SI485i : NLP

Set 7
Syntax and Parsing
Syntax

• Grammar, or syntax:
  • The kind of implicit knowledge of your native language that you had mastered by the time you were 3 years old
  • Not the kind of stuff you were later taught in “grammar” school

• Verbs, nouns, adjectives, etc.
• Rules: “verbs take noun subjects”…
Example

- “Fed raises interest rates”
Example 2

“I saw the man on the hill with a telescope.”
Example 3

- “I saw her duck”
Syntax

Linguists like to argue

• Phrase-structure grammars, transformational syntax, X-bar theory, principles and parameters, government and binding, GPSG, HPSG, LFG, relational grammar, minimalism.... And on and on.
Syntax

Why should you care?

• Email recovery … n-grams only made local decisions.
• Author detection … couldn’t model word structure
• Sentiment … don’t know what sentiment is targeted at

• Many many other applications:
  • Grammar checkers
  • Dialogue management
  • Question answering
  • Information extraction
  • Machine translation
Syntax

1. Key notions that we’ll cover
   - Part of speech
   - Constituency
   - Ordering
   - Grammatical Relations

2. Key formalism
   - Context-free grammars

3. Resources
   - Treebanks
Word Classes, or *Parts of Speech*

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

- Lots of debate within linguistics about the number, nature, and universality of these
  - We’ll completely ignore this debate.
### POS examples

<table>
<thead>
<tr>
<th>Category</th>
<th>词性</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>noun</td>
<td>chair, bandwidth, pacing</td>
</tr>
<tr>
<td>V</td>
<td>verb</td>
<td>study, debate, munch</td>
</tr>
<tr>
<td>ADJ</td>
<td>adjective</td>
<td>purple, tall, ridiculous</td>
</tr>
<tr>
<td>ADV</td>
<td>adverb</td>
<td>unfortunately, slowly</td>
</tr>
<tr>
<td>P</td>
<td>preposition</td>
<td>of, by, to</td>
</tr>
<tr>
<td>PRO</td>
<td>pronoun</td>
<td>I, me, mine</td>
</tr>
<tr>
<td>DET</td>
<td>determiner</td>
<td>the, a, that, those</td>
</tr>
</tbody>
</table>
POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a collection.

<table>
<thead>
<tr>
<th>word</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>koala</td>
<td>N</td>
</tr>
<tr>
<td>put</td>
<td>V</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>keys</td>
<td>N</td>
</tr>
<tr>
<td>on</td>
<td>P</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>table</td>
<td>N</td>
</tr>
</tbody>
</table>
POS Tags Vary on Context

He will *refuse* to *lead*.

There is *lead* in the *refuse*.
Open and Closed Classes

- **Closed class**: a small fixed membership
  - Usually *function words* (short common words which play a role in grammar)

- **Open class**: new ones created all the time
  - English has 4: Nouns, Verbs, Adjectives, Adverbs
    - Many languages have these 4, but not all!
  - Nouns are typically where the bulk of the action is with respect to new items
Closed Class Words

Examples:

- prepositions: *on, under, over, …*
- particles: *up, down, on, off, …*
- determiners: *a, an, the, …*
- pronouns: *she, who, I, ..*
- conjunctions: *and, but, or, …*
- auxiliary verbs: *can, may should, …*
- numerals: *one, two, three, third, …*
Open Class Words

• **Nouns**
  • Proper nouns (Boulder, Granby, Beyoncé, Port-au-Prince)
    • English capitalizes these.
  • Common nouns (the rest)
  • Count nouns and mass nouns
    • Count: have plurals, get counted: goat/goats, one goat, two goats
    • Mass: don’t get counted (snow, salt, communism) (*two snows)

• **Adverbs**: tend to modify things
  • Unfortunately, John walked home extremely slowly yesterday
  • Directional/locative adverbs (here, home, downhill)
  • Degree adverbs (extremely, very, somewhat)
  • Manner adverbs (slowly, slinkily, delicately)

• **Verbs**
  • In English, have morphological affixes (eat/eats/eaten)
POS: Choosing a Tagset

- Many potential distinctions we can draw
- We need *some* standard set of tags to work with

- We could pick very coarse tagsets
  - N, V, ADJ, ADV

- The finer grained, Penn TreeBank tags (45 tags)
  - VBG, VBD, VBN, PRP$, WRB, WP$
  - Even more fine-grained tagsets exist

