Parsing: Earley & PCFGs

Ling 571
Deep Processing Techniques for NLP
January 19, 2011

Roadmap

- Motivation I: CKY limitations
- The Earley algorithm

- Motivation II: Ambiguity
- Probabilistic Context-free Grammars (PCFGs)

CKY Algorithm

- Dynamic programming approach
  - Yields parsing in cubic time in length of input string
  - Fairly efficient

- Issues:
- \

CKY Algorithm

- Dynamic programming approach
  - Yields parsing in cubic time in length of input string
  - Fairly efficient

- Issues:
  - Requires CNF conversion
    - Limits expressiveness

CKY Algorithm

- Dynamic programming approach
  - Yields parsing in cubic time in length of input string
  - Fairly efficient

- Issues:
  - Requires CNF conversion
    - Limits expressiveness
  - Maintains ambiguity

Earley Parsing

- Avoid repeated work/recursion problem
  - Dynamic programming
    - Store partial parses in “chart”
      - Compactly encodes ambiguity
    - $O(N^3)$
- Chart entries:
  - Subtree for a single grammar rule
  - Progress in completing subtree
  - Position of subtree wrt input

Earley Algorithm

- Uses dynamic programming to do parallel top-down search in (worst case) $O(N^3)$ time
- First, left-to-right pass fills out a chart with N +1 states
  - Think of chart entries as sitting between words in the input string keeping track of states of the parse at these positions
  - For each word position, chart contains set of states representing all partial parse trees generated to date. E.g. chart[0] contains all partial parse trees generated at the beginning of the sentence

Chart Entries

Represent three types of constituents:

- predicted constituents
- in-progress constituents
- completed constituents

Parse Progress

- Represented by Dotted Rules
- Position of • indicates type of constituent
- $_0$ Book $_1$ that $_2$ flight $_3$
  - S → • VP, [0,0] (predicted)
  - NP → Det • Nom, [1,2] (in progress)
  - VP →V NP •, [0,3] (completed)
- [x,y] tells us what portion of the input is spanned so far by this rule
- **Each State s_i:**
  <dotted rule>, [<back pointer>,<current position>]

$_0$ Book $_1$ that $_2$ flight $_3$

S → • VP, [0,0]

- First 0 means S constituent begins at the start of input
- Second 0 means the dot here too
- So, this is a top-down prediction

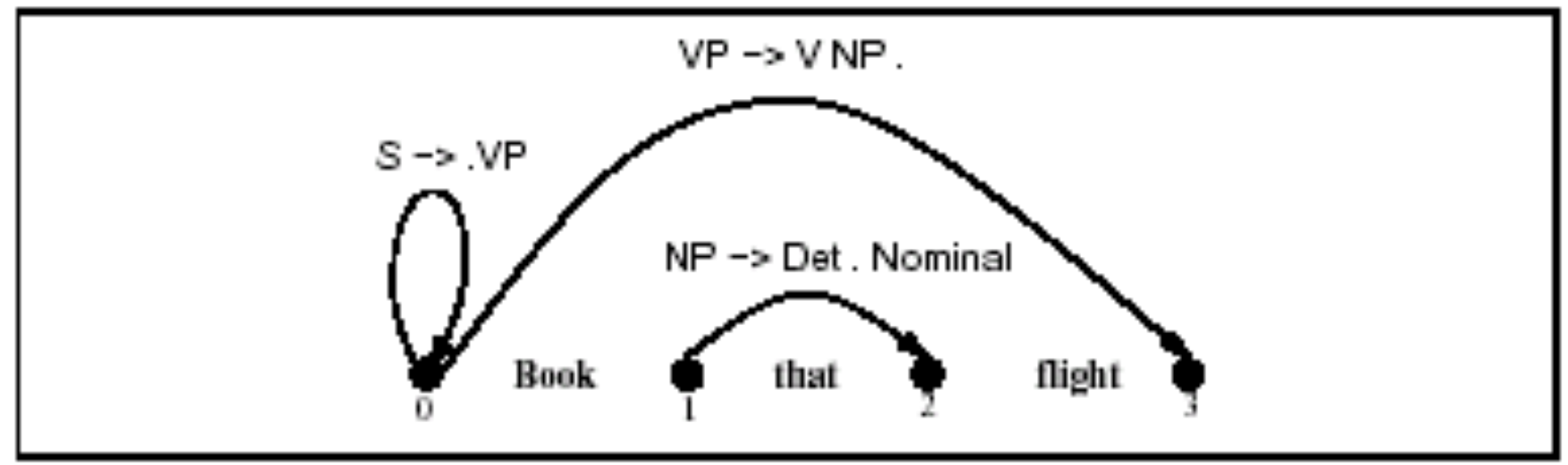
NP → Det • Nom, [1,2]

- the NP begins at position 1
- the dot is at position 2
- so, Det has been successfully parsed
- Nom predicted next

$_0$ Book $_1$ that $_2$ flight $_3$ (continued)

VP → V NP •, [0,3]

- Successful VP parse of entire input



Successful Parse

- Final answer found by looking at last entry in chart
- If entry resembles $S \rightarrow \alpha \bullet [0,N]$ then input parsed successfully
- Chart will also contain record of all possible parses of input string, given the grammar

Parsing Procedure for the Earley Algorithm

- Move through each set of states in order, applying one of three operators to each state:
  - **predictor:** add predictions to the chart
  - **scanner:** read input and add corresponding state to chart
  - **completer:** move dot to right when new constituent found
- Results (new states) added to current or next set of states in chart
- No backtracking and no states removed: keep complete history of parse

States and State Sets

- **Dotted Rule s_i** represented as
  <dotted rule>, [<back pointer>, <current position>]

- **State Set S_j** to be a collection of states s_i with the same <current position>.

