Probabilistic Context-Free Grammars

Informatics 2A: Lecture 22

Shay Cohen

7 November, 2018

Last class

- The Earley algorithm (three important operators that are run until we find a parse: PREDICTOR, SCANNER and COMPLETER)
- The complexity of CYK and the Earley algorithm: $O(n^3)$. (But, grammar constants can be large!)
- The example that I went through is available online. Go step by step through it to understand it

Last class

Questions to think about to understand Earley's algorithm better:

- How does Earley's algorithm solve the problem of left corner parsing?
- How is Earley's algorithm different than Recursive Descent Parsing?
- Why is the complexity of Earley's algorithm $O(n^3)$?
- What happens if we run Earley's algorithm on a grammar in Chmosky normal form, and how does that relate to CYK?

Motivation

Probabilistic Context-Free Grammars
- Definition
- Conditional Probabilities
- Applications
- Probabilistic CYK

Why Probability?

Probability is one of the most important tools in reasoning under uncertainty

We build a *probabilistic model* of a phenomenon in the world

The model allows more than one outcome for a given scenario

That way, we can reason in various contexts depending on the situation. In our case, probability mostly helps managing ambiguity

Often, we will be interested in the most likely outcome

We already saw a case of that with part-of-speech tagging

Motivation

Three things motivate the use of probabilities in grammars and parsing:

1. Syntactic disambiguation. Ambiguity is unavoidable in natural language (grammars).
2. Coverage. What if there is a production that we haven't seen before? Might want to allow with small probability (Seemingly extreme but actually common use case: all productions are possible!).
3. Representativeness. Suppose we want to adapt a parser to a new domain, where some words have different usage, hence different part-of-speech.

Motivation 1: Ambiguity

- Amount of ambiguity increases with sentence length.
- Real sentences are fairly long (avg. sentence length in the *Wall Street Journal* is 25 words).
- The amount of (unexpected!) ambiguity increases rapidly with sentence length. This poses a problem, even for chart parsers, if they have to keep track of all possible analyses.
- It would reduce the amount of work required if we could ignore improbable analyses.

A second provision passed by the Senate and House would eliminate a rule allowing companies that post losses resulting from LBO debt to receive refunds of taxes paid over the previous three years. [wsj_1822] (33 words)

Motivation 2: Coverage

- It is actually very difficult to write a grammar that covers all the constructions used in ordinary text or speech.
- Typically hundreds of rules are required in order to capture both all the different linguistic patterns and all the different possible analyses of the same pattern. (How many grammar rules did we have to add to cover three different analyses of *You made her duck*?)
- Ideally, one wants to induce (learn) a grammar from a corpus.
- Grammar induction requires probabilities.

Motivation 3: Representativeness

The likelihood of a particular construction can vary, depending on:

- register (formal vs. informal): eg, *greenish, alot*, subject-drop (*Want a beer?*) are all more probable in informal than formal register;
- genre (newspapers, essays, mystery stories, jokes, ads, Twitter, etc.): PoS-taggers trained on different genres often have very different distributions.
- domain (biology, patent law, football, etc.).

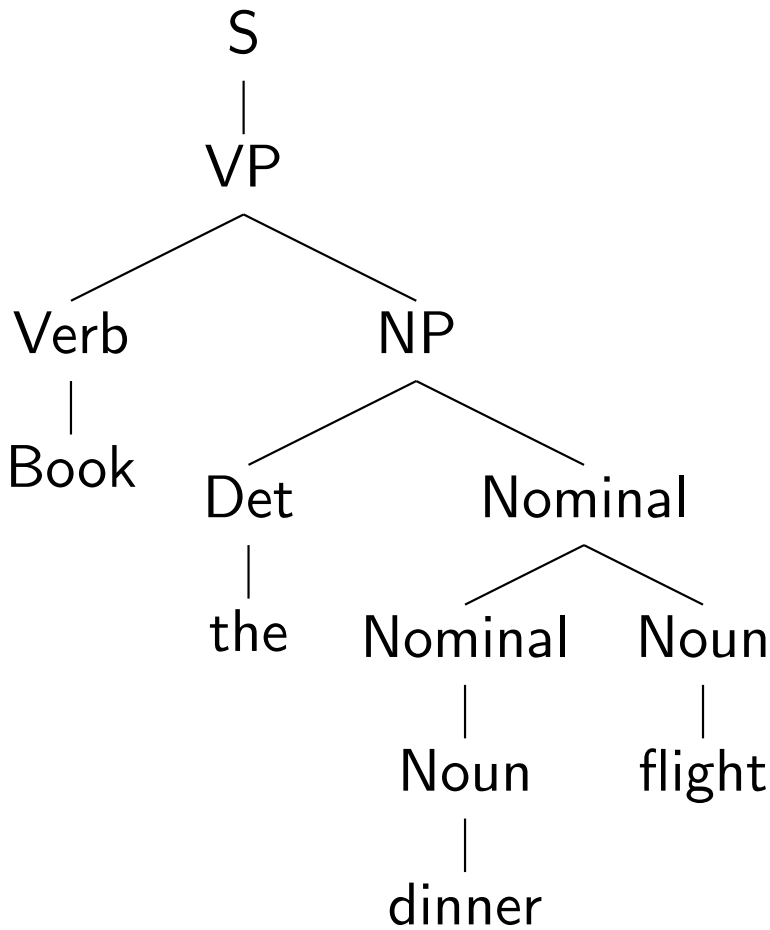
Probabilistic grammars and parsers can reflect these kinds of distributions.

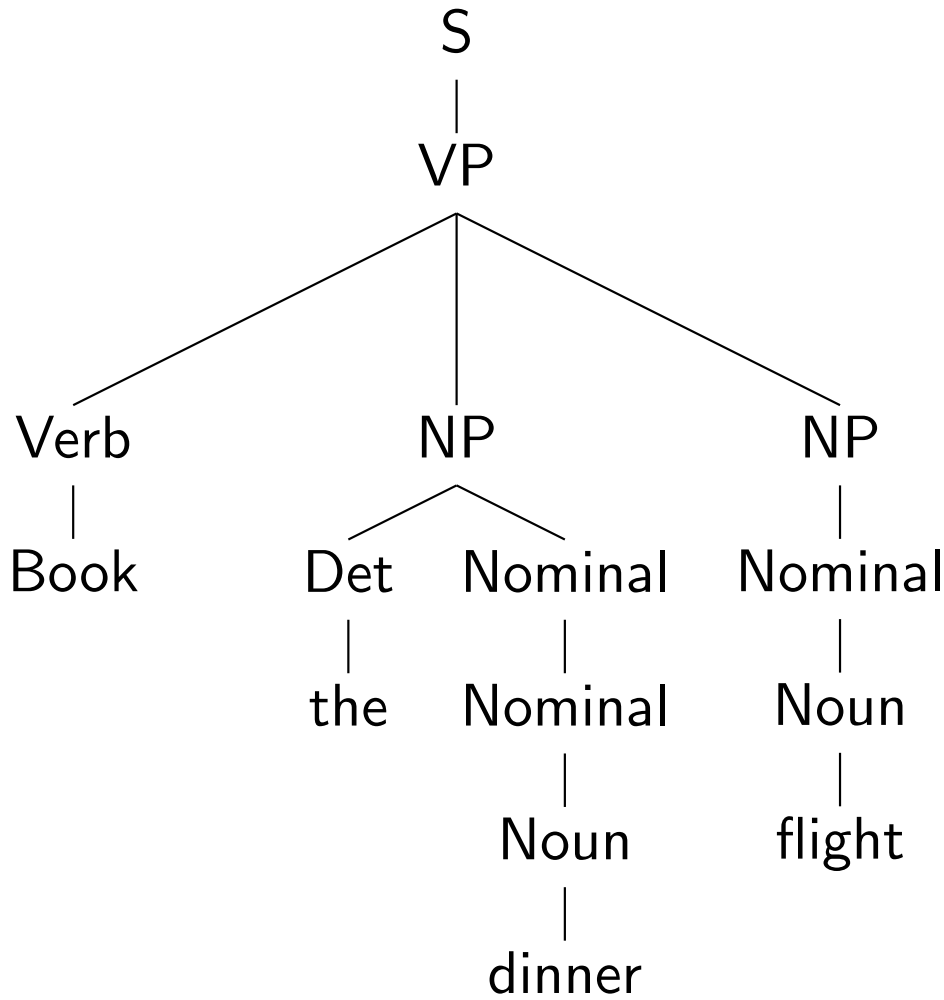
What Have We Achieved So Far?

- A zoo of parsing algorithms: Recursive descent parsing, Shift-Reduce parsing, CYK algorithm, Earley algorithm
- They all work as recognisers, or with backpointers: parsers that return all possible parses
- Is there a notion of best parse? Or is it better to keep all of these parses?

Example Parses for an Ambiguous Sentence

Book the dinner flight.





Probabilistic Context-Free Grammars

A PCFG $\langle N, \Sigma, R, S \rangle$ is defined as follows:

N	is the set of non-terminal symbols
Σ	is the terminals (disjoint from N)
R	is a set of rules of the form $A \rightarrow \beta[p]$ where $A \in N$ and $\beta \in (\sigma \cup N)*$, and p is a number between 0 and 1
S	a start symbol, $S \in N$

A PCFG is a CFG in which each rule is associated with a probability.

More about PCFGS

What does the p associated with each rule express?

It expresses the probability that the LHS non-terminal will be expanded as the RHS sequence.

- $P(A \rightarrow \beta | A)$
- $\sum_{\beta} P(A \rightarrow \beta | A) = 1$
- The sum of the probabilities associated with all of the rules expanding the non-terminal A is required to be 1.

$$A \rightarrow \beta \ [p] \quad \text{or} \quad P(A \rightarrow \beta | A) = p \quad \text{or} \quad P(A \rightarrow \beta) = p$$

Example Grammar

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow the$	[.10]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Det \rightarrow a$	[.90]
$S \rightarrow VP$	[.05]	$Noun \rightarrow book$	[.10]
$NP \rightarrow Pronoun$	[.35]	$Noun \rightarrow flight$	[.30]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Noun \rightarrow dinner$	[.60]
$NP \rightarrow Det\ Nominal$	[.20]	$Proper\text{-}Noun \rightarrow Houston$	[.60]
$NP \rightarrow Nominal$	[.15]	$Proper\text{-}Noun \rightarrow NWA$	[.40]
$Nominal \rightarrow\ Noun$	[.75]	$Aux \rightarrow does$	[.60]
$Nominal \rightarrow Nominal\ Noun$	[.05]	$Aux \rightarrow can$	[.40]
$VP \rightarrow Verb$	[.35]	$Verb \rightarrow book$	[.30]
$VP \rightarrow Verb\ NP$	[.20]	$Verb \rightarrow include$	[.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	$Verb \rightarrow prefer$	[.20]
$VP \rightarrow Verb\ PP$	[.15]	$Verb \rightarrow sleep$	[.20]

PCFGs as a random process

Start with the root node, and at each step, probabilistically expand the nodes until you hit a terminal symbol:

PCFGs and consistency

Question: Does this process always have to terminate?

PCFGs and consistency

Question: Does this process always have to terminate?

Consider the grammar, for some $\epsilon > 0$:

Example

$S \to S\,S$ with probability $0.5 + \epsilon$
$S \to a$ with probability $0.5 - \epsilon$

PCFGs and consistency

Question: Does this process always have to terminate?

Consider the grammar, for some $\epsilon > 0$:

Example

$S \to S\,S$ with probability $0.5 + \epsilon$
$S \to a$ with probability $0.5 - \epsilon$

Whenever a nonterminal is expanded, it is more probable that the result will **increase** rather than **decrease** the number of nonterminals in the intermediate string.

Hence, the probability of seeing a finite tree is less than one!

Fortunately, most (but not all) common methods of assigning probabilities do not have this problem.

HMMs and PCFGs

- We know that context-free formalisms are more powerful than finite state transducers.
- As such, hidden Markov models can be represented as probabilistic context-free grammars:
  - For each emission $p(w \mid T)$ include a rule $T[\mathrm{pre}] \rightarrow w$ with the corresponding probability
  - For each transition $p(T \mid T')$ include a rule $T \rightarrow T[\mathrm{pre}]\,T'$ with the corresponding probability
  - Add the starting rule $S \rightarrow T$ for each T with probabilities $p(T \mid \mathrm{start})$
  - Add the rules $T \rightarrow \epsilon$ with probabilities $p(\mathrm{stop} \mid T)$
- We will get right branching trees

Independence assumptions in random process of PCFGs

We have a "Markovian" process here (limited memory of history)

Everything above a given node in the tree is conditionally independent of everything below that node if we know what is the nonterminal in that node

Another way to think about it: once we get to a new nonterminal and continue from there, we forget the whole derivation up to that point, and focus on that nonterminal as if it is a new root node

Too strong independence assumptions for natural language data.

PCFGs and disambiguation

- A PCFG assigns a probability to every parse tree or derivation associated with a sentence.
- This probability is the product of the rules applied in building the parse tree.

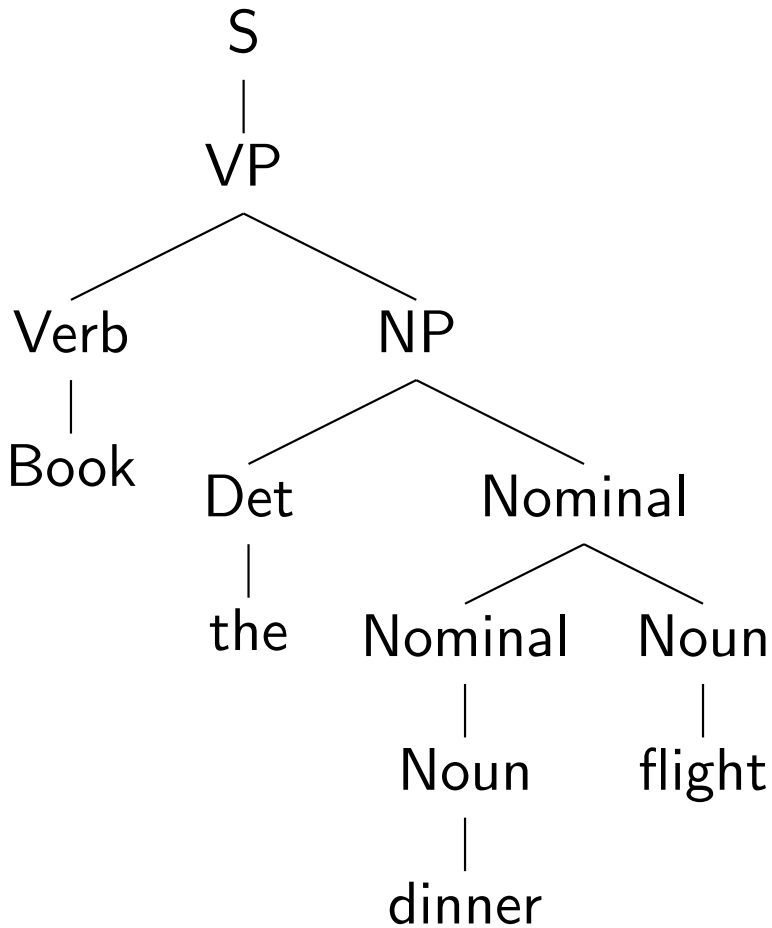
$$P(T,S) = \prod_{i=1}^{n} P(A_i \rightarrow \beta_i) \quad n \text{ is number of rules in } T$$

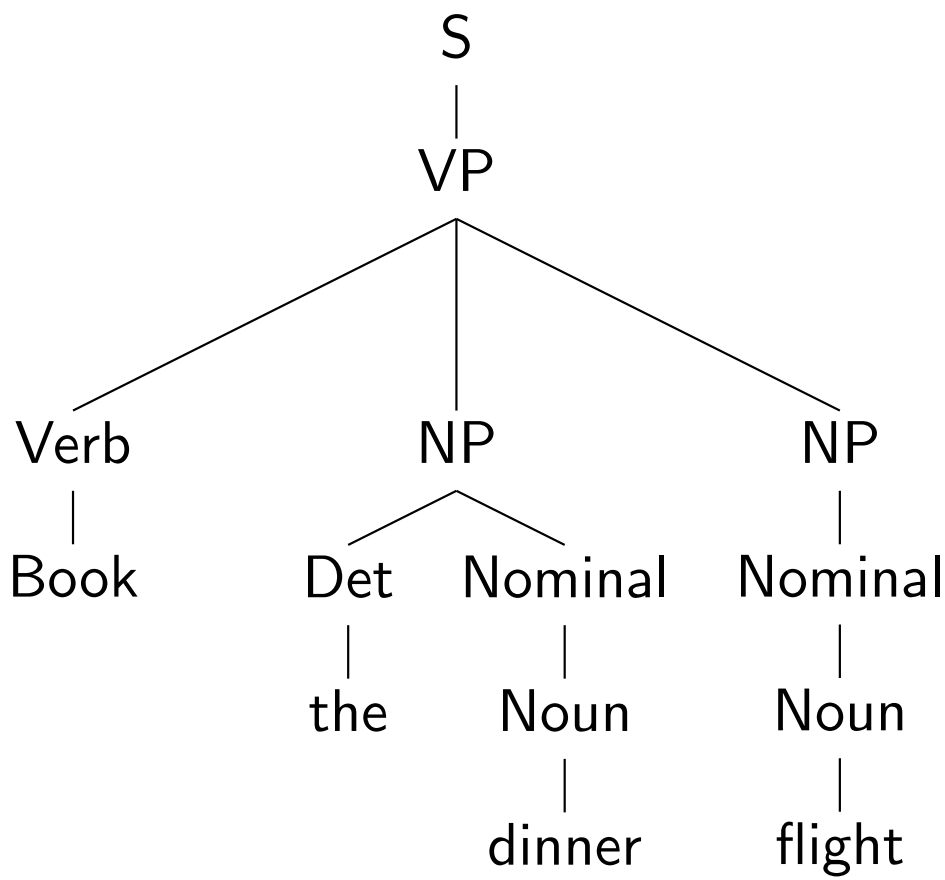
$$P(T,S) = P(T)P(S|T) = P(S)P(T|S) \text{ by definition}$$

