CSC 594 Topics in AI – Applied Natural Language Processing

Fall 2009/2010

4. Grammar and Parsing

Lexical Categories: Parts-of-Speech (POS)

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.

• N	noun	*chair, bandwidth, pacing*
• V	verb	*study, debate, munch*
• ADJ	adjective	*purple, tall, ridiculous*
• ADV	adverb	*unfortunately, slowly*
• P	preposition	*of, by, to*
• PRO	pronoun	*I, me, mine*
• DET	determiner	*the, a, that, those*

Language Structure and Meaning

We want to know how meaning is mapped onto what language structures. Commonly in English in ways like this:

- [THING The dog] is [PLACE in the garden]
- [THING The dog] is [PROPERTY fierce]
- [ACTION [THING The dog] is chasing [THING the cat]]
- [STATE [THING The dog] was sitting [PLACE in the garden] [TIME yesterday]]
- [ACTION [THING We] ran [PATH out into the water]]
- [ACTION [THING The dog] barked [PROPERTY/MANNER loudly]]
- [ACTION [THING The dog] barked [PROPERTY/AMOUNT nonstop for five hours]]

Source: Andrew McCallum, UMass Amherst

Constituency

- Sentences have parts, some of which appear to have subparts. These groupings of words that go together we will call **constituents**.

I hit the man with a cleaver
I hit [the man with a cleaver]
I hit [the man] with a cleaver

You could not go to her party
You [could not] go to her party
You could [not go] to her party

Source: Andrew McCallum, UMass Amherst

4

Constituent Phrases

- For constituents, we usually name them as phrases based on the word that **heads** the constituent:
  - Noun phrase – e.g. [the man], [the man [with a cleaver]]
  - Verb phrase – e.g. [hit the man], [hit the man with a cleaver]
  - Adjective phrase – e.g. [extremely significant]
  - Prepositional phrase – e.g. [with a cleaver], [in my room]

5

Grammar

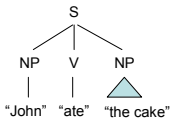
- A formal specification of how a constituent should be made of and combined with other constituents (to form a larger constituent).
- Unlike programming languages (which are *formal languages*), natural languages are born without formal specifications.
- So, grammars for natural languages are empirically derived by observation, which is also subjective.
  ➔ There are many different grammars for a given language.

6

Grammars and Sentence Structure

- The most common way of representing the structure of a sentence is as a **tree** – a *syntax tree*.

"John ate the cake"



7

Context-Free Grammar (CFG)

- The most common way of specifying natural language grammars (because most natural languages are context-free in the generative capacity).
- Formally, a CFG consists of:

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol**

8

Grammaticality

- A CFG defines a formal language = the set of all sentences (strings of words) that can be derived by the grammar.
- Sentences in this set said to be **grammatical**.
- Sentences outside this set said to be **ungrammatical**.

Source: Andrew McCallum, UMass Amherst

9

An Example CFG

G = <N, ∑, S, R> where

N = {S, NP, NOM, VP, Det, Noun, Verb, Aux}
∑ = {*that, this, a, the, man, book, flight, meal, include, read, does*}
S = S
R = {

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

}

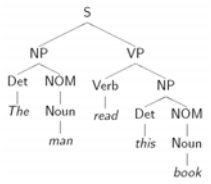
Application of grammar rewrite rules

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

(By top-down derivation)

S → NP VP
→ Det NOM VP
→ *The* NOM VP
→ *The* Noun VP
→ *The man* VP
→ *The man* Verb NP
→ *The man read* NP
→ *The man read* Det NOM
→ *The man read this* NOM
→ *The man read this* Noun
→ *The man read this book*

Parse tree



Chomsky Hierarchy

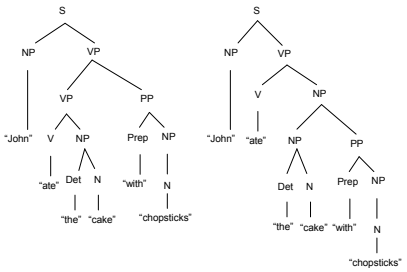
- Type 0 Languages / Grammars
  Rewrite rules $\alpha \rightarrow \beta$
  where α and β are any string of terminals and nonterminals
- Context-sensitive Languages / Grammars
  Rewrite rules $\alpha X \beta \rightarrow \alpha \gamma \beta$
  where X is a non-terminal, and α, β, γ are any string of terminals and nonterminals, (γ must be non-empty).
- Context-free Languages / Grammars
  Rewrite rules $X \rightarrow \gamma$
  where X is a nonterminal and γ is any string of terminals and nonterminals
- Regular Languages / Grammars
  Rewrite rules $X \rightarrow \alpha Y$
  where X, Y are single nonterminals, and α is a string of terminals; Y might be missing.

