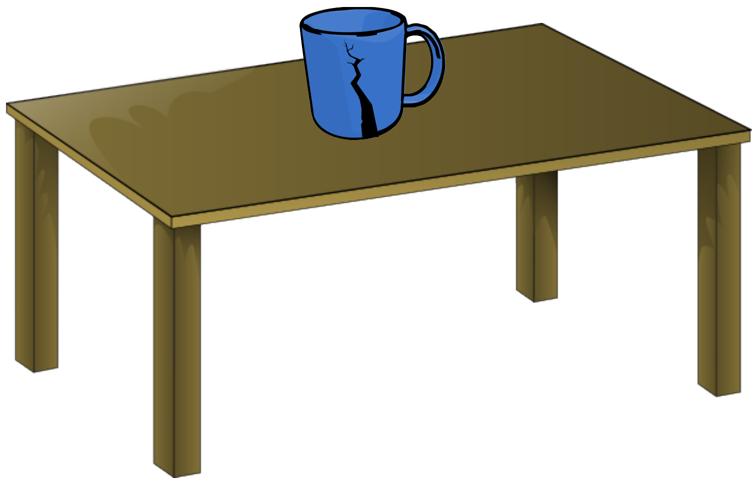
- Joint parsing and image recognition



the mug on the table with a crack

Visual Recognition Cues

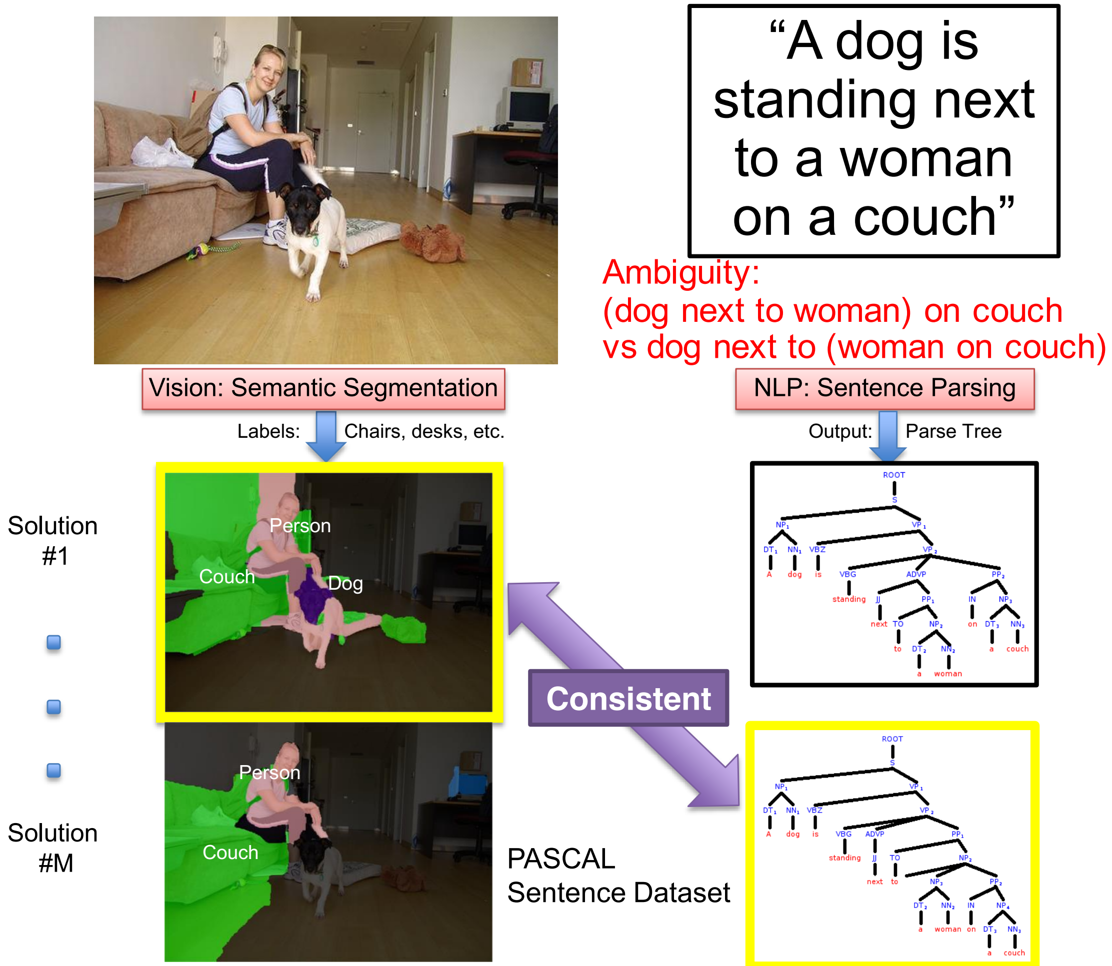
- Joint parsing and image recognition



the mug on the table with a crack

