Natural Language Syntax and Parsing

Natural Language Syntax is described often like a formal language, through a context-free grammar:
- the Start-Symbol $S \equiv$ sentence
- Non-Terminals $NT \equiv$ syntactic constituents
- Terminals $T \equiv$ lexical entries/words
- Productions $P \subseteq NT \times (NT \cup T)^+$ \equiv grammar rules

Parsing
- derive the **syntactic structure** of a sentence based on a language model (**grammar**)
- construct a **parse tree**, i.e. the **derivation** of the sentence based on the grammar (**rewrite system**)

Sample Grammar

$$S \subseteq NT, \text{ Part-of-Speech} \subseteq NT, \text{ Constituents} \subseteq NT, \text{ Words} \subseteq T, \text{ Rules:}$$
- $S \rightarrow NP \ VP \quad $ *statement*
- $S \rightarrow \text{Aux} \ NP \ VP \quad $ *question*
- $S \rightarrow VP \quad $ *command*
- $NP \rightarrow \text{Det} \ Nominal$
- $NP \rightarrow \text{Proper-Noun}$
- $Nominal \rightarrow \text{Noun} \mid \text{Noun Nominal} \mid \text{Nominal PP}$
- $VP \rightarrow \text{Verb} \mid \text{Verb} \ NP \mid \text{Verb} \ PP \mid \text{Verb} \ NP \ PP$
- $PP \rightarrow \text{Prep} \ NP$
- $\text{Det} \rightarrow$ that | this | a
- $\text{Noun} \rightarrow$ book | flight | meal | money
- $\text{Proper-Noun} \rightarrow$ Houston | American Airlines | TWA
- $\text{Verb} \rightarrow$ book | include | prefer
- $\text{Aux} \rightarrow$ does
- $\text{Prep} \rightarrow$ from | to | on
Sample Parse Tree

Task: Parse "Does this flight include a meal?"

Problems in Parsing - Ambiguity

Ambiguity
"One morning, I shot an elephant in my pajamas. How he got into my pajamas, I don’t know.”
Groucho Marx
syntactical/structural ambiguity – several parse trees are possible e.g. above sentence
semantic/lexical ambiguity – several word meanings e.g. bank (where you get money) and (river) bank
even different word categories possible (interim) e.g. “He books the flight.” vs. “The books are here.” or “Fruit flies from the balcony” vs. “Fruit flies are on the balcony.”

Problems in Parsing - Attachment

Attachment
in particular PP (prepositional phrase) binding; often referred to as ‘binding problem’
“One morning, I shot an elephant in my pajamas.”
(S ... (NP (PNoun l)) (VP (Verb shot) (NP (Det an (Nominal (Noun elephant))) (PP in my pajamas))...)
rule VP → Verb NP PP
(S ... (NP (PNoun l)) (VP (Verb shot) (NP (Det an) (Nominal (Nominal (Noun elephant)) (PP in my pajamas))... )
rule VP → Verb NP and NP → Det Nominal and Nominal → Nominal PP and Nominal → Noun

Bottom-up and Top-down Parsing

Bottom-up – from word-nodes to sentence-symbol
Top-down Parsing – from sentence-symbol to words
Problems with Bottom-up and Top-down Parsing

Problems with left-recursive rules like NP → NP PP: don’t know how many times recursion is needed.

Pure Bottom-up or Top-down Parsing is inefficient because it generates and explores too many structures which in the end turn out to be invalid (several grammar rules applicable → ‘interim’ ambiguity).

Combine top-down and bottom-up approach:
- Start with sentence; use rules top-down (look-ahead); read input; try to find shortest path from input to highest unparsed constituent (from left to right).
- → Chart-Parsing / Earley-Parser

Chart Parsing / Early Algorithm

**Earley-Parser** based on Chart-Parsing

**Essence:** Integrate top-down and bottom-up parsing. Keep recognized sub-structures (sub-trees) for shared use during parsing.

**Top-down:** Start with S-symbol. Generate all applicable rules for S. Go further down with left-most constituent in rules and add rules for these constituents until you encounter a left-most node on the RHS which is a word category (POS).

**Bottom-up:** Read input word and compare. If word matches, mark as recognized and move parsing on to the next category in the rule(s).

---

**Chart**

Sequence of *n* input words; *n+1* nodes marked 0 to *n*.

Arrows indicate recognized part of RHS of rule. The • indicates recognized constituents in rules.

![Directed acyclic graph representation of the three dotted lines](Jurafsky_Martin_Figure_10.15_p.380)

---

**Chart Parsing / Earley Parser 1**

**Chart**

Sequence of input words; *n+1* nodes marked 0 to *n*.

States in chart represent possible rules and recognized constituents, with arcs.