Earley Algorithm from Book

```
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 from Book

procedure PREDICTOR($(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$)
  for each $(B \rightarrow \gamma)$ **in** GRAMMAR-RULES-FOR($B, grammar$) **do**
    ENQUEUE($(B \rightarrow \bullet\ \gamma,\ [j, j]), chart[j]$)
  end

procedure SCANNER($(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$)
  if B $\subset$ PARTS-OF-SPEECH($word[j]$) **then**
    ENQUEUE($(B \rightarrow word[j],\ [j, j+1]), chart[j+1]$)

procedure COMPLETER($(B \rightarrow \gamma \bullet,\ [j,k])$)
  for each $(A \rightarrow \alpha \bullet B\ \beta,\ [i, j])$ **in** $chart[j]$ **do**
    ENQUEUE($(A \rightarrow \alpha\ B \bullet \beta,\ [i,k]), chart[k]$)
  end

Earley Algorithm (simpler!)

1. Add Start → · S, [0,0] to state set 0
   Let i=1

2. **Predict** all states you can, adding new predictions to state set 0

3. **Scan** input word i—add all matched states to state set S_i.
   Add all new states produced by **Complete** to state set S_i
   Add all new states produced by **Predict** to state set S_i
   Let i = i + 1
   Unless i=*n*, repeat step 3.

4. At the end, see if state set *n* contains Start → S ·, [0,n]

3 Main Sub-Routines of Earley Algorithm

- **Predictor**: Adds predictions into the chart.
- **Completer**: Moves the dot to the right when new constituents are found.
- **Scanner**: Reads the input words and enters states representing those words into the chart.

Predictor

- Intuition: create new state for top-down prediction of new phrase.