red chair and table
light green table

Visual Recognition Cues

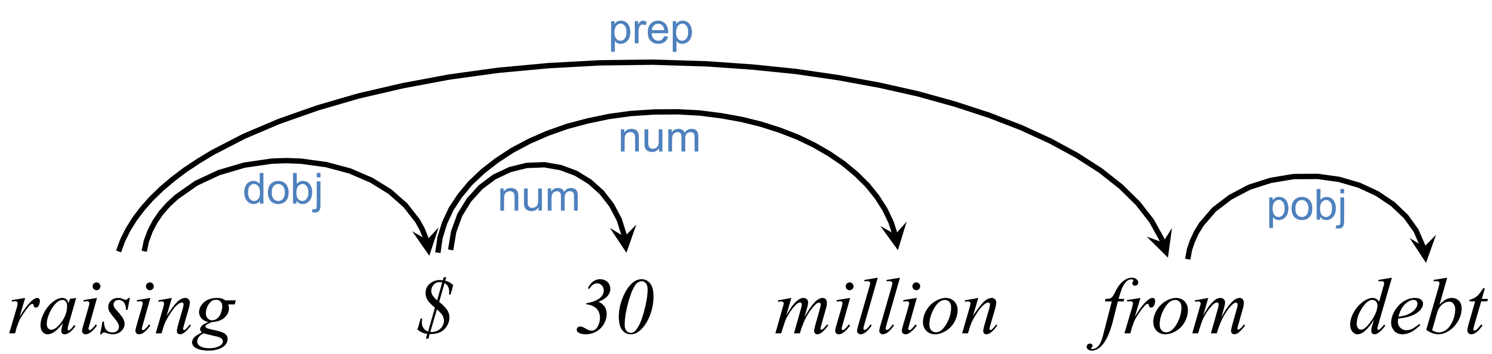


[Christie et al., 2016]

Dependency Parsing

Dependency Parsing

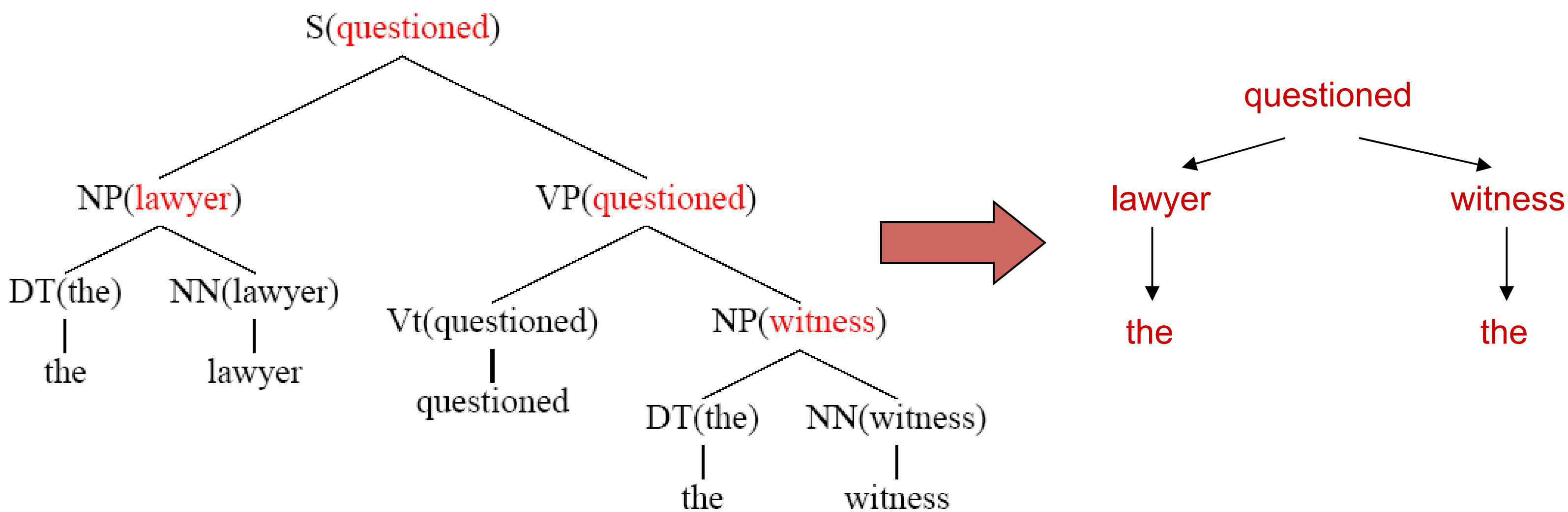
- Predicting directed head-modifier relationship pairs



