Natural Language Processing

3007/7059 Artificial Intelligence

Slides by Wei Zhang

School of Computer Science

University of Adelaide

Outline

- **Natural Language Understanding**
- **Grammars**

Note: Parts of this lecture are based on material from: Steven Bird, Ewan Klein, and Edward Loper, Natural Language Processing with Python: http://www.nltk.org/book/

Parts of this lecture are based on material from: AI, COMP329, Macquarie University

What is Natural Language Processing

"*Natural language processing is a range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like languages processing for a range of particular tasks or applications*."
by Liddy (1998)

Some other names:

- Computational Linguistics
- Natural Language Engineering
- Speech and Text Processing

What is Natural Language Processing

The Basics of Natural Language Processing

https://www.youtube.com/watch?v=d4gGtcobq8M

Applications of NLP

- Spelling & grammar checking
- Speech recognition & synthesis
- Sentiment analysis
- Automatic summarization
- Machine translation
- Information retrieval
- Information extraction
- …

Natural Language Understanding

- When you hear the declarative sentence
  The cat sits on the mat.
  you know when it is true.
- When you hear the instruction
- *Remove the head from the herring and fillet the fish.*
  you know what action is required.
- When you hear the question
  What time is it?
  you know that it is a request to give information.

Natural Language Understanding

- When you hear the sentence

  I was born in Zurich.

  then you acquire information that allows you to draw conclusions about the speaker.

- When you hear the sentence

  Yolande went to Rome by train.

you can conclude that Yolande bought a ticket and that she is not longer in the city she started her journey from.

Natural Language Understanding

- Understanding an utterance means
  - relating new information to what we already know,
  - drawing conclusions from the revised information,
  - knowing what an appropriate response to that utterance is.

Natural Language Understanding

- Understanding depends on
  - the result of parsing,
  - the lexical information,
  - the actual context, and
  - commonsense reasoning.

Natural Language Understanding

NLU is hard

- Ambiguity
  - Lexical ambiguity
  - Syntactic ambiguity
  - Semantic ambiguity
  - Anaphoric ambiguity
  - Literal/non-literal meaning
- Vagueness
- …

Lexical Ambiguity

- Words have multiple meanings.
- *I saw a bat.*
  - bat = flying mammal / wooden club?
  - saw = past tense of *see* / present tense of *saw* (to cut with a saw.)

Syntactic Ambiguity

- Syntactic ambiguity is a situation where an analysis may result in more than one parse tree.
- Particularly common sources of ambiguity in English are:
  - prepositional phrase (pp) attachment
  - conjunction
  - noun group structure

Syntactic Ambiguity

- *Mary ate a salad with spinach from Italy for lunch on Wednesday.*
  - *with spinach* can attach to *salad* or *ate*;
  - *from Italy* can attach to *spinach*, *salad*, or *ate*;
  - *for lunch* can attach to *Italy*, *spinach*, *salad*, or *ate*;
  - *on Wednesday* can attach to *lunch*, *Italy*, *spinach*, *salad* or *ate*.
- There exist 42 possible different parse trees for this sentence.

Stanford Parser

One possible parse tree:

```
(ROOT
  (S
    (NP (NNP Mary))
    (VP (VBD ate)
      (NP (DT a) (NN salad))
      (PP (IN with)
        (NP (NN spinach)))
      (PP (IN from)
        (NP
          (NP (NNP Italy))
          (PP (IN for)
            (NP
              (NP (NN lunch))
              (PP (IN on)
                (NP (NNP Wednesday))))))))
    (. .)))
```

http://nlp.stanford.edu:8080/parser/index.jsp

Stanford Parser

Another possible parse tree:

```
(S (NP Mary)
   (VP ate
       (NP a salad)
       (PP with
           (NP spinach))
       (PP from
           (NP Italy))
       (PP for
           (NP lunch))
       (PP on
           (NP Wednesday)))
   .)
```

http://www.link.cs.cmu.edu/link/submit-sentence-4.html

Semantic Ambiguity

- Even after the lexical ambiguity of the individual words and the syntactic structure have been resolved, there are two ways of reading the following sentences:
  - *Every man loves a woman.*
  - *John and Mary are married.*
  - *John kissed his wife, and so did Sam.*

Anaphoric Ambiguity

- A phrase or word refers to something previously mentioned, but there is more than one antecedent.
- *Margaret invited Susan for a visit, and she gave her a good lunch.* (she = Margaret; her = Susan)
- *Margaret invited Susan for a visit, but she told her she had to go to work.* (she = Susan; her = Margaret.)