Almost all NLPers use these.
## Penn TreeBank POS Tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>coordin. conjunction</td>
<td><em>and, but, or</em></td>
<td>SYM</td>
<td>symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>cardinal number</td>
<td><em>one, two, three</em></td>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>DT</td>
<td>determiner</td>
<td><em>a, the</em></td>
<td>UH</td>
<td>interjection</td>
<td><em>ah, oops</em></td>
</tr>
<tr>
<td>EX</td>
<td>existential ‘there’</td>
<td><em>there</em></td>
<td>VB</td>
<td>verb, base form</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>FW</td>
<td>foreign word</td>
<td><em>mea culpa</em></td>
<td>VBD</td>
<td>verb, past tense</td>
<td><em>ate</em></td>
</tr>
<tr>
<td>IN</td>
<td>preposition/sub-conj</td>
<td><em>of, in, by</em></td>
<td>VBG</td>
<td>verb, gerund</td>
<td><em>eating</em></td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
<td><em>yellow</em></td>
<td>VBN</td>
<td>verb, past participle</td>
<td><em>eaten</em></td>
</tr>
<tr>
<td>JJR</td>
<td>adj., comparative</td>
<td><em>bigger</em></td>
<td>VBP</td>
<td>verb, non-3sg pres</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>JJS</td>
<td>adj., superlative</td>
<td><em>wildest</em></td>
<td>VBZ</td>
<td>verb, 3sg pres</td>
<td><em>eats</em></td>
</tr>
<tr>
<td>LS</td>
<td>list item marker</td>
<td><em>1, 2, One</em></td>
<td>WDT</td>
<td>wh-determiner</td>
<td><em>which, that</em></td>
</tr>
<tr>
<td>MD</td>
<td>modal</td>
<td><em>can, should</em></td>
<td>WP</td>
<td>wh-pronoun</td>
<td><em>what, who</em></td>
</tr>
<tr>
<td>NN</td>
<td>noun, sing. or mass</td>
<td><em>llama</em></td>
<td>WP$</td>
<td>possessive wh-</td>
<td><em>whose</em></td>
</tr>
<tr>
<td>NNS</td>
<td>noun, plural</td>
<td><em>llamas</em></td>
<td>WRB</td>
<td>wh-adverb</td>
<td><em>how, where</em></td>
</tr>
<tr>
<td>NNP</td>
<td>proper noun, singular</td>
<td><em>IBM</em></td>
<td>$</td>
<td>dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>NNPS</td>
<td>proper noun, plural</td>
<td><em>Carolinatas</em></td>
<td>#</td>
<td>pound sign</td>
<td>#</td>
</tr>
<tr>
<td>PDT</td>
<td>predeterminer</td>
<td><em>all, both</em></td>
<td>“</td>
<td>left quote</td>
<td>‘ or “</td>
</tr>
<tr>
<td>POS</td>
<td>possessive ending</td>
<td><em>’s</em></td>
<td>”</td>
<td>right quote</td>
<td>’ or ”</td>
</tr>
<tr>
<td>PRP</td>
<td>personal pronoun</td>
<td><em>I, you, he</em></td>
<td>(</td>
<td>left parenthesis</td>
<td>[, (, {, &lt;</td>
</tr>
<tr>
<td>PRP$</td>
<td>possessive pronoun</td>
<td><em>your, one’s</em></td>
<td>)</td>
<td>right parenthesis</td>
<td>], ), }, &gt;</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
<td><em>quickly, never</em></td>
<td>,</td>
<td>comma</td>
<td>,</td>
</tr>
<tr>
<td>RBR</td>
<td>adverb, comparative</td>
<td><em>faster</em></td>
<td>.</td>
<td>sentence-final punc</td>
<td>. ! ?</td>
</tr>
<tr>
<td>RBS</td>
<td>adverb, superlative</td>
<td><em>fastest</em></td>
<td>:</td>
<td>mid-sentence punc</td>
<td>: ; ... -- -</td>
</tr>
</tbody>
</table>
Important! Not 1-to-1 mapping!

- Words often have more than one POS
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB

- Part of the challenge of *Parsing* is to determine the POS tag for a particular instance of a word. This can change the entire parse tree.

These examples from Dekang Lin
Exercise!

Label each word with its Part of Speech tag!
(look back 2 slides at the POS tag list for help)

1. The bat landed on a honeydew.

2. Parrots were eating under the tall tree.

3. His screw cap holder broke quickly after John sat on it.
Word Classes and Constituency

• Words can be part of a word class (part of speech).