- Demos: https://corenlp.run/

Dependency Parsing

- Can convert (lexicalized) constituent tree to dependency tree (each local binary tree gives us a dependency attachment from head to modifier)



Constituent Parse (with head words)

Dependency Parse

Dependency Parsing

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

[Jurafsky-Martin-SLP3]

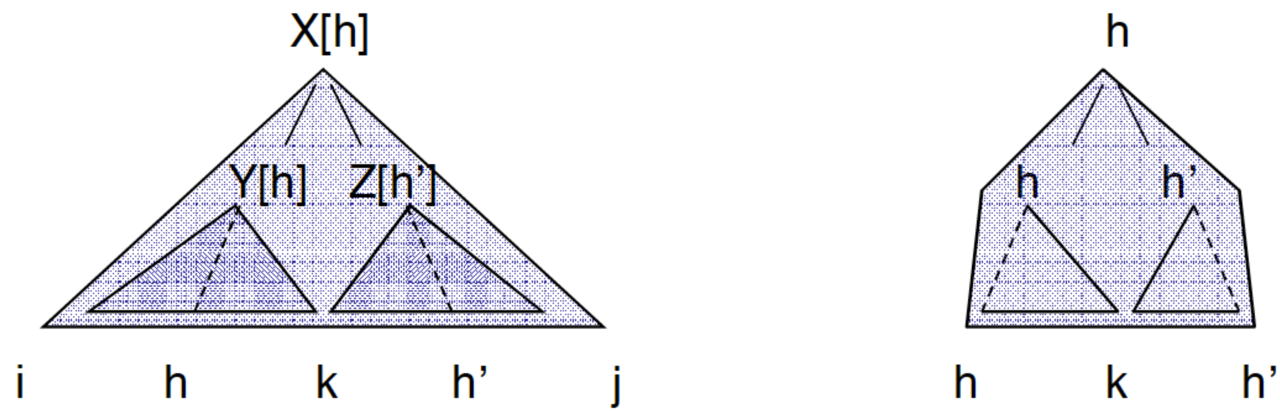
Dependency Parsing

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

[Jurafsky-Martin-SLP3]

