CS 181: Natural Language Processing

Lecture 12: Statistical Parsing, Features, and Unification

Kim Bruce
Pomona College
Spring 2008

Disclaimer: Slide contents borrowed from many sources on web!

Problems with PCFG's

Problems with Statistical Parsing

- Independence Assumptions:
  - Rules assume probabilities for rules same, no matter where they occur.
- No Lexical Conditioning:
  - Specific words in different subcategories result in different probabilities.

Need to look outside for context, inside for subcategory information!

Independence Assumptions

- NP's that are
  - subjects are pronouns 91% of the time
  - objects are pronouns 34% of the time
  - *Introduce new referents in object, subjects refer to those already introduced.*
  - overall NP's expand to pronouns 25% of time, and to Det NN 28%
  - Must annotate parents to capture info
- Come back to this ...

Lexical Dependencies

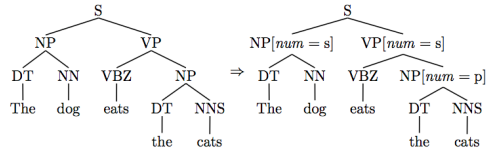
- Prepositional phrase attachment:
  - Attach to object or verb?
  - *John saw the man with the hat.*
    - VP → VBD NP, NP → NP PP
  - *John saw the moon with the telescope.*
    - VP → VBD NP PP
- How can we tell which is preferred?
- Depends on lexical items, not parts of speech. *Annotation key to solution*

Improving PCFG's

- Annotate nodes w/ name of parent
  - E.g., NP^S vs NP^VP
  - 1st is subject, 2nd is object
  - Adverbs similar: RB^ADVP vs RB^VP vs. RB^NP.
- Can split in other ways if distinguishing characteristic occurs elsewhere
- Different tagging systems can help problem.

Subcategorization Helps

- Penn Treebank distinguishes singular/ plural nouns by NN vs NNS and verbs by VBZ vs. VB.
- Propagate up tree



Subcategorization Helps

- Learns high probability for
  - S → NP[num=s] VP[num = s]
- but not
  - S → NP[num=p] VP[num = s]
- Won't make much distinction between
  - VP → VP[num=s] NP[num = s]
  - VP → VP[num=s] NP[num = p]

Subcategorization Helps

- Use subcategorization for sisters
- Learns high probability for
  - VBZ[subcat=NP] → eats
- but not
  - VBZ[subcat=NP] → exists

Data on Rules & Verbs

	come	take	think	want
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%

Disadvantages

- Increasing #tags increases size of grammar
- Need more training data
- Converting to binary rules may help w/ sparseness issues.
- Petrov et al *split and merge* algorithms is best on Treebank data.

Head of phrase

- Key idea in linguistics
  - X-bar theory, Head-driven phase structure grammar.
- Intuitions (12.4.4)
  - Central subconstituent of rule
  - Grammatically most important
  - Semantic predicate of rule
- See pg 27 of Chap 12 for rules.

Lexicalized CFG's

- Annotate tree by lexical heads (key words in phrase)
  - P(VP → V NP NP) likelihood depends on verb: *gave* vs. *ran*

Indicating Heads in Rules

- Add annotations specifying the "head"

Internal Rules
S → NP VP
VP → VBD NP PP
NP → DT NN NP → NNS
PP → P NP

Lexical Rules
VBD → dumped
NNS → sacks NNS → workers
DT → the
P → into
NN → bin

Each rule has one head

Using Your Head

Push head up tree

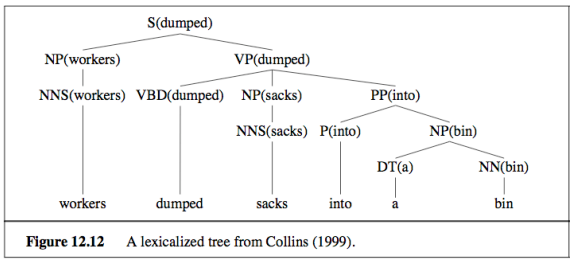
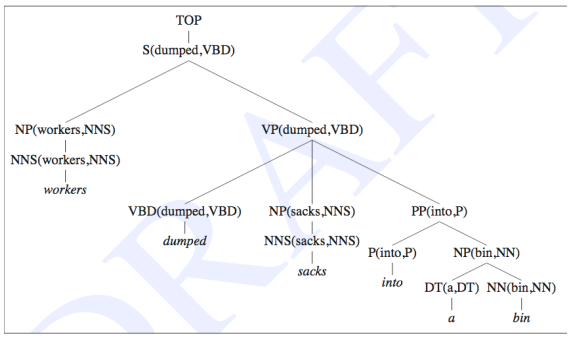


Figure 12.12 A lexicalized tree from Collins (1999).

