What is “sentence processing”?

Sentence processing is the study of how humans comprehend and produce sentences (and words within sentences, and sequences of sentences, etc.) in real time.
Realistic models of human sentence processing must account for:

- Robustness to arbitrary input
- Accurate disambiguation
- Inference on basis of incomplete input (Tanenhaus et al., 1995; Altmann and Kamide, 1999; Kaiser and Trueswell, 2004)
- Processing difficulty is *differential* and *localized*
Robustness

Real linguistic input is not always totally well-formed...

I think when she finally came to the realization that, you know, no, I can not, I can not take care of myself.

... I mean, for somebody who is, you know, for most of their life has, has, uh, not just merely had a farm but had ten children had a farm, ran everything because her husband was away in the coal mines. And, you know, facing that situation, it’s, it’s quite a dilemma.

... but usually we come to understand it pretty well anyway.
Robustness

Real linguistic input is not always totally well-formed...

I think when she finally came to the realization that, you know, no, I can not, I can not take care of myself.

... I mean, for somebody who is, you know, for most of their life has, has, uh, not just merely had a farm but had ten children had a farm, ran everything because her husband was away in the coal mines.
And, you know, facing that situation, it’s, it’s quite a dilemma.
(The woman is facing being put in a resting home.)

...but usually we come to understand it pretty well anyway.
Robustness

Real linguistic input is not always totally well-formed. . .

I think when she finally came to the realization that, you know, no, I can not, I can not take care of myself.

. . .

I mean, for somebody who is, you know, for most of their life has, has, uh, not just merely had a farm but had ten children had a farm, ran everything because her husband was away in the coal mines.
And, you know, facing that situation, it’s, it’s quite a dilemma.
(The woman is facing being put in a resting home.)

. . . but usually we come to understand it pretty well anyway.
Accurate disambiguation

Most sentences are ambiguous in ways we do not even notice:

Mary forgot the pitcher...
Accurate disambiguation

Most sentences are ambiguous in ways we do not even notice:

Mary forgot the pitcher...
Accurate disambiguation

Most sentences are ambiguous in ways we do not even notice:

*Mary forgot the pitcher of water sitting near the stove.*
Accurate disambiguation

Most sentences are ambiguous in ways we do not even notice:

Mary forgot the pitcher of water sitting near the stove.
Accurate disambiguation

Most sentences are ambiguous in ways we do not even notice:

Mary forgot the pitcher of water sitting near the stove.

That’s probably not what you were thinking of...
Inference on the basis of incomplete input

Comprehenders do not wait until the whole sentence has been heard to make inferences about what it means or will wind up meaning:

(Altmann and Kamide, 1999)
Inference on the basis of incomplete input

Comprehenders do not wait until the whole sentence has been heard to make inferences about what it means or will wind up meaning:

(Altmann and Kamide, 1999)
Inference on the basis of incomplete input

Comprehenders do not wait until the whole sentence has been heard to make inferences about what it means or will wind up meaning:

“The boy will eat/move the cake…”

(Altmann and Kamide, 1999)
Inference on the basis of incomplete input

Comprehenders do not wait until the whole sentence has been heard to make inferences about what it means or will wind up meaning:

“The boy will eat/move the cake…”

(Altmann and Kamide, 1999)
Inference on the basis of incomplete input

Comprehenders do not wait until the whole sentence has been heard to make inferences about what it means or will wind up meaning:

“The boy will *eat/move* the cake…”

That is, comprehension is *incremental*

*(Altmann and Kamide, 1999)*
Using multiple relative clauses in a sentence can make processing difficult:

This is the malt that the rat that the cat that the dog worried killed ate.

It’s not the meaning of the sentence, or the use of relative clauses, that makes it hard:

This is the malt that was eaten by the rat that was killed by the cat that was worried by the dog.
Processing difficulty is differential

Using multiple relative clauses in a sentence can make processing difficult:

This is the malt that the rat that the cat that the dog worried killed ate.

It’s not the meaning of the sentence, or the use of relative clauses, that makes it hard:

This is the malt that was eaten by the rat that was killed by the cat that was worried by the dog.
Processing difficulty is differential

Using multiple relative clauses in a sentence can make processing difficult:

This is the malt that the rat that the cat that the dog worried killed ate.

It’s not the meaning of the sentence, or the use of relative clauses, that makes it hard:

This is the malt that was eaten by the rat that was killed by the cat that was worried by the dog.
Processing difficulty is localized

[Self-paced reading demo, Example1]

(Grodner and Gibson, 2005)
Processing difficulty is localized

[Self-paced reading demo, Example1]

Word-by-word reading times for sentences with different types of relative clauses (RCs)

(Grodner and Gibson, 2005)
Goal of modeling sentence processing

- We need to account for these properties (robustness, disambiguation, inference from incomplete input, and differential difficulty) together
- Probabilistic models are an interesting way to do this
- You are probably already convinced about robustness, disambiguation, and inference!
- I’ll talk largely about some interesting tie-ins with differential difficulty
Goal of modeling sentence processing

- We need to account for these properties (robustness, disambiguation, inference from incomplete input, and differential difficulty) together
- Probabilistic models are an interesting way to do this
  - You are probably already convinced about robustness, disambiguation, and inference!
  - I’ll talk largely about some interesting tie-ins with differential difficulty
Goal of modeling sentence processing

- We need to account for these properties (robustness, disambiguation, inference from incomplete input, and differential difficulty) together
- Probabilistic models are an interesting way to do this
- You are probably already convinced about robustness, disambiguation, and inference!
- I’ll talk largely about some interesting tie-ins with differential difficulty
Goal of modeling sentence processing

- We need to account for these properties (robustness, disambiguation, inference from incomplete input, and differential difficulty) together
- Probabilistic models are an interesting way to do this
- You are probably already convinced about robustness, disambiguation, and inference!
- I’ll talk largely about some interesting tie-ins with differential difficulty
Try to guess the next word in the sentence

Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)

Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to...

- Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
- Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to... chat? get warm? talk? eat? rest?

► Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
► Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to... chat? get warm? talk? eat? rest?
The children went outside to...

- Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
- Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to... chat? get warm? talk? eat? rest?
The children went outside to... play

> Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
> Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to... chat? get warm? talk? eat? rest?
The children went outside to... play

- Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
- Why would this be the case?
Try to guess the next word in the sentence

My brother came inside to... chat? get warm? talk? eat? rest?
The children went outside to... play

- Empirically, it’s been shown that more highly predictable words are read more quickly (Ehrlich and Rayner, 1981)
- Why would this be the case?
Surprisal as a possible metric for processing difficulty

An event’s surprisal is simply its negative log conditional probability

$$\log \frac{1}{P(x|\text{Context})}$$

Intuitively, this is a measure of the amount of information contained in the event
Four proposals for surprisal as a measure of processing difficulty/time:

- Surprisal of a word as primitive measure of processing (Mandelbrot, 1953; Attneave, 1959; Hale, 2001)
- Kullback-Leibler divergence (*relative entropy*) as size of update that the word induces for distribution over interpretations of input (Levy, 2005, 2008)
  - independently proposed as a measure of surprise in visual scene perception (Itti and Baldi, 2005)
- Surprisal as optimal perceptual discrimination (Norris, 2006)
- Surprisal as an optimal solution to the speed/resource tradeoff in language comprehension (Smith, 2006)
Probabilistic grammars for estimating surprisal

- Comprehenders’ expectations about upcoming words should reflect structural distributional regularities of the language
- Hence, probabilistic grammars are a good candidate
- We’ll use *probabilistic context-free grammars* (PCFGs) as a model of language users’ grammatical knowledge
Context-free Grammars

A context-free grammar (CFG) consists of a tuple \((N, V, S, R)\) such that:

- \(N\) is a finite set of non-terminal symbols;
- \(V\) is a finite set of terminal symbols;
- \(S\) is the start symbol;
- \(R\) is a finite set of rules of the form \(X \rightarrow \alpha\) where \(X \in N\) and \(\alpha\) is a sequence of symbols drawn from \(N \cup V\).

A CFG derivation is the recursive expansion of non-terminal symbols in a string by rules in \(R\), starting with \(S\), and a derivation tree \(T\) is the history of those rule applications.
Let our grammar (the rule-set \( R \)) be

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow Det \ N \\
NP & \rightarrow NP \ PP \\
PP & \rightarrow P \ NP \\
VP & \rightarrow V \\
\end{align*}
\]

Det \rightarrow the
N \rightarrow dog
N \rightarrow cat
P \rightarrow near
V \rightarrow growled

The nonterminal set \( N \) is \{ \( S \), \( NP \), \( VP \), \( Det \), \( N \), \( P \), \( V \) \}, the terminal set \( V \) is \{ \text{the, dog, cat, near, growled} \}, and our start symbol \( S \) is \( S \).
Context-free Grammars: an example II

\[ S \rightarrow \text{NP } \text{VP} \]
\[ \text{NP} \rightarrow \text{Det } \text{N} \]
\[ \text{NP} \rightarrow \text{NP } \text{PP} \]
\[ \text{PP} \rightarrow \text{P } \text{NP} \]
\[ \text{VP} \rightarrow \text{V} \]

\[ \text{Det} \rightarrow \text{the} \]
\[ \text{N} \rightarrow \text{dog} \]
\[ \text{N} \rightarrow \text{cat} \]
\[ \text{P} \rightarrow \text{near} \]
\[ \text{V} \rightarrow \text{growled} \]

Here is a *derivation* and the resulting *derivation tree*:
Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
NP $\rightarrow$ NP PP
PP $\rightarrow$ P NP
VP $\rightarrow$ V

Det $\rightarrow$ the
N $\rightarrow$ dog
N $\rightarrow$ cat
P $\rightarrow$ near
V $\rightarrow$ growled

Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow Det \ N \\
NP & \rightarrow NP \ PP \\
PP & \rightarrow P \ NP \\
VP & \rightarrow V
\end{align*}
\]

Det \rightarrow the \\
N \rightarrow dog \\
N \rightarrow cat \\
P \rightarrow near \\
V \rightarrow growled

Here is a *derivation* and the resulting *derivation tree*:

```
S
  / \   /
NP  VP
 /   / /
NP PP
 /   / /
Det N  VP
 |   | /   /
 the dog P NP
   |   |   /
   near Det N
     |   |   /
     the cat
growled
```

Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:

```
S
  NP
    Det
      the
    N
      dog
  PP
    P
      near
    NP
      Det
        the
      N
        cat
  VP
    V
      growled
```
Context-free Grammars: an example II

S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
NP $\rightarrow$ NP PP
PP $\rightarrow$ P NP
VP $\rightarrow$ V

Det $\rightarrow$ the
N $\rightarrow$ dog
N $\rightarrow$ cat
P $\rightarrow$ near
V $\rightarrow$ growled

Here is a derivation and the resulting derivation tree:

```
S
 NP
  NP
    Det N P NP
      the dog near
      Det N
        the cat
  PP
    V
    growled
```
Context-free Grammars: an example II

\[
\begin{align*}
S & \rightarrow \text{NP } \text{VP} \\
\text{NP} & \rightarrow \text{Det } \text{N} \\
\text{NP} & \rightarrow \text{NP } \text{PP} \\
\text{PP} & \rightarrow \text{P } \text{NP} \\
\text{VP} & \rightarrow \text{V} \\
\text{Det} & \rightarrow \text{the} \\
\text{N} & \rightarrow \text{dog} \\
\text{N} & \rightarrow \text{cat} \\
\text{P} & \rightarrow \text{near} \\
\text{V} & \rightarrow \text{growled}
\end{align*}
\]

Here is a derivation and the resulting derivation tree:

\[
\begin{align*}
\text{S} & \rightarrow \text{NP } \text{VP} \\
\text{NP} & \rightarrow \text{Det } \text{N} \\
\text{NP} & \rightarrow \text{NP } \text{PP} \\
\text{PP} & \rightarrow \text{P } \text{NP} \\
\text{VP} & \rightarrow \text{V} \\
\text{Det} & \rightarrow \text{the} \\
\text{N} & \rightarrow \text{dog} \\
\text{N} & \rightarrow \text{cat} \\
\text{P} & \rightarrow \text{near} \\
\text{V} & \rightarrow \text{growled}
\end{align*}
\]
Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:
Here is a derivation and the resulting derivation tree:
Context-free Grammars: an example II

S → NP VP
NP → Det N
NP → NP PP
PP → P NP
VP → V

Det → the
N → dog
N → cat
P → near
V → growled

Here is a derivation and the resulting derivation tree:
A probabilistic context-free grammar (PCFG) consists of a tuple \((N, V, S, R, P)\) such that:

- \(N\) is a finite set of non-terminal symbols;
- \(V\) is a finite set of terminal symbols;
- \(S\) is the start symbol;
- \(R\) is a finite set of rules of the form \(X \rightarrow \alpha\) where \(X \in N\) and \(\alpha\) is a sequence of symbols drawn from \(N \cup V\);
- \(P\) is a mapping from \(R\) into probabilities, such that for each \(X \in N\),

\[
\sum_{[X \rightarrow \alpha] \in R} P(X \rightarrow \alpha) = 1
\]

PCFG derivations and derivation trees are just like for CFGs. The probability \(P(T)\) of a derivation tree is simply the product of the probabilities of each rule application.
Example PCFG

1 S → NP VP
0.8 NP → Det N
0.2 NP → NP PP
1 PP → P NP
1 VP → V

1 Det → the
0.5 N → dog
0.5 N → cat
1 P → near
1 V → growled

P(T) = 1 \times 0.2 \times 0.8 \times 1 \times 0.5 \times 0.8 \times 1 \times 0.8 \times 1 \times 0.5 \times 1 \times 1
= 0.032
We just learned how to calculate the *probability of a tree*