Recursive Rules, Syntactic Ambiguities

Grammar

S → NP VP
NP → NP PP
NP → Det N
NP → N
VP → VP PP
VP → V NP
PP → Prep NP
NP → "John"
V → "ate"
Det → "the"
N → "cake"
N → "chopsticks"
Prep → "with"

PP attachment ambiguity
"John ate the cake with chopsticks"



13

Agreement

- **Agreement** is a constraint that holds between constituents to make the sentence grammatical.
- For example, in English, determiners and the head nouns in NPs have to agree in their number.
  - This flight
  - (*) This flights
  - Those flights
  - (*) Those flight
- Another example: the head nouns in the subject NPs must agree with the person of the main verb (if 3rd person singular, present tense).
  - He thinks ..
  - (*) He think

14

Problem

- Our earlier NP rules are clearly deficient since they don't capture the agreement constraints
  - *NP → Det Nom*
    - Accepts, and assigns correct structures, to grammatical examples (*this flight*)
    - But its also happy with incorrect examples (*these flight)
  - Such a rule is said to overgenerate → causes ambiguity

15

Verb Subcategorization

- English VPs consist of a head verb along with 0 or more following constituents which we'll call *arguments*.
  - Find: Please find [a flight to NY]$_{NP}$
  - Give: Give [me]$_{NP}$[a cheaper fare]$_{NP}$
  - Help: Can you help [me]$_{NP}$[with a flight]$_{PP}$
  - Prefer: I prefer [to leave earlier]$_{TO\text{-}VP}$
  - Told: I was told [United has a flight]$_{S}$
  - (*) John sneezed the book
  - (*) I prefer United has a flight
  - (*) Give with a flight

Possible CFG Solution

- Possible solution for agreement.
- Can use the same trick for all the verb/VP classes.

- SgS -> SgNP SgVP
- PlS -> PlNp PlVP
- SgNP -> SgDet SgNom
- PlNP -> PlDet PlNom
- PlVP -> PlV NP
- SgVP ->SgV Np
- …

CFG Solution for Agreement

- It works and stays within the power of CFGs
- But its ugly
- And it doesn't scale all that well because of the interaction among the various constraints explodes the number of rules in our grammar.

The Point

- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.
- But there are problems
  - That can be dealt with adequately, although not elegantly, by staying within the CFG framework.
- There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)
  - Lexical Functional Grammar (LFG)
  - Head-driven Phrase Structure Grammar (HPSG)
  - Construction grammar
  - X-Tree Adjoining Grammar (XTAG)
  - Unification Grammar
  - etc.

Source: Jurafsky & Martin "Speech and Language Processing"

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree.
- These are generally created
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.

Source: Jurafsky & Martin "Speech and Language Processing"

Penn Treebank

- Penn TreeBank is a widely used treebank.
  - Most well known is the Wall Street Journal section of the Penn TreeBank.
  - 1 M words from the 1987-1989 Wall Street Journal.

```
( ((S (`` ``)
   (S-TPC-2
     (NP-SBJ-1 (PRP We) )
     (VP (MD would)
       (VP (VB have)
         (S
           (NP-SBJ (-NONE- *-1) )
           (VP (TO to)
             (VP (VB wait)
               (SBAR-TMP (IN until)
                 (S
                   (NP-SBJ (PRP we) )
                   (VP (VBP have)
                     (VP (VBN collected)
                       (PP-CLR (IN on)
                         (NP (DT those)(NNS assets))))))))))))
  (, ,) (`` ``)
  (NP-SBJ (PRP he) )
  (VP (VBD said)
    (S (-NONE- *T*-2) ))
  (. .) ))
```

Source: Jurafsky & Martin "Speech and Language Processing"

Treebank Grammars

- Treebanks implicitly define a grammar for the language covered in the treebank.
- Not complete, but if you have decent size corpus, you'll have a grammar with decent coverage.
- Grammar rules tend to be 'flat' – to avoid recursive rules.
- For example, the Penn Treebank has 4500 different rules for VPs. Among them...

```
VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
```

Source: Jurafsky & Martin "Speech and Language Processing"

22

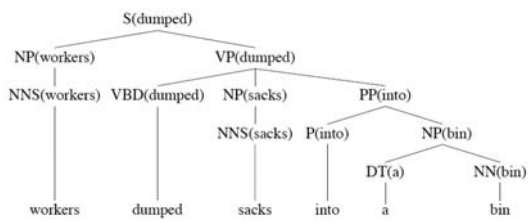
Heads in Trees

- Finding heads in treebank trees is a task that arises frequently in many applications.
  - Particularly important in statistical parsing
- We can visualize this task by annotating the nodes of a parse tree with the heads of each corresponding node.

Source: Jurafsky & Martin "Speech and Language Processing"

23

Lexically Decorated Tree

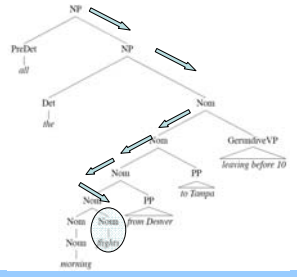


Source: Jurafsky & Martin "Speech and Language Processing"

24

Head Finding

- The standard way to do head finding is to use a simple set of tree traversal rules specific to each non-terminal in the grammar.