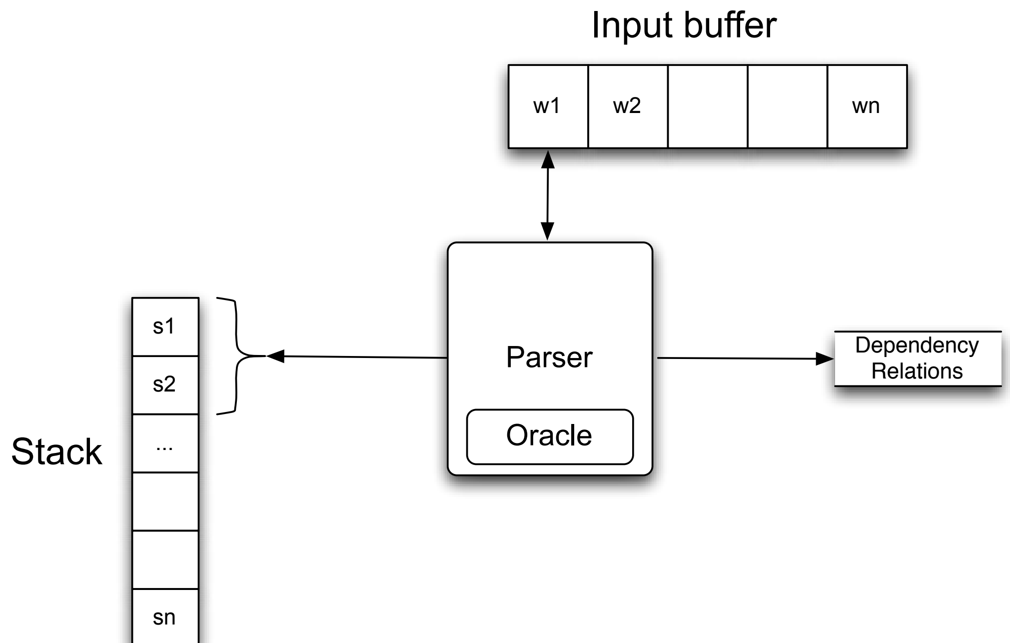
Dependency Parsing

- Pure (projective, 1st order) dependency parsing is only cubic [Eisner, 1996]



- Non-projective dependency parsing useful for Czech & other languages – MST algorithms [McDonald et al., 2005]

Transition-based Dependency Parsing



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

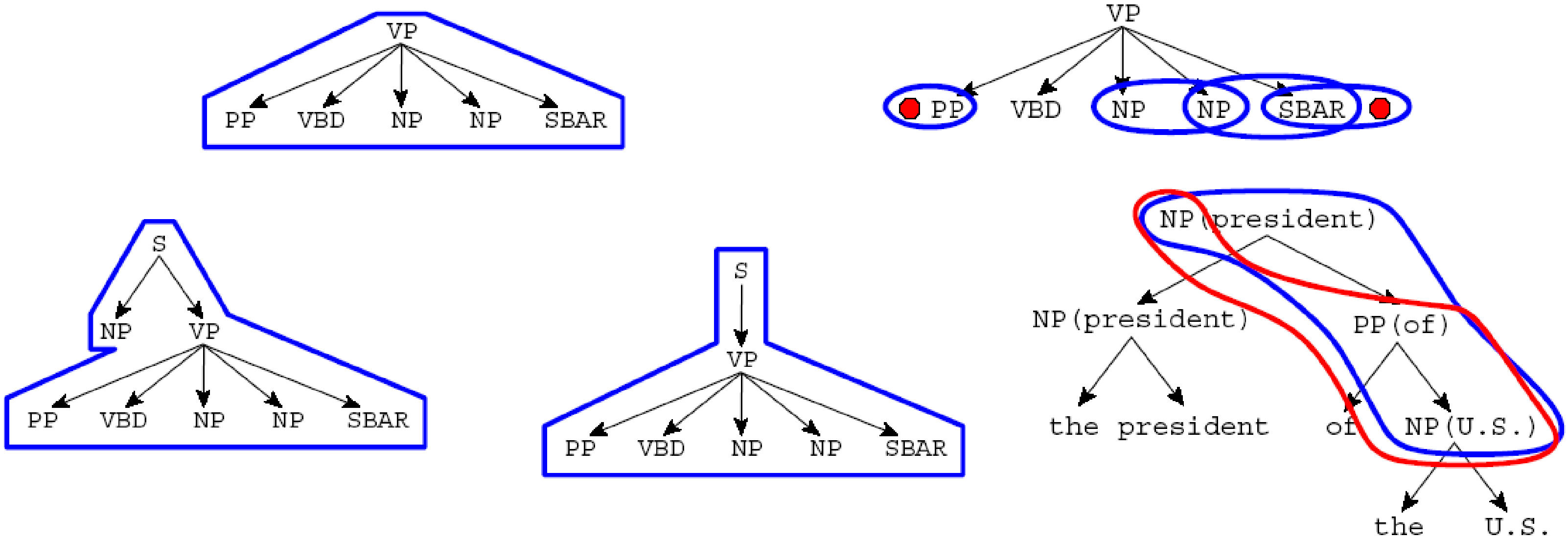
Parsing: Other Models and Methods

- Combinatory Categorial Grammar [Steedman, 1996, 2000; Clark and Curran, 2004] (next week for semantic parsing)
- Transition-based Dependency Parsing [Yamada and Matsumoto, 2003; Nivre, 2003]
- Tree-Insertion Grammar, DOP [Schabes and Waters, 1995; Hwa, 1998; Scha, 1990; Bod, 1993; Goodman, 1996; Bansal and Klein, 2010]
- Tree-Adjoining Grammar [Resnik, 1992; Joshi and Schabes, 1998; Chiang, 2000]
- Shift-Reduce Parser [Nivre and Scholz, 2004; Sagae and Lavie, 2005]
- Other: Reranking, A*, K-Best, Self-training, Co-training, System Combination, Cross-lingual Transfer [Sarkar, 2001; Steedman et al., 2003; Charniak and Johnson, 2005; Hwa et al., 2005; Huang and Chiang, 2005; McClosky et al., 2006; Fossum and Knight, 2009; Pauls and Klein, 2009; McDonald et al., 2011]