• Words can also join others to form groups!
  • Often called **phrases**
  • Groups of words that share properties is **constituency**

Noun Phrase
“the big blue ball”
Constituency

• Groups of words within utterances act as single units

• These units form coherent classes that can be shown to behave in similar ways
  • With respect to their internal structure
  • And with respect to other units in the language
Constituency

• Internal structure
  • Manipulate the phrase in some way, is it consistent across all constituent members?
    • For example, noun phrases can insert adjectives

• External behavior
  • What other constituents does this one commonly associate with (follows or precedes)?
    • For example, noun phrases can come before verbs
Constituency

• For example, the following are all *noun phrases* in English...

<table>
<thead>
<tr>
<th>Harry the Horse</th>
<th>a high-class spot such as Mindy’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>the Broadway coppers</td>
<td>the reason he comes into the Hot Box</td>
</tr>
<tr>
<td>they</td>
<td>three parties from Brooklyn</td>
</tr>
</tbody>
</table>

• Why? One piece of (external) evidence is that they can all precede verbs.
Exercise!

Try some constituency tests!

1. “eating”
   1. Is this a Verb phrase or Noun phrase? Why?

2. “termite eating”
   1. Is this a Verb phrase or Noun phrase? Why?

3. “eating”
   1. Can this be used as an adjective? Why?
Grammars and Constituency

• There’s nothing easy or obvious about how we come up with right set of constituents and the rules that govern how they combine...

• That’s why there are so many different theories

• Our approach to grammar is generic (and doesn’t correspond to a modern linguistic theory of grammar).
Context-Free Grammars

- Context-free grammars (CFGs)
  - Phrase structure grammars
  - Backus-Naur Form (CNF)

- Consist of
  - Rules
  - Terminals
  - Non-terminals

So…we’ll make CFG rules for all valid noun phrases.
Definition

- Formally, a CFG (you should know this already)

\( N \) a set of **non-terminal symbols** (or variables)
\( \Sigma \) 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,
\( \beta \) is a string of symbols from the infinite set of strings \( (\Sigma \cup N)^* \)
\( S \) a designated **start symbol**
Context-Free Grammars

- **Terminals**
  - We’ll take these to be words (for now)

- **Non-Terminals**
  - The constituents in a language
    - Like noun phrase, verb phrase and sentence

- **Rules**
  - Rules consist of a single non-terminal on the left and any number of terminals and non-terminals on the right.
Some NP Rules

• Here are some rules for our noun phrases

\[
\begin{align*}
NP & \rightarrow \text{Det Nominal} \\
NP & \rightarrow \text{ProperNoun} \\
\text{Nominal} & \rightarrow \text{Noun} \mid \text{Nominal Noun}
\end{align*}
\]

• These describe two kinds of NPs.
  • One that consists of a determiner followed by a nominal
  • One that says that proper names are NPs.
  • The third rule illustrates two things
    • An explicit disjunction (Two kinds of nominals)
    • A recursive definition (Same non-terminal on the right and left)
# Example Grammar

<table>
<thead>
<tr>
<th>Grammar Rules</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP \ VP$</td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>I</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>$NP \rightarrow Det Nominal$</td>
<td>a + flight</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Nominal Noun</td>
<td>morning + flight</td>
</tr>
<tr>
<td>Nominal $\rightarrow$ Noun</td>
<td>flights</td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td>do</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP$</td>
<td>want + a flight</td>
</tr>
<tr>
<td>$VP \rightarrow Verb NP PP$</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>$VP \rightarrow Verb PP$</td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td>$PP \rightarrow Preposition NP$</td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>
Generativity

- As with FSAs and FSTs, you can view these rules as either analysis or synthesis engines
  - Generate strings in the language
  - Reject strings not in the language
  - Impose structures (trees) on strings in the language
Derivations

- A *derivation* is a sequence of rules applied to a string that *accounts* for that string
  - Covers all the elements in the string
  - Covers only the elements in the string
Parsing

- Parsing is the process of taking a string and a grammar and returning parse tree(s) for that string.
Sentence Types

• Declaratives:  
  \[ S \rightarrow NP \ VP \]
  \[ A \ plane \ left. \]

• Imperatives:  
  \[ S \rightarrow VP \]
  \[ Leave! \]

• Yes-No Questions:  
  \[ S \rightarrow Aux \ NP \ VP \]
  \[ Did \ the \ plane \ leave? \]

• WH Questions:  
  \[ S \rightarrow WH-NP \ Aux \ NP \ VP \]
  \[ When \ did \ the \ plane \ leave? \]
Phrases and Agreement
Noun Phrases

• Let’s consider the following rule in more detail...

