CMPT-413: Computational Linguistics

Anoop Sarkar

anoop@cs.sfu.ca

www.sfu.ca/˜anoop/courses/CMPT-413-Spring-2003.html

Parts of Speech

- We have seen that individual words can be classified into groups or classes that we call **parts of speech**
  - Determiners: *a, the*
  - Verbs: *arrive, attracts, love, sit*
  - Prepositions: *of, by, in, outside, on*
  - Nouns: *he, she, it, San, Diego*
- But these individual words can group together to form larger groups which possess meaning when put together, e.g. *San Diego*, *the man outside the building*

Constituents

- Let’s consider the grouping of words into **noun phrases**
  - *three parties from Brooklyn*
  - *a high class spot such as Mindy’s*
  - *they*
  - *Harry the Horse*
  - *the fact that he came into the Hot Box*
  - *swimming on a hot day*

Constituents

- These *noun phrases* are selected by verbs as a whole unit:
  - three parties from Brooklyn *arrived* . . .
  - ∗ three from *arrived* . . .
  - a high class spot such as Mindy's *attracts* . . .
  - they *sit* . . .
  - they *like* swimming on a hot day

Constituents

- These *noun phrases* are selected by verbs as a whole unit:
  - three parties from Brooklyn *arrived* . . .
  - $*$ three from *arrived* . . .
  - a high class spot such as Mindy's *attracts* . . .
  - they *sit* . . .
  - they *like* swimming on a hot day

Testing for constituents

- Things that can be moved around together: *preposed* or *postposed* elements in a sentence.
  - *On Sept 17th, I'd like to fly to Toronto*
  - *I'd like to fly, On Sept 17th, to Toronto*
  - *I'd like to fly to Toronto On Sept 17th*
  - ∗ *On I'd like to fly Sept to Toronto 17th*

Testing for constituents

- Things that can be questioned:
  - Who came to the negotiating table?
    three parties from Brooklyn
  - Where would a high roller like Deckard go?
    a high class spot such as Mindy's
  - What is it that Mary would like to do when she visits?
    swimming on a hot day

Testing for constituents

- Things that can be coordinated:
  - *John and Mary*
  - *the barrier islands and frogs that provide hallucinations when you lick them*
  - *swimming on a hot day and taking a long skiing lesson*
- Can you think of some cases that do not pass all three of these tests?

Finding Noun Phrases

- Finding noun phrases is often called **chunking**
- Instance of finding a sequence – can be thought of as another application for regular expressions
- First part of speech tags should be assigned so that we can write regular expressions that do not refer to words
- Then write a *regular grammar* of noun phrase chunks: use Perl regexps

Chunking Noun Phrases: Not as easy as it seems

- *(NNP San) (NNP Diego)*
- *(NNPS Wednesdays)*
- *(DT the) (NN company) (POS ’s) (VBN refocused) (NN direction)*
- *(DT the) (NN government) (VBZ ’s) (VBG dawdling)*
- $*$ *(DT The) (NNP Dow) (NNP Jones) (VBZ is) (VBG swimming) (IN in) (NN tech) (NNS stocks)*

Chunking as Part-of-Speech Tagging

```
>> [ John/NNP Smith/NNP ] ,/, [ president/NN ]
      I         I          O        I
        of/IN [ IBM/NNP ] ./.
         O       I         O

>> [ Pundits/NNS ] condemned/VBD [ terrorism/NN ]
         I             O             I
        and [ assassination/NN ] ./.
         O          I            O
```

Chunking as Part-of-Speech Tagging

```
>> [ John/NNP ] saw/VBD [the/DT cat/NN]
      I           O         I      I
        [the/DT dog/NN] liked/VBD ./.
          B      I       O         O
```

Recursion in Regular Languages

- Consider a regular expression for arithmetic expressions:
  $2 + 3 * 4,\ 8 * 10 + -24$

  ```
  ^\s*-?\s*\d+\s*((\+|\*)\s*-?\s*\d+\s*)*$
  ```

- *Can we compute the* meaning *of these expressions?*

Recursion in Regular Languages

- Construct the finite state automata and associate the meaning with the state sequence (just like in part of speech tagging)

- Or think of it as a transducer that produces the final result from the sequence

- However, this solution is missing something crucial about arithmetic expressions – *what is it?*

Recursion in Regular Languages

- Going back to noun phrases (NP, for short): let's attempt to provide a regular expression grammar for a subset of all the possible noun phrases
- Consider the noun phrases: *the man in the park*, *the person with the big head in the park*, *the unicorn in the garden inside the dream with a strange mark on the head*, . . .
- These are simple noun phrases that have prepositional phrases (PP, for short) modifying nouns. PPs are another example of a constituent, but now we need to combine them with NPs

Recursion in Regular Languages

- Consider the noun phrases: *the man in the park*, *the person with the big head in the park*, *the unicorn in the garden inside the dream with a strange mark on the head*, . . .