- Other Demos: http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo, http://4.easy-ccg.appspot.com/

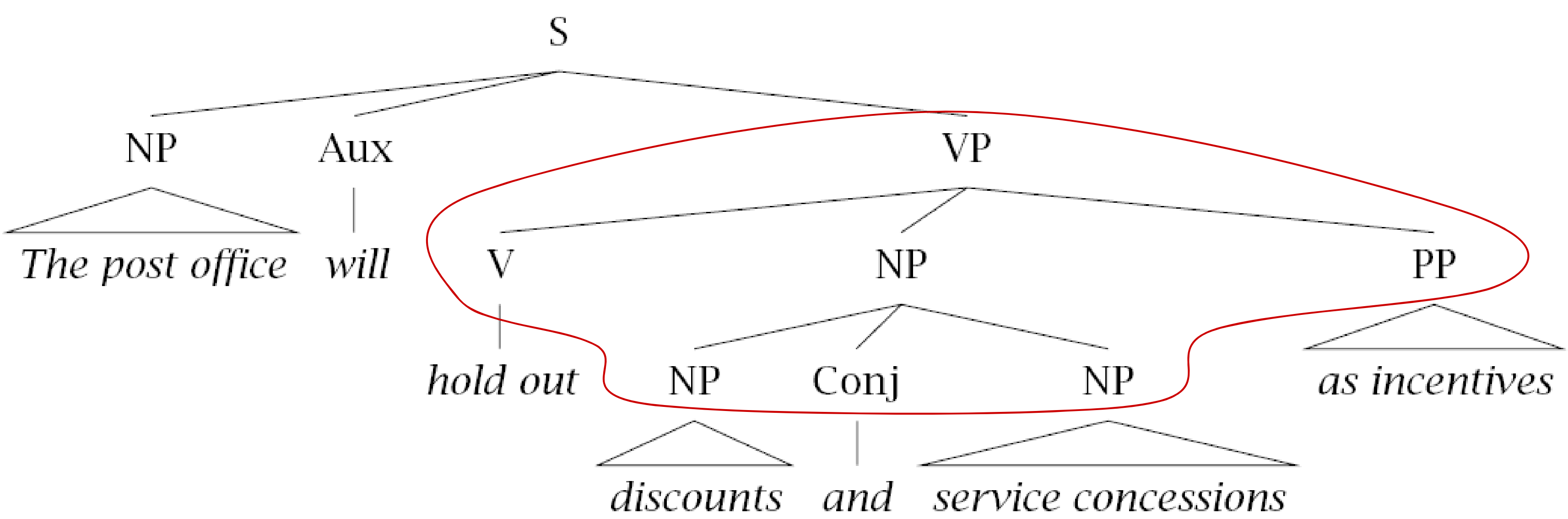
Parser Reranking

- Can first get the *k*-best list of parses based on parser probability
- Then we can fire features on full tree (as opposed to local features in the parser's dynamic program)
- Can fire non-local, global features like tree depth, width, right-branching vs left-branching, etc.
- See [Charniak and Johnson, 2005] for feature list.