- Applied when non part-of-speech non-terminals are to the right of a dot: **S → • VP [0,0]**
- Adds new states to *current* chart
  - One new state for each expansion of the non-terminal in the grammar
    VP → • V [0,0]
    VP → • V NP [0,0]
- Formally:
  S_j: A → α · B β, [i,j]
  S_j: B → · γ, [j,j]

Scanner

- Intuition: Create new states for rules matching part of speech of next word.
- Applicable when part of speech is to the right of a dot: VP → • V NP [0,0] 'Book...'
- Looks at current word in input
- If match, adds state(s) to *next* chart
  VP → V • NP [0,1]
- Formally:
  S_j: A → α · B β, [i,j]
  S_{j+1}: A → α B · β, [i,j+1]

Completer

- Intuition: parser has finished a new phrase, so must find and advance states all that were waiting for this
- Applied when dot has reached right end of rule
  NP → Det Nom • [1,3]
- Find all states w/dot at 1 and expecting an NP:
  VP → V • NP [0,1]
- Adds new (completed) state(s) to *current* chart :
  VP → V NP • [0,3]
- Formally: S_k: B → δ ·, [j,k]
  S_k: A → α B · β , [i,k],
  where: S_j: A → α · B β , [i,j].

Chart[0]

S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor

Note that given a grammar, these entries are the same for all inputs; they can be pre-loaded.

Chart[1]

S12	*Verb* → *book* •	[0,1]	Scanner
S13	*VP* → *Verb* •	[0,1]	Completer
S14	*VP* → *Verb* • *NP*	[0,1]	Completer
S15	*VP* → *Verb* • *NP PP*	[0,1]	Completer
S16	*VP* → *Verb* • *PP*	[0,1]	Completer
S17	*S* → *VP* •	[0,1]	Completer
S18	*VP* → *VP* • *PP*	[0,1]	Completer
S19	*NP* → • *Pronoun*	[1,1]	Predictor
S20	*NP* → • *Proper-Noun*	[1,1]	Predictor
S21	*NP* → • *Det Nominal*	[1,1]	Predictor