More Rules ...

- Equivalent to having multiple copies of each rule.
- Sometimes annotate w/ POS of head as well!

Lexicalized Tree w/ Head Tags



Problems

- Specialized rules way too sparse!!
- Must figure out a way of calculating probabilities based on simpler components

Evaluating Parsers

Evaluation

- Recall = $\frac{\text{\# correct labelings}}{\text{total \# correct labelings in gold std}}$
- Precision = $\frac{\text{\# correct labelings in parse}}{\text{total \# constituents in parse}}$
- F-score = $\frac{2PR}{P+R}$
- Cross-brackets # bracketings which cross between reference & hypothetical parses

Features & Unification

Proliferation of Tags

- Do we want 3sgV and other specialized tags?
- Makes sparse data problem worse
- Instead associate properties with tags and require agreement where necessary
  - Nominative/Accusative, gender, tense, singular/plural, comparative/superlative, ...
- Takes us beyond CFG's.

Feature Structures

- Associate feature name with its value

CAT:	NP
Number:	SG
Person:	3
...	...

- Can also nest them:

CAT:	NP	
Agreement:	Number:	SG
	Person:	3

- Refer to <Agreement Number>, etc

Add Notation to CFG

- Constrain grammar w/ agreement
- Write $A_0 \rightarrow A_1 \ldots A_n$ only if
  - <A_i feature path> = Atomic value
  - <A_i feature path> = <A_j feature path>
- Example: S → NP VP only if
  - <NP Number> = <VP number>

Agreement

- Subject-Verb agreement
  - S → NP VP only if
    - <NP agreement> = <VP agreement>
  - Takes into account both number and person
  - S → Aux NP VP only if
    - <NP agreement> = <Aux agreement>
- Determiner-Nominal agreement
  - NP → Det Nom iff
    - <Det Agreement> = <Nom Agreement>

Assigning Values to Features

- Preterminals features come from lexicon:
  - Aux → do
    - <Aux Agreement Number> = PL
    - <Aux Agreement Person> = 3
  - Aux → does
    - <Aux Agreement Number> = SG
    - <Aux Agreement Person> = 3
  - Det → this
    - <Aux Agreement Number> = SG
  - Det → these
    - <Aux Agreement Number> = PL

Using your Head to Move Up the Parse Tree

- NP → Det NOM
  - <NP Agreement> = <NOM Agreement>
- Typically, the features copied are from the head of the phrase.
- VP → Verb NP
  - <VP Agreement> = <Verb Agreement>

Subcategorization

- Subcategorization labels for verbs can be added as features
- Also move up and down tree
  - Verb → hits
    - <Verb Head Agreement Number> = SG
    - <Verb Head Subcat> = DITRANS
    - also INTRANS, TRANS, ... as Subcat

Using Subcategorization

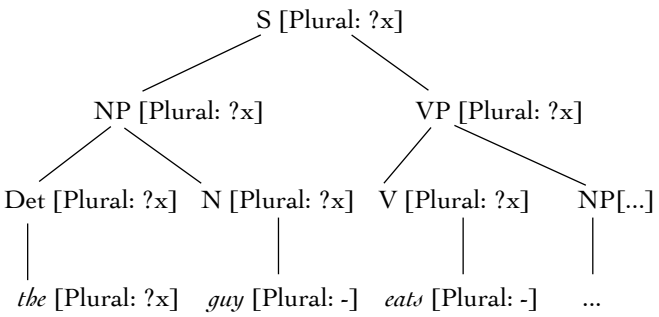
- VP → Verb
  - <VP Head> = <Verb Head>
  - <VP Head Subcat> = INTRANS
- VP → Verb NP
  - <VP Head> = <Verb Head>
  - <VP Head Subcat> = TRANS
- VP → Verb NP NP
  - <VP Head> = <Verb Head>
  - <VP Head Subcat> = DITRANS