Literal/non-literal Meaning

- The literal meaning of a sentence is closely related to the meaning of its individual words.
- The literal meaning does not reflect the context in which words occur.
- Examples of non-literal meaning:
  - *The White House announced today that ...*

    (*White House* = the President's staff, Metonymy)
  - *The price of tomatoes has gone through the roof.*

    (= increased greatly, Metaphor).

Vagueness

- A concept closely related to ambiguity is vagueness.
- Vagueness does not give rise to multiple interpretations.
- An expression is vague with respect to certain semantic features which it leaves unspecified.
- Consider:

  John is tall.

- The meaning of the adjective *tall* is vague.
- The precise degree of tallness is indeterminable.

What else make NLU difficult?

- Non-standard English
- Segmentation issues
- Idioms
- Neologisms
- World knowledge
- Tricky entity names
- …

Outline

- **Natural Language Understanding**
- **Grammars**

Grammars

Grammars

- Context-free Grammars
- Recursive Descent Parser
- Probabilistic Context-free Grammars

Context-free Grammars

- A context-free grammar (CFG) is a set of recursive rewriting rules (or productions) used to generate patterns of strings.
- A CFG consists of the following components:
  - a set of terminal symbols
  - a set of non-terminal symbols
  - a set of productions
  - a start symbol.

Context-free Grammars

- Start symbol: S
- Production rule: {S -> NP VP, …}
- Non-terminal symbols: {S, NP, VP, PP, Det, N, V, P, …}
- Terminal symbols: {"dog", "man", "a", "saw", "in", …}

Syntactic Categories

Symbol	**Meaning**	**Example**
S	sentence	*the dog saw a man*
NP	noun phrase	*the dog*
VP	verb phrase	*saw a man*
PP	prepositional phrase	*in the park*
Det	determiner	*the*
N	noun	*dog*
V	verb	*saw*
P	preposition	*in*

A Context-Free Grammar in Python

```
from nltk import CFG
cfg = CFG.fromstring("""
S -> NP VP
NP -> Det N | Det N PP
VP -> V NP | V NP PP
PP -> P NP
Det -> "the" | "a"
N -> "man" | "dog" | "park"
V -> "saw"
P -> "in"
""")
print(cfg)
```

Output

```
Grammar with 13 productions (start state = S)
    S -> NP VP
    NP -> Det N
    NP -> Det N PP
    VP -> V NP
    VP -> V NP PP
    PP -> P NP
    Det -> 'the'
    Det -> 'a'
    N -> 'man'
    N -> 'dog'
    N -> 'park'
    V -> 'saw'
    P -> 'in'
```

Recursive Descent Parser

- The recursive descent parser builds a parse tree in a top-down manner during this process.
- With the initial goal (find an **S**), the S root node is created.
- As the process recursively expands its goals using the production rules of the grammar, the parse tree is extended downwards.
- During this process, the parser is often forced to choose between alternative productions.
- When a particular production does not work, then the parser has to backtrack.

Recursive Descent Parsing

Initial stage:

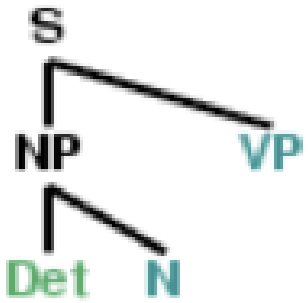
S

```
S -> NP VP
    NP -> Det N
    NP -> Det N PP
    VP -> V NP
    VP -> V NP PP
    PP -> P NP
    Det -> 'the'
    Det -> 'a'
    N -> 'man'
    N -> 'dog'
    N -> 'park'
    V -> 'saw'
    P -> 'in'
```

the dog saw a man in the park

Recursive Descent Parsing

Second production:




```
S -> NP VP
    NP -> Det N
    NP -> Det N PP
    VP -> V NP
    VP -> V NP PP
    PP -> P NP
    Det -> 'the'
    Det -> 'a'
    N -> 'man'
    N -> 'dog'
    N -> 'park'
    V -> 'saw'
    P -> 'in'
```