S22	*PP* → • *Prep NP*	[1,1]	Predictor

Charts[2] and [3]

S23	$Det \rightarrow that \bullet$	[1,2]	Scanner
S24	$NP \rightarrow Det \bullet Nominal$	[1,2]	Completer
S25	$Nominal \rightarrow \bullet Noun$	[2,2]	Predictor
S26	$Nominal \rightarrow \bullet Nominal\ Noun$	[2,2]	Predictor
S27	$Nominal \rightarrow \bullet Nominal\ PP$	[2,2]	Predictor
S28	$Noun \rightarrow flight \bullet$	[2,3]	Scanner
S29	$Nominal \rightarrow Noun \bullet$	[2,3]	Completer
S30	$NP \rightarrow Det\ Nominal \bullet$	[1,3]	Completer
S31	$Nominal \rightarrow Nominal \bullet Noun$	[2,3]	Completer
S32	$Nominal \rightarrow Nominal \bullet PP$	[2,3]	Completer
S33	$VP \rightarrow Verb\ NP \bullet$	[0,3]	Completer
S34	$VP \rightarrow Verb\ NP \bullet PP$	[0,3]	Completer
S35	$PP \rightarrow \bullet Prep\ NP$	[3,3]	Predictor
S36	$S \rightarrow VP \bullet$	[0,3]	Completer
S37	$VP \rightarrow VP \bullet PP$	[0,3]	Completer

How do we retrieve the parses at the end?

- Augment the Completer to add pointers to prior states it advances as a field in the current state
  - i.e. what state did we advance here?
  - Read the pointers back from the final state

- What about ambiguity?

- What about ambiguity?
- CKY/Earley can represent it

- What about ambiguity?
- CKY/Earley can represent it
- Can’t resolve it

Probabilistic Parsing

- Provides strategy for solving disambiguation problem
  - Compute the probability of all analyses
  - Select the most probable

- Employed in language modeling for speech recognition
  - N-gram grammars predict words, constrain search
  - Also, constrain generation, translation

PCFGs