Dependency Grammars

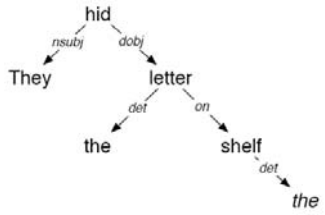
- In CFG-style phrase-structure grammars the main focus is on *constituents.*
- But it turns out you can get a lot done with just binary relations among the words in an utterance.
- In a **dependency grammar** framework, a parse is a tree where
  - the nodes stand for the words in an utterance
  - The links between the words represent dependency relations between pairs of words.
    - Relations may be typed (labeled), or not.

Dependency Relations

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Dependency Tree

"They hid the letter on the shelf"



Source: Jurafsky & Martin "Speech and Language Processing"

Dependency Parsing

- The dependency approach has a number of advantages over full phrase-structure parsing.
  - Deals well with free word order languages where the constituent structure is quite fluid
  - Parsing is much faster than CFG-bases parsers
  - Dependency structure often captures the syntactic relations needed by later applications
    - CFG-based approaches often extract this same information from trees anyway.

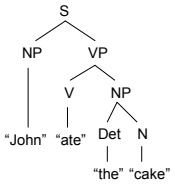
Source: Jurafsky & Martin "Speech and Language Processing"

Syntactic Parsing

- The process of deriving the phrase structure of a sentence is called "parsing".
- The structure (often represented by a Context Free parse tree) is based on the grammar.

Grammar

R0: S → NP VP
R1: NP → Det N
R2: VP → VG NP
R3: VG → V
R4: NP → "John"
R5: V → "ate"
R6: Det → "the"
R7: N → "cake"



Parsing Algorithms

- Top-down Parsing -- (top-down) derivation
- Bottom-up Parsing
- Chart Parsing
- Earley's Algorithm – most efficient, **O(n^3)**
- Left-corner Parsing – optimization of Earley's
- and lots more…

(Bottom-up) Chart Parsing

"John ate the cake"
0 1 2 3 4

Grammar
S → NP VP
NP → Det N
VP → VG NP
VG → V
NP → "John"
V → "ate"
Det → "the"
N → "cake"

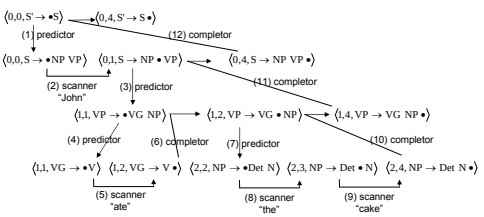
(11) reduce ⟨0,4, S → NP VP •⟩			
---	(10) reduce ⟨1,4, VP → VG NP •⟩		
---	(5) shift 2 ⟨1,2, VP → VG • NP⟩	(9) reduce ⟨2,4, NP → Det N •⟩	
(2) shift 2 ⟨0,1, S → NP • VP⟩	(4) shift 2 ⟨1,2, VG → V •⟩	(7) shift 2 ⟨2,3, NP → Det • N⟩	---
(1) shift 1 "John" ⟨0,1, NP → "John" •⟩	(3) shift 1 "ate" ⟨1,2, V → "ate" •⟩	(6) shift 1 "the" ⟨2,3, Det → "the" •⟩	(8) shift 1 "cake" ⟨3,4, N → "cake" •⟩

0 1 2 3 4

Earley's Algorithm

"John ate the cake"
0 1 2 3 4

Grammar
S → NP VP
NP → Det N
VP → VG NP
VG → V
NP → "John"
V → "ate"
Det → "the"
N → "cake"



Demo using my CF parser

34

Probabilistic Parsing

- For ambiguous sentences, we'd like to know which parse tree is more likely than others.
- So we must assign probability to each parse tree … but how?
- A probability of a parse tree *t* is

$$p(t) = \sum_r p(r)$$ where *r* is a rule used in *t*.

and *p(r)* is obtained from a (annotated) corpus.

35

Partial Parsing

- Parsing fails when the coverage of the grammar is not complete – but it's almost impossible to write out all legal syntax (without accepting ungrammatical sentences).
- We'd like to at least get pieces even when full parsing fails.
- Why not abandon full parsing and aim for partial parsing from the start…

36

Semantic Analysis (1)

- Derive the meaning of a sentence.
- Often applied on the result of syntactic analysis.

  "John ate the cake."

  NP V NP

  ((action INGEST) ; syntactic verb
  (actorJOHN-01) ; syntactic subj
  (object FOOD)) ; syntactic obj
- To do semantic analysis, we need a (semantic) dictionary (e.g. WordNet, http://www.cogsci.princeton.edu/~wn/).

37

Semantic Analysis (2)

- Semantics is a double-edged sword...
  - Can resolve syntactic ambiguity
    - "I saw a man on the hill with a telescope"
    - "I saw a man on the hill with a hat"
  - But introduces semantic ambiguity
    - "She walked towards the bank"
- But in human sentence processing, we seem to resolve both types of ambiguities simultaneously (and in linear time)...

38

Demo using my Unification parser

39