- The *probability of a string* $w_1 \cdots n$ is the sum of the probabilities of all trees whose yield is $w_1 \cdots n$
- The *probability of a string prefix* $w_1 \cdots i$ is the sum of the probabilities of all trees whose yield begins with $w_1 \cdots i$
- If we had the probabilities of two string prefixes $w_1 \cdots i-1$ and $w_1 \cdots i$, we could calculate the conditional probability $P(w_i \mid w_1 \cdots i-1)$ as their ratio:

\[
P(w_i \mid w_1 \cdots i-1) = \frac{P(w_1 \cdots i)}{P(w_1 \cdots i-1)}
\]
Inference over infinite tree sets

Consider the following noun-phrase grammar:

\[
\begin{align*}
\text{NP} & \rightarrow \text{Det N} \\
\text{NP} & \rightarrow \text{NP PP} \\
\text{PP} & \rightarrow \text{P NP} \\
\text{Det} & \rightarrow \text{the} \\
\text{N} & \rightarrow \text{dog} \\
\text{N} & \rightarrow \text{cat} \\
\text{P} & \rightarrow \text{near}
\end{align*}
\]

Question: given a sentence starting with the... what is the probability that the next word is dog? Intuitively, the answers to this question should be

\[P(\text{dog}|\text{the}) = \frac{2}{3}\]

because the second word HAS to be either dog or cat.
Consider the following noun-phrase grammar:

\[ \begin{align*}
2 \quad & \text{NP} \rightarrow \text{Det N} \\
3 \quad & \text{NP} \rightarrow \text{NP PP} \\
3 \quad & \text{PP} \rightarrow \text{P NP} \\
1 \quad & \text{Det} \rightarrow \text{the} \\
2 \quad & \text{N} \rightarrow \text{dog} \\
1 \quad & \text{P} \rightarrow \text{near} \\
3 \quad & \text{N} \rightarrow \text{cat}
\end{align*} \]

Question: given a sentence starting with \textit{the}… what is the probability that the next word is \textit{dog}?

Intuitively, the answers to this question should be 

\[ P(\text{dog}|\text{the}) = \frac{2}{3} \]

because the second word HAS to be either \textit{dog} or \textit{cat}. 
Consider the following noun-phrase grammar:

\[
\begin{align*}
&2 \text{ NP } \rightarrow \text{ Det N} \\
&3 \text{ NP } \rightarrow \text{ NP PP} \\
&1 \text{ PP } \rightarrow \text{ P NP} \\
&2 \text{ Det } \rightarrow \text{ the} \\
&3 \text{ N } \rightarrow \text{ dog} \\
&3 \text{ N } \rightarrow \text{ cat} \\
&1 \text{ P } \rightarrow \text{ near}
\end{align*}
\]

Question: given a sentence starting with \textit{the}…

what is the probability that the next word is \textit{dog}?

Intuitively, the answers to this question should be

\[
P(\text{dog}|\text{the}) = \frac{2}{3}
\]

because the second word HAS to be either \textit{dog} or \textit{cat}.
Consider the following noun-phrase grammar:

\[
\begin{align*}
\text{NP} & \rightarrow \text{Det N} \\
\text{NP} & \rightarrow \text{NP PP} \\
\text{PP} & \rightarrow \text{P NP} \\
\text{Det} & \rightarrow \text{the} \\
\text{N} & \rightarrow \text{dog} \\
\text{N} & \rightarrow \text{cat} \\
\text{P} & \rightarrow \text{near}
\end{align*}
\]

Question: given a sentence starting with \textit{the...}

what is the probability that the next word is \textit{dog}?

Intuitively, the answers to this question should be

\[P(\text{dog}|\text{the}) = \frac{2}{3}\]

because the second word HAS to be either \textit{dog} or \textit{cat}. 

Inference over infinite tree sets
We “should” just enumerate the trees that cover *the dog* . . . , and divide their total probability by that of *the* . . .

. . . but there are infinitely many trees.
We “should” just enumerate the trees that cover *the dog* . . . , and divide their total probability by that of *the* . . .

. . . but there are infinitely many trees.
Inference over infinite tree sets (2)

We “should” just enumerate the trees that cover *the dog* . . . , and divide their total probability by that of *the* . . .

. . . but there are infinitely many trees.
We “should” just enumerate the trees that cover the dog . . . , and divide their total probability by that of the . . .

. . . but there are infinitely many trees.
Shortcut 1: you can think of a *partial* tree as marginalizing over all completions of the partial tree.
It has a corresponding marginal probability in the PCFG.
Shortcut 1: you can think of a *partial* tree as marginalizing over all completions of the partial tree. It has a corresponding marginal probability in the PCFG.
Shortcut 1: you can think of a *partial* tree as marginalizing over all completions of the partial tree. It has a corresponding marginal probability in the PCFG.
Problem 2: there are still an infinite number of incomplete trees covering a partial input.

\[
P(\text{the dog} \ldots) = \frac{4}{9} + \frac{4}{27} + \frac{4}{81} + \frac{4}{243} + \cdots
\]

\[
= \frac{4}{9} \prod_{i=0}^{\infty} \left(\frac{1}{3}\right)^i
\]

\[
= \frac{2}{3}
\]

... which matches the original rule probability

\[
\frac{2}{3} \ N \rightarrow \text{dog}
\]
Problem 2: there are still an infinite number of incomplete trees covering a partial input.

\[
\begin{array}{cccccccc}
\text{NP} & \text{NP} & \text{NP} & \text{NP} & \text{NP} & \text{NP} & \text{NP} \\
\text{Det} & \text{N} & \text{NP} & \text{PP} & \text{Det} & \text{N} & \text{PP} \\
\text{the dog} & \text{the dog} \\
\frac{4}{9} & \frac{4}{27} \\
\end{array}
\]

BUT! These tree probabilities form a geometric series:

\[
P(\text{the dog . . .}) = \frac{4}{9} + \frac{4}{27} + \frac{4}{81} + \frac{4}{243} + \cdots
\]

\[
= \frac{4}{9} \prod_{i=0}^{\infty} \left(\frac{1}{3}\right)^i
\]

\[
= \frac{2}{3}
\]

… which matches the original rule probability

\[
\frac{2}{3} N \rightarrow \text{dog}
\]
Problem 2: there are still an infinite number of incomplete trees covering a partial input.

\[ \frac{4}{9} \quad \frac{4}{27} \quad \frac{4}{81} \quad \frac{4}{243} \ldots \]