- Probabilistic Context-free Grammars
  - Augmentation of CFGs

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta\ [p]$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$, and p is a number between 0 and 1 expressing $P(\beta \mid A)$
S	a designated **start symbol**

PCFGs

- Augments each production with probability that LHS will be expanded as RHS
  - P(A->B) or P(A->B|A), p(RHS|LHS)
  - Sum over all possible expansions is 1

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

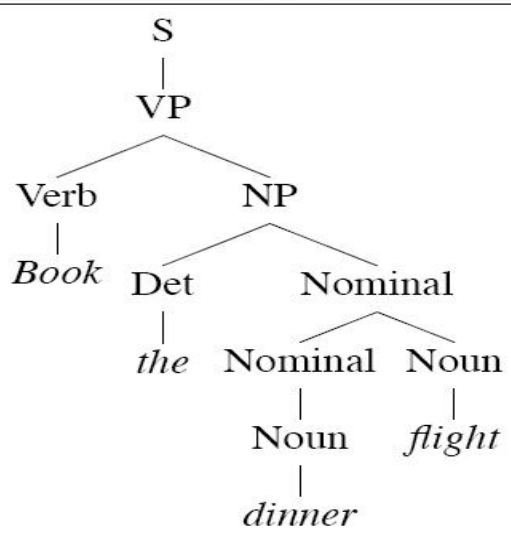
- A PCFG is consistent if sum of probabilities of all sentences in language is 1.
  - Recursive rules often yield inconsistent grammars

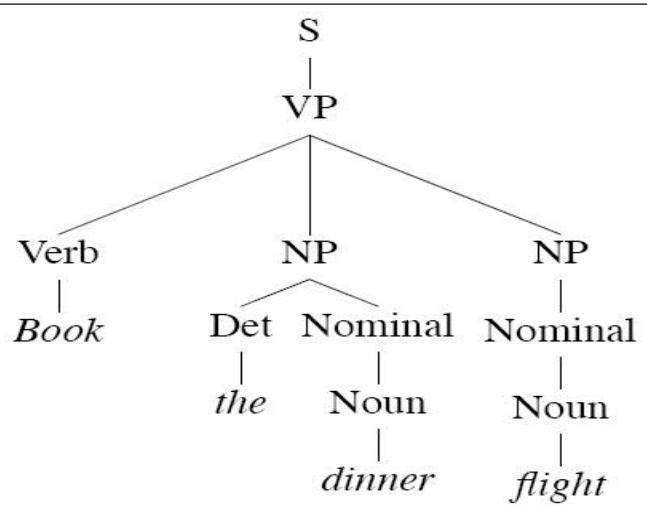
Disambiguation

- A PCFG assigns probability to each parse tree T for input S.
  - Probability of T: product of all rules to derive T

$$P(T,S) = \prod_{i=1}^{n} P(RHS_i \mid LHS_i)$$

$$P(T,S) = P(T)P(S \mid T) = P(T)$$

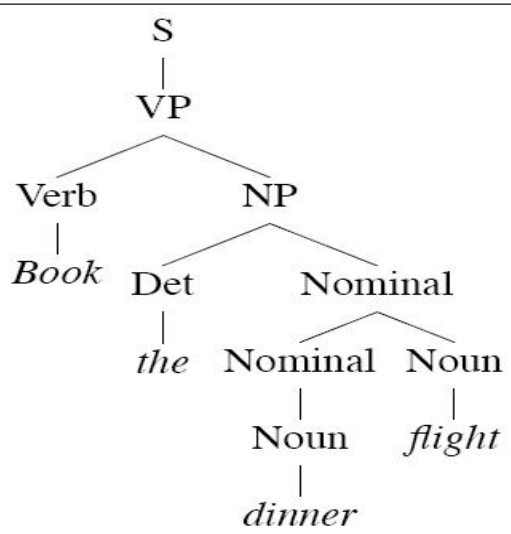


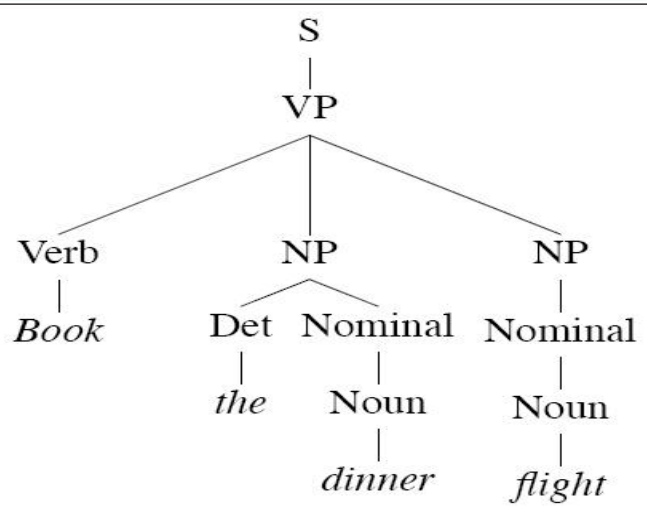


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05

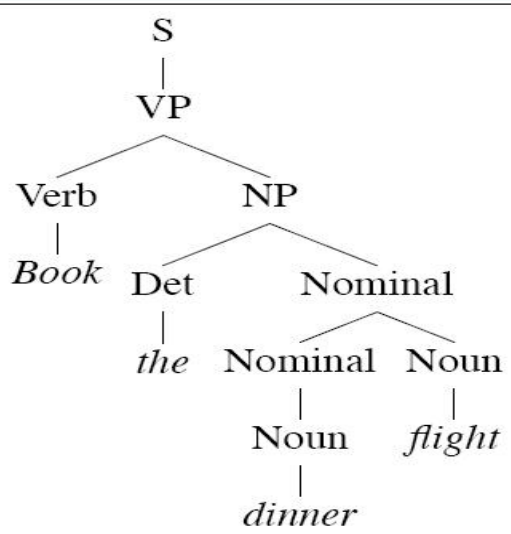


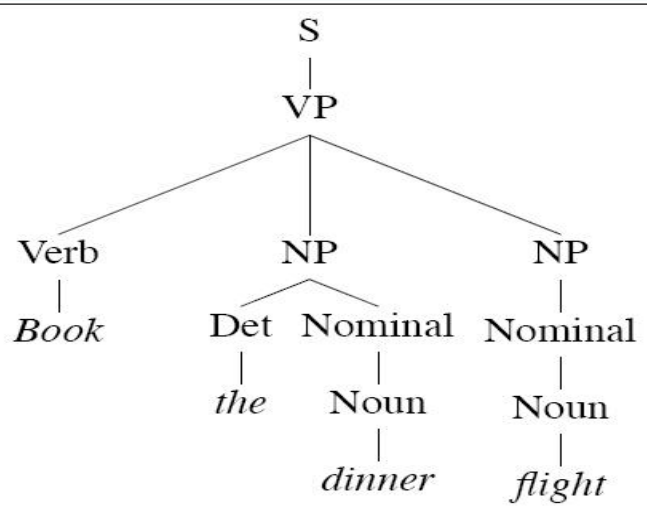


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2

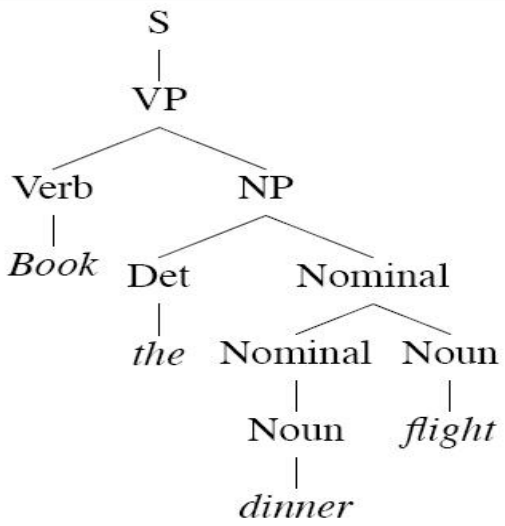


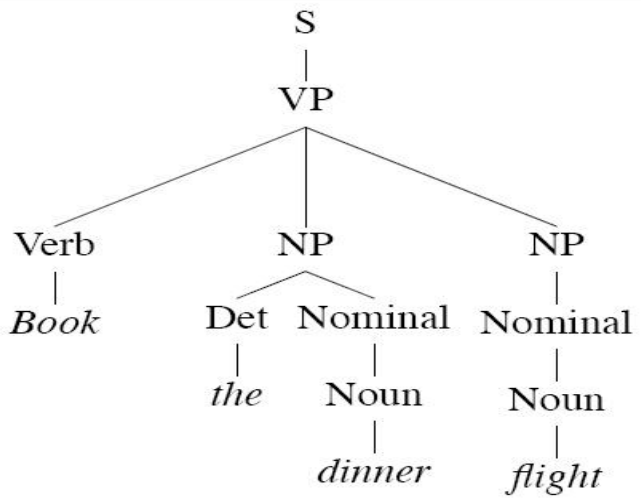


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2

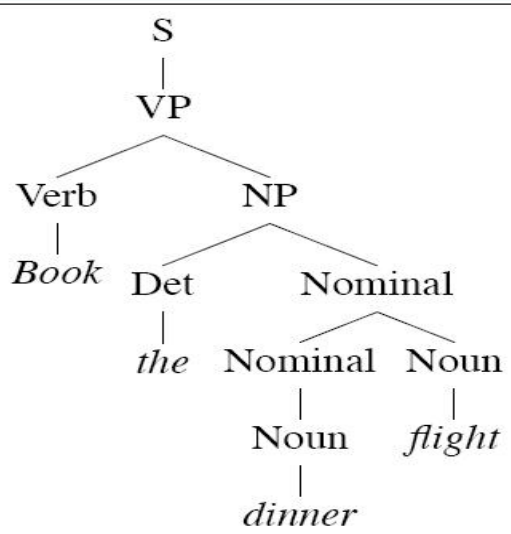


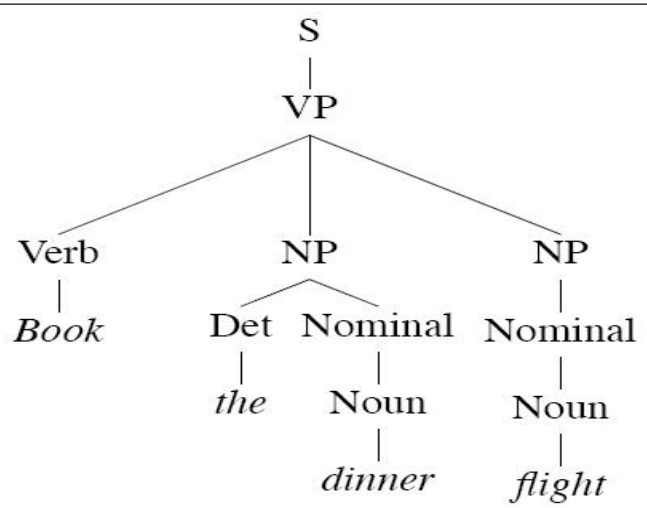


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2

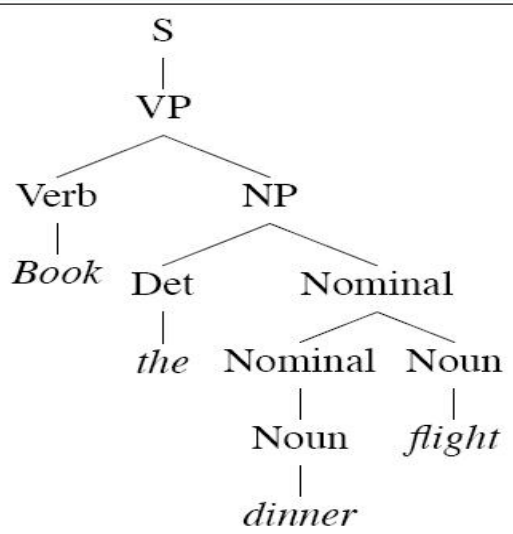


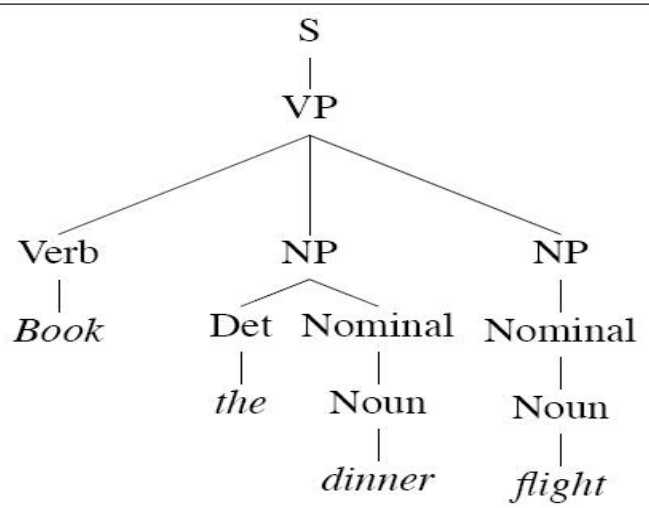