$$\text{But } P(S|T) = 1 \quad \text{because } S \text{ is determined by } T$$

$$\text{So } P(T,S) = P(T)$$

Application 1: Disambiguation

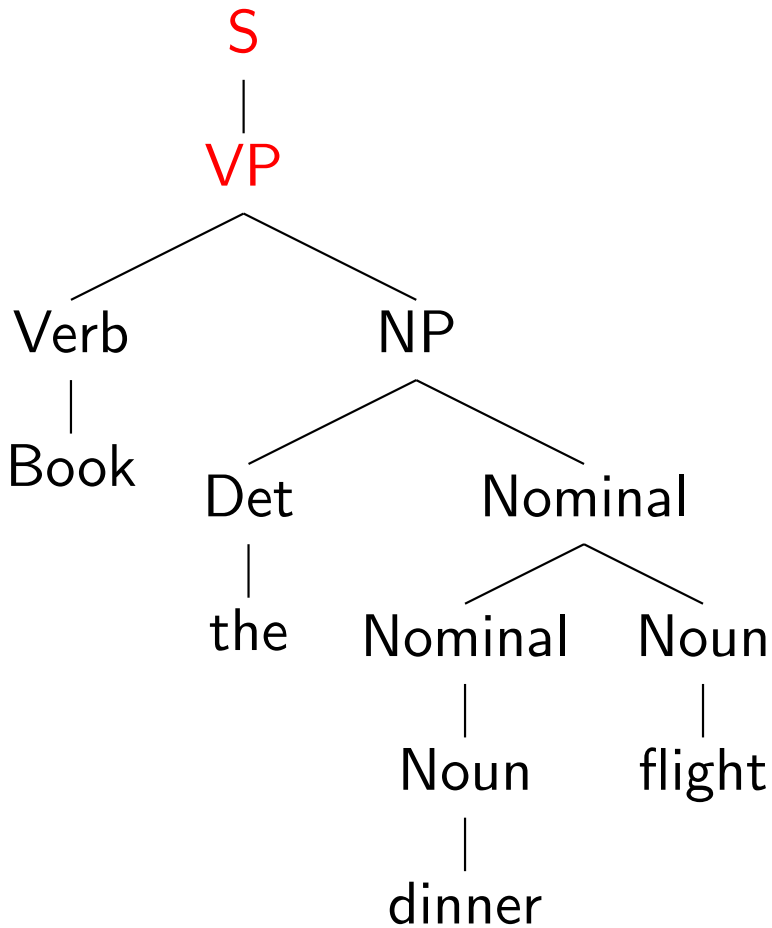


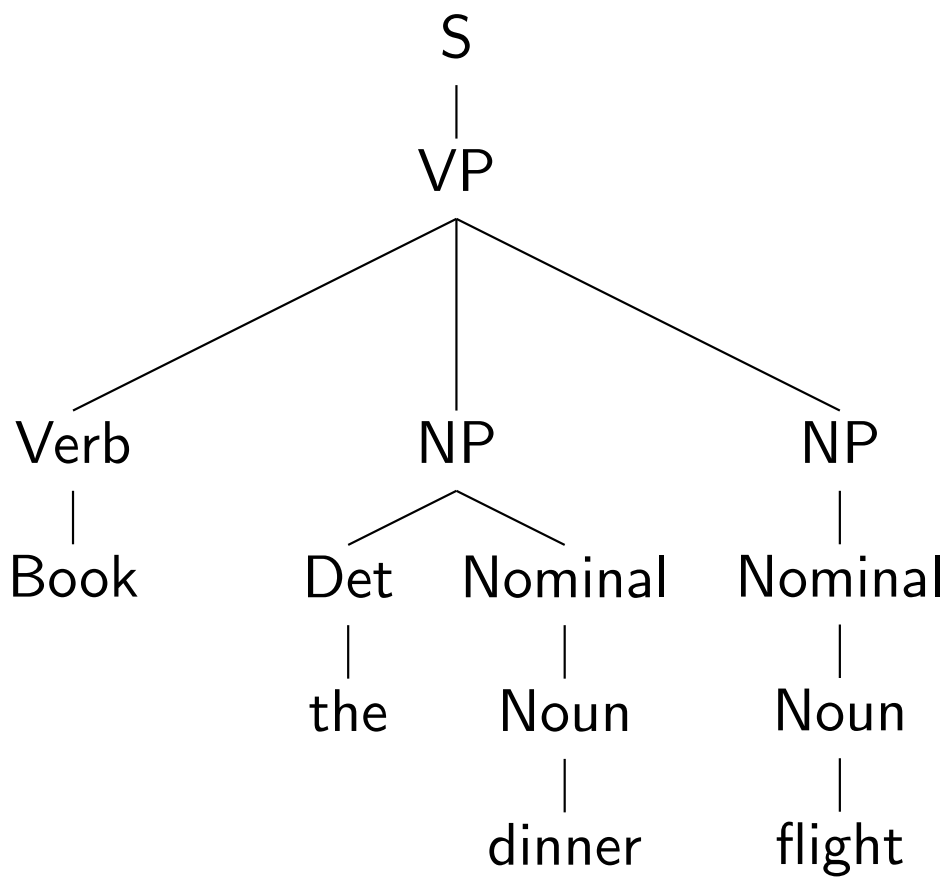


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

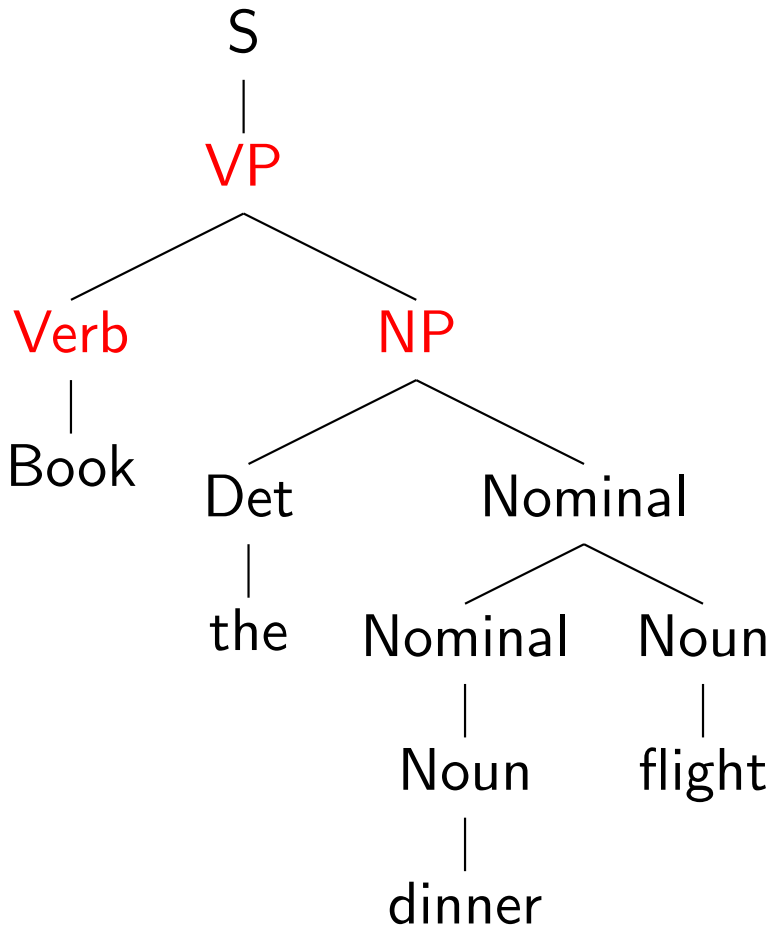


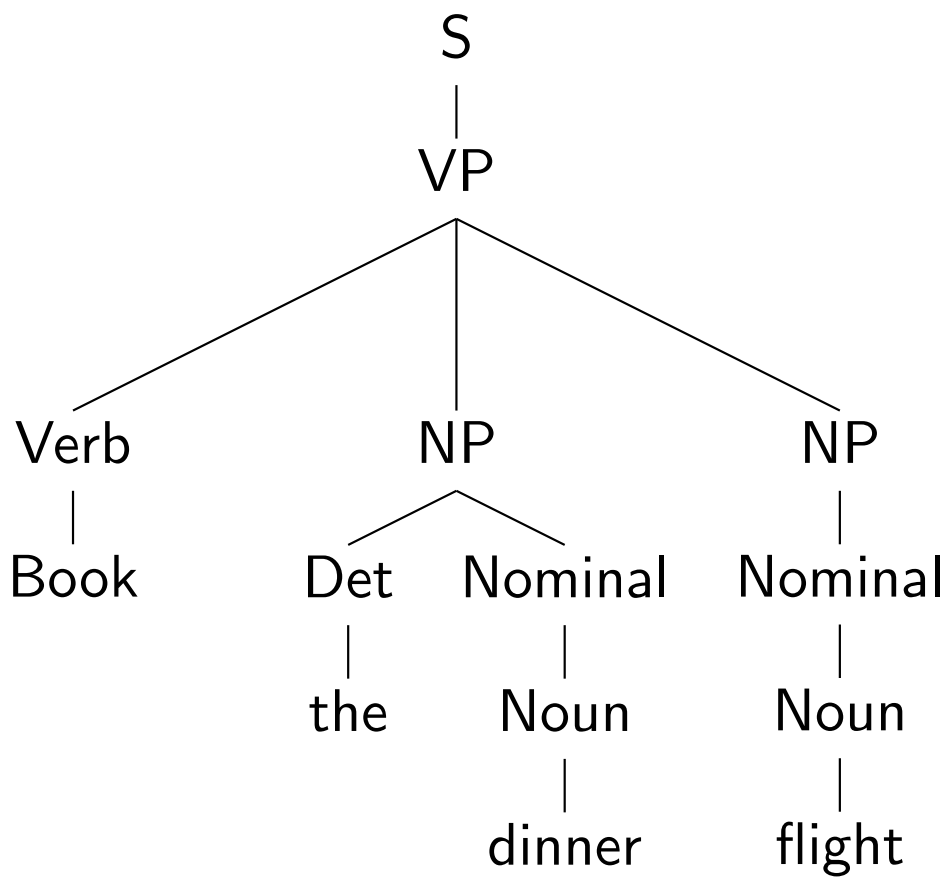


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

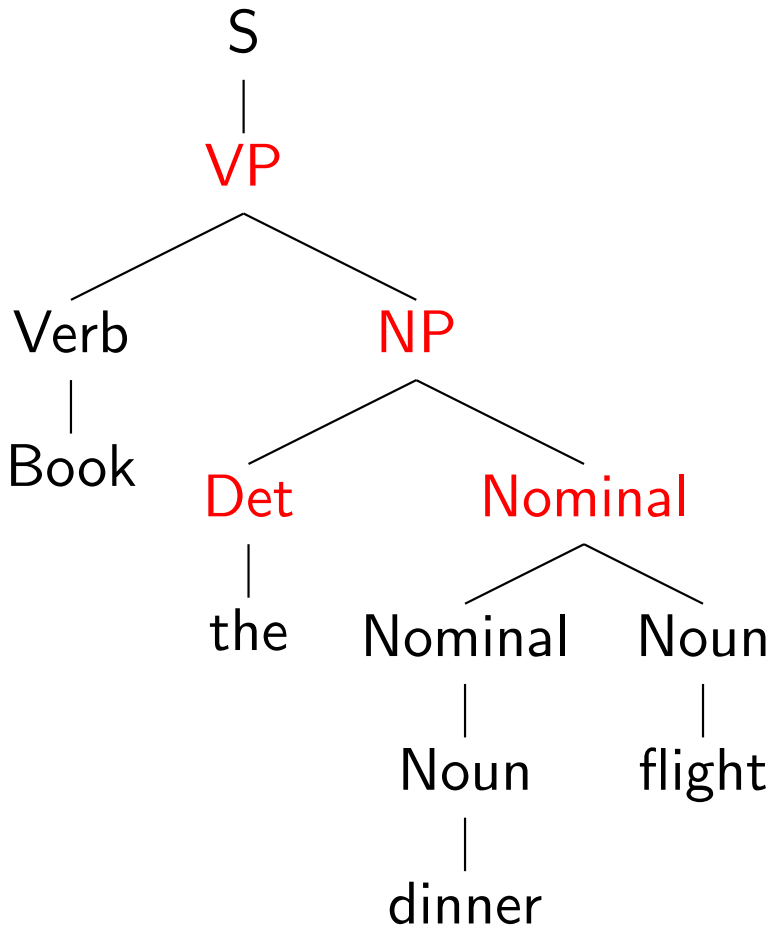


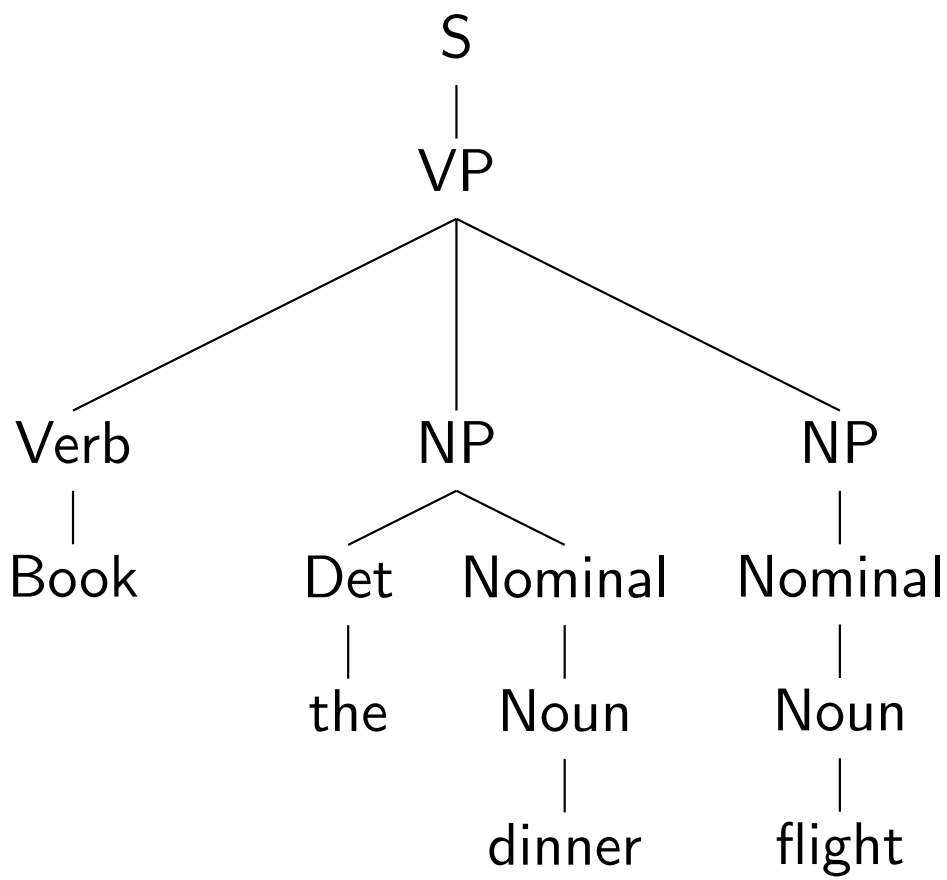


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

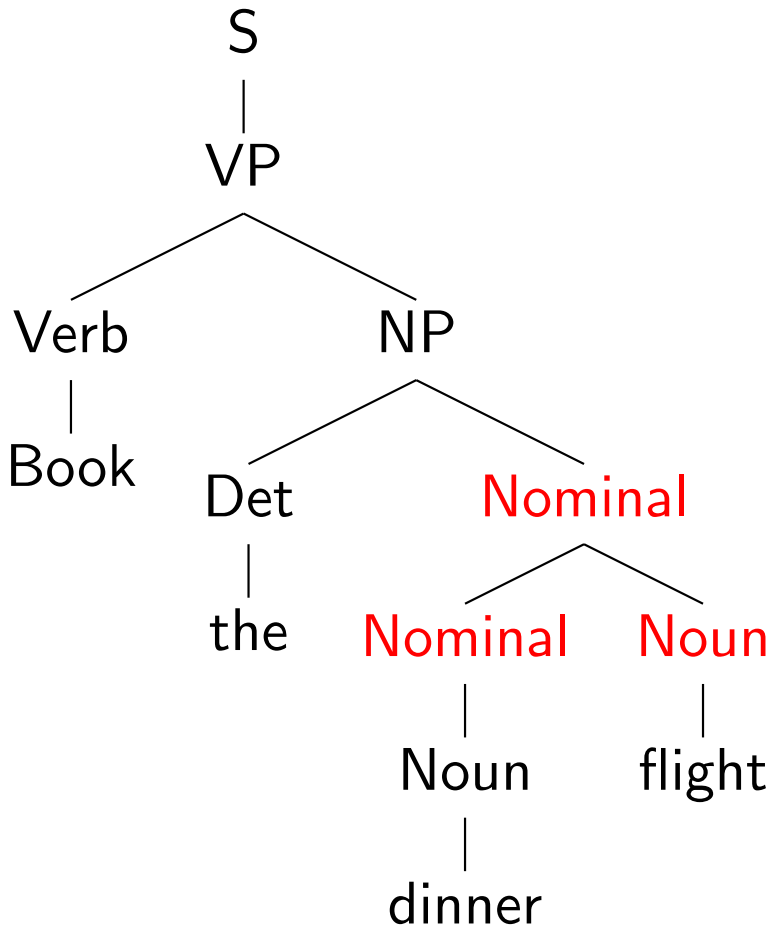
Application 1: Disambiguation

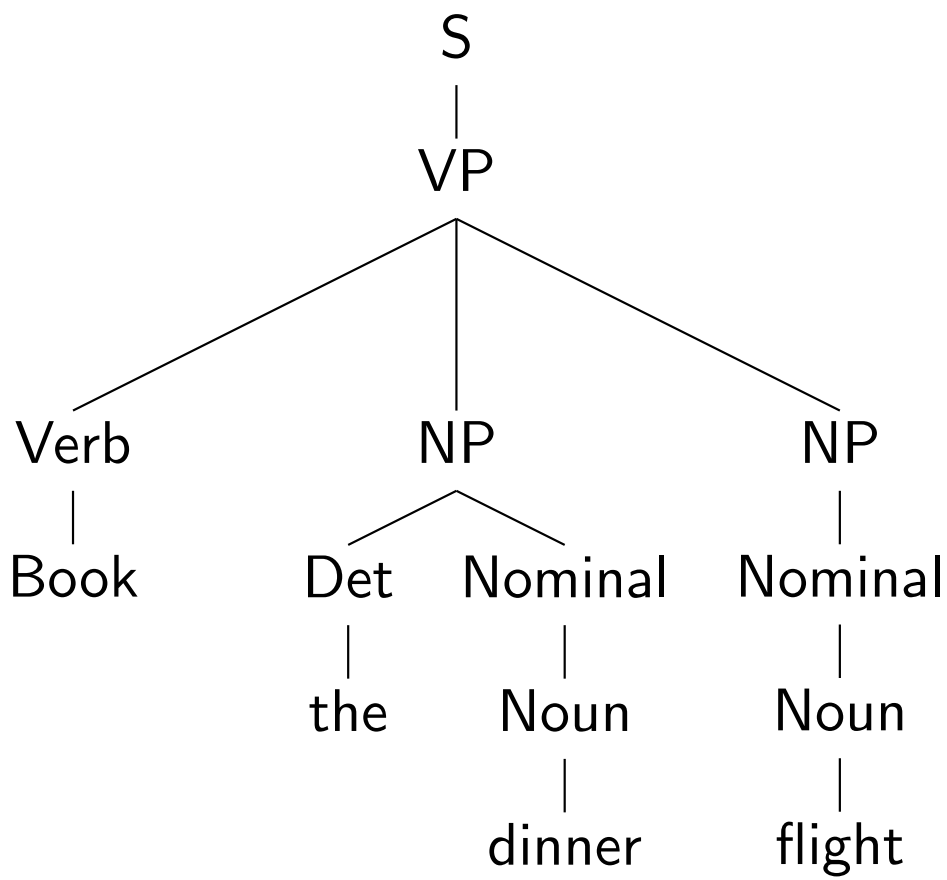




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

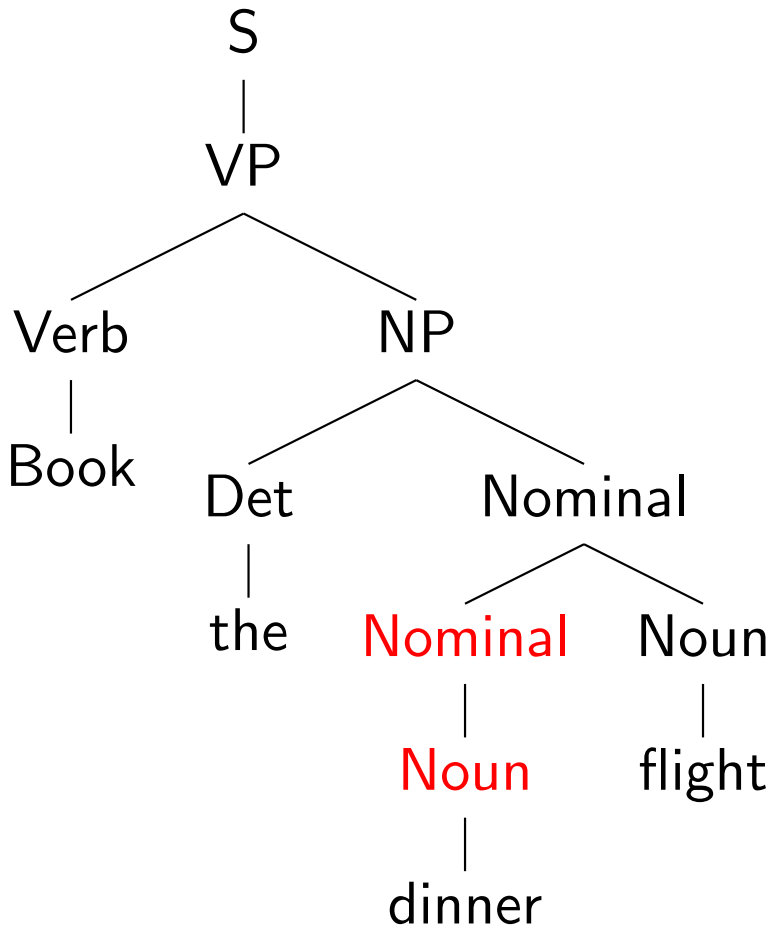
Application 1: Disambiguation

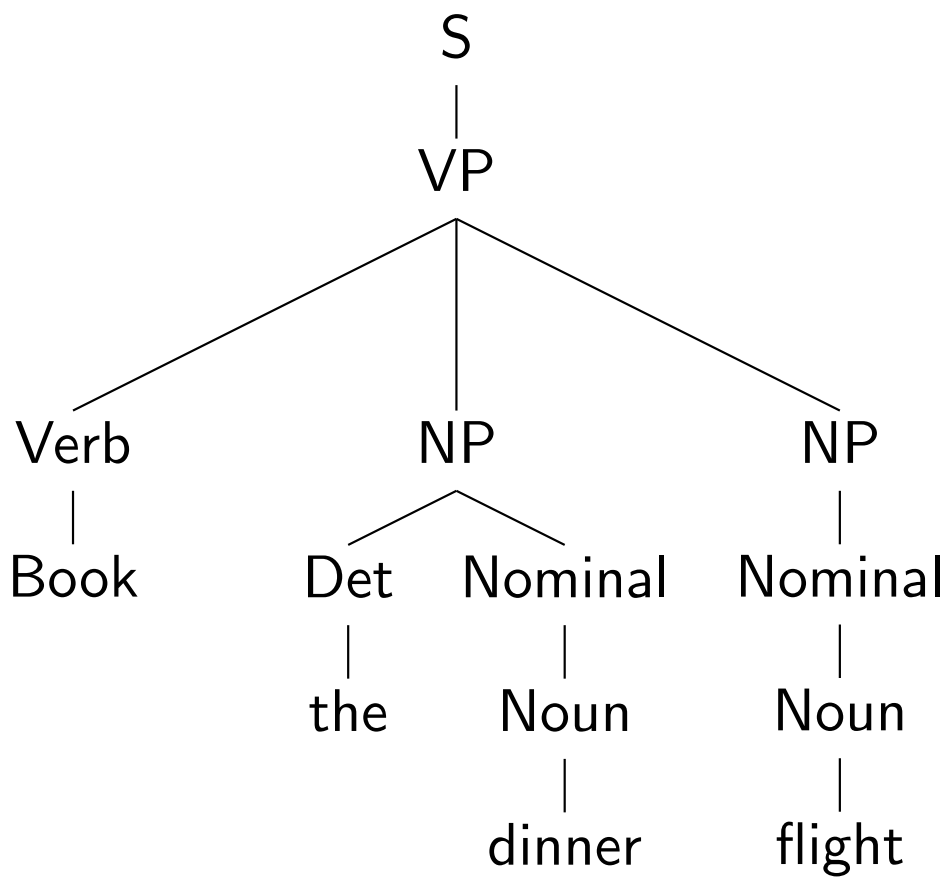




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Application 1: Disambiguation