\[ NP \rightarrow \text{Det Nominal} \]

• Most of the complexity of English noun phrases is hidden inside this one rule.
Determiners

• Noun phrases can start with determiners...
• Determiners can be
  • Simple lexical items: the, this, a, an, etc.
    • A car
  • Or simple possessives
    • John’s car
  • Or complex recursive versions of that
    • John’s sister’s husband’s son’s car
Nominals

- Contains the main noun and any pre- and post-modifiers of the head.
  - Pre-
    - Quantifiers, cardinals, ordinals...
      - Three cars
    - Adjectives and Aps
      - large cars
    - Ordering constraints
      - Three large cars
      - ?large three cars
Agreement

• By *agreement*, we have in mind constraints that hold among various constituents that take part in a rule or set of rules.
• For example, in English, determiners and the head nouns in NPs have to agree in their number.

This flight  
Those flights

*This flights  
*Those flight
Verb Phrases

- English VPs consist of a head verb along with 0 or more following constituents which we’ll call arguments.

\[
\begin{align*}
VP & \rightarrow \text{Verb} \quad \text{disappear} \\
VP & \rightarrow \text{Verb NP} \quad \text{prefer a morning flight} \\
VP & \rightarrow \text{Verb NP PP} \quad \text{leave Boston in the morning} \\
VP & \rightarrow \text{Verb PP} \quad \text{leaving on Thursday}
\end{align*}
\]
Subcategorization

• Not all verbs are allowed to participate in all those VP rules.
• We can *subcategorize* the verbs in a language according to the sets of VP rules that they participate in.
• This is just a variation on the traditional notion of transitive/intransitive.
• Modern grammars may have 100s of such classes
Subcategorization

• Sneeze: John sneezed
• 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-VP}
• Told: I was told [United has a flight]_{S}
• ...
Programming Analogy

• Verbs are like functions

• Each verb takes a certain number and type of parameters

• A verb’s subcategorization frame specifies the number, position and types.
Subcategorization

• *John sneezed the book
• *I prefer United has a flight
• *Give with a flight

• As with agreement phenomena, we need a way to formally express these facts
Why subcategorization?

• Right now, the various rules for VPs overgenerate.
  • They permit the presence of strings containing verbs and arguments that don’t go together

• Overgeneration example
  • VP -> V NP
  • Sneezed the book is a VP since “sneeze” is a verb and “the book” is a valid NP
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 it is ugly
- It doesn’t scale because the interaction among constraints explodes the number of rules in our grammar.
The Ugly Reality

• CFGs 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)
  • LFG, HPSG, Construction grammar, XTAG, etc.
  • Chapter 15 explores the unification approach in more detail
CFG $\rightarrow$ PCFG
What do we as computer scientists?

- Stop trying to hardcode all possibilities.
- Find a bunch of sentences and parse them by hand.
- Build a *probabilistic* CFG over the parse trees, implicitly capturing these nasty constraints with probabilities.
Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree.
- These are auto-manually created:
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary.

- This requires detailed annotation guidelines, a POS tagset, and a grammar and instructions for how to deal with particular grammatical constructions.
Penn Treebank

- Penn TreeBank is a widely used treebank.

Most well known part is the Wall Street Journal section of the Penn TreeBank.

Create a Treebank Grammar

- Use labeled trees as your grammar!

- Simply take the local rules that make up all sub-trees
  - The WSJ section gives us about 12k rules if you do this

- Not complete, but if you have decent size corpus, you’ll have a grammar with decent coverage.
Learned Treebank Grammars

• Such grammars tend to be very flat due to the fact that they tend to avoid recursion.
  • To ease the annotators' burden, among things
• The Penn Treebank has ~4500 different rules for VPs. Among them...

\[
\begin{align*}
  \text{VP} & \rightarrow \text{VBD} \quad \text{PP} \\
  \text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \\
  \text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \quad \text{PP} \\
  \text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \quad \text{PP} \quad \text{PP}
\end{align*}
\]
Lexically Decorated Tree

S(dumped)
  NP(workers)
    NNS(workers)
       workers
    VBD(dumped)
       dumped
  VP(dumped)
    NP(sacks)
      NNS(sacks)
        sacks
      P(into)
        into
    PP(into)
      NP(bin)
        DT(a)
          a
        NN(bin)
          bin
Treebank Uses

- Treebanks are particularly critical to the development of statistical parsers
  - Chapter 14
- Also valuable to *Corpus Linguistics*
  - Investigating the empirical details of various constructions in a given language