BUT! These tree probabilities form a geometric series:

\[
P(\text{the dog} \ldots) = \frac{4}{9} + \frac{4}{27} + \frac{4}{81} + \frac{4}{243} + \cdots
\]

\[
= \frac{4}{9} \prod_{i=0}^{\infty} \left( \frac{1}{3} \right)^i
\]

\[
= \frac{2}{3}
\]

\[
\ldots \text{which matches the original rule probability}
\]

\[
\frac{2}{3} \ N \rightarrow \ dog
\]
Problem 2: there are still an infinite number of incomplete trees covering a partial input.

\[\begin{align*}
\text{NP} & \quad \text{NP} \\
\text{Det} & \quad \text{N} \\
\text{the dog} & \quad \text{the dog} \\
\frac{4}{9} & \quad \frac{4}{27} \\
\end{align*}\]

BUT! These tree probabilities form a geometric series:

\[P(\text{the dog . . .}) = \frac{4}{9} + \frac{4}{27} + \frac{4}{81} + \frac{4}{243} + \cdots = \frac{4}{9} \prod_{i=0}^{\infty} \left(\frac{1}{3}\right)^i = \frac{2}{3}\]

\[\frac{2}{3} \ N \rightarrow \text{dog}\]
Problem 2: there are still an infinite number of incomplete trees covering a partial input.

\[
\begin{align*}
\text{NP} & \quad \text{NP} \\
\text{Det} & \quad \text{N} & \quad \text{Det} & \quad \text{N} & \quad \text{Det} & \quad \text{N} \\
\text{.} & \quad \text{the dog} & \quad \text{.} & \quad \text{the dog} & \quad \text{.} & \quad \text{the dog} \\
\frac{4}{9} & \quad \frac{4}{27} & \quad \frac{4}{81} & \quad \frac{4}{243}
\end{align*}
\]

BUT! These tree probabilities form a geometric series:

\[
P(\text{the dog . . .}) = \frac{4}{9} + \frac{4}{27} + \frac{4}{81} + \frac{4}{243} + \cdots
\]

\[
= \frac{4}{9} \prod_{i=0}^{\infty} \left( \frac{1}{3} \right)^i
\]

\[
= \frac{2}{3}
\]

\[
\frac{2}{3} N \rightarrow \text{dog}
\]

\[
\]
Generalizing the geometric series induced by rule recursion

In general, these infinite tree sets arise due to \textit{left recursion} in a probabilistic grammar

\[
A \rightarrow B \alpha \\
B \rightarrow A \beta
\]

We can formulate a stochastic \textit{left-corner matrix} of transitions between categories:

\[
P_L = \begin{bmatrix}
A & 0.3 & 0.7 & \cdots & 0 \\
B & 0.1 & 0.1 & \cdots & 0.2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
K & 0.2 & 0.1 & \cdots & 0.2
\end{bmatrix}
\]

and solve for its closure \( R_L = (I - P_L)^{-1} \).

\textit{(Stolcke, 1995)}
Generalizing the geometric series induced by rule recursion

In general, these infinite tree sets arise due to *left recursion* in a probabilistic grammar

\[ A \rightarrow B \alpha \quad B \rightarrow A \beta \]

We can formulate a stochastic *left-corner matrix* of transitions between categories:

\[
P_L = \begin{bmatrix}
A & B & \ldots & K \\
A & 0.3 & 0.7 & \ldots & 0 \\
B & 0.1 & 0.1 & \ldots & 0.2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
K & 0.2 & 0.1 & \ldots & 0.2
\end{bmatrix}
\]

and solve for its closure \( R_L = (I - P_L)^{-1} \).

*(Stolcke, 1995)*
Generalizing the geometric series induced by rule recursion

In general, these infinite tree sets arise due to *left recursion* in a probabilistic grammar

$$A \rightarrow B \alpha \qquad B \rightarrow A \beta$$

We can formulate a stochastic *left-corner matrix* of transitions between categories:

$\begin{pmatrix}
A & B & \ldots & K \\
A & 0.3 & 0.7 & \ldots & 0 \\
B & 0.1 & 0.1 & \ldots & 0.2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
K & 0.2 & 0.1 & \ldots & 0.2
\end{pmatrix}$

and solve for its closure $R_L = (I - P_L)^{-1}$.  

*(Stolcke, 1995)*
Generalizing the geometric series

\[
\begin{align*}
\frac{2}{3} & \quad \text{NP} \rightarrow \text{Det N} & 1 & \quad \text{Det} \rightarrow \text{the} \\
\frac{1}{3} & \quad \text{NP} \rightarrow \text{NP PP} & \frac{2}{3} & \quad \text{N} \rightarrow \text{dog} \\
\frac{1}{3} & \quad \text{PP} \rightarrow \text{P NP} & \frac{1}{3} & \quad \text{N} \rightarrow \text{cat} \\
1 & \quad \text{P} \rightarrow \text{near} & 1 & \quad \text{P} \rightarrow \text{near}
\end{align*}
\]

The closure of our left-corner matrix is

\[
\begin{pmatrix}
1.5 & 0 & 1.0 & 0 & 0 \\
0 & 1.0 & 0 & 0 & 1.0 \\
0 & 0 & 1.0 & 0 & 0 \\
0 & 0 & 0 & 1.0 & 0 \\
0 & 0 & 0 & 0 & 1.0 \\
\end{pmatrix}
\]

Note that the \(\frac{3}{2}\) “bonus” accrued for left-recursion of NPs appears in the (NP,NP) cell of the matrix.
Generalizing the geometric series

\[
\begin{align*}
\frac{2}{3} & \quad \text{NP} \rightarrow \text{Det N} & 1 & \quad \text{Det} \rightarrow \text{the} \\
\frac{1}{3} & \quad \text{NP} \rightarrow \text{NP PP} & \frac{2}{3} & \quad \text{N} \rightarrow \text{dog} \\
1 & \quad \text{PP} \rightarrow \text{P NP} & \frac{1}{3} & \quad \text{N} \rightarrow \text{cat} \\
1 & \quad \text{P} \rightarrow \text{near} & 1 & \quad \text{P} \rightarrow \text{near}
\end{align*}
\]

The closure of our left-corner matrix is

\[
\begin{pmatrix}
1.5 & 0 & 1.0 & 0 & 0 \\
0 & 1.0 & 0 & 0 & 1.0 \\
0 & 0 & 1.0 & 0 & 0 \\
0 & 0 & 0 & 1.0 & 0 \\
0 & 0 & 0 & 0 & 1.0
\end{pmatrix}
\]

Note that the \(\frac{3}{2}\) “bonus” accrued for left-recursion of NPs appears in the (NP,NP) cell of the matrix.
Efficient incremental parsing: the probabilistic Earley algorithm

We can use the Earley algorithm (Earley, 1970) in a probabilistic incarnation (Stolcke, 1995) to deal with these infinite tree sets.