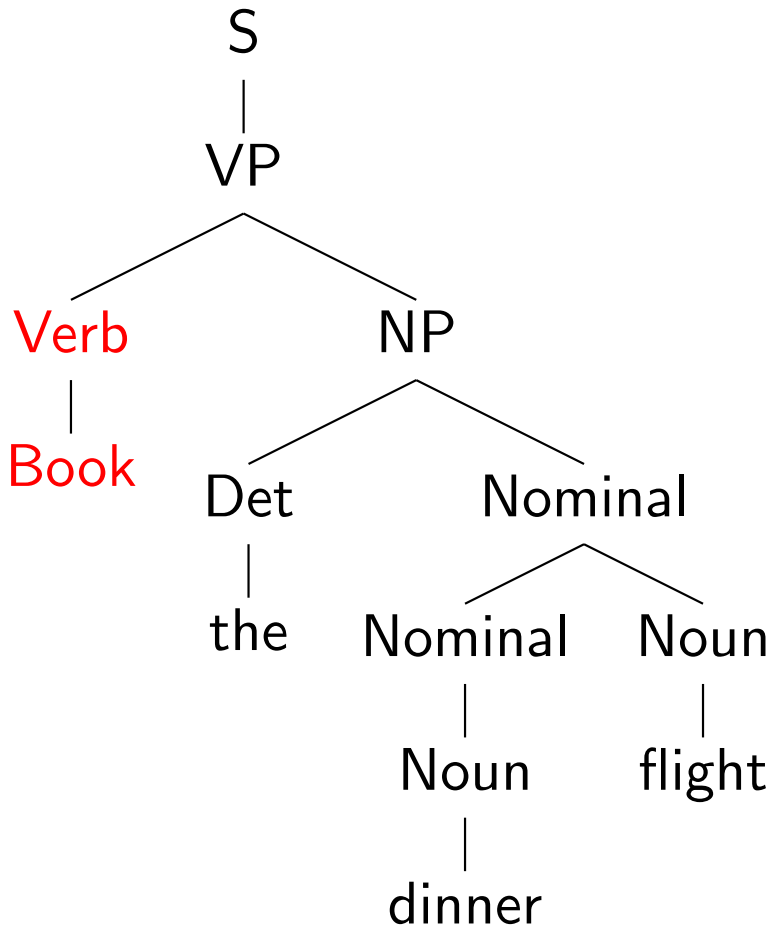


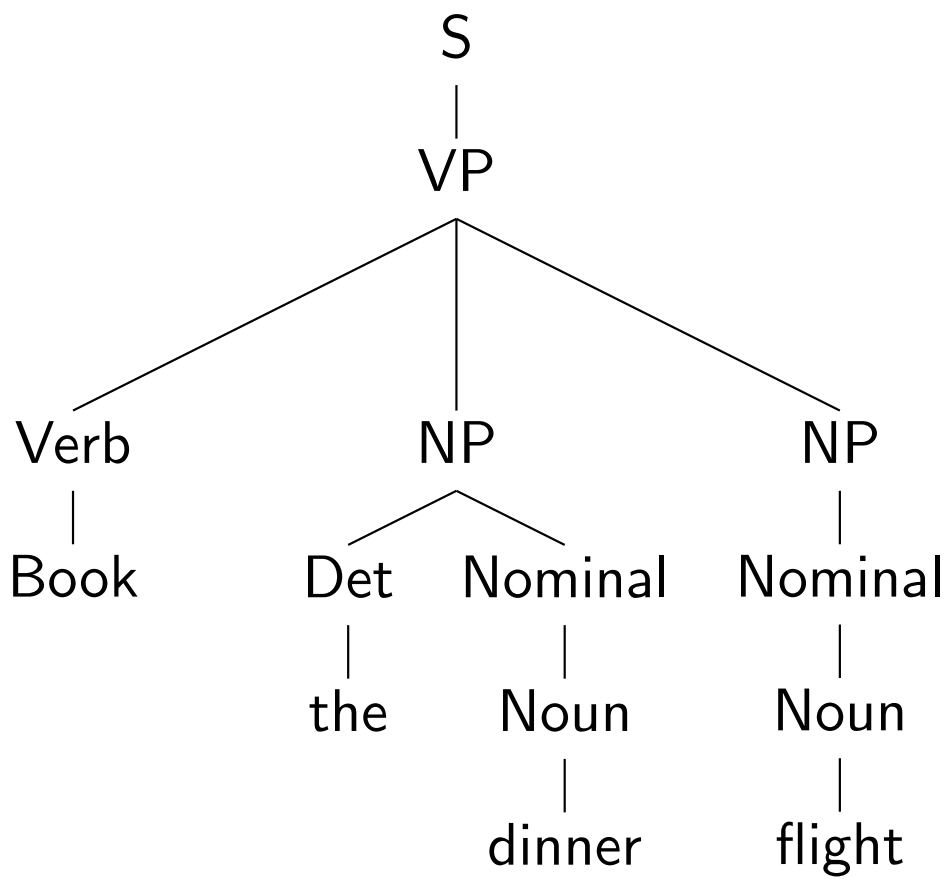


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

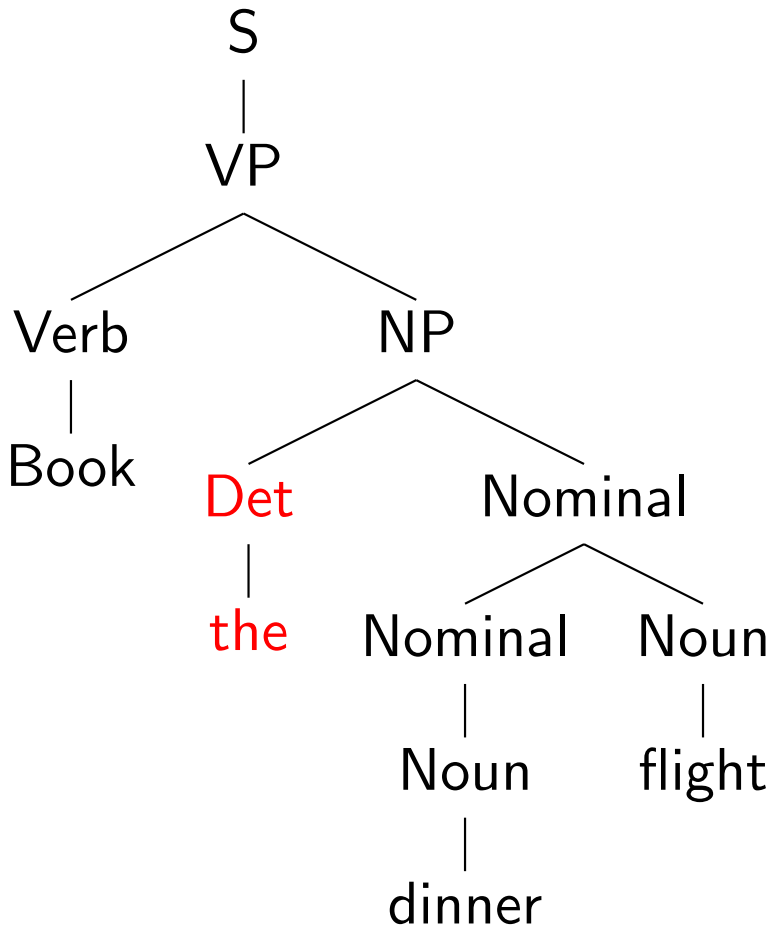


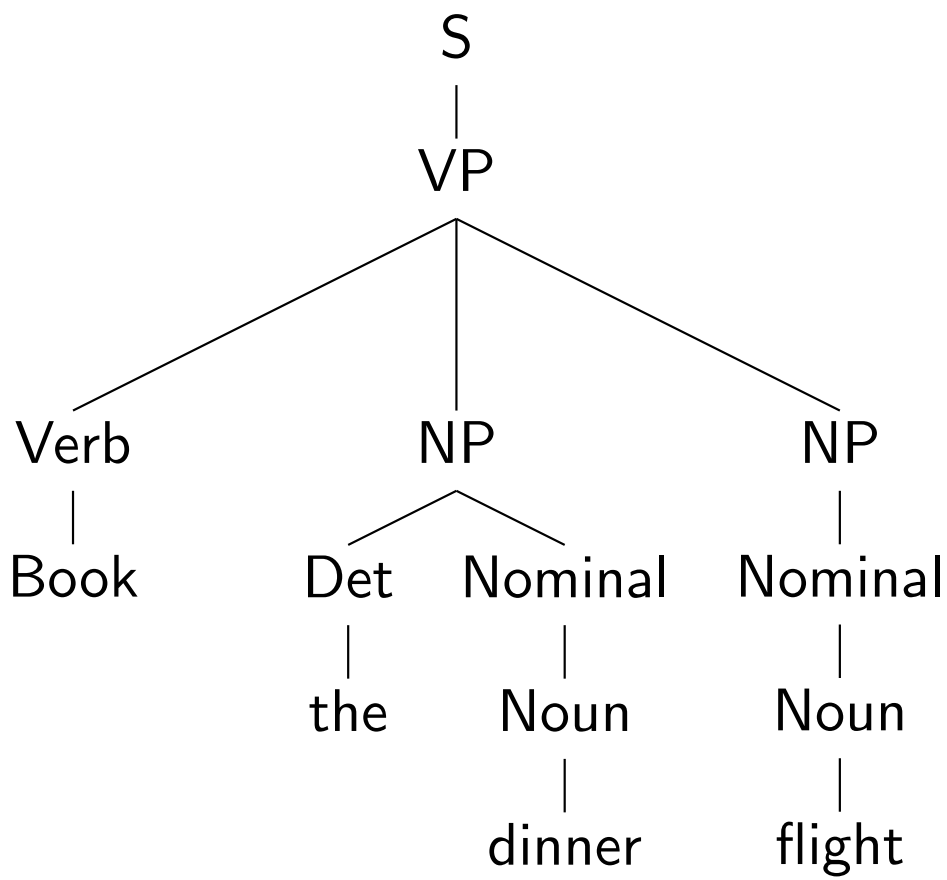


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

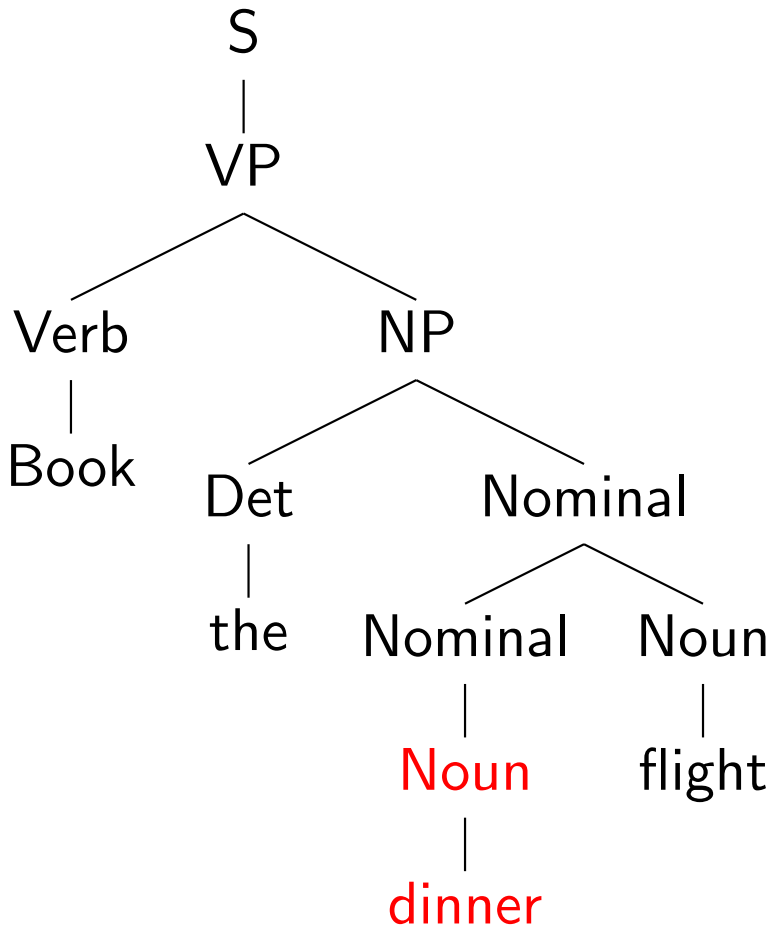
Application 1: Disambiguation

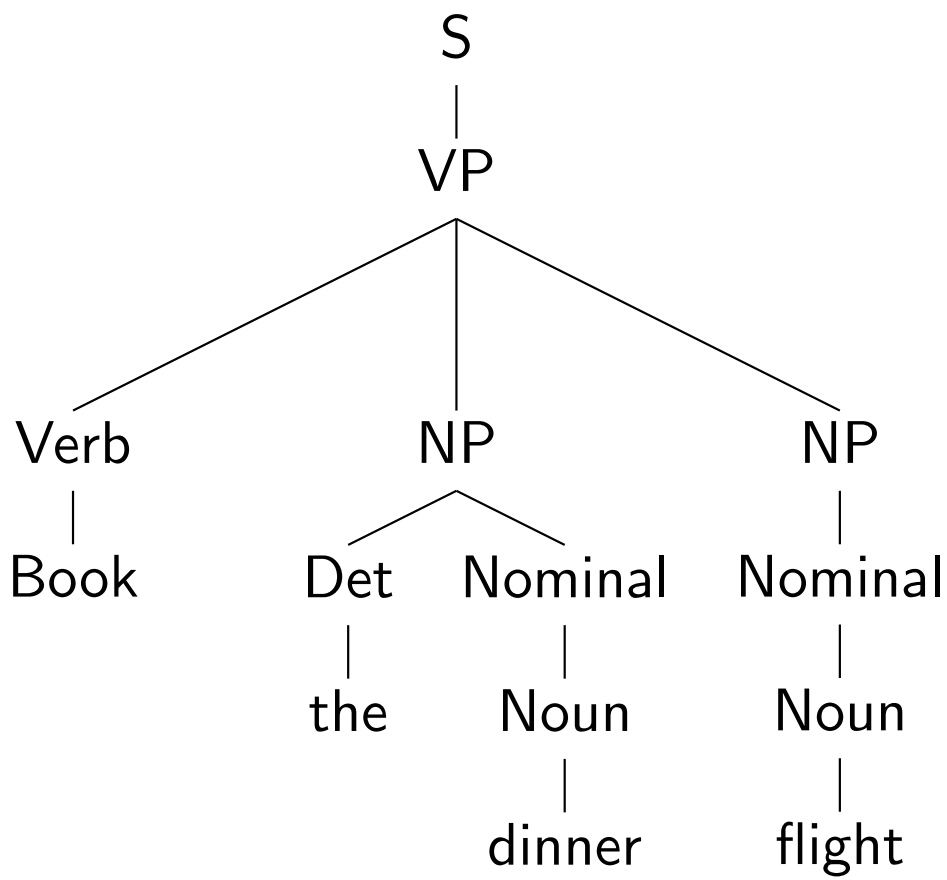




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

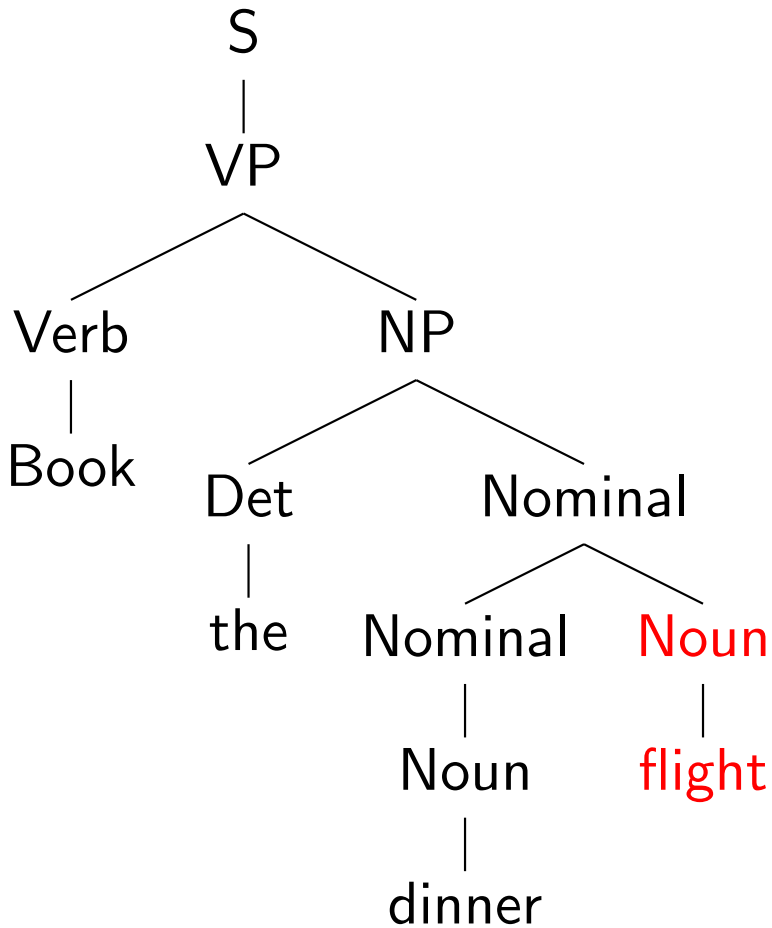
Application 1: Disambiguation

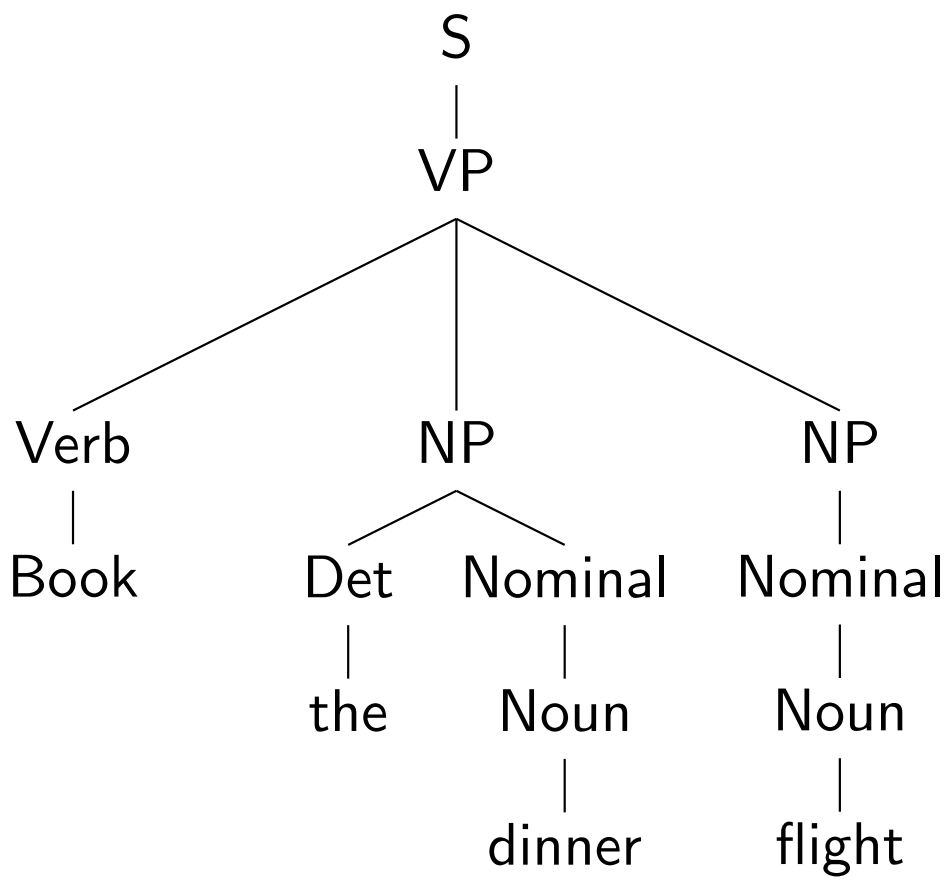




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Application 1: Disambiguation