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75

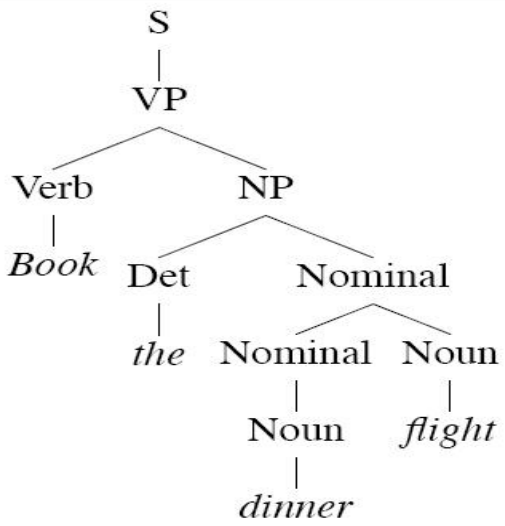


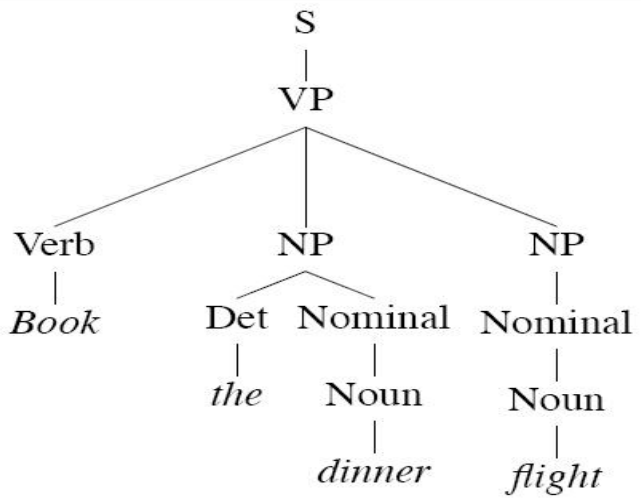


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3

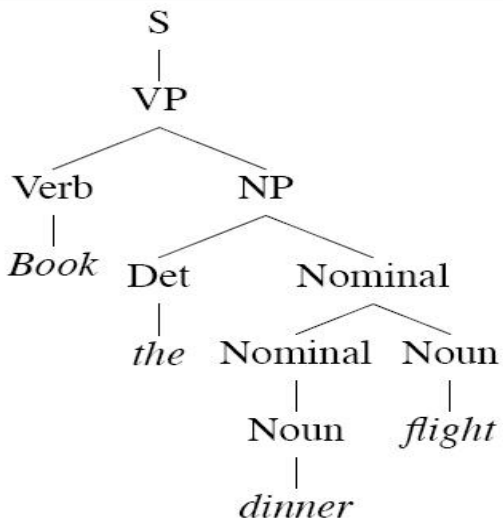


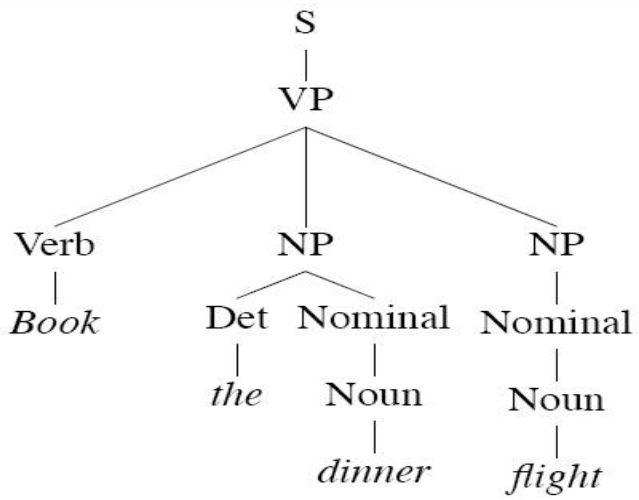


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6

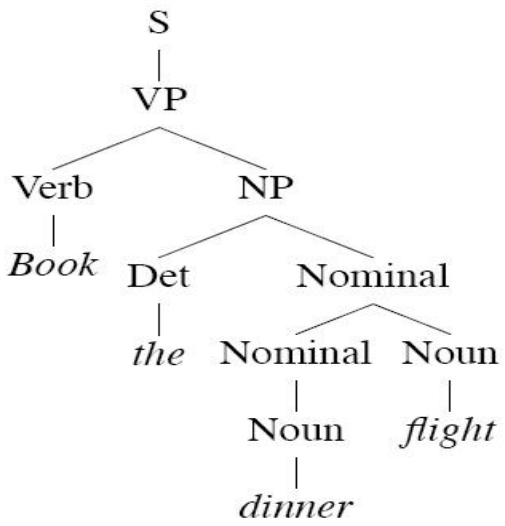


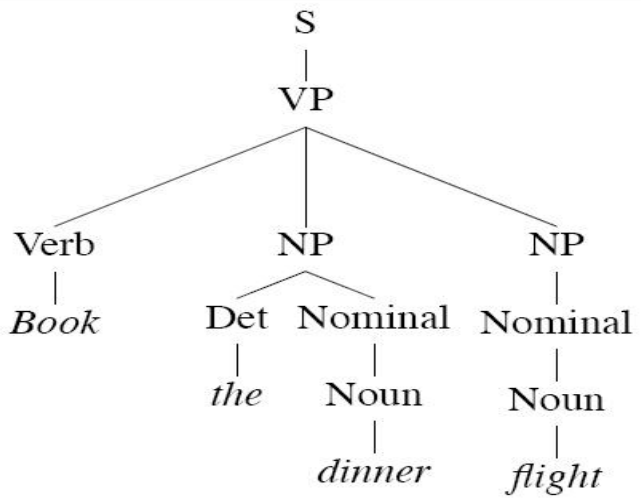


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1

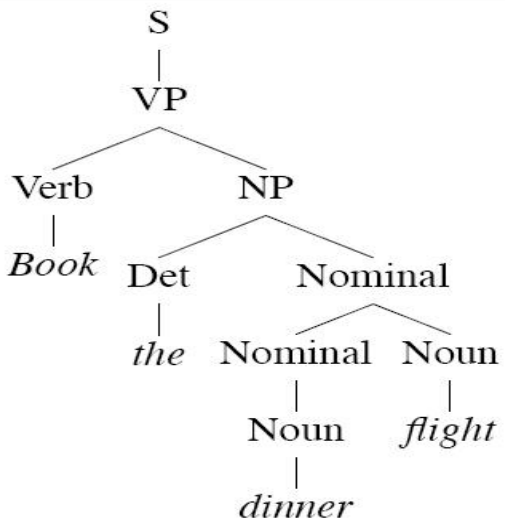


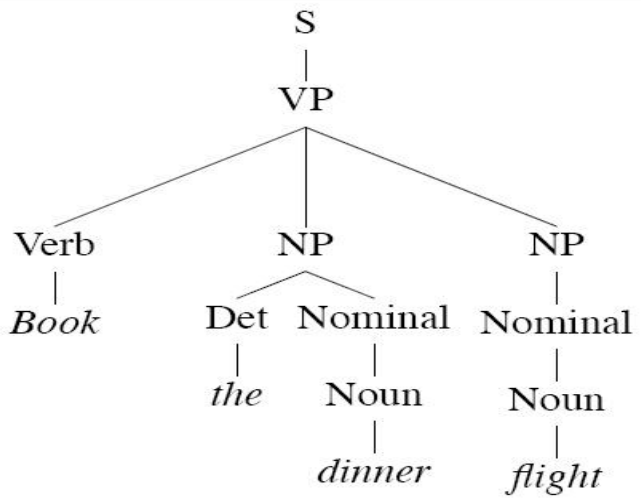


Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6



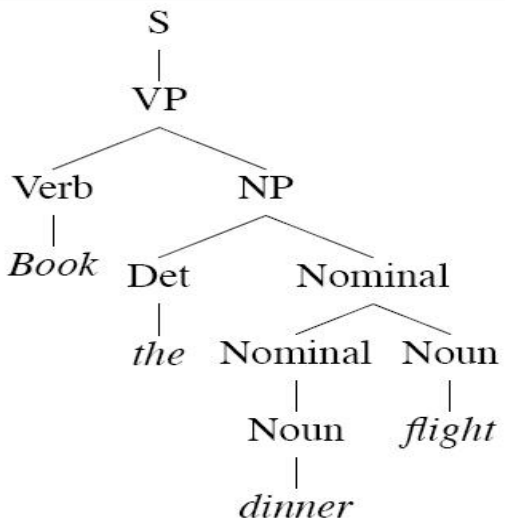


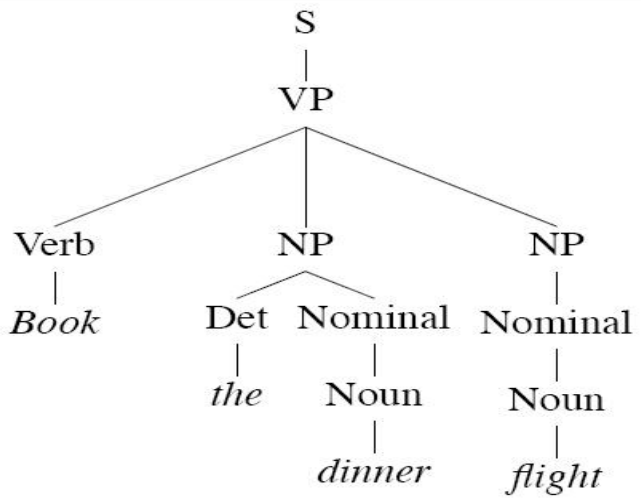
Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05



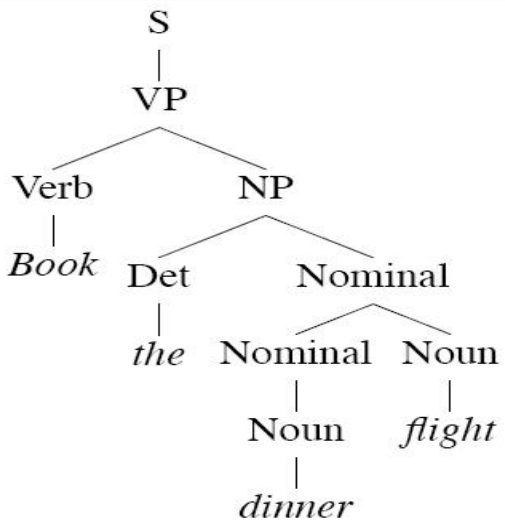


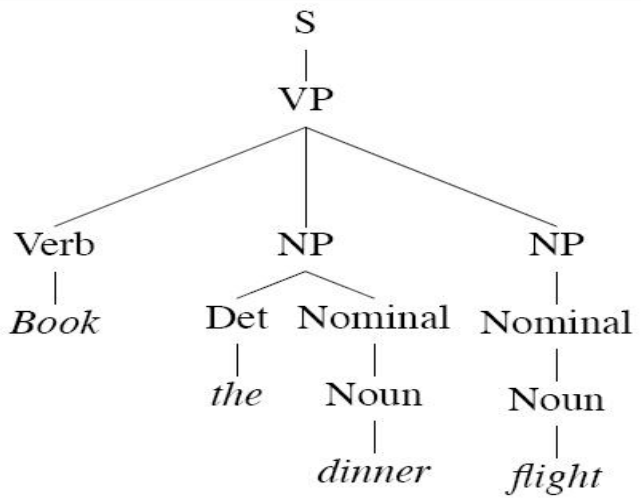
Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1



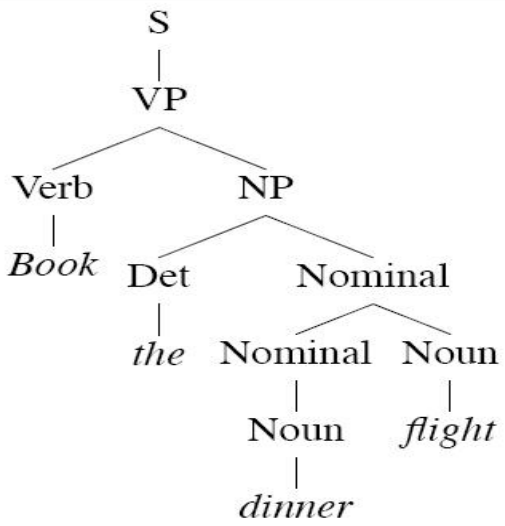


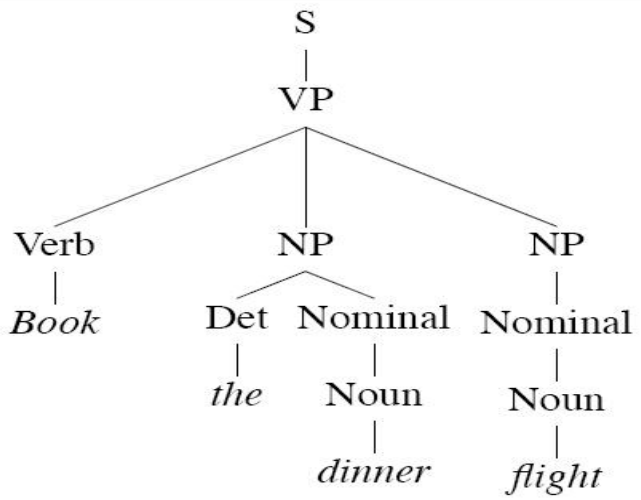
Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1*0.15



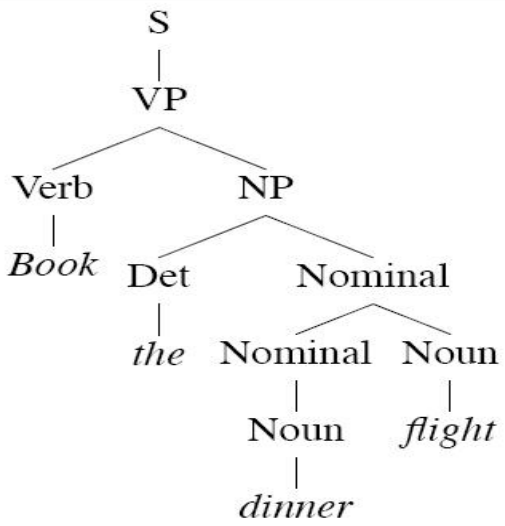


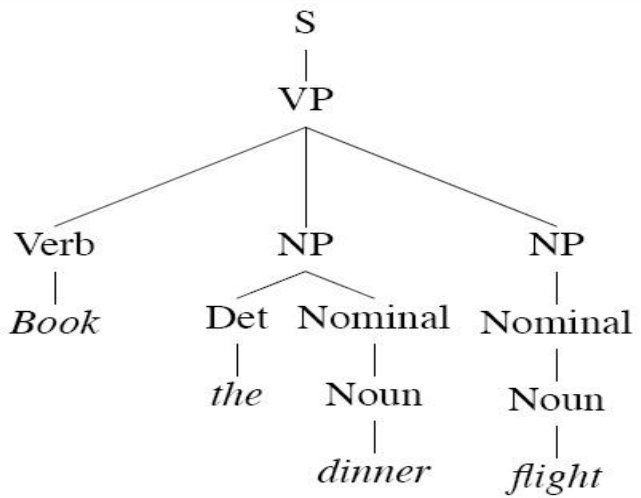
Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1*0.15*0.75



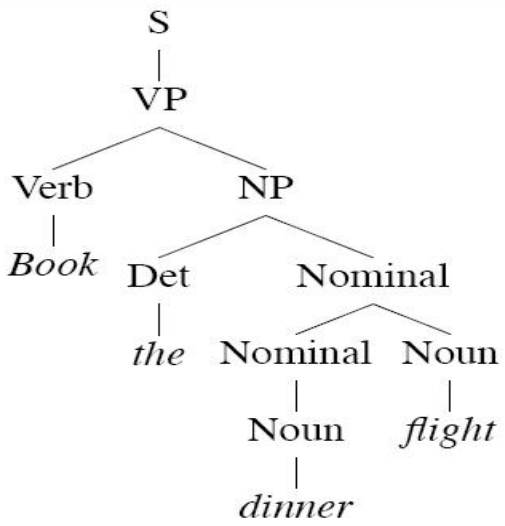


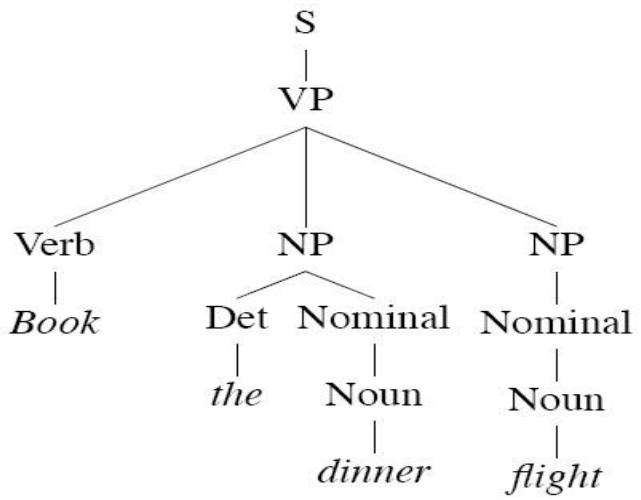
Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1*0.15*0.75*0.75*





Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flights	.40

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1*0.15*0.75*0.75*
0.3*0.6*0.1*0.4=6.1x10^-7

Formalizing Disambiguation

- Select T such that:

$$\hat{T}(S) = \underset{Ts.t, S=yield(T)}{\operatorname{argmax}} P(T)$$

  - String of words S is *yield* of parse tree over S
  - Select tree that maximizes probability of parse

Learning Probabilities

- Simplest way:
  - Treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times non-terminal is expanded
    - Number of times non-terminal is expanded by given rule

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

  - Alternative: Learn probabilities by re-estimating
    - (Later)

Example PCFG

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\ [.10] \mid a\ [.30] \mid the\ [.60]$
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book\ [.10] \mid flight\ [.30]$
$S \rightarrow VP$	[.05]	$\mid meal\ [.15] \mid money\ [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flights\ [.40] \mid dinner\ [.10]$
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book\ [.30] \mid include\ [.30]$
$NP \rightarrow Det\ Nominal$	[.20]	$\mid prefer;[.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I\ [.40] \mid she\ [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me\ [.15] \mid you\ [.40]$
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston\ [.60]$
$Nominal \rightarrow Nominal\ PP$	[.05]	$\mid NWA\ [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does\ [.60] \mid can\ [40]$
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from\ [.30] \mid to\ [.30]$
$VP \rightarrow Verb\ NP\ PP$	[.10]	$\mid on\ [.20] \mid near\ [.15]$
$VP \rightarrow Verb\ PP$	[.15]	$\mid through\ [.05]$
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

Issues with PCFGs

- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