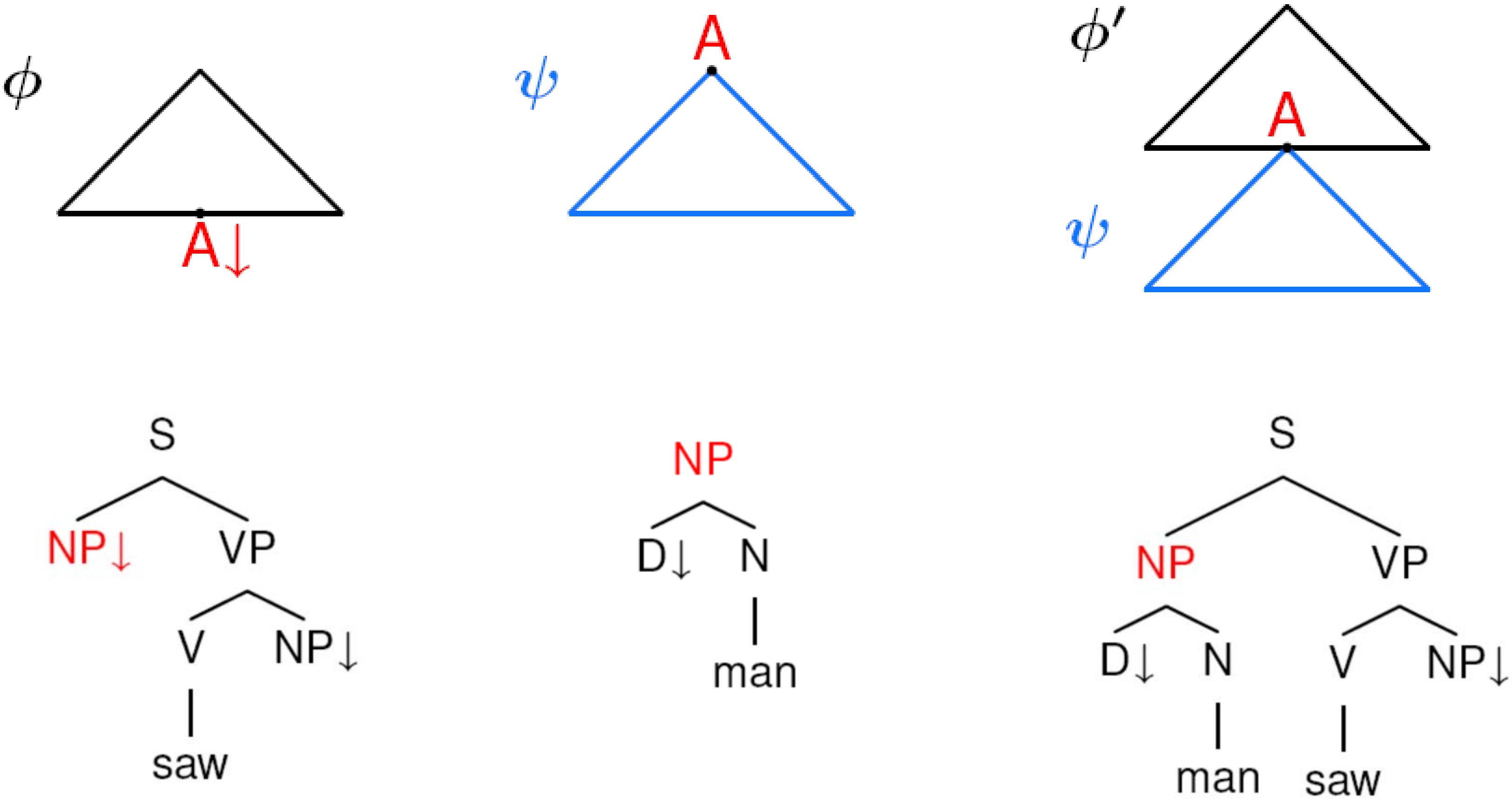


Data Oriented Parsing (TIGs)

- DOP is formally a Tree-Insertion Grammar, i.e., we can rewrite a large subtree in a single step
- Hence, this brings in derivational ambiguity



Data Oriented Parsing (TIGs)

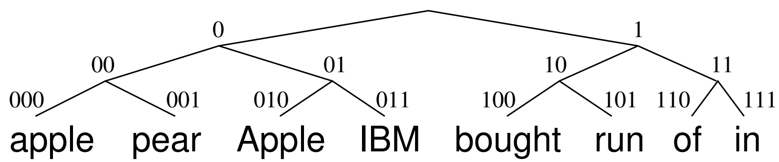


Neural Models for Parsing

Word Embeddings for Parsing

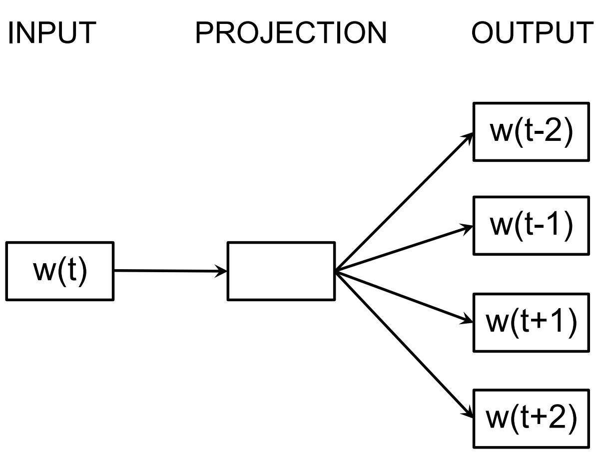
- Discrete or continuous, trained on large amounts of context