The (slightly oversimplified) probabilistic Earley algorithm has two fundamental types of operations:

- **Prediction**: if $X$ is a possible goal, choose a rule $X \rightarrow Y_\alpha$ and set up $Y_\alpha$ as a new sequence of possible sub-goals of $X$.
- **Completion**: if $X$ is a possible goal and we encounter a completed $X$, absorb it and move on to the next sub-goal in the sequence.
Efficient incremental parsing: the probabilistic Earley algorithm

- Parsing consists of constructing a chart of states (items)
- A state has the following structure:

$$X \rightarrow \alpha \circ \beta$$

- The forward probability is the total probability of getting from the root at the start of the sentence through to this state.
- The inside probability is the “bottom-up” probability of the state.
Efficient incremental parsing: the probabilistic Earley algorithm

<table>
<thead>
<tr>
<th>NP → NP ( \circ ) PP</th>
<th>NP → Det ( \circ ) N</th>
<th>PP → P ( \circ ) NP</th>
<th>NP → Det ( \circ ) N</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{2}{3} ) ( \frac{1}{3} ) ( \frac{2}{9} )</td>
<td>( \frac{2}{3} ) ( \frac{4}{9} )</td>
<td>( \frac{2}{9} ) ( \frac{1}{9} ) ( \frac{2}{3} )</td>
<td>( \frac{2}{6} ) ( \frac{1}{3} )</td>
</tr>
</tbody>
</table>
Efficient incremental parsing: the probabilistic Earley algorithm

Det → the
1 1

NP → NP ⊕ PP
\[ \frac{2}{3} \quad \frac{1}{3} \quad \frac{2}{9} \]

NP → Det N
\[ \frac{2}{3} \quad \frac{4}{9} \]

NP → Det N
\[ \frac{2}{3} + \frac{1}{3} \quad \frac{2}{3} \]

NP → NP ⊕ PP
\[ \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{3} \]

ROOT → NP
1 1

N → cat
\[ \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \]

P → near
\[ \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \]

NP → NP ⊕ PP
\[ \frac{1}{6} \quad \frac{1}{3} \]

NP → Det N
\[ \frac{1}{3} \quad \frac{2}{9} \]

Det → the
1 1

N → dog
\[ \frac{2}{3} \quad \frac{2}{3} \]

P → near
\[ \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \]

Det → the
1 1

NP → Det N
\[ \frac{1}{3} \quad \frac{2}{3} \]

P → near
\[ \frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3} \]

Det → the
1 1

the dog near the
Efficient incremental parsing: the probabilistic Earley algorithm

Det → • the
1 1

NP → • Det N
\[\frac{2}{3} + \frac{1}{3} \cdot \frac{2}{3}\]

NP → • NP PP
1 \[\begin{array}{ll}
\frac{1}{2} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3}
\end{array}\]

ROOT → • NP
1 1

N → • dog
\[\begin{array}{ll}
\frac{2}{3} & \frac{2}{3}
\end{array}\]

P → • near
\[\begin{array}{ll}
\frac{1}{3} & \frac{2}{9}
\end{array}\]

NP → • NP PP
\[\begin{array}{ll}
\frac{1}{6} & \frac{1}{3}
\end{array}\]

Det → • the
1 1

NP → • Det N
\[\begin{array}{ll}
\frac{1}{3} & \frac{2}{3}
\end{array}\]

N → • dog
\[\begin{array}{ll}
\frac{1}{3} & \frac{2}{3}
\end{array}\]

P → • near
\[\begin{array}{ll}
\frac{1}{3} & \frac{2}{3}
\end{array}\]

Det → • the
1 1
Efficient incremental parsing: the probabilistic Earley algorithm

Det → the
1 1
NP → NP \circ PP
\frac{1}{3} \quad \frac{2}{9}

NP → Det N
\frac{2}{3} \quad \frac{1}{3} \quad \frac{2}{3}

NP → Det N
\frac{2}{3} \quad \frac{4}{9}

\overline{NP → Det N}
\overline{\frac{2}{9} + \frac{1}{9} \quad \frac{2}{3}}

NP → NP PP
\overline{1} \quad \frac{1}{3} \quad \frac{1}{3}

NP → NP PP
\overline{\frac{1}{6} \quad \frac{1}{3}}

PP → P NP
\overline{1} \quad \frac{1}{3} \quad \frac{2}{9}

N → cat
\overline{\frac{1}{3} \quad 1}

N → dog
\overline{\frac{1}{3} \quad \frac{2}{9}}

P → near
\overline{\frac{1}{3} \quad 1}

P → near
\overline{\frac{1}{3} \quad 1}

the

dog

near

the
Efficient incremental parsing: the probabilistic Earley algorithm

- **Det**: the
  - 1
- **NP**: the dog near the
  - NP → Det N
    - 1
  - NP → Det N
  - 2/3
  - NP → NP PP
    - 1
- **ROOT**: NP → NP PP
  - 1
- **P**: near
  - 1/3
  - P → P NP
    - 1
- **N**: dog
  - 2/3
  - N → dog
    - 1/3
  - N → dog
    - 1
  - N → dog
    - 2/3

Efficient incremental parsing: the probabilistic Earley algorithm

<table>
<thead>
<tr>
<th>Non-terminal</th>
<th>Production Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Det</td>
<td>→ the</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>→ the</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>NP</td>
<td>→ Det N</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>NP</td>
<td>→ NP PP</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>NP</td>
<td>→ NP PP</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>NP</td>
<td>→ Det N</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>PP</td>
<td>→ P NP</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>P</td>
<td>→ near</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>N</td>
<td>→ dog</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>N</td>
<td>→ dog</td>
<td>2/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>N</td>
<td>→ dog</td>
<td>2/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>N</td>
<td>→ dog</td>
<td>2/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>P</td>
<td>→ near</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2/3</td>
</tr>
<tr>
<td>Det</td>
<td>→ the</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>→ the</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>→ the</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td>→ the</td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td>→ the</td>
<td>2/3</td>
</tr>
<tr>
<td>NP</td>
<td>→ Det N</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td>→ Det N</td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td>→ Det N</td>
<td>2/3</td>
</tr>
<tr>
<td>PP</td>
<td>→ P NP</td>
<td>1/3</td>
</tr>
<tr>
<td>1</td>
<td>→ P NP</td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td>→ P NP</td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td>→ P NP</td>
<td>2/3</td>
</tr>
</tbody>
</table>
Efficient incremental parsing: the probabilistic Earley algorithm

```
Det → o the
1 1

NP → o Det N
2/3 + 1/3 2/3

NP → o NP PP
1/2 1/3 1/3 1/3

ROOT → o NP
1 1 2/3 2/3

NP → Det o N
1 2/3

Det → the o
1 1

NP → NP o PP
1/3 2/9

NP → Det N o
2/3 4/9

Det → o the
1/3 1

P → o near
1/3 1

NP → o Det N
2/9 + 1/9 2/3

N → o dog

PP → o P NP
1/3 2/9

NP → o NP PP
1/6 1/3

PP → P o NP
1/3 2/9

NP → Det o N
1/3 2/3

P → near o
1/3 1

Det → the o
1 1

N → dog o
2/3 2/3

near

the
```
Efficient incremental parsing: the probabilistic Earley algorithm