- (NP) (PP)* $\rightarrow$ (Det N) (PP)$_*$ $\rightarrow$ (Det N) (P NP)*

- (Det N) (P (Det N)) PP* $\rightarrow$ (Det N) (P (Det N))*

- So, it's possible, but it gets ugly fast, let's widen our view of what can occur inside NPs.

Recursion in Regular Languages

- Let’s call (Det N) a **basal** NP and now consider that (Det N) is not the only base NP that is possible: (N) or (A N) or (A^+ N) or even: (D A^* N POS N) *the short man ’s dream* . . .
- So this means that we can now have (P (N)) or (P (A N)) or (P (A^+ N)) or . . .
- Each former type of NP can be modified by each latter type of PP
- What is the only way to rescue the regular expression approach? combinatorial explosion of combinations

Context-Free Languages

- Clearly, this and other issues with the kind of recursion possible in regular languages is a problem if we want to describe natural languages
  Recall our morphological FSA which over-generated and produced bogus words like *demonizableable* **because** of recursion

- We need to look at a class of formal languages that generalizes regular languages: **Context-Free Languages**

Context-Free Grammars

- Recall the trinity of regular expressions, finite state automata and regular languages

- Now we generalize to context free grammars, pushdown automata and context-free languages

- Just like before, certain closure properties hold, the union of two CFLs is also a CFL, etc.
  except for one crucial property that is true in RLs but not in CFLs

Context-Free Grammars

- Determinization is also not always possible for pushdown automata surprising fact about CFGs is that you can construct one that is *inherently* ambiguous
- Particular relevance for natural languages, compare with artificial grammars that we use routinely when we use a programming language (what happens in cases of ambiguity in finite state automata?)
- Deterministic vs. non-deterministic parsing (more on this later)

Context-Free Grammars

- A CFG is a 4-tuple: (N, T, P, S), where
  - N is a set of non-terminal symbols,
  - T is a set of terminal symbols which can include the empty string ϵ. T is analogous to Σ the alphabet in FSAs.
  - P is a set of rules of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in \{N \cup T\}^*$
  - S is a set of start symbols, $S \in N$

Context-Free Grammars

- Here's an example of a CFG, let's call this one G:

  1. $S \rightarrow a\ S\ b$
  2. $S \rightarrow \epsilon$

- What is the language of this grammar, which we will call $L(G)$, the set of strings *generated* by this grammar How?
  Notice that there cannot be any FSA that corresponds exactly to this set of strings $L(G)$ Why?

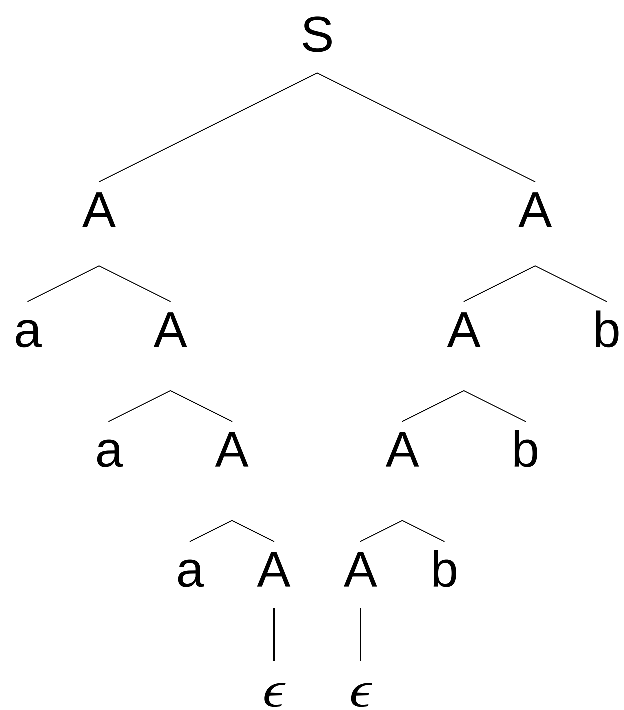
- What is the *tree set* or derivations produced by this grammar?