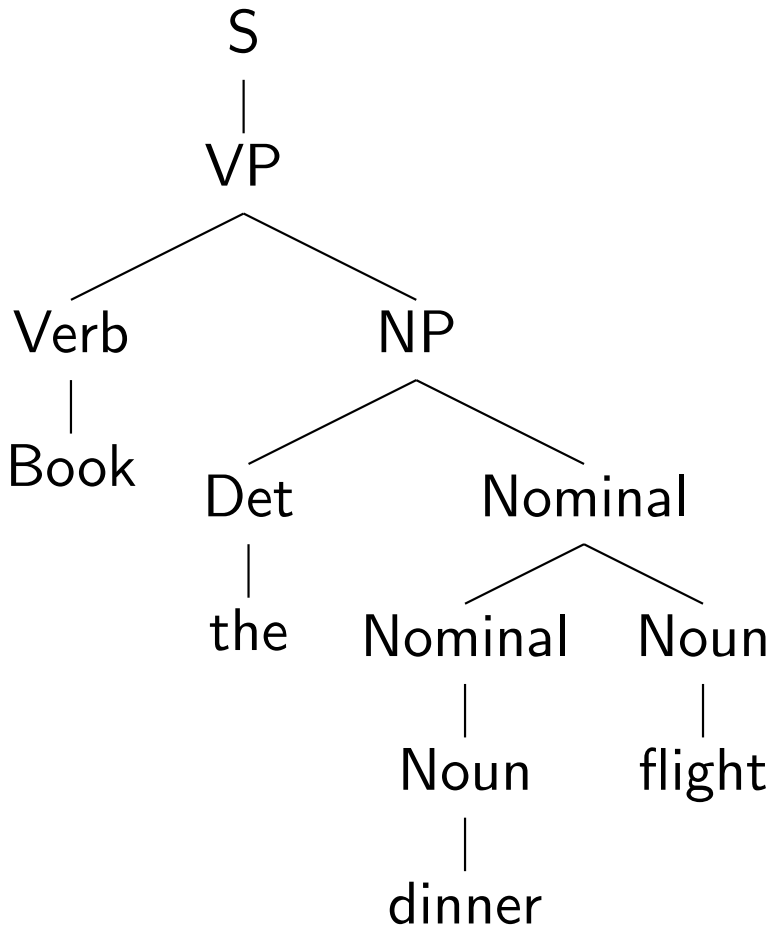


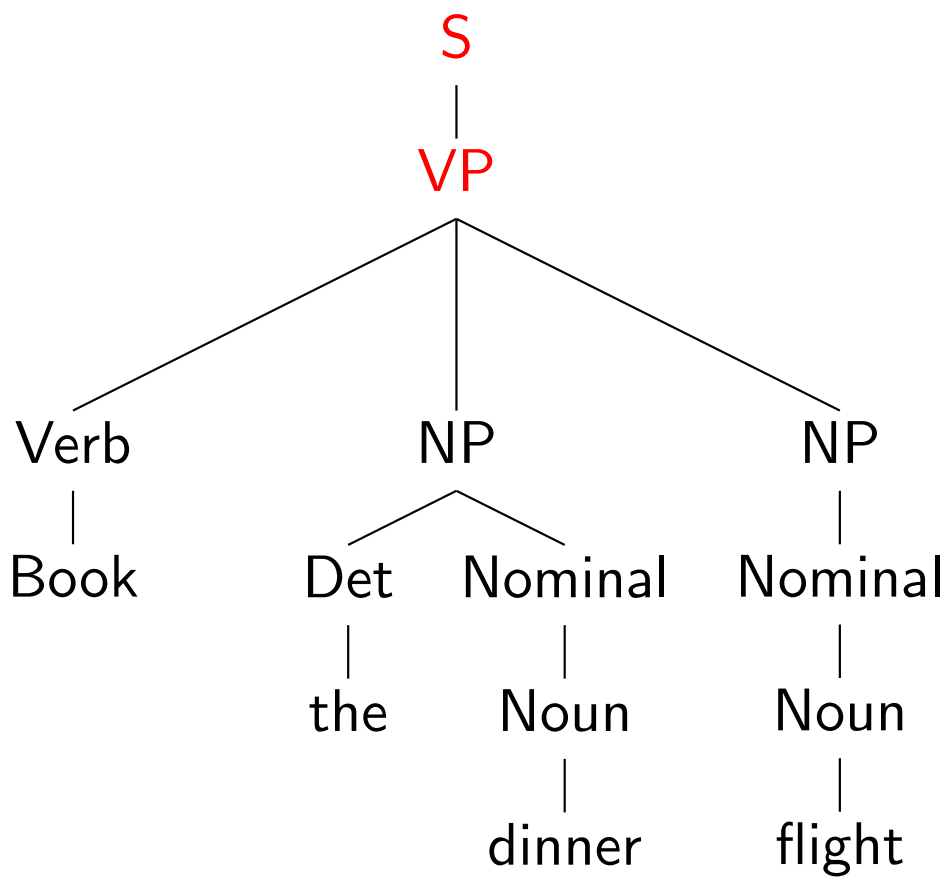


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

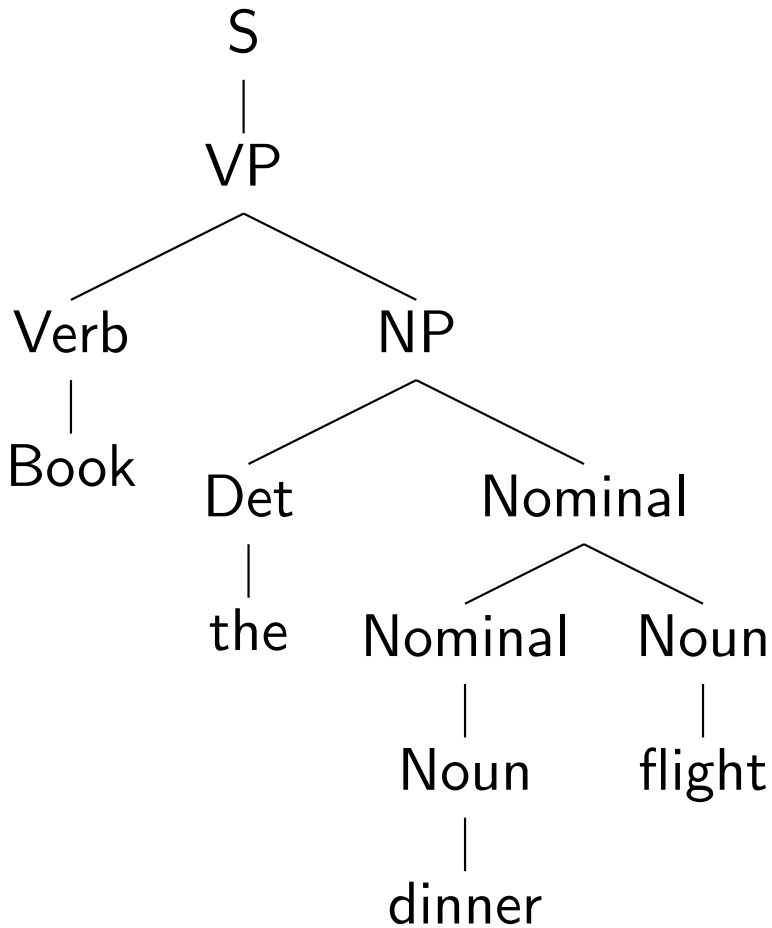


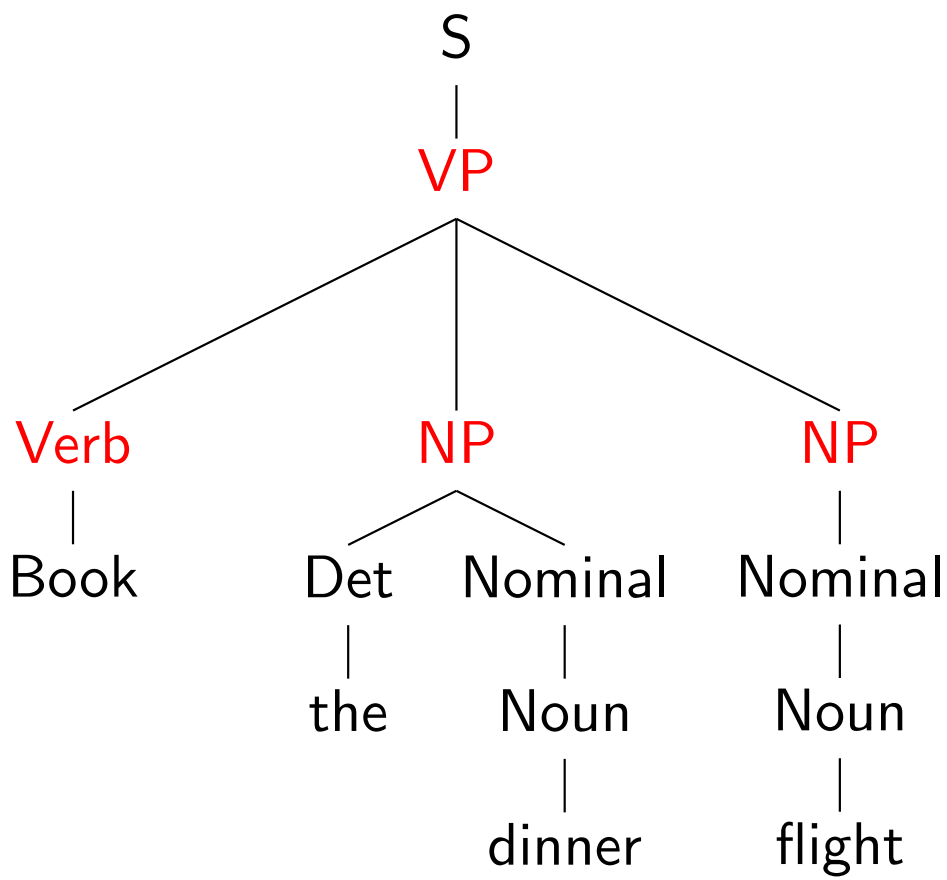


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

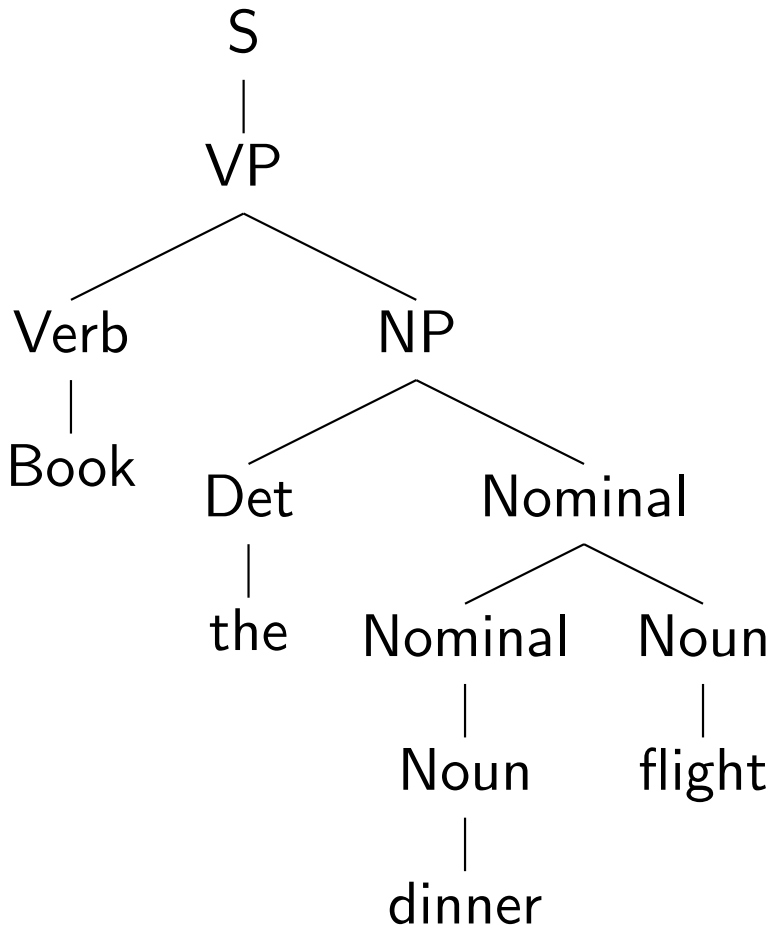
Application 1: Disambiguation

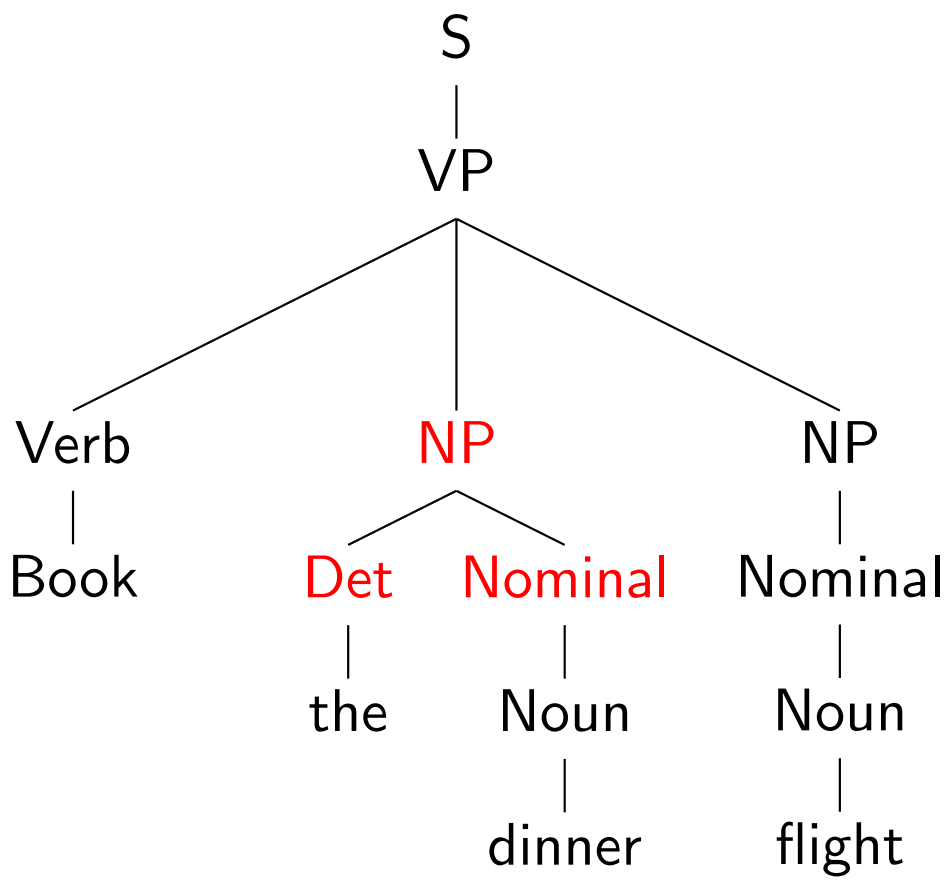




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Application 1: Disambiguation