Det → ◦ the
1  1

NP → ◦ Det N
\frac{2}{3} + \frac{1}{3} = \frac{2}{3}

NP → ◦ NP PP
\frac{1}{2} \times \frac{1}{3} + \frac{1}{3} = \frac{1}{3}

ROOT → ◦ NP
1  1

NP → ◦ N
\frac{2}{3} \times \frac{2}{3} = \frac{2}{3}

NP → ◦ N
\frac{1}{3} \times \frac{2}{3} = \frac{2}{9}

NP → ◦ Det N
\frac{1}{3} \times \frac{2}{3} = \frac{2}{9}

NP → ◦ NP PP
\frac{1}{3} \times \frac{2}{3} = \frac{2}{9}

the
dog

near

the
Efficient incremental parsing: the probabilistic Earley algorithm

\[
\begin{align*}
\text{Det} & \rightarrow \circ \text{the} \\
1 & 1 \\
\text{NP} & \rightarrow \circ \text{NP} \circ \text{PP} \\
\frac{1}{3} & \frac{2}{9} \\
\text{NP} & \rightarrow \circ \text{Det} \circ \text{N} \\
\frac{2}{3} + \frac{1}{3} & \frac{2}{3} \\
\text{NP} & \rightarrow \circ \text{NP} \circ \text{PP} \\
\frac{1}{2} & \frac{1}{3} & \frac{1}{3} \\
\text{N} & \rightarrow \circ \text{cat} \\
\frac{1}{3} & 1 \\
\text{P} & \rightarrow \circ \text{near} \\
\frac{2}{9} + \frac{1}{9} & \frac{2}{3} \\
\text{NP} & \rightarrow \circ \text{NP} \circ \text{PP} \\
\frac{1}{6} & \frac{1}{3} \\
\text{NP} & \rightarrow \circ \text{Det} \circ \text{N} \\
\frac{1}{3} & \frac{2}{9} \\
\text{P} & \rightarrow \circ \text{near} \\
\frac{1}{3} & 1 \\
\text{Det} & \rightarrow \circ \text{the} \\
\frac{1}{3} & 1 \\
\text{NP} & \rightarrow \circ \text{NP} \circ \text{PP} \\
\frac{1}{3} & \frac{2}{9} \\
\text{NP} & \rightarrow \circ \text{Det} \circ \text{N} \\
\frac{1}{3} & \frac{2}{3} \\
\text{P} & \rightarrow \circ \text{near} \\
\frac{1}{3} & 1 \\
\text{Det} & \rightarrow \circ \text{the} \\
\frac{1}{3} & 1
\end{align*}
\]
Efficient incremental parsing: the probabilistic Earley algorithm

Det $\rightarrow$ the

$\frac{2}{3} + \frac{1}{3} \frac{2}{3}$

NP $\rightarrow$ Det N

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ NP PP

$\frac{1}{3} \frac{1}{3}$

N $\rightarrow$ cat

$\frac{1}{3} \frac{1}{3}$

NP $\rightarrow$ NP

$\frac{2}{3} \frac{2}{3}$

ROOT $\rightarrow$ NP

$\frac{1}{3} \frac{1}{3}$

N $\rightarrow$ dog

$\frac{2}{3} \frac{2}{3}$

NP $\rightarrow$ Det N

$\frac{2}{3} \frac{2}{3}$

Det $\rightarrow$ the

$\frac{1}{3} \frac{1}{3}$

N $\rightarrow$ dog

$\frac{2}{3} \frac{2}{3}$

PP $\rightarrow$ P NP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{2}{3} \frac{2}{3}$

P $\rightarrow$ near

$\frac{1}{3} \frac{1}{3}$

PP $\rightarrow$ P NP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{1}{3} \frac{2}{3}$

Det $\rightarrow$ the

$\frac{1}{3} \frac{1}{3}$

NP $\rightarrow$ NP PP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{2}{3} \frac{2}{3}$

P $\rightarrow$ near

$\frac{1}{3} \frac{1}{3}$

NP $\rightarrow$ NP PP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{1}{3} \frac{2}{3}$

Det $\rightarrow$ the

$\frac{1}{3} \frac{1}{3}$

NP $\rightarrow$ NP PP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{2}{3} \frac{2}{3}$

P $\rightarrow$ near

$\frac{1}{3} \frac{1}{3}$

NP $\rightarrow$ NP PP

$\frac{1}{3} \frac{2}{9}$

NP $\rightarrow$ Det N

$\frac{1}{3} \frac{2}{3}$

Det $\rightarrow$ the

$\frac{1}{3} \frac{1}{3}$
Efficient incremental parsing: the probabilistic Earley algorithm

\[
\begin{align*}
\text{Det} & \rightarrow \circ \text{the} \\
1 & \quad 1 \\
\text{NP} & \rightarrow \circ \text{Det} \ N \\
\frac{2}{3} & \quad \frac{1}{3} \\
\frac{2}{3} & \quad \frac{2}{3} \\
\text{NP} & \rightarrow \circ \text{NP} \ PP \\
\frac{2}{3} & \quad \frac{1}{3} \\
\frac{1}{3} & \quad \frac{1}{3} \\
\text{ROOT} & \rightarrow \circ \text{NP} \\
1 & \quad 1 \\
\text{NP} & \rightarrow \circ \text{Det} \circ N \\
1 & \quad \frac{2}{3} \\
\text{Det} & \rightarrow \circ \text{the} \\
1 & \quad 1 \\
\text{N} & \rightarrow \circ \text{dog} \\
\frac{2}{3} & \quad \frac{2}{3} \\
\text{P} & \rightarrow \circ \text{near} \\
\frac{1}{3} & \quad 1 \\
\frac{2}{3} & \quad \frac{2}{9} \\
\text{NP} & \rightarrow \circ \text{NP} \ PP \\
\frac{2}{3} & \quad \frac{1}{3} \\
\frac{1}{3} & \quad \frac{2}{9} \\
\text{PP} & \rightarrow \circ \text{P} \ NP \\
\frac{2}{3} & \quad \frac{1}{3} \\
\frac{1}{3} & \quad \frac{1}{3} \\
\text{NP} & \rightarrow \circ \text{Det} \circ N \\
1 & \quad \frac{2}{3} \\
\text{Det} & \rightarrow \circ \text{the} \\
1 & \quad 1 \\
\text{N} & \rightarrow \circ \text{dog} \\
\frac{2}{3} & \quad \frac{2}{3} \\
\text{P} & \rightarrow \circ \text{near} \\
\frac{1}{3} & \quad 1 \\
\frac{2}{3} & \quad \frac{2}{3} \\
\text{Det} & \rightarrow \circ \text{the} \\
1 & \quad 1 \\
\text{the} & \quad \text{dog} \\
\text{near} & \quad \text{the} 
\end{align*}
\]
Efficient incremental parsing: the probabilistic Earley algorithm