**Interim state**

S → • VP, [0,0]

- top-down look at rule S → VP
- nothing of RHS of rule yet recognized (• is far left)
- arc at beginning, no coverage (covers no input word; beginning of arc at 0 and end of arc at 0)
Chart Parsing / Earley Parser 2

Interim states

NP → Det • Nominal, [1,2]
- top-down look with rule NP → Det • Nominal
- Det recognized (• after Det)
- arc covers one input word which is between nodes 1 and 2
- look next for Nominal

NP → Det Nominal • , [1,3]
- Nominal was recognized, move • after Nominal
- move end of arc to cover Nominal (change 2 to 3)
- structure is completely recognized; arc is inactive; mark NP as recognized in other rules (move • ).
Chart - 3a

S → VP
VP → V NP
NP → Det Nom
Nom → Noun

VP
V
Book
Det
this
Noun
flight

Chart - 3b

S → VP
VP → V NP
NP → Det Nom
Nom → Noun

VP
V
Book
det
this
Noun
flight

Chart - 3c

S → VP
VP → V NP
NP → Det Nom
Nom → Noun

VP
V
Book
Det
this
Noun
flight

Chart - 3d

S → VP
VP → V NP
NP → Det Nom
Nom → Noun

VP
V
Book
Det
this
Noun
flight
Chart - All States

$S \rightarrow VP$

$S \rightarrow VP$

$VP \rightarrow V NP$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$Nom \rightarrow Noun$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$VP \rightarrow V NP$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$

Chart - Final States

$S \rightarrow VP$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$

Chart 0 with two S- and two VP-Rules

$S \rightarrow VP$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$

$S \rightarrow VP NP$

Chart 1a with two S- and two VP-Rules

$S \rightarrow VP$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$S \rightarrow VP NP$

$VP \rightarrow V NP$

$NP \rightarrow Det Nom$

$Nom \rightarrow Noun$

$V \rightarrow Book$

$Det \rightarrow this$

$Noun \rightarrow flight$
Chart 1b with two S- and two VP-Rules

S → VP .
VP → V . NP
VP → V .
S → VP . NP

Chart 2 with two S- and two VP-Rules

S → VP .
VP → V . NP
S → VP . NP
VP → V . NP
NP → Det . Nom
Nom → . Noun

Chart 3 with two S- and two VP-Rules

S → VP .
VP → V NP .
VP → V .
S → VP NP .
NP → Det Nom .
Nom → Noun .

Final Chart - with two S-and two VP-Rules

S → VP .
VP → V . NP
S → VP NP .
NP → Det Nom .
Nom → Noun .
Nom → . Noun
Earley Algorithm

Earley Algorithm - Functions

**predictor**
generates new rules for partly recognized RHS with constituent right of • (top-down generation)

**scanner**
if word category (POS) is found right of the •, the Scanner reads the next input word and adds a rule for it to the chart (bottom-up mode)

**completer**
if rule is completely recognized (the • is far right), the recognition state of earlier rules in the chart advances: the • is moved over the recognized constituent (bottom-up recognition).

---

Earley-Algorithm 1

```
function EARLEY-PARSE (words, grammar) returns chart
  ENQUEUE((γ → • S, [0,0]), chart [0])
  for i from 0 to LENGTH (words) do
    for each state in chart [i] do
      if INCOMPLETE?(state) and NEXT-CAT(state) is not a part of speech then PREDICTOR(state)
      elseif INCOMPLETE?(state) and NEXT-CAT(state) is a part of speech then SCANNER(state)
      else COMPLETER(state)
    end
  end
  return(chart)
```

---

Earley-Algorithm 2

```
function PREDICTOR((A → α • B β, [i, j]))
  for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar)
    do ENQUEUE((B → γ [j, j], chart [j])
end

procedure SCANNER ((A → α • B β, [i, j]))
  if B ∈ PARTS-OF-SPEECH(word [j])
    then ENQUEUE((B → word [j], [j, j+1]), chart [j+1])
end

procedure COMPLETER ((B → γ •, [j, k]))
  for each (A → α • B β, [i, j]) in chart [j]
    do ENQUEUE((A → αB • β, [i,k]), chart [k])
end

procedure ENQUEUE(state, chart-entry)
  if state is not already in chart-entry
    then PUSH(state, chart-entry)
end
```
Earley Algorithm
- Figures -

Jurafsky & Martin, Figures 10.16, 10.17, 10.18

function EARLEY-PARSE(words, grammar) returns chart
  ENQUEUE(\$ \rightarrow \$) [0, 0]
  for i = 0 to LENGTH(words) do
    for each state in chart[i] do
      if state is not a part of speech then
        PREDICT(state)
      end
      NEXT-CAT(state) is a part of speech then
        SCANNER(state)
      end
    end
  end
  return(chart)

procedure PREDICT(state \rightarrow \$) [0, 0]
  for each (B \rightarrow \$) in GRAMMAR_RULES FOR B grammar do
    ENQUEUE(B \rightarrow \$) [0, 0], chart[i]
  end
end

procedure SCANNER(state \rightarrow \$) [0, 0]
  if B is PREDICTED(state) then
    ENQUEUE(B \rightarrow \$) [0, 0], chart[i+1]
  end
end

procedure PREDICT(state \rightarrow \$) [0, 0]
  for each (B \rightarrow \$) in GRAMMAR_RULES FOR B grammar do
    ENQUEUE(B \rightarrow \$) [0, 0], chart[i]
  end
end

procedure ENQUEUE(state chart-entry)
  if state is not already in chart-entry then
    PUSH(state, chart-entry)
  end
end
Additional References


**Earley Algorithm**
Jurafsky & Martin, Figure 10.16, p.384

**Earley Algorithm - Examples**
Jurafsky & Martin, Figures 10.17 and 10.18