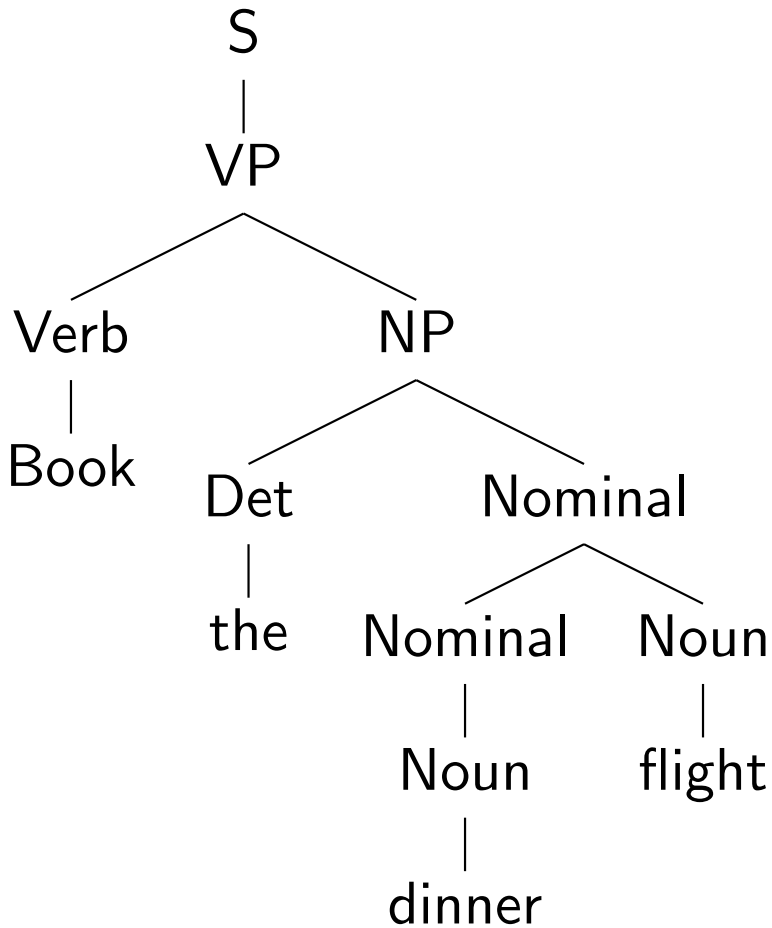


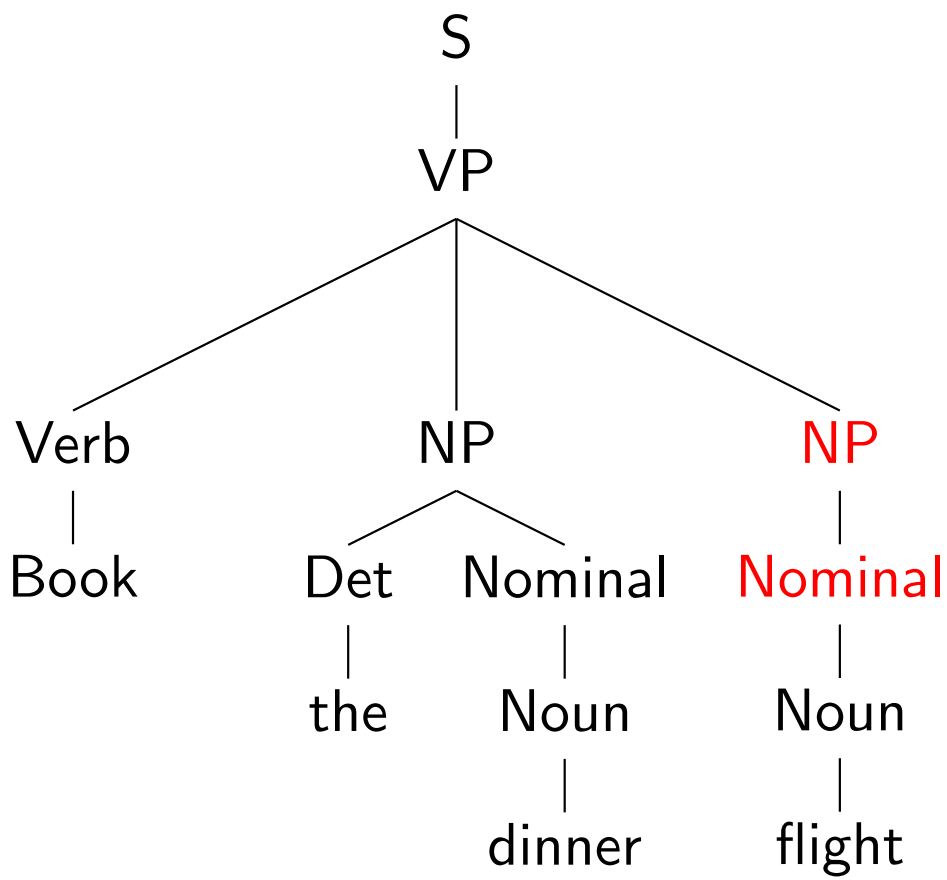


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

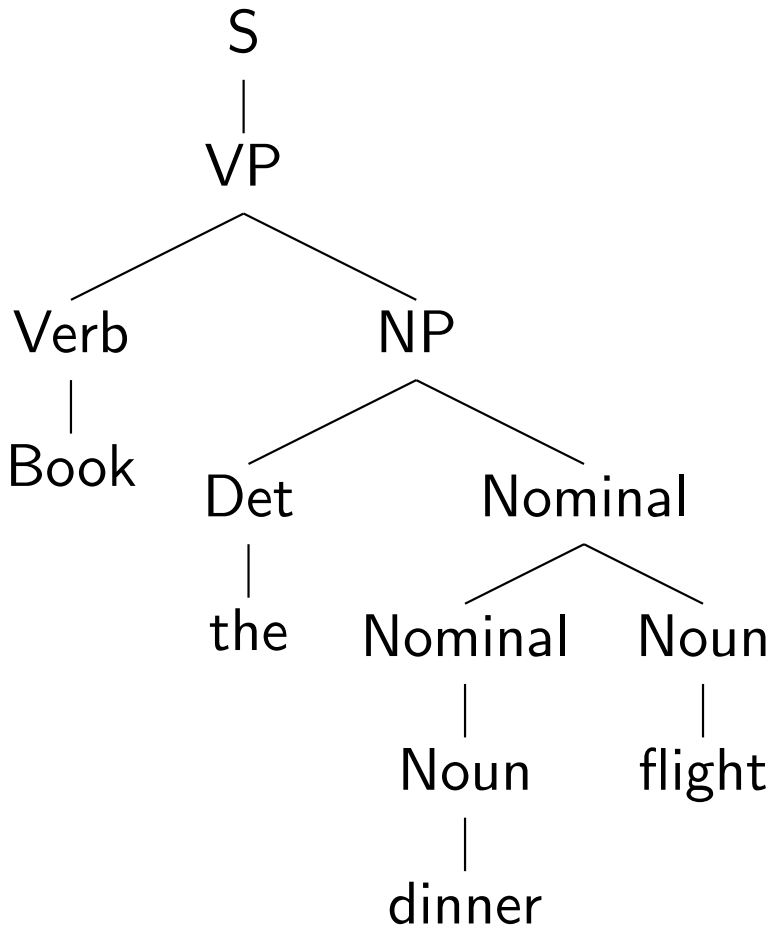


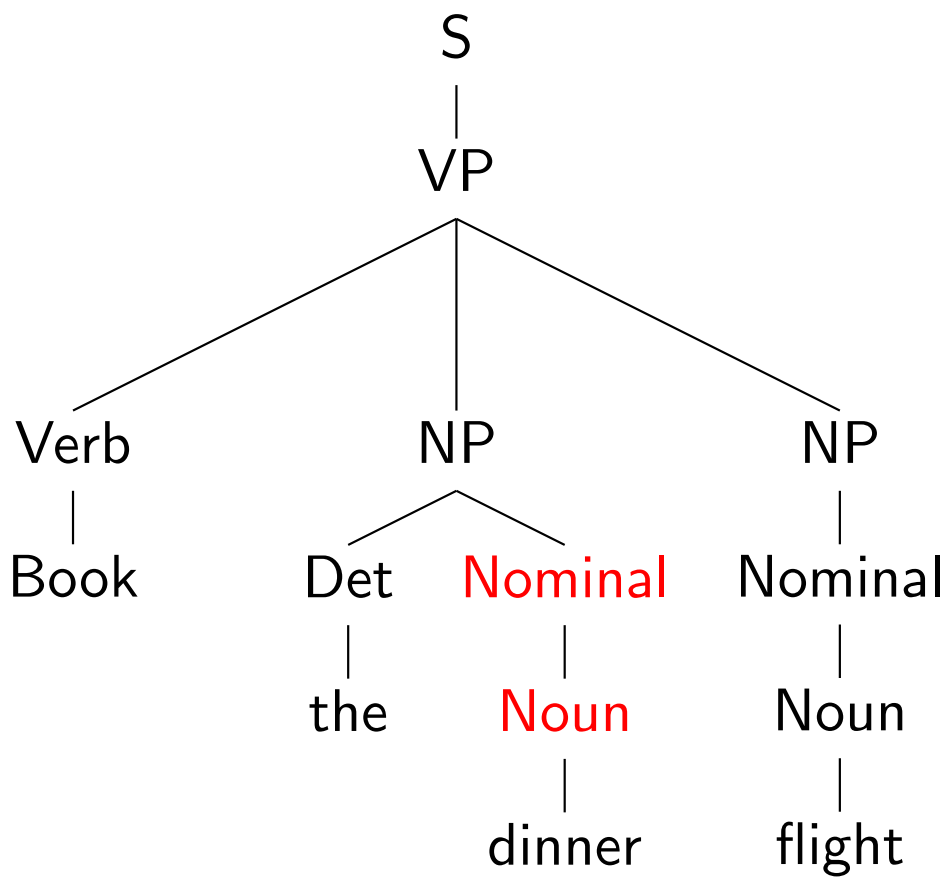


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

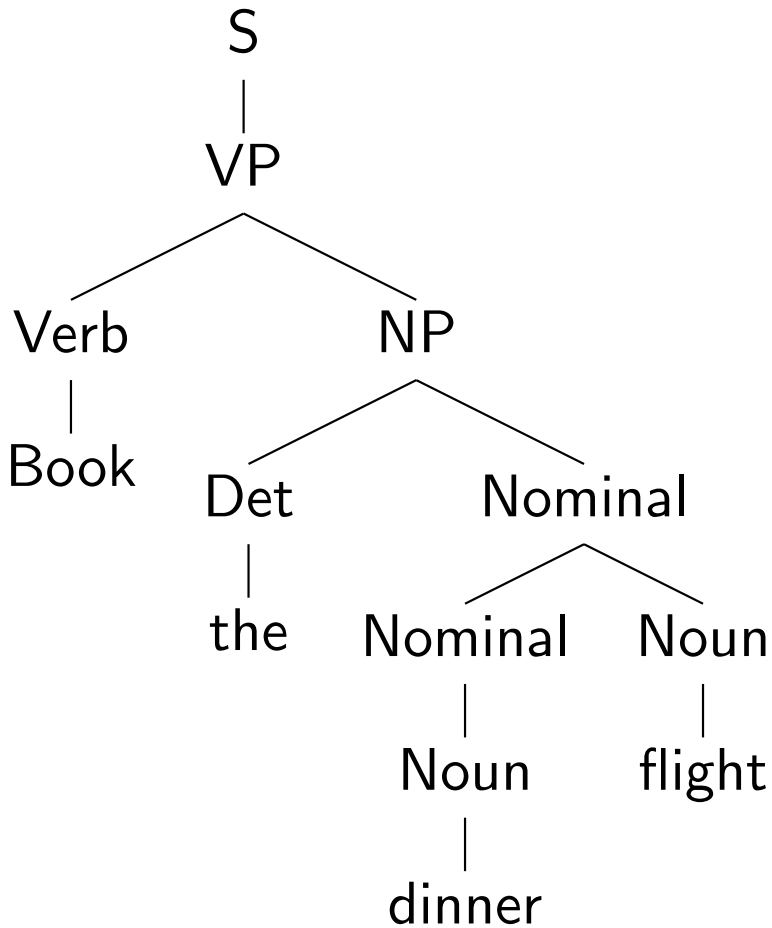


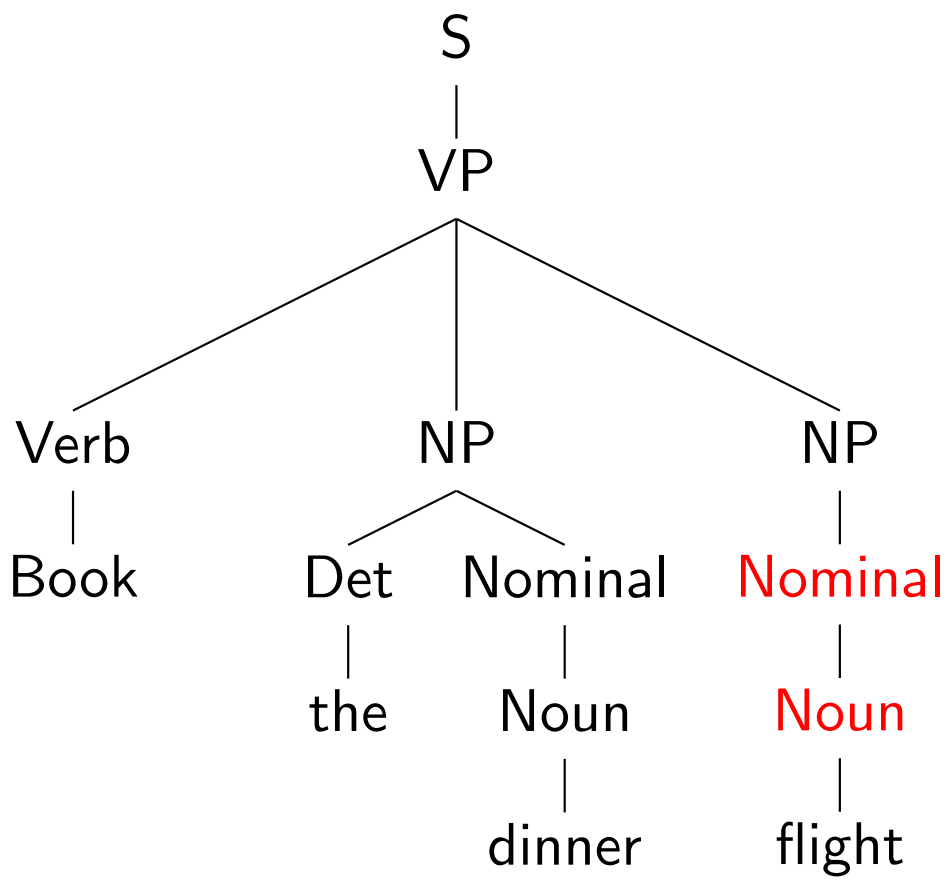


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

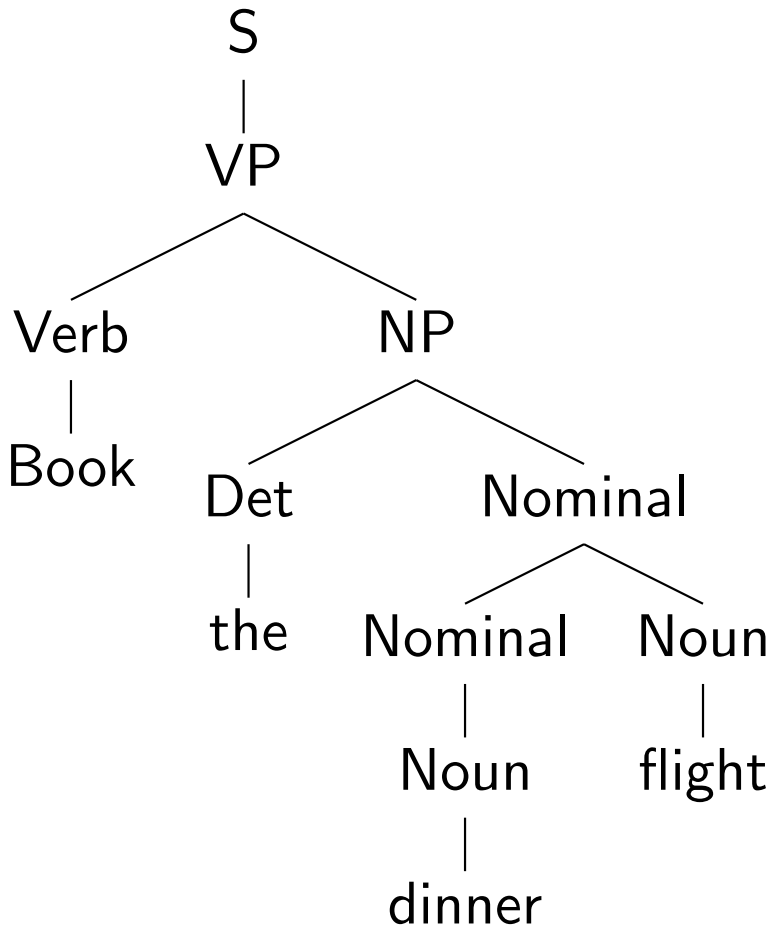