\[ \text{Det} \rightarrow \circ \text{the} \]
\[ \frac{1}{3} \quad \frac{2}{9} \]
\[ \text{NP} \rightarrow \circ \text{NP} \circ \text{PP} \]
\[ \frac{2}{3} + \frac{1}{3} = \frac{2}{3} \]
\[ \text{ROOT} \rightarrow \circ \text{NP} \]
\[ \frac{1}{3} \quad \frac{2}{3} \]

\[ \text{Det} \rightarrow \circ \text{the} \]
\[ \frac{1}{3} \quad 1 \]
\[ \text{NP} \rightarrow \circ \text{Det} \circ \text{N} \]
\[ \frac{2}{9} + \frac{1}{9} = \frac{2}{3} \]
\[ \text{PP} \rightarrow \circ \text{P} \circ \text{NP} \]
\[ \frac{1}{6} \quad \frac{1}{3} \]

\[ \text{NP} \rightarrow \circ \text{Det} \circ \text{N} \]
\[ \frac{1}{3} \quad \frac{2}{9} \]
\[ \text{P} \rightarrow \circ \text{near} \]
\[ \frac{1}{3} \quad 1 \]
\[ \text{Det} \rightarrow \circ \text{the} \]
\[ \frac{1}{3} \quad 1 \]
Efficient incremental parsing: the probabilistic Earley algorithm

Det → ◦ the
  1 1

NP → ◦ NP PP
  \[ \begin{array}{lll}
  1/3 & 2/9 \\
  \end{array} \]

NP → ◦ Det N
  \[ \begin{array}{ll}
  2/3 + 1/3 & 2/3 \\
  \end{array} \]

NP → ◦ NP PP
  \[ \begin{array}{lll}
  1/2 & 1/3 \\
  \end{array} \]

ROOT → ◦ NP
  \[ \begin{array}{ll}
  1/3 & 1 \\
  \end{array} \]

NP → ◦ Det N
  \[ \begin{array}{ll}
  2/3 & 1 \\
  \end{array} \]

NP → ◦ cat
  \[ \begin{array}{ll}
  1/3 & 1/3 \\
  \end{array} \]

P → ◦ near
  \[ \begin{array}{ll}
  1/3 & 1 \\
  \end{array} \]

NP → ◦ NP PP
  \[ \begin{array}{lll}
  1/6 & 1/3 \\
  \end{array} \]

NP → ◦ Det N
  \[ \begin{array}{ll}
  1/3 & 2/3 \\
  \end{array} \]

N → ◦ dog
  \[ \begin{array}{ll}
  1/3 & 2/3 \\
  \end{array} \]

PP → ◦ P NP
  \[ \begin{array}{ll}
  1/3 & 2/9 \\
  \end{array} \]

PP → ◦ NP
  \[ \begin{array}{ll}
  1/3 & 2/9 \\
  \end{array} \]

NP → ◦ Det N
  \[ \begin{array}{ll}
  1/3 & 2/3 \\
  \end{array} \]

P → ◦ near
  \[ \begin{array}{ll}
  1/3 & 1 \\
  \end{array} \]

Det → ◦ the
  \[ \begin{array}{ll}
  1/3 & 1 \\
  \end{array} \]
Efficient incremental parsing: the probabilistic Earley algorithm

\[
\begin{align*}
\text{Det} & \rightarrow \circ \text{the} & 1 & 1 & \text{NP} & \rightarrow \text{NP} \circ \text{PP} & 1 & 2 \\
& & & & & & 2 & 9 \\
\text{NP} & \rightarrow \circ \text{Det} \text{N} & 2 & 3 & 2 & 3 \\
& & & & & & 3 & 4 & 9 \\
\text{NP} & \rightarrow \circ \text{NP} \text{ PP} & 1 & 2 & 1 & 3 \\
& & & & & & 1 & 3 & 3 & 3 \\
\text{ROOT} & \rightarrow \circ \text{NP} & 1 & 1 & \text{N} & \rightarrow \circ \text{dog} & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
& & & & & & 2 & 3 & 2 & 3 \\
\text{NP} & \rightarrow \text{Det} \circ \text{N} & 1 & 2 & 3 \\
& & & & & & 2 & 3 \\
\text{Det} & \rightarrow \circ \text{the} & 1 & 1 & \text{N} & \rightarrow \circ \text{dog} & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
& & & & & & 2 & 3 & 2 & 3 \\
\text{PP} & \rightarrow \circ \text{P} \text{ NP} & 1 & 2 & 3 \\
& & & & & & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
\text{PP} & \rightarrow \circ \text{NP} \text{ PP} & 1 & 6 & 1 \\
& & & & & & 3 & 2 & 3 \\
\text{NP} & \rightarrow \text{Det} \circ \text{N} & 1 & 2 & 3 \\
& & & & & & 2 & 3 \\
\text{Det} & \rightarrow \circ \text{the} & 1 & 1 & \text{N} & \rightarrow \circ \text{dog} & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
& & & & & & 2 & 3 & 2 & 3 \\
\text{PP} & \rightarrow \circ \text{P} \text{ NP} & 1 & 6 & 1 \\
& & & & & & 3 & 2 & 3 \\
\text{NP} & \rightarrow \text{Det} \circ \text{N} & 1 & 2 & 3 \\
& & & & & & 2 & 3 \\
\text{Det} & \rightarrow \circ \text{the} & 1 & 1 & \text{N} & \rightarrow \circ \text{dog} & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
& & & & & & 2 & 3 & 2 & 3 \\
\text{PP} & \rightarrow \circ \text{NP} \text{ PP} & 1 & 6 & 1 \\
& & & & & & 3 & 2 & 3 \\
\text{Det} & \rightarrow \circ \text{the} & 1 & 1 & \text{N} & \rightarrow \circ \text{dog} & 2 & 3 & 2 \\
& & & & & & 3 & 2 & 3 \\
& & & & & & 2 & 3 & 2 & 3 \\
\text{PP} & \rightarrow \circ \text{P} \text{ NP} & 1 & 6 & 1 \\
& & & & & & 3 & 2 & 3 \\
\text{NP} & \rightarrow \text{Det} \circ \text{N} & 1 & 2 & 3 \\
& & & & & & 2 & 3 \\
\end{align*}
\]
Efficient incremental parsing: the probabilistic Earley algorithm