Serves as constraint and for copying up

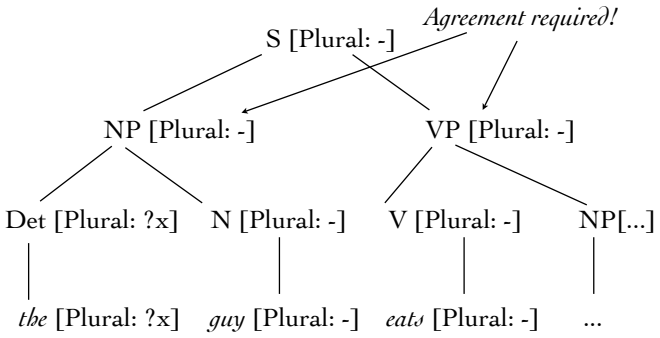
Constraint Solving

- Features are assigned to members of lexicon (may be ambiguity)
  - *Sometimes use +,- for feature values*
- Project up the tree to be used later
- Used to force agreement with sister nodes
- Walk up and down tree to solve constraints

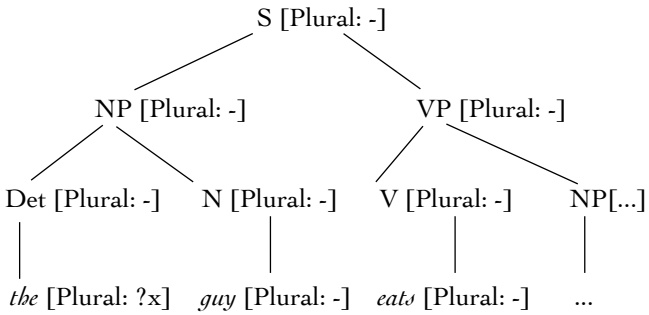
Solving Constraints



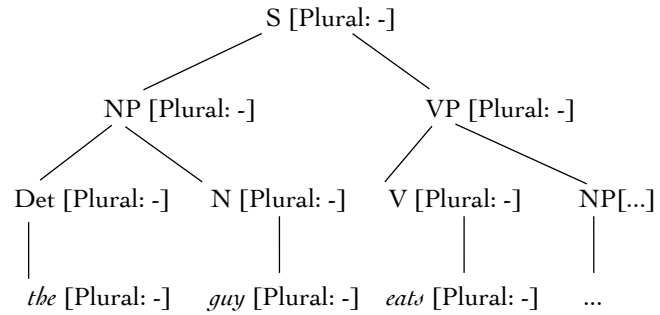
Copying Up



Sister Agreement



Copying Down



Success Not Guaranteed

- Previous would have failed if either NP VP disagreement or Det NP disagreement
- Like type-checking
- Need to formalize feature structures so that we can determine whether can solve unification problem.

Formal Definition

- Feature structures are feature-value pairs where
  - Features are atomic symbols
  - Values are atomic symbols or feature structures

Features for Words

- Some feature structures
  - sleep = {[Cat V], [Plural -], [Person 1]}
  - sleep = {[Cat V], [Plural +], [Person 1]}
  - ...
  - sleeps = {[Cat V], [Plural -], [Person 3]}

Feature Structures

- May have shared features
  - Two paths to same value
  - DAG rather than tree
- In diagrams, indicate replication by shared index for second and later occurrences

Agreement:	①
Subject:	[Agreement: ①]

Operations on Features

- Check consistency
- Merge info in structures
- Unification -- increase information

Examples

- Feature combinations:
  1. [Agreement: [Plural: -,Person: 1st]]
  2. [Agreement: [Plural: -, Nominative: +]]
  3. [Agreement: [Plural: -, Person: 3rd]]
- Unify 1 & 2, 2& 3, but not 1 & 3.

Unification of Features

- As discover more about sentence, add new features from different paths -- unification
- Requires same labels have unifiable values
  - Either same or one is specialization of other
  - [Plural: -] is unifiable with [Plural: Null] but not with [Plural: +]
- Write Null in other ways: ?x, ?y, []

Example

Agreement:	①
Subject:	[Agreement: ①]

∪

Subject:	Agreement:	Person:	3
		Plural:	-

=

Agreement:	①		
Subject:	Agreement: ①	Person:	3
		Plural:	-

Subsumption

- A less specific (more abstract) feature F *subsumes* (written $\subseteq$) another feature G iff
  - For every feature x in F, $F(x) \subseteq G(x)$
  - For all paths p and q in F such that $F(p) = F(q)$, it is also the case that $G(p) = G(q)$
- Can add features or fill in more details, but can't change constraints when go to bigger one. *More information. Semilattice*
- Define $F \cup G$ to be smallest H subsumed by both F and G

Any Questions?