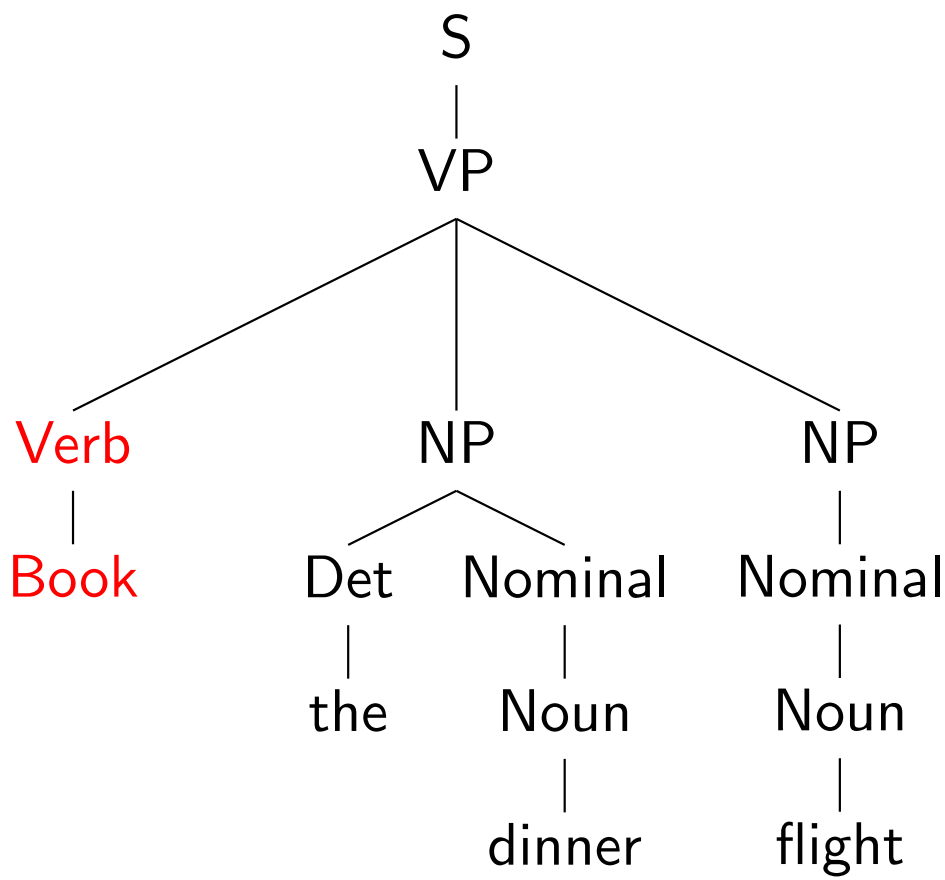


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

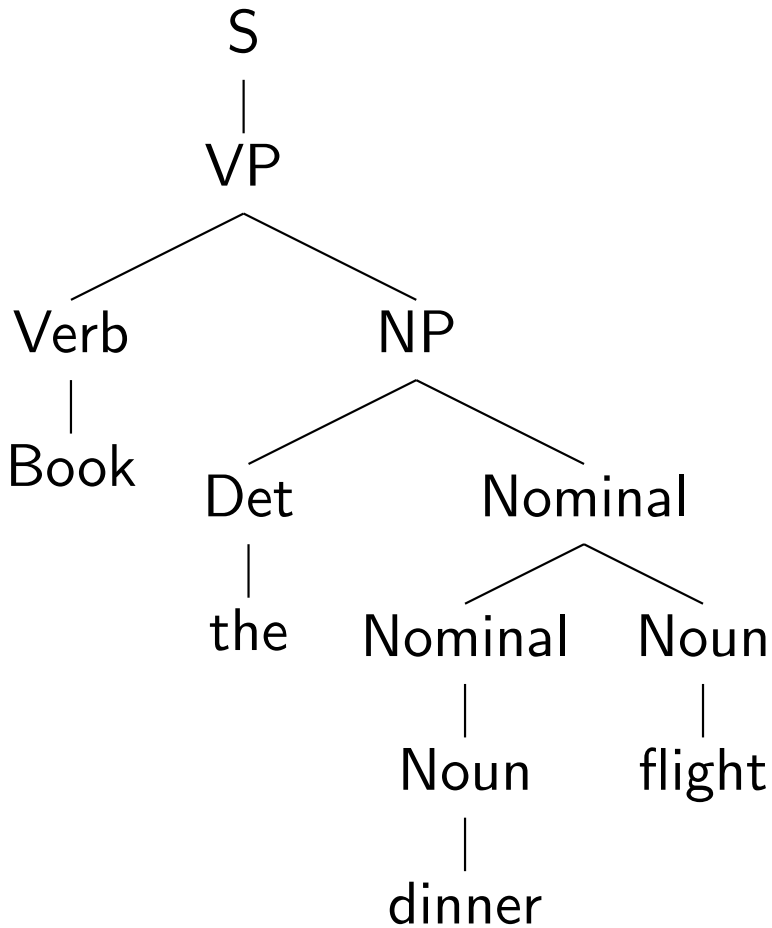


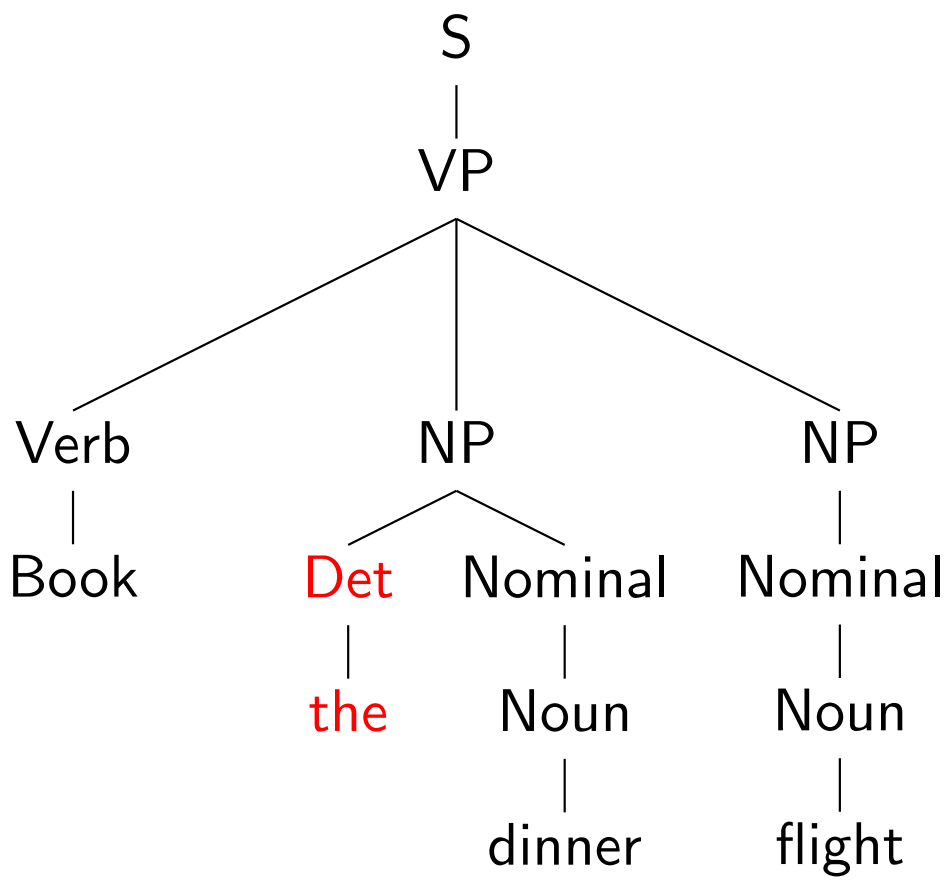


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

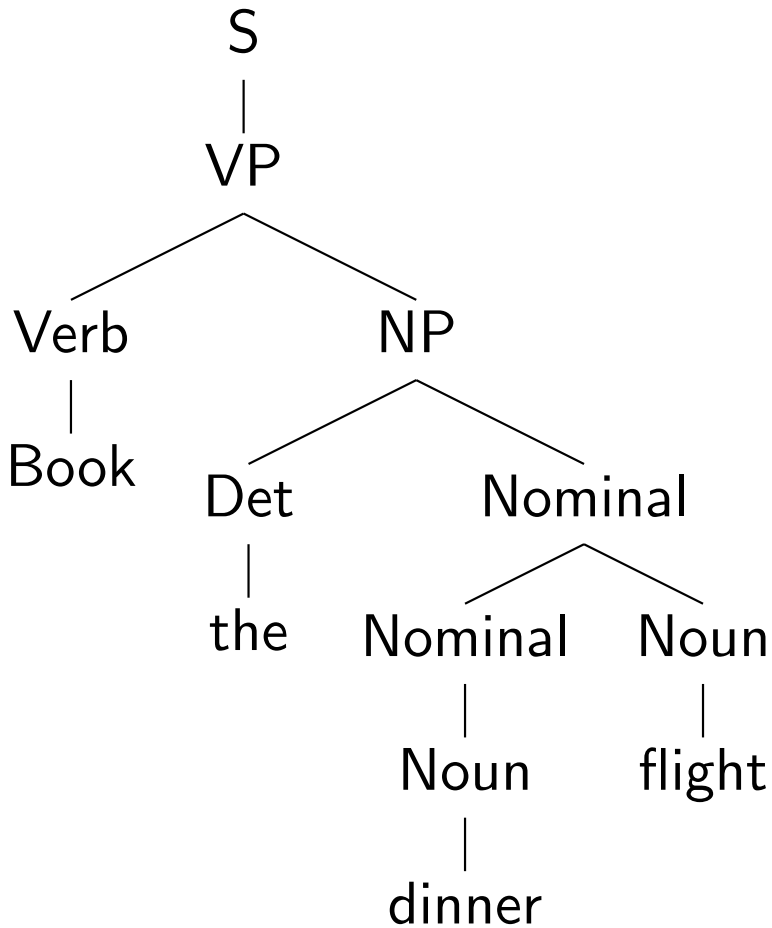
Application 1: Disambiguation

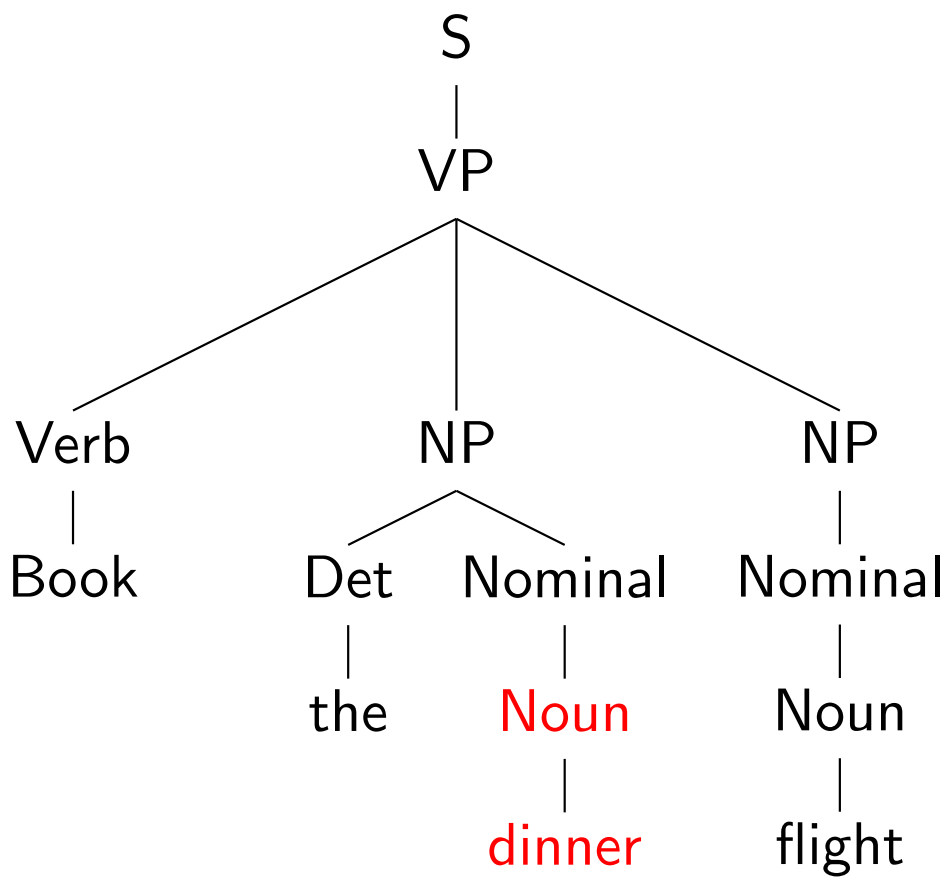




$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$
$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Application 1: Disambiguation