Issues with PCFGs

- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

	Pronoun	Non-pronoun
Subject	91%	9%
Object		

Issues with PCFGs

- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

	Pronoun	Non-pronoun
Subject	91%	9%
Object	34%	66%

Issues with PCFGs

- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

	Pronoun	Non-pronoun
Subject	91%	9%
Object	34%	66%

    - In Treebank: roughly equi-probable
  - How can we handle this?

Issues with PCFGs

- Independence assumptions:
  - Rule expansion is context-independent
    - Allows us to multiply probabilities
  - Is this valid?

	Pronoun	Non-pronoun
Subject	91%	9%
Object	34%	66%

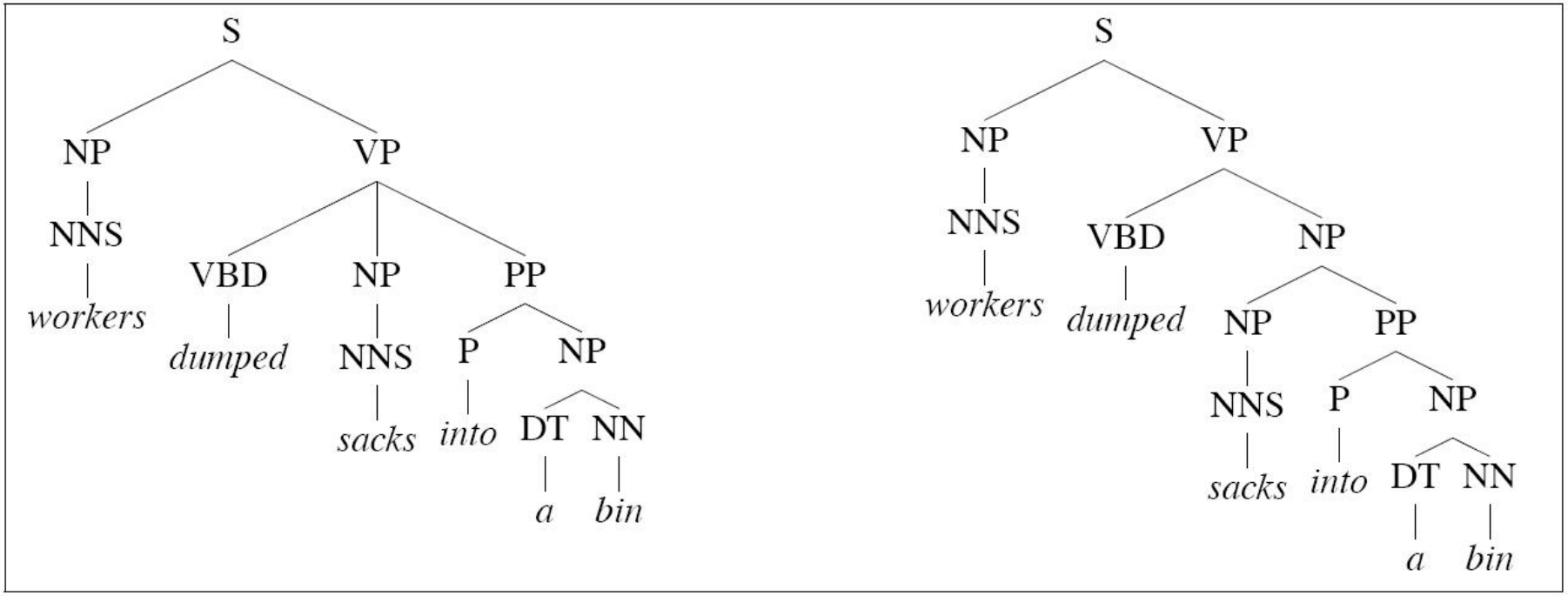
    - In Treebank: roughly equi-probable
  - How can we handle this?
    - Condition on Subj/Obj with parent annotation

Issues with PCFGs

- Insufficient lexical conditioning
  - Present in pre-terminal rules
- Are there cases where other rules should be conditioned on words?

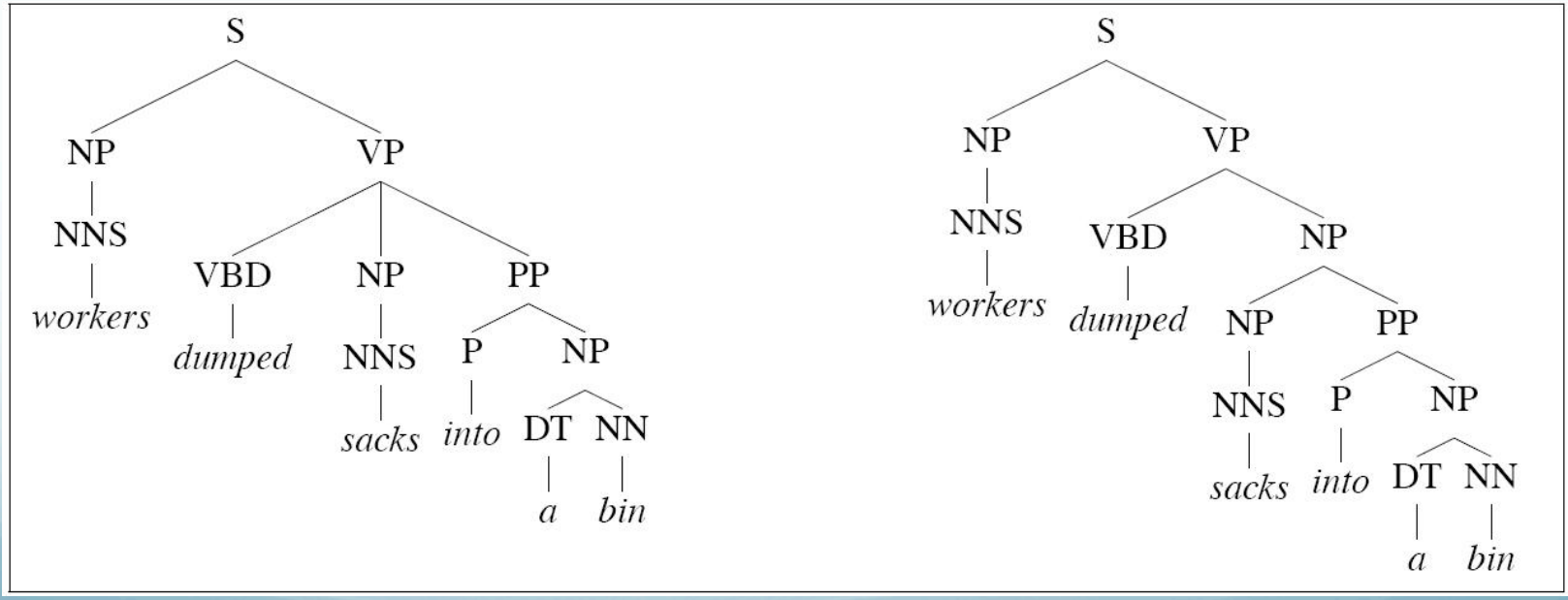
Issues with PCFGs

- Insufficient lexical conditioning
  - Present in pre-terminal rules
- Are there cases where other rules should be conditioned on words?



Issues with PCFGs

- Insufficient lexical conditioning
  - Present in pre-terminal rules
- Are there cases where other rules should be conditioned on words?



Different verbs & prepositions have different attachment preferences

PCFGs

- Augment parsers to handle probabilities
- Adapt PCFGs to handle
  - Structural dependencies
    - By splitting nodes
  - Lexical dependencies
    - By lexicalizing PCFGs