the dog saw a man in the park

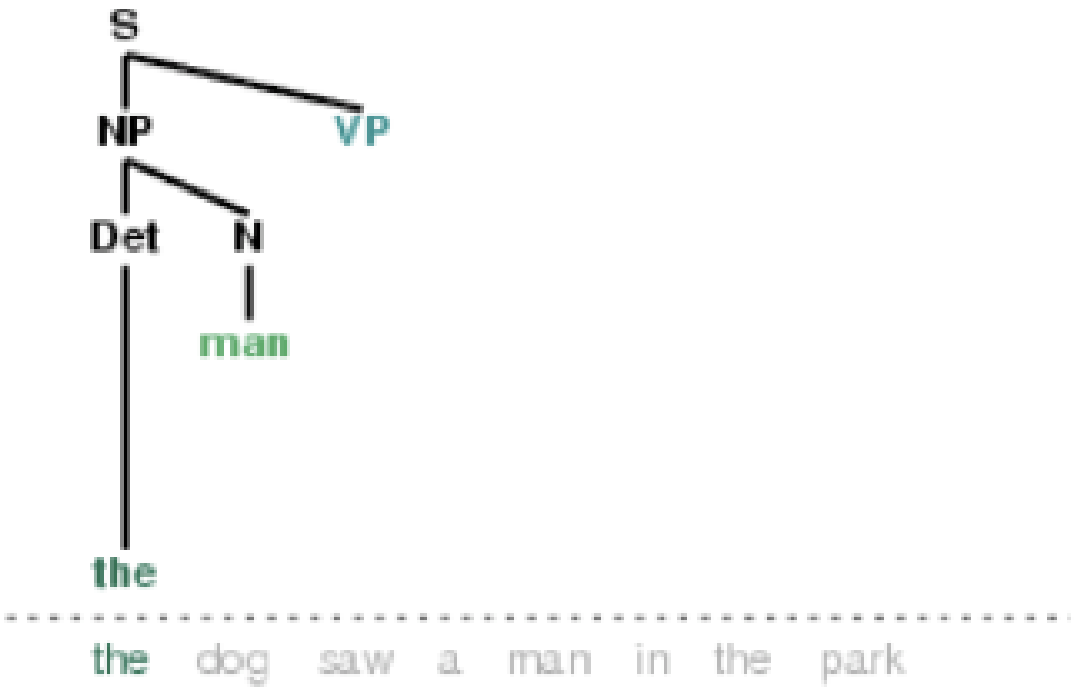
Recursive Descent Parsing

Matching determiner *the*:


```
S -> NP VP
     NP -> Det N
     NP -> Det N PP
     VP -> V NP
     VP -> V NP PP
     PP -> P NP
     Det -> 'the'
     Det -> 'a'
     N -> 'man'
     N -> 'dog'
     N -> 'park'
     V -> 'saw'
     P -> 'in'
```

Recursive Descent Parsing

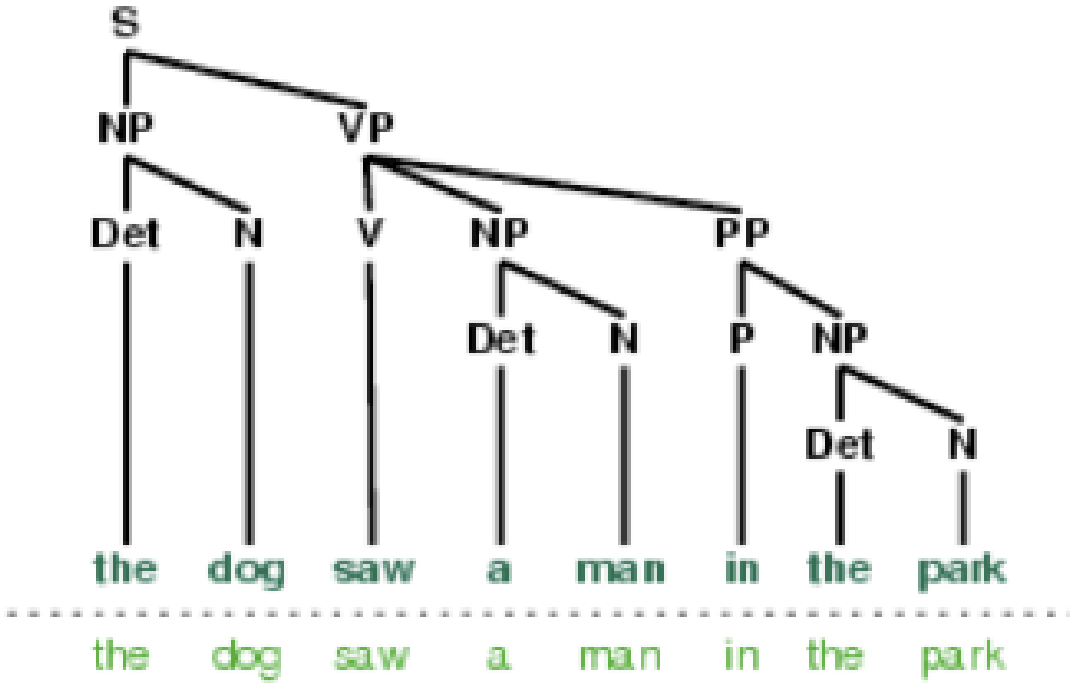
Cannot match *man*:




```
S -> NP VP
     NP -> Det N
     NP -> Det N PP
     VP -> V NP
     VP -> V NP PP
     PP -> P NP
     Det -> 'the'
     Det -> 'a'
     N -> 'man'
     N -> 'dog'
     N -> 'park'
     V -> 'saw'
     P -> 'in'
```