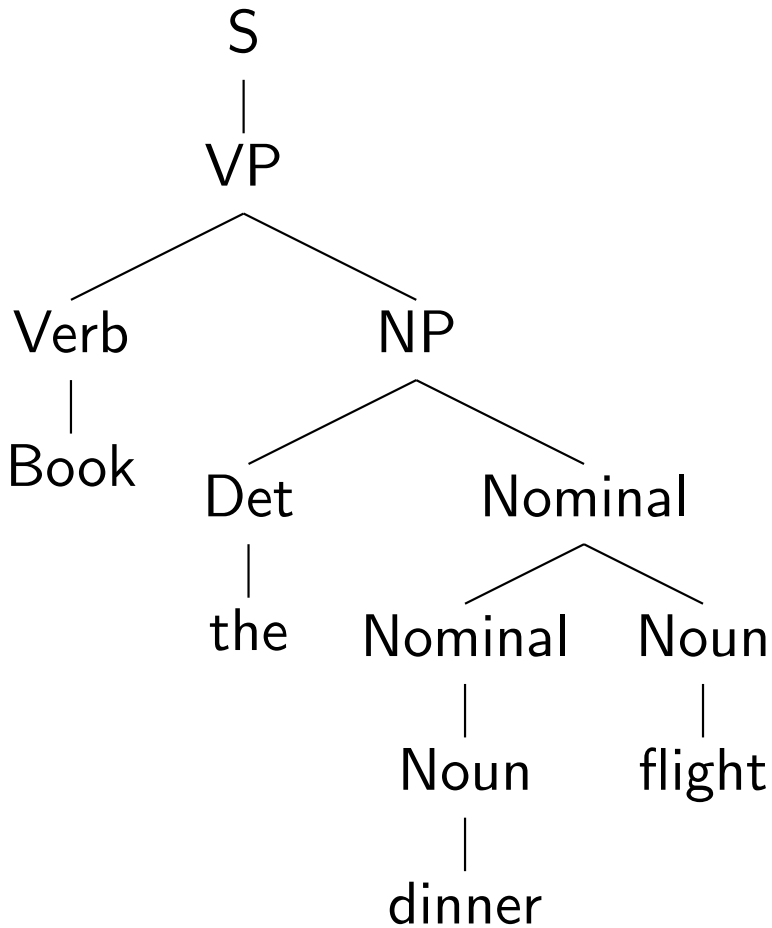


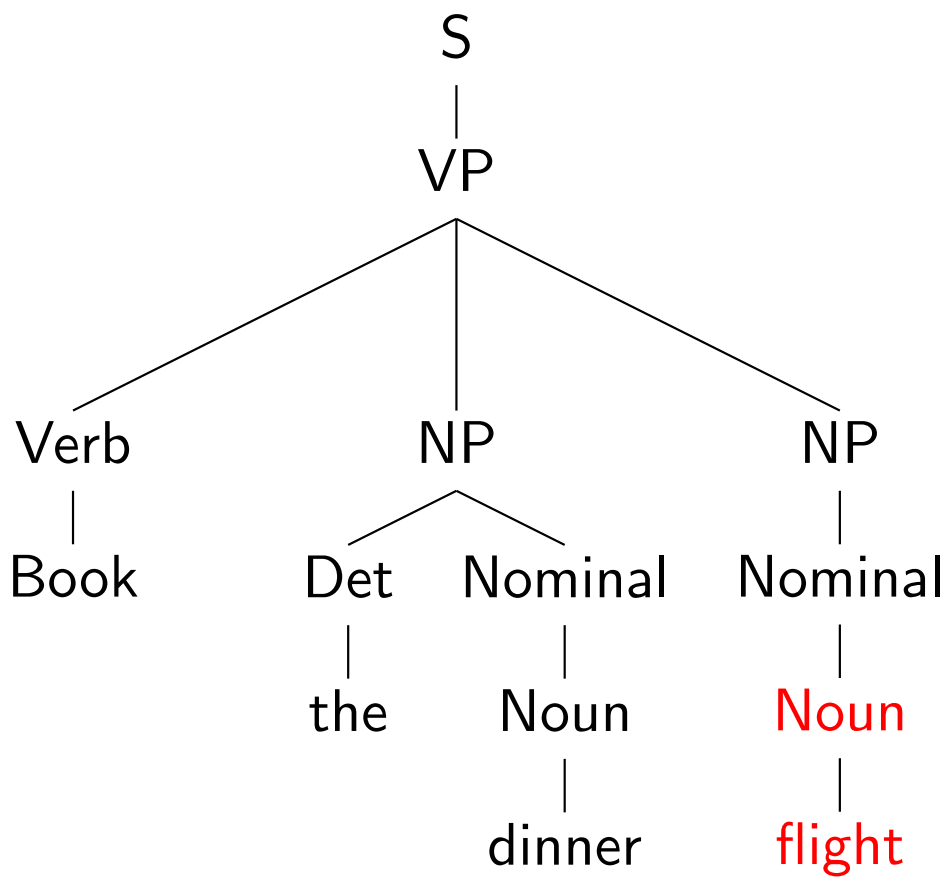


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation

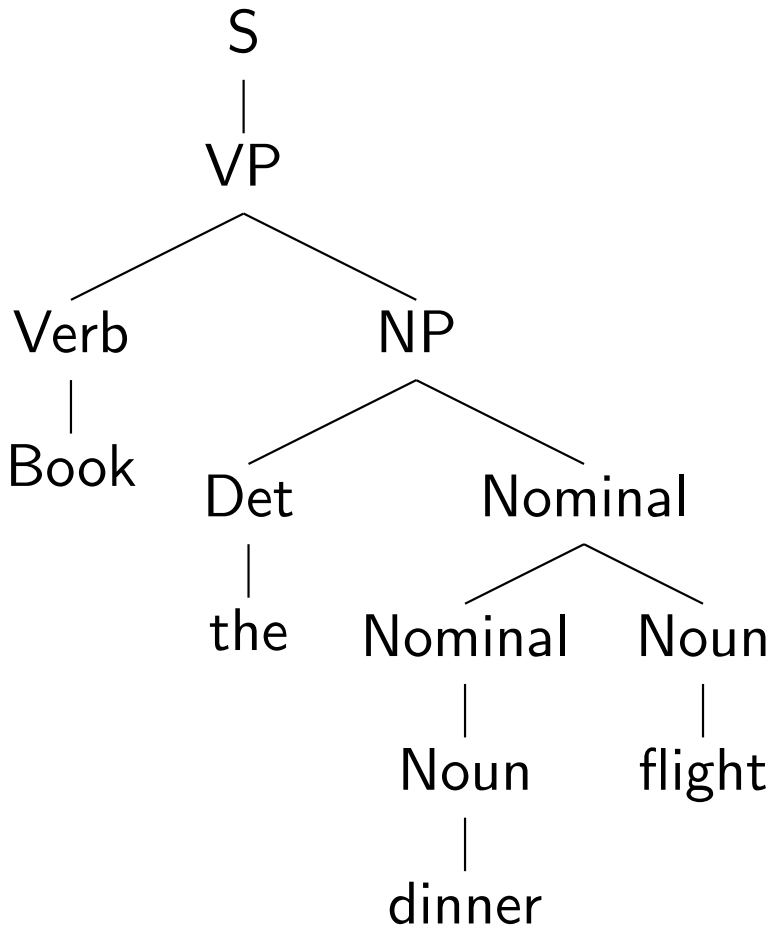


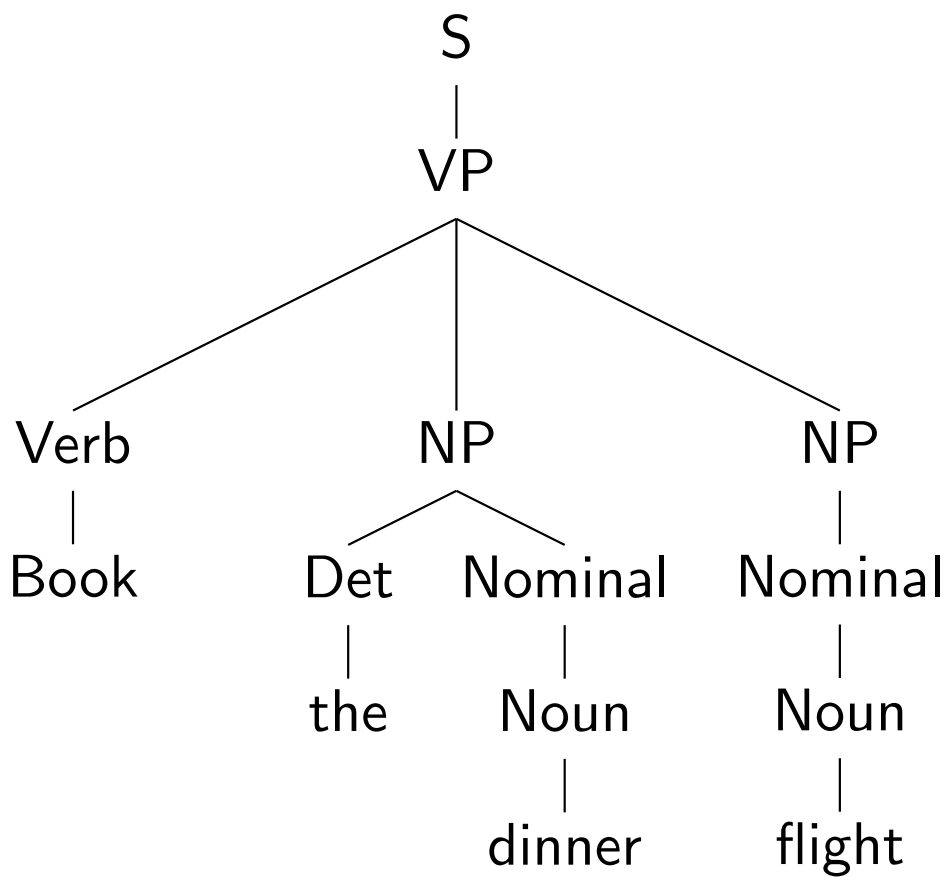


$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Application 1: Disambiguation





$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$

$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$

Notice that shared parts have same probability!

Application 2: Language Modelling

As well as assigning probabilities to parse trees, a PCFG assigns a probability to every sentence generated by the grammar. This is useful for language modelling.

The probability of a sentence is the sum of the probabilities of each parse tree associated with the sentence:

$$P(S) = \sum_{Ts.t.yield(T)=S} P(T, S)$$

$$P(S) = \sum_{s.t.yield(T)=S} P(T)$$

When is it useful to know the probability of a sentence?
When ranking the output of speech recognition, machine translation, and error correction systems.

Probabilistic Parsing

- With HMMs, we were interested in the most likely sequence of tags given the sentence
- With PCFGs, we are interested in the most likely *derivation* given the sentence:

$$\arg\max_T P(T \mid S) = \arg\max_T \frac{P(T,S)}{P(S)} = \arg\max_T P(T,S)$$

- Therefore, due to Bayes' rule, we need to find the most likely derivation for a sentence, without having to compute the probability of a sentence (summing over all derivations)

Probabilistic CYK

Many probabilistic parsers use a probabilistic version of the CYK bottom-up chart parsing algorithm.

Sentence S of length n and CFG grammar with V non-terminals

Ordinary CYK

2-$d(n+1) * (n+1)$ array where a value in cell (i, j) is list of non-terminals spanning position i through j in S.

Probabilistic CYK

3-$d(n+1) * (n+1) * V$ array where a value in cell (i, j, K) is probability of non-terminal K spanning position i through j in S

As with regular CYK, probabilistic CYK assumes that the grammar is in Chomsky-normal form (rules $A \rightarrow B\ C$ or $A \rightarrow w$).

Recursive description of probabilistic CYK

Call $\mathrm{Chart}[A, i, j]$ the probability of the highest-probability derivation of $w_{i+1} \ldots w_j$ from A. Stated mathematically:

$$\mathrm{Chart}[A, i, i+i] = p(A \rightarrow w_{i+1})$$

$$\mathrm{Chart}[A, i, j] = \max_{\{k : i<k<j\}} \max_{\{B, C : A \rightarrow B\ C \in G\}} \mathrm{Chart}[B, i, k] \times \mathrm{Chart}[C, k, j] \times p(A \rightarrow B\ C)$$

Recursive description of probabilistic CYK

Call $\mathrm{Chart}[A, i, j]$ the probability of the highest-probability derivation of $w_{i+1} \ldots w_j$ from A. Stated mathematically:

$$\begin{aligned}\mathrm{Chart}[A, i, i+i] =& p(A \rightarrow w_{i+1}) \\ \mathrm{Chart}[A, i, j] =& \max_{\{k : i<k<j\}} \max_{\{B, C : A \rightarrow B\ C \in G\}} \\ & \mathrm{Chart}[B, i, k] \times \mathrm{Chart}[C, k, j] \times p(A \rightarrow B\ C)\end{aligned}$$

Seem familiar?

Recursive description of probabilistic CYK

Call $\mathrm{Chart}[A, i, j]$ the probability of the highest-probability derivation of $w_{i+1} \ldots w_j$ from A. Stated mathematically:

$$\begin{aligned}\mathrm{Chart}[A, i, i+i] =& p(A \to w_{i+1}) \\ \mathrm{Chart}[A, i, j] =& \max_{\{k : i<k<j\}} \max_{\{B, C : A \to B\ C \in G\}} \\ & \mathrm{Chart}[B, i, k] \times \mathrm{Chart}[C, k, j] \times p(A \to B\ C)\end{aligned}$$

Seem familiar?

Remember CYK (where $\mathrm{Chart}[A, i, j]$ was simply true or false):

$$\begin{aligned}\mathrm{Chart}[A, i, j] =& \textsc{True} \text{ iff } A \to w_{i+1} \in G \\ \mathrm{Chart}[A, i, j] =& \bigvee_{k=i+1}^{j-1} \bigvee_{A \to B\ C} \mathrm{Chart}[B, i, k] \wedge \mathrm{Chart}[C, k, j]\end{aligned}$$

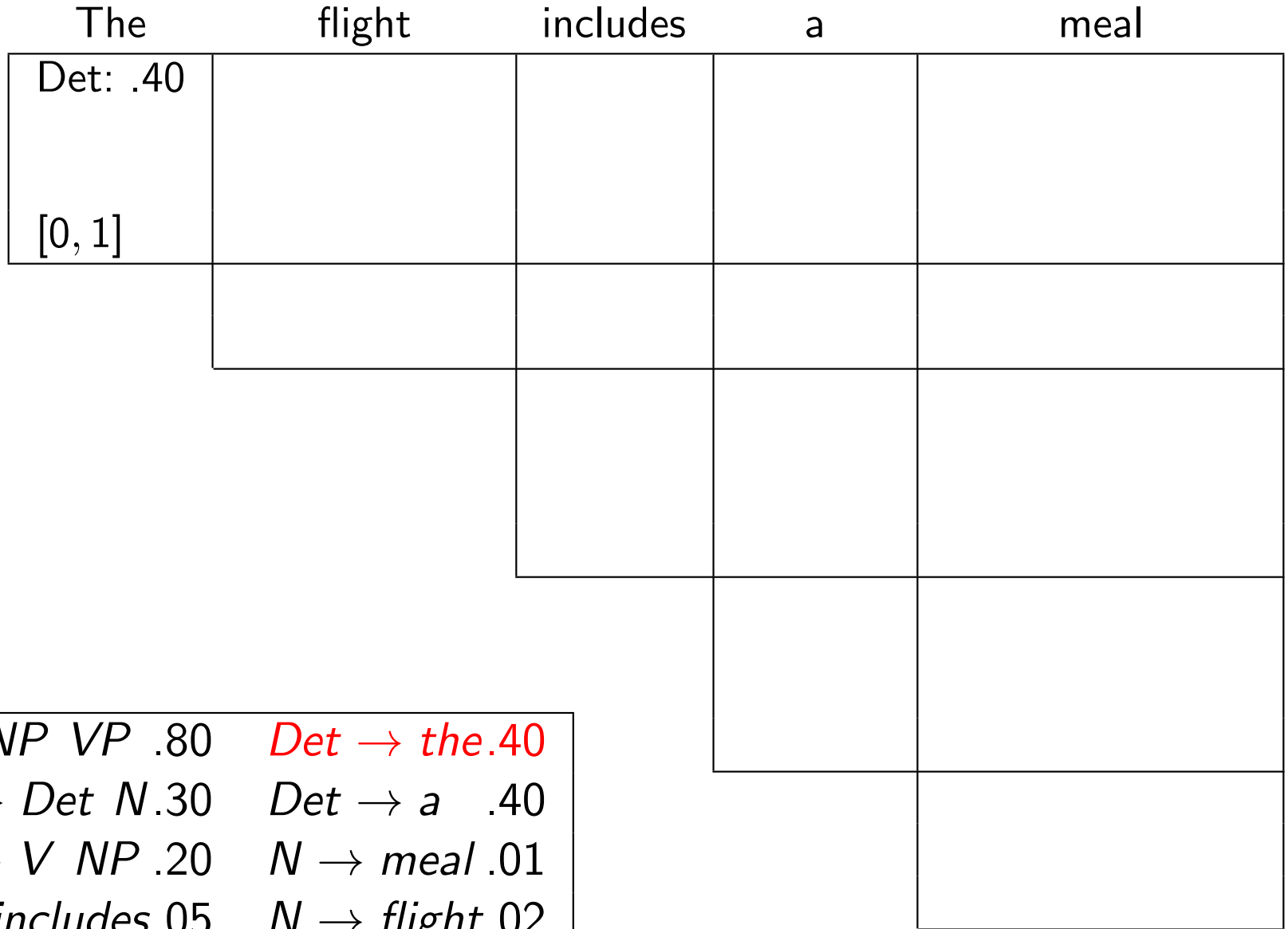
Powerful abstraction: probabilistic CYK is just CYK with a different *semiring*.