Context-Free Grammars

- This notion of generating both the strings and the trees is an important one for Computational Linguistics

- Consider the trees for the grammar G':
  $P = \{S \rightarrow A\,A, A \rightarrow aA, A \rightarrow A\,b, A \rightarrow \epsilon\}$,
  $\Sigma = \{a, b\}, N = \{S, A\}, T = \{a, b, \epsilon\}, S = \{S\}$

- Why is it called *context-free* grammar?



Can the grammar G' produce only trees of the kind shown above?

Context-Free Grammars

- We will come back to this issue when we try to figure out whether human languages are more powerful than CFLs.

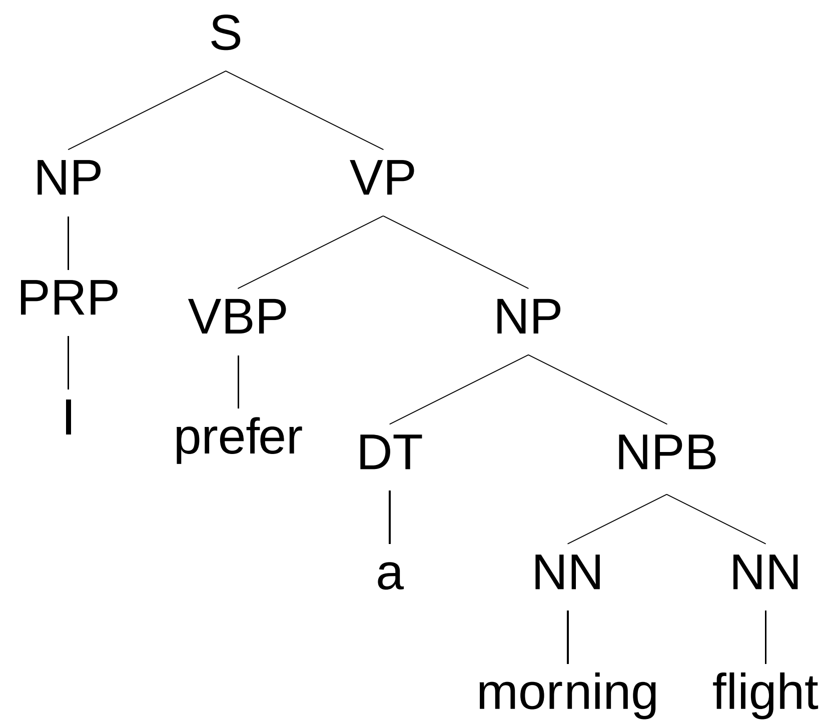
- The distinction between strings and the trees (or any kind of structural description) is called *weak* vs. *strong* generative capacity.

Parse Trees

Consider the grammar with rules:

$$
\begin{array}{rcl}
S & \rightarrow & \textit{NP VP} \\
\textit{NP} & \rightarrow & \textit{PRP} \\
\textit{NP} & \rightarrow & \textit{DT NPB} \\
\textit{VP} & \rightarrow & \textit{VBP NP} \\
\textit{NPB} & \rightarrow & \textit{NN NN} \\
\textit{PRP} & \rightarrow & \textit{I} \\
\textit{VBP} & \rightarrow & \textit{prefer} \\
\textit{DT} & \rightarrow & \textit{a} \\
\textit{NN} & \rightarrow & \textit{morning} \\
\textit{NN} & \rightarrow & \textit{flight}
\end{array}
$$

Parse Trees



Parse Trees: Equivalent Representations

- (S (NP (PRP I)) (VP (VBP prefer) (NP (DT a) (NPB (NN morning) (NN flight)))))

- $[_{S}$ $[_{NP}$ $[_{PRP}$ I]] $[_{VP}$ $[_{VBP}$ prefer] $[_{NP}$ $[_{DT}$ a] $[_{NPB}$ $[_{NN}$ morning] $[_{NN}$ flight]]]]]

Inherently Ambiguous Grammars

- $S \rightarrow S\ S$
- $S \rightarrow a$
- Given the above rules, consider the input *aaa*, what are the valid parse trees?
- Now consider the input *aaaa*