Recursive Descent Parsing

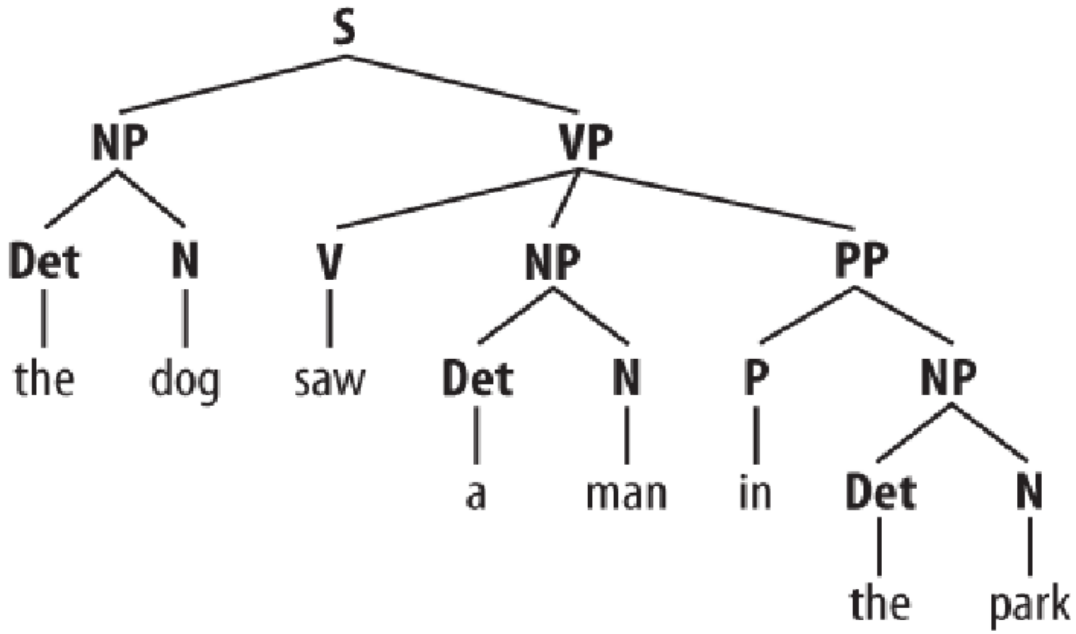
Complete parse:




```
S -> NP VP
     NP -> Det N
     NP -> Det N PP
     VP -> V NP
     VP -> V NP PP
     PP -> P NP
     Det -> 'the'
     Det -> 'a'
     N -> 'man'
     N -> 'dog'
     N -> 'park'
     V -> 'saw'
     P -> 'in'
```