Probabilistic CYK

function Probabilistic-CYK(*words*, *grammar*) **returns** most probable parse and its probability

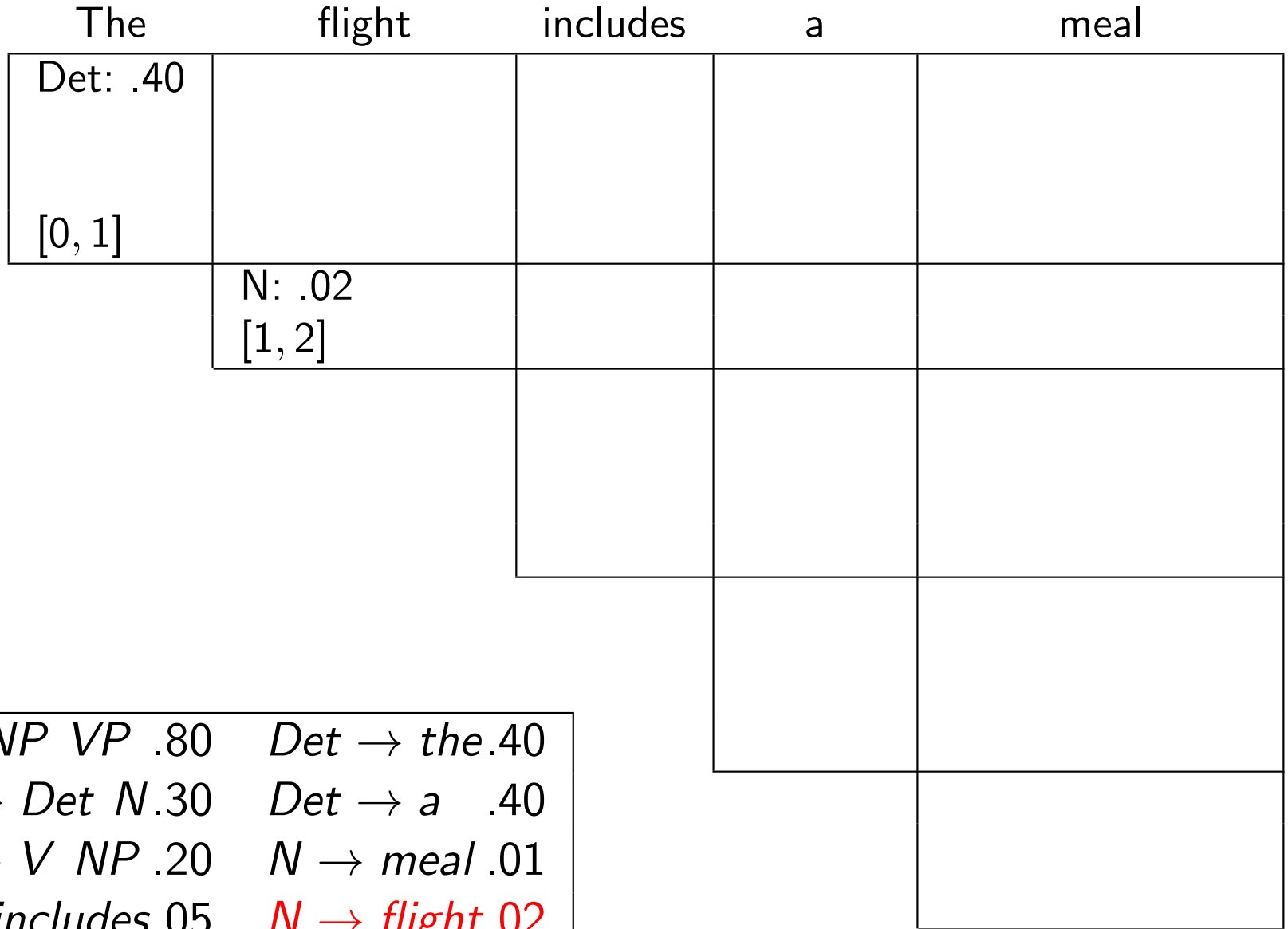
for $j \leftarrow$**from** 1 **to** LENGTH(*words*) **do**
  for all $\{A|A \rightarrow words[j] \in grammar\}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$**from** $j-2$ **downto** 0 **do**
    for all $\{A|A \rightarrow BC \in grammar,$
        and $table[i, k, B] > 0$ **and** $table[k, j, C] > 0\}$
      if$(table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C])$**then**
        $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$
        $back[i, j, A] \leftarrow \{k, B, C\}$
return
BUILD_TREE(*back*[1,LENGTH(*words*), *S*]), *table*[1,LENGTH(*words*), *S*]

Visualizing the Chart



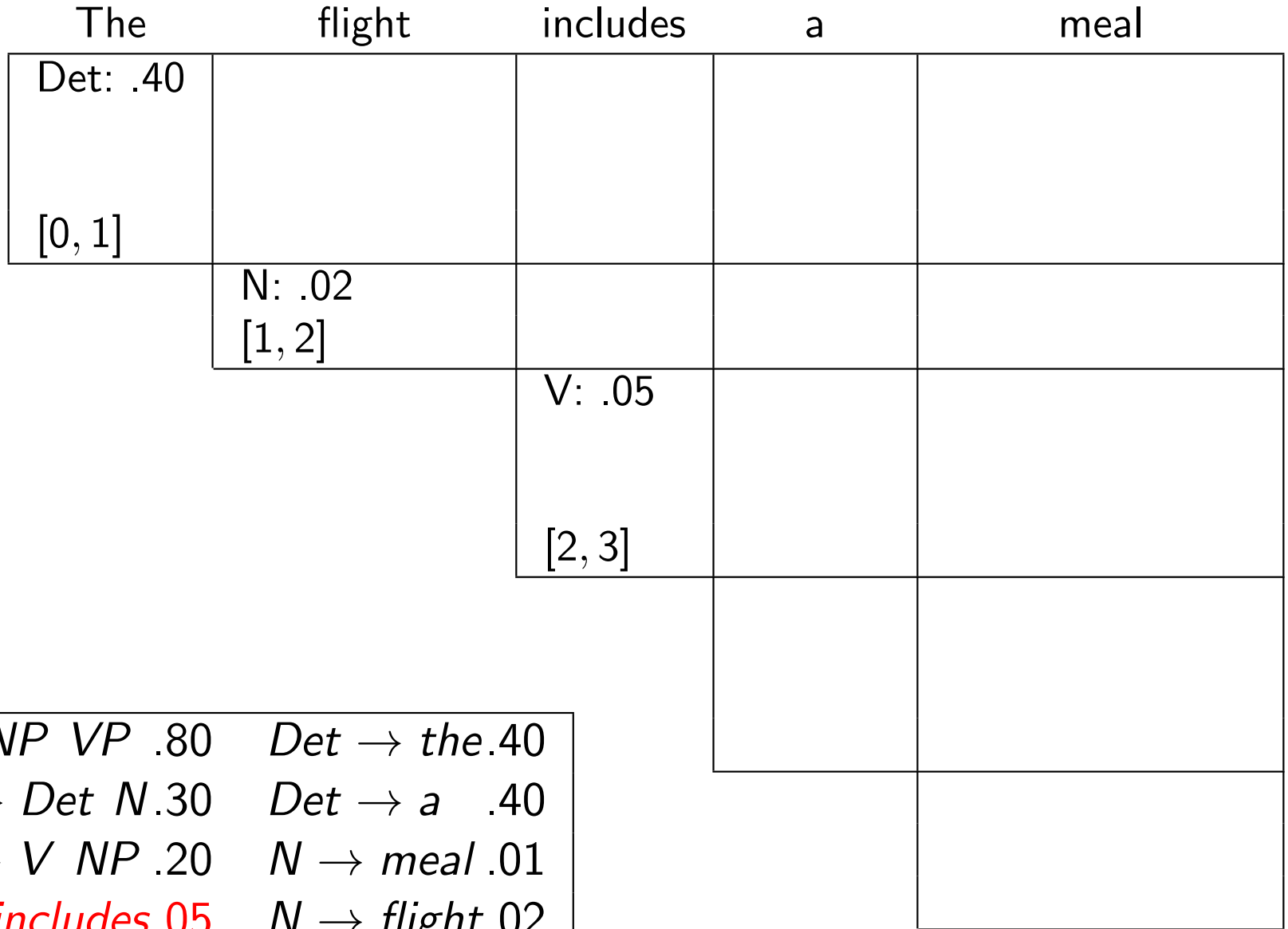
$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart



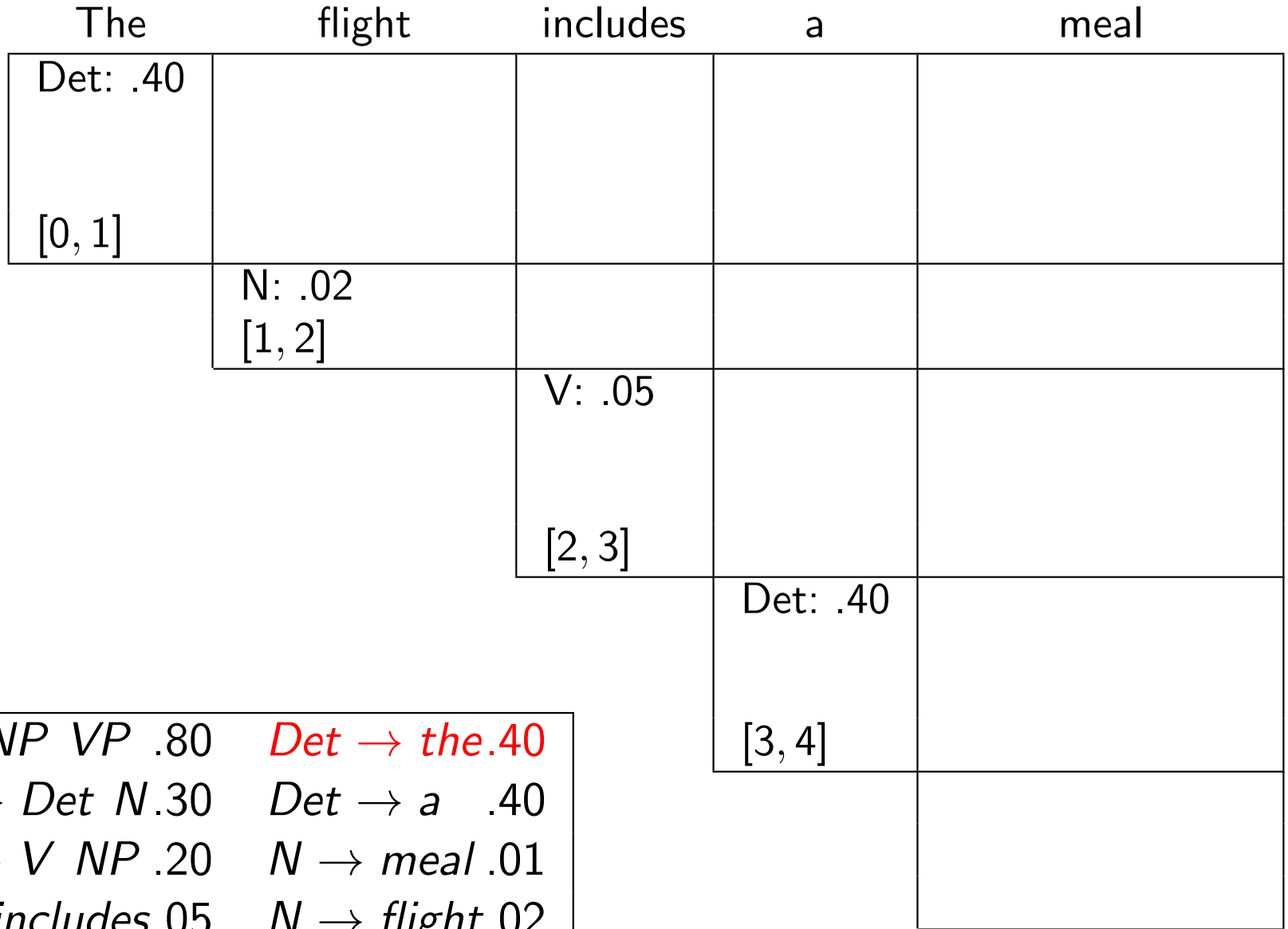
$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart



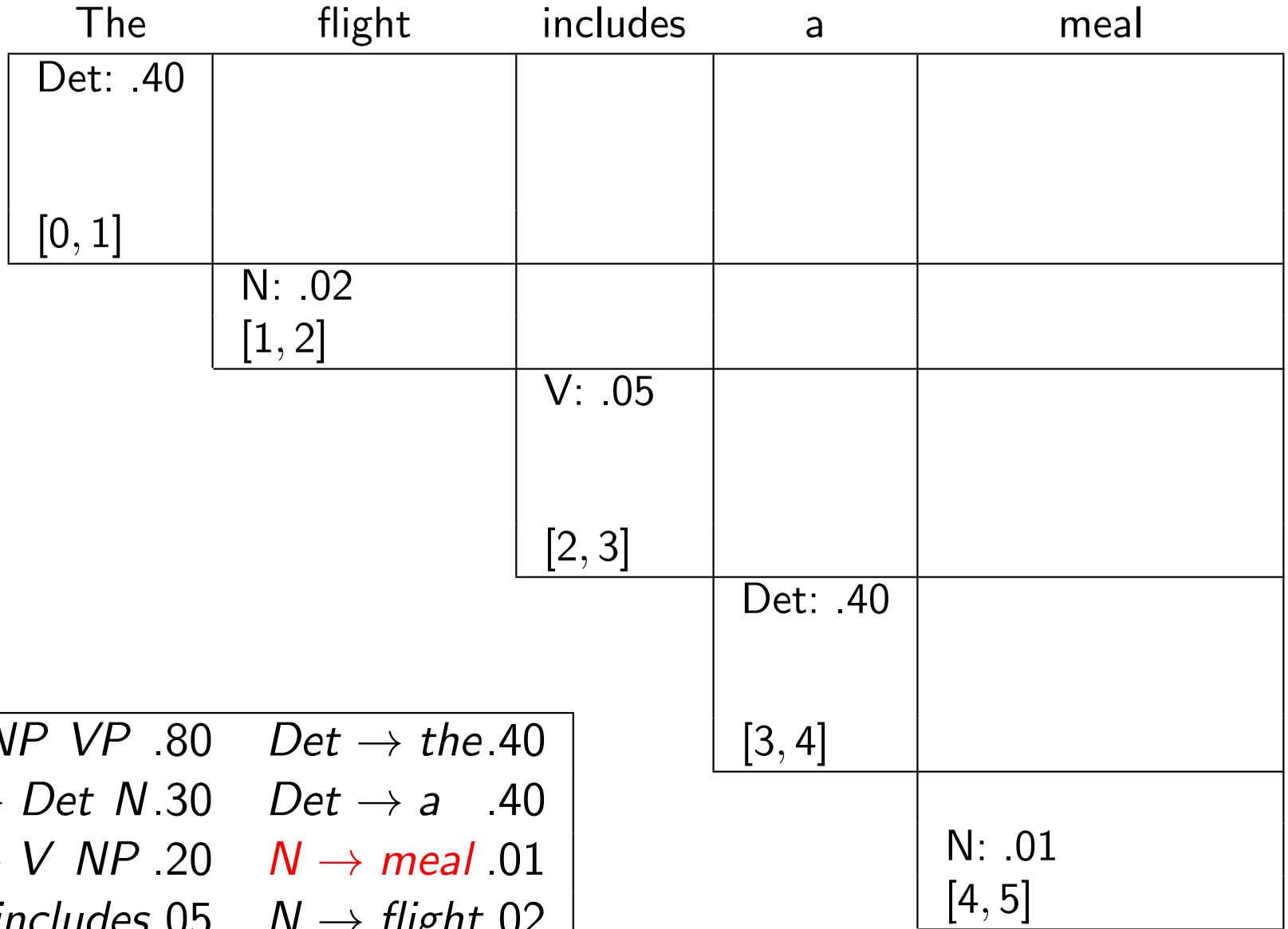
$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart



$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$.40
$NP \rightarrow Det\ N$.30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$.05	$N \rightarrow flight$.02

Visualizing the Chart



$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]			
	N: .02 [1, 2]			
		V: .05 [2, 3]		
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]			
	N: .02 [1, 2]	[1, 3]		
		V: .05 [2, 3]		
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]		
	N: .02 [1, 2]	[1, 3]		
		V: .05 [2, 3]		
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]		
	N: .02 [1, 2]	[1, 3]		
		V: .05 [2, 3]	[2, 4]	
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]		
	N: .02 [1, 2]	[1, 3]	[1, 4]	
		V: .05 [2, 3]	[2, 4]	
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]	[0, 4]	
	N: .02 [1, 2]	[1, 3]	[1, 4]	
		V: .05 [2, 3]	[2, 4]	
			Det: .40 [3, 4]	
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]	[0, 4]	
	N: .02 [1, 2]	[1, 3]	[1, 4]	
		V: .05 [2, 3]	[2, 4]	
			Det: .40 [3, 4]	NP: .30 × .40 × .01 = 0.0012 [3, 5]
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]	[0, 4]	
	N: .02 [1, 2]	[1, 3]	[1, 4]	
		V: .05 [2, 3]	[2, 4]	VP: .20 × .05 × 0.0012 = 0.000012 [2, 5]
			Det: .40 [3, 4]	NP: .30 × .40 × .01 = 0.0012 [3, 5]
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]	[0, 4]	
	N: .02 [1, 2]	[1, 3]	[1, 4]	[1, 5]
		V: .05 [2, 3]	[2, 4]	VP: .20 × .05 × 0.0012 = 0.000012 [2, 5]
			Det: .40 [3, 4]	NP: .30 × .40 × .01 = 0.0012 [3, 5]
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Visualizing the Chart

The	flight	includes	a	meal
Det: .40 [0, 1]	NP: .30×.40 × .02 = .0024 [0, 2]	[0, 3]	[0, 4]	S: .80 × .0024 × .000012 = .000000023 [0, 5]
	N: .02 [1, 2]	[1, 3]	[1, 4]	[1, 5]
		V: .05 [2, 3]	[2, 4]	VP: .20 × .05 × 0.0012 = 0.000012 [2, 5]
			Det: .40 [3, 4]	NP: .30 × .40 × .01 = 0.0012 [3, 5]
				N: .01 [4, 5]

$S \rightarrow NP\ VP$ .80	$Det \rightarrow the$ .40
$NP \rightarrow Det\ N$ .30	$Det \rightarrow a$ .40
$VP \rightarrow V\ NP$ .20	$N \rightarrow meal$ .01
$V \rightarrow includes$ .05	$N \rightarrow flight$ .02

Probabilistic CYK: more tricky example

S	$\rightarrow$	NP VP (1.0)
NP	$\rightarrow$	N (0.6) | A NP (0.2) | NP N (0.2)
VP	$\rightarrow$	V (0.8) | V Adv (0.2)
N	$\rightarrow$	orange (0.3) | tree (0.5) | blossoms (0.2)
A	$\rightarrow$	orange (1.0)
V	$\rightarrow$	blossoms (1.0)
Adv	$\rightarrow$	early (1.0)

(Not quite in CNF, but never mind.) We'll parse:

orange tree blossoms early

The probabilistic CYK-style chart

	orange	tree	blossoms	early
orange	N (0.3) A (1.0) NP (0.18)	NP (0.06)	S (0.048) NP (0.0024)	S (0.012)
tree		N (0.5) NP (0.3)	NP (0.012)	S (0.06)
blossoms			N (0.2) V (1.0) NP (0.12) VP (0.8)	VP (0.2)
early				Adv(1.0)

The probabilistic CYK-style chart: some comments

- The phrase orange tree gets 0.06 for its best analysis *as an NP*, since

  $$\begin{aligned} 0.06 &= 0.2*1.0*0.3 \quad (\text{for NP} \rightarrow \text{A NP}) \\ \text{beats } 0.018 &= 0.18*0.5*0.2 \quad (\text{for NP} \rightarrow \text{NP N}). \end{aligned}$$

  Only the higher probability is recorded in the chart.
- For orange tree blossoms, there are now two analyses as NP, each with probability 0.0024.
- There is also an analysis of orange tree blossoms as S. This doesn't compete with its analysis as NP, so both are recorded.

How to turn Earley into Probabilistic Earley?

- Each item carries a probability as a weight.
- Maintain backpointers as usual.
- When scanning a word in an RHS with a lexical rule, multiply in the rule probability into the item.
- When completing a constituent, multiply in the rule probability into the complete item.
- If you ever need to "re-create" an item that was already created, replace its backpointers with the new ones only if the new re-created item has a larger probability.

Earley parsing can also be represented using dynamic programming equations.

Summary

- A PCFG is a CFG with each rule annotated with a probability;
- the sum of the probabilities of all rules that expand the same non-terminal must be 1;
- probability of a parse tree is the product of the probabilities of all the rules used in this parse;
- probability of sentence is sum of probabilities of all its parses;
- applications for PCFGs: disambiguation, language modeling;
- Probabilistic CYK algorithm.

Next lecture: But where do the rule probabilities come from?