Introduction to NLP

*Introduction to Parsing*
# Parsing programming languages

```c
#include <stdio.h>

int main()
{
    int n, reverse = 0;

    printf("Enter a number to reverse\n");
    scanf("%d", &n);

    while (n != 0)
    {
        reverse = reverse * 10;
        reverse = reverse + n % 10;
        n = n / 10;
    }

    printf("Reverse of entered number is = %d\n", reverse);

    return 0;
}
```
Parsing human languages

• Rather different than computer languages
  – Can you think in which ways?
Parsing human languages

• Rather different than computer languages
  – No types for words
  – No brackets around phrases
  – Ambiguity
    • Words
    • Parses
  – Implied information
The parsing problem

• Parsing means associating tree structures to a sentence, given a grammar (often a CFG)
  – There may be exactly one such tree structure
  – There may be many such structures
  – There may be none

• Grammars (e.g., CFG) are declarative
  – They don’t specify how the parse tree will be constructed
Syntactic ambiguities

• PP attachment
  – I saw the man with the telescope

• Gaps
  – Mary likes Physics but hates Chemistry

• Coordination scope
  – Small boys and girls are playing

• Particles vs. prepositions
  – She ran up a large bill

• Gerund vs. adjective
  – Frightening kids can cause trouble
Applications of parsing

• Grammar checking
  – I want to return this shoes.

• Question answering
  – How many people in sales make $40K or more per year?

• Machine translation
  – E.g., word order – SVO vs. SOV

• Information extraction
  – *Breaking Bad* takes place in New Mexico.

• Speech generation

• Speech understanding
Introduction to NLP

Context-free grammars
Context-free grammars

• A context–free grammar is a 4–tuple \((N, \Sigma, R, S)\)
  – \(N\): non–terminal symbols
  – \(\Sigma\): terminal symbols (disjoint from \(N\))
  – \(R\): rules \((A \rightarrow \beta)\), where \(\beta\) is a string from \((\Sigma \cup N)^*\)
  – \(S\): start symbol from \(N\)
Example

["the", "child", "ate", "the", "cake", "with", "the", "fork"]

S  ->  NP  VP
NP  ->  DT  N  |  NP  PP
PP  ->  PRP  NP
VP  ->  V  NP  |  VP  PP
DT  ->  'a'  |  'the'
DT  ->  'a'  |  'the'
N  ->  'child'  |  'cake'  |  'fork'
PRP  ->  'with'  |  'to'
V  ->  'saw'  |  'ate'
Example

["the", "child", "ate", "the", "cake", "with", "the", "fork"]

S  -> NP  VP
NP  -> DT  N  |  NP  PP
PP  -> PRP  NP
VP  -> V  NP  |  VP  PP
DT  -> 'a'  |  'the'
N  -> 'child'  |  'cake'  |  'fork'
PRP  -> 'with'  |  'to'
V  -> 'saw'  |  'ate'

Heads marked in bold face
Phrase-structure grammars (1/2)

• Sentences are not just bags of words
  – Alice bought Bob flowers
  – Bob bought Alice flowers

• Context–free view of language
  – A prepositional phrase looks the same whether it is part of the subject NP or part of the VP

• Constituent order
  – SVO (subject verb object)
  – SOV (subject object verb)
Phrase-structure grammars (2/2)

- Auxiliary verbs
  - The dog may have eaten my homework
- Imperative sentences
  - Leave the book on the table
- Interrogative sentences
  - Did the customer have a complaint?
- Negative sentences
  - The customer didn’t have a complaint
A longer example

S -> NP VP | Aux NP VP | VP
NP -> PRON | Det Nom
Nom -> N | Nom N | Nom PP
PP -> PRP NP
VP -> V | V NP | VP PP
Det -> 'the' | 'a' | 'this'
PRON -> 'he' | 'she'
N -> 'book' | 'boys' | 'girl'
PRP -> 'with' | 'in'
V -> 'takes' | 'take'

What changes were made to the grammar?
A longer example

S \rightarrow \text{NP VP} \mid \text{Aux NP VP} \mid \text{VP}
\text{NP} \rightarrow \text{PRON} \mid \text{Det Nom}
\text{Nom} \rightarrow \text{N} \mid \text{Nom N} \mid \text{Nom PP}
\text{PP} \rightarrow \text{PRP NP}
\text{VP} \rightarrow \text{V} \mid \text{V NP} \mid \text{VP PP}
\text{Det} \rightarrow \text{'}the' \mid \text{'}a' \mid \text{'}this'
\text{PRON} \rightarrow \text{'}he' \mid \text{'}she'
\text{N} \rightarrow \text{'}book' \mid \text{'}boys' \mid \text{'}girl'
\text{PRP} \rightarrow \text{'}with' \mid \text{'}in'
\text{V} \rightarrow \text{'}takes' \mid \text{'}take'
A longer example

S → NP VP | Aux NP VP | VP
NP → PRON | Det Nom
Nom → N | Nom N | Nom PP
PP → PRP NP
VP → V | V NP | VP PP
Det → 'the' | 'a' | 'this'
PRON → 'he' | 'she'
N → 'book' | 'boys' | 'girl'
PRP → 'with' | 'in'
V → 'takes' | 'take'
( (S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
  )
  (ADJP
    (NP (CD 61) (NNS years))
    (JJ old)
    (, ,)
  )
  (VP (MD will)
    (VP (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director))
        (NP-TMP (NNP Nov.) (CD 29))
      )
    )
    (, .)
  )
) ( (S
  (NP-SBJ (NNP Mr.) (NNP Vinken))
  (VP (VBZ is)
    (NP-PRD
      (NP (NN chairman))
      (PP (IN of)
        (NP
          (NP (NNP Elsevier) (NNP N.V.))
          (, ,)
          (NP (DT the) (NNP Dutch) (VBG publishing) (NN group)))))
    (, .)
  )
)
Center Embedding

• Center Embedding
  – The rat ate the seed.
  – The rat that the cat ate ate the seed.
  – The rat that the cat that the dog ate ate ate the seed.
  – ...

• Is this language a CFL?

• Notes
  – CFG cannot describe bounded recursion
  – Competence vs. performance
CFGs are equivalent to PDAs

• Example
  – Consider the language $x^n y^n$
  – stack is empty, input=xxxxyyy
  – push * onto stack, input=xxyyyy
  – push * onto stack, input=xyyyy
  – push * onto stack, input=yyyy
  – pop * from stack, input=yy
  – pop * from stack, input=y
  – pop * from stack, input=""
Leftmost derivation

• A leftmost derivation is a sequence of strings $s_1, s_2, ..., s_n$
  
  – $s_1 = S$, the start symbol
  
  – $s_n$ includes only terminal symbols

• Example:
  
  – [S]
  
  – [S] [NP VP]
  
  – [S] [NP VP] [DT N VP]
  
  – ...
  
  – [S] [NP VP] [DT N VP] ... [the child ate the cake with the fork]
the child ate the cake with the fork.

Leftmost derivation