First Syntax Tree

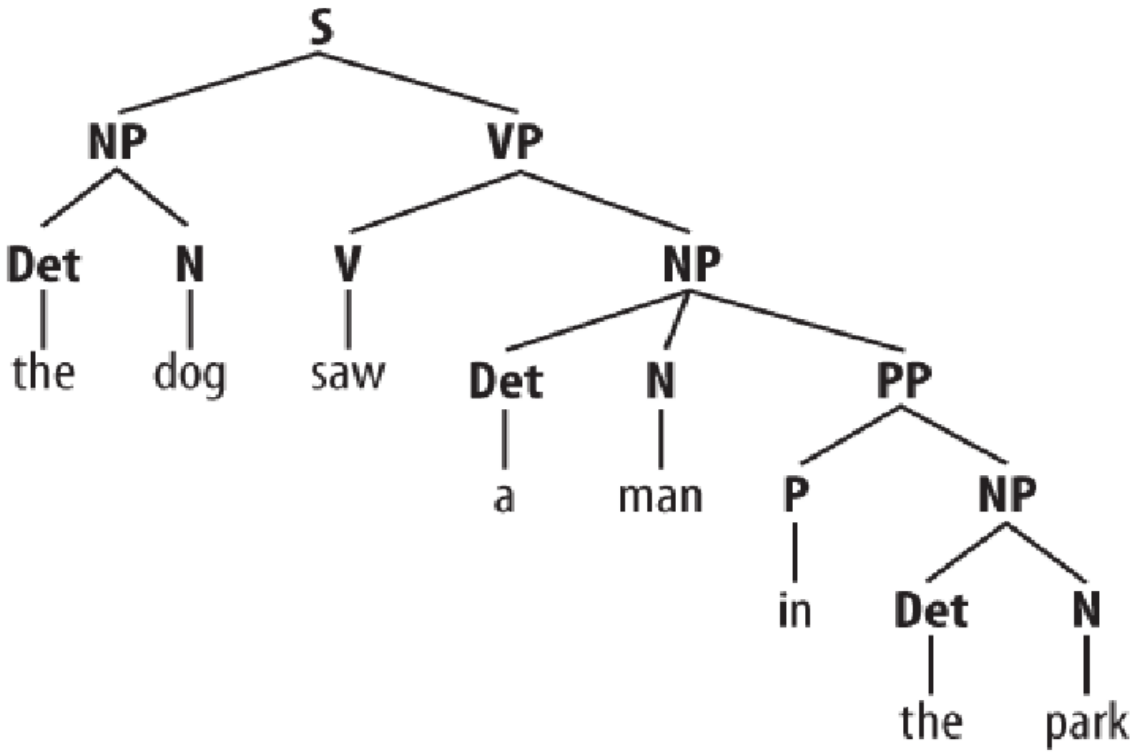
Ambiguity: prepositional phrase attachment




```
S -> NP VP
  NP -> Det N
  NP -> Det N PP
  VP -> V NP
  VP -> V NP PP
  PP -> P NP
  Det -> 'the'
  Det -> 'a'
  N -> 'man'
  N -> 'dog'
  N -> 'park'
  V -> 'saw'
  P -> 'in'
```

Second Syntax Tree

Ambiguity: prepositional phrase attachment




```
S -> NP VP
NP -> Det N
NP -> Det N PP
VP -> V NP
VP -> V NP PP
PP -> P NP
Det -> 'the'
Det -> 'a'
N -> 'man'
N -> 'dog'
N -> 'park'
V -> 'saw'
P -> 'in'
```

A Recursive Descent Parser in Python

```
# Recursive Descent Parser (top-down, depth-first)

from nltk import RecursiveDescentParser

sent = "The dog saw the man in the park".split()

rd_parser = RecursiveDescentParser(cfg)

for tree in rd_parser.parse(sent):
    print(tree)
```

Output

```
(S
  (NP (Det the) (N dog))
  (VP
    (V saw)
    (NP (Det a) (N man) (PP (P in) (NP (Det the) (N park))))))
(S
  (NP (Det the) (N dog))
  (VP
    (V saw)
    (NP (Det a) (N man))
    (PP (P in) (NP (Det the) (N park)))))
```

Probabilistic Context-Free Grammars

- A probabilistic context-free grammar (or *PCFG*) is a context-free grammar that associates a probability with each of its productions.
- It generates the same set of parses for a text that the corresponding context-free grammar does, and assigns a probability to each parse.
- The probability of a parse generated by a PCFG is simply the product of the probabilities of the productions used to generate it.

A PCFG in Python

```
from nltk import PCFG
pcfg = PCFG.fromstring("""
S -> NP VP      [1.0]
NP -> Det N     [0.7]
NP -> Det N PP  [0.3]
VP -> V NP      [0.6]
VP -> V NP PP   [0.4]
PP -> P NP      [1.0]
Det -> "the"    [0.6]
Det -> "a"      [0.4]
N -> "man"      [0.4]
N -> "dog"      [0.3]
N -> "park"     [0.3]
V -> "saw"      [1.0]
P -> "in"       [1.0]
""")
```

Viterbi Parser in Python

```
from nltk import ViterbiParser

sent = "the dog saw a man in the park".split()

viterbi_parser = ViterbiParser(pcfg, trace=2)

for tree in viterbi_parser.parse(sent):
    print(tree)
```

Output

```
(S
  (NP (Det the) (N dog))
  (VP
    (V saw)
    (NP (Det a) (N man))
    (PP (P in) (NP (Det the) (N park)))))
(p=0.000711245)
```

Summary

- NLU
  - Understanding of natural language involves different aspects (syntax, semantics, pragmatics).
  - Ambiguity gives rise to multiple interpretations.
  - Vagueness lets certain semantic features unspecified.
- Grammar
  - A context-free grammar (CFG) is a set of recursive rewriting rules (or productions) used to generate patterns of strings.
  - A recursive descent parser is a top-down parser.
  - Most natural language sentences are syntactically ambiguous and result in more than one syntax tree.
  - A PCFG is a context-free grammar that associates a probability with each of its production rules.