- BROWN (Brown et al., 1992):



apple	→	000
pear	→	001
Apple	→	010

- SKIPGRAM (Mikolov et al., 2013):



apple	→	[0.65 0.15 -0.21 0.15 0.70 -0.90]
pear	→	[0.51 0.05 -0.32 0.20 0.80 -0.95]
Apple	→	[0.11 0.33 0.51 -0.05 -0.41 0.50]

[Koo et al., 2008; Bansal et al., 2014]

Word Embeddings for Parsing

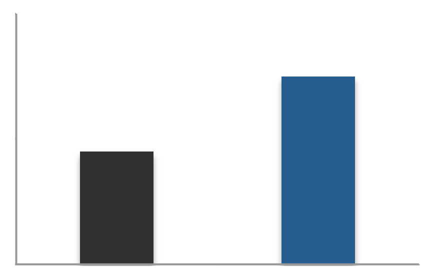
- Condition on dependency context instead of linear, then convert each dependency to a tuple:

dep label	grandparent	parent	child	dep label
$[PMOD_{<L>}$	$regulation_{<G>}$	of	$safety$	$PMOD_{<L>}]$

[*Mr., Mrs., Ms., Prof., III, Jr., Dr.*]
[*Jeffrey, William, Dan, Robert, Stephen, Peter, John, Richard, ...*]
[*Portugal, Iran, Cuba, Ecuador, Greece, Thailand, Indonesia, ...*]

[*his, your, her, its, their, my, our*]
[*Your, Our, Its, My, His, Their, Her*]
[*truly, wildly, politically, financially, completely, potentially, ...*]

- 10% rel. error reduction over 90-92% parsers



[Bansal et al., 2014]

Neural Dependency Parser

[Chen and Manning, 2014; CS224n]

Softmax probabilities

Output layer y

$y = \text{softmax}(Uh + b_2)$

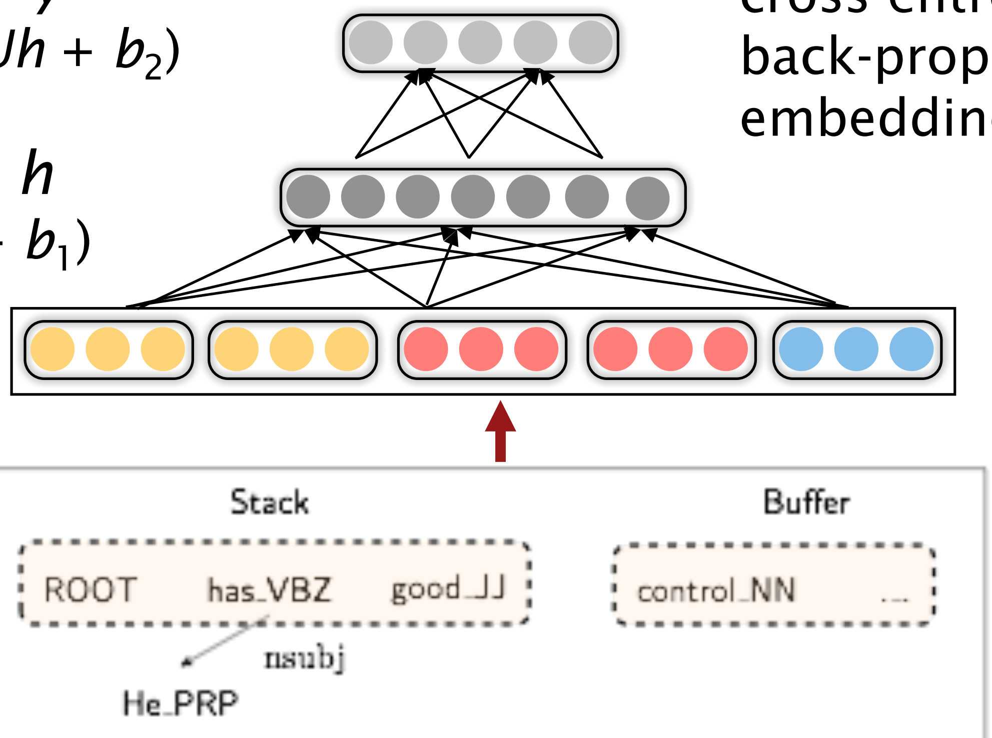
Hidden layer h

$h = \text{ReLU}(Wx + b_1)$

Input layer x

lookup + concat

cross-entropy error will be back-propagated to the embeddings.



https://www.aclweb.org/anthology/D14-1082.pdf

Neural Dependency Parser

[CS224n]

Further developments in transition-based neural dependency parsing

This work was further developed and improved by others, including in particular at Google

- Bigger, deeper networks with better tuned hyperparameters
- Beam search
- Global, conditional random field (CRF)-style inference over the decision sequence

Leading to SyntaxNet and the Parsey McParseFace model

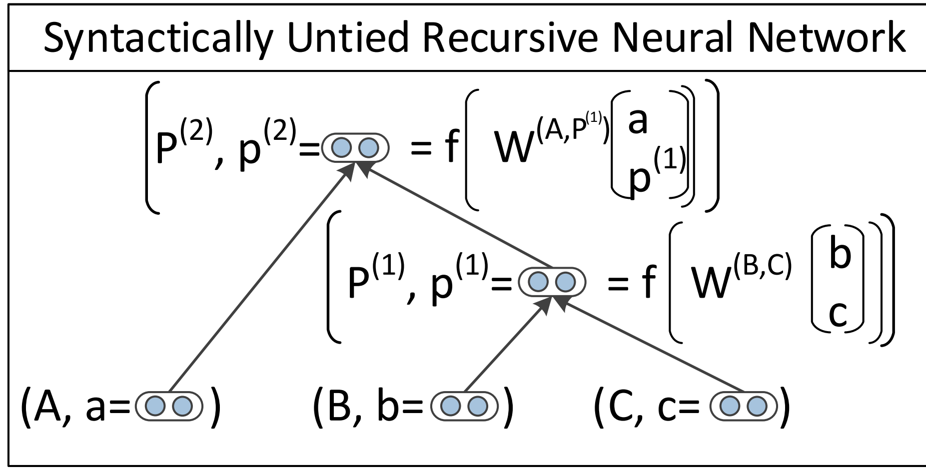
https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Method	UAS	LAS (PTB WSJ SD 3.3
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

Neural Constituent Parser

[Socher et al., 2013; CS224n]

- Compositional Vector Grammar (CVG)



$$p^{(1)} = f\left(W^{(B,C)}\begin{bmatrix} b \\ c \end{bmatrix}\right)$$

$$s\left(p^{(1)}\right) = \left(v^{(B,C)}\right)^T p^{(1)} + \log P(P_1 \rightarrow B \;\; C)$$

$$P((P_1, p_1) \rightarrow (B, b)(C, c))$$
$$= P(p_1 \rightarrow b \;\; c | P_1 \rightarrow B \;\; C) P(P_1 \rightarrow B \;\; C)$$

$$p^{(2)} = f\left(W^{(A,P_1)}\begin{bmatrix} a \\ p^{(1)} \end{bmatrix}\right)$$

$$s\left(p^{(2)}\right) = \left(v^{(A,P_1)}\right)^T p^{(2)} + \log P(P_2 \rightarrow A \;\; P_1)$$

Goodness of a tree is measured in terms of its score and the CVG score of a complete tree is the sum of the scores at each node →

$$s(\texttt{CVG}(\theta, x, \hat{y})) = \sum_{d \in N(\hat{y})} s\left(p^d\right)$$

https://www.aclweb.org/anthology/P13-1045.pdf

Other Recent Neural Parsers

- Recurrent Neural Network Grammars. https://www.aclweb.org/anthology/N16-1024.pdf
- An Improved Non-monotonic Transition System for Dependency Parsing. https://www.aclweb.org/anthology/D15-1162.pdf
- Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. https://www.aclweb.org/anthology/D15-1041.pdf
- Many Languages, One Parser. https://www.aclweb.org/anthology/Q16-1031.pdf
- Cross-lingual Dependency Parsing Based on Distributed Representations. https://www.aclweb.org/anthology/P15-1119.pdf
- Transition-Based Dependency Parsing with Stack Long Short-Term Memory. https://www.aclweb.org/anthology/P15-1033.pdf
- A Fast Unified Model for Parsing and Sentence Understanding. https://www.aclweb.org/anthology/P16-1139.pdf
- Constituency Parsing with a Self-Attentive Encoder. https://www.aclweb.org/anthology/P18-1249.pdf