\[
\begin{align*}
\text{Det} & \rightarrow \circ \text{the} \\
& 1 \quad 1 \\
\text{NP} & \rightarrow \circ \text{Det N} \\
& \frac{2}{3} \quad \frac{1}{3} \quad \frac{2}{3} \\
\text{NP} & \rightarrow \circ \text{NP PP} \\
& \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{3} \\
\text{ROOT} & \rightarrow \circ \text{NP} \\
& 1 \quad 1 \\
\text{NP} & \rightarrow \circ \text{Det} \circ N \\
& 1 \quad \frac{2}{3} \\
\text{Det} & \rightarrow \circ \text{the} \\
& 1 \quad 1 \\
\text{NP} & \rightarrow \circ \text{NP} \circ \text{PP} \\
& \frac{1}{3} \quad \frac{2}{9} \\
\text{PP} & \rightarrow \circ \text{P NP} \\
& \frac{1}{6} \quad \frac{1}{3} \\
\text{NP} & \rightarrow \circ \text{Det} \circ N \\
& \frac{1}{3} \quad \frac{2}{9} \\
\text{Det} & \rightarrow \circ \text{the} \\
& 1 \quad \frac{1}{3} \quad \frac{2}{3} \\
\text{NP} & \rightarrow \circ \text{Det} \circ N \\
& \frac{1}{6} \quad \frac{1}{3} \\
\text{NP} & \rightarrow \circ \text{NP PP} \\
& \frac{1}{2} \quad \frac{1}{3} \quad \frac{2}{3}
\end{align*}
\]
Efficient incremental parsing: the probabilistic Earley algorithm

Det \rightarrow \text{the}  
\begin{align*}  
1 & \quad 1 
\end{align*}

NP \rightarrow \text{NP} \circ \text{PP}  
\begin{align*}  
\frac{1}{3} & \quad \frac{2}{9} 
\end{align*}

NP \rightarrow \text{Det} \circ \text{N}  
\begin{align*}  
\frac{2}{3} & \quad \frac{2}{9} 
\end{align*}

\frac{2}{3} + \frac{1}{3} = \frac{2}{3}

NP \rightarrow \text{NP} \circ \text{PP}  
\begin{align*}  
\frac{1}{2} & \quad \frac{1}{3} 
\end{align*}

ROOT \rightarrow \text{NP}  
\begin{align*}  
1 & \quad 1 
\end{align*}

N \rightarrow \text{cat}  
\begin{align*}  
\frac{1}{3} & \quad \frac{1}{3} 
\end{align*}

P \rightarrow \text{near}  
\begin{align*}  
\frac{1}{3} & \quad 1 
\end{align*}

\frac{2}{9} + \frac{1}{9} = \frac{2}{3}

NP \rightarrow \text{NP} \circ \text{PP}  
\begin{align*}  
\frac{1}{6} & \quad \frac{1}{3} 
\end{align*}

Det \rightarrow \text{the}  
\begin{align*}  
\frac{1}{3} & \quad 1 
\end{align*}

NP \rightarrow \text{Det} \circ \text{N}  
\begin{align*}  
\frac{2}{3} & \quad \frac{2}{9} 
\end{align*}

\frac{2}{3} + \frac{1}{3} = \frac{2}{3}

NP \rightarrow \text{NP} \circ \text{PP}  
\begin{align*}  
\frac{1}{3} & \quad \frac{2}{9} 
\end{align*}

NP \rightarrow \text{Det} \circ \text{N}  
\begin{align*}  
\frac{1}{3} & \quad \frac{2}{3} 
\end{align*}

Det \rightarrow \text{the}  
\begin{align*}  
\frac{1}{3} & \quad 1 
\end{align*}

the \quad \text{dog} \quad \text{near} \quad \text{the}
Efficient incremental parsing: the probabilistic Earley algorithm

\[
\text{Det} \rightarrow \circ \text{the} \\
\begin{array}{c|c|c}
\text{NP} & \text{NP} \circ \text{NP} & \frac{1}{3} \\
\text{NP} & \text{Det} \circ \text{N} & \frac{2}{9}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{Det} \circ \text{N} \\
\begin{array}{c|c|c}
\frac{2}{3} + \frac{1}{3} & \frac{2}{3}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{NP} \circ \text{PP} \\
\begin{array}{c|c|c}
\frac{1}{3} & \frac{1}{3}
\end{array}
\]

\[
\text{ROOT} \rightarrow \circ \text{NP} \\
\begin{array}{c|c|c}
\frac{2}{3} & \frac{2}{3}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{Det} \circ \text{N} \\
\begin{array}{c|c|c}
\frac{2}{3} & \frac{2}{3}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{NP} \circ \text{PP} \\
\begin{array}{c|c|c}
\frac{1}{3} & \frac{2}{9}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{NP} \circ \text{PP} \\
\begin{array}{c|c|c}
\frac{1}{6} & \frac{1}{3}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{Det} \circ \text{N} \\
\begin{array}{c|c|c}
\frac{2}{9} & \frac{1}{9}
\end{array}
\]

\[
\text{NP} \rightarrow \circ \text{NP} \circ \text{PP} \\
\begin{array}{c|c|c}
\frac{2}{3} & \frac{1}{3}
\end{array}
\]

\[
\text{Det} \rightarrow \circ \text{the} \\
\begin{array}{c|c|c}
\frac{1}{3} & 1
\end{array}
\]

\[
\text{Det} \rightarrow \circ \text{the} \\
\begin{array}{c|c|c}
\frac{2}{3} & 1
\end{array}
\]

\[
\text{N} \rightarrow \circ \text{dog} \\
\begin{array}{c|c|c}
\frac{2}{3} & \frac{2}{3}
\end{array}
\]

\[
\text{N} \rightarrow \circ \text{cat} \\
\begin{array}{c|c|c}
\frac{1}{3} & \frac{1}{3}
\end{array}
\]

\[
\text{P} \rightarrow \circ \text{near} \\
\begin{array}{c|c|c}
\frac{1}{3} & 1
\end{array}
\]

\[
\text{P} \rightarrow \circ \text{near} \\
\begin{array}{c|c|c}
\frac{1}{3} & 1
\end{array}
\]

\[
\text{PP} \rightarrow \circ \text{P} \circ \text{NP} \\
\begin{array}{c|c|c}
\frac{1}{3} & \frac{2}{9}
\end{array}
\]

\[
\text{PP} \rightarrow \circ \text{P} \circ \text{NP} \\
\begin{array}{c|c|c}
\frac{1}{3} & \frac{2}{9}
\end{array}
\]

\[
\text{the} \\
\begin{array}{c|c|c}
1 & \frac{2}{3}
\end{array}
\]

\[
\text{dog} \\
\begin{array}{c|c|c}
\frac{2}{3} & \frac{2}{3}
\end{array}
\]

\[
\text{near} \\
\begin{array}{c|c|c}
\frac{1}{3} & 1
\end{array}
\]

\[
\text{the} \\
\begin{array}{c|c|c}
\frac{1}{3} & 1
\end{array}
\]
Efficient incremental parsing: the probabilistic Earley algorithm

Det → ◦ the
\[ \frac{1}{3} \quad \frac{2}{9} \]

NP → Det N
\[ \frac{2}{3} + \frac{1}{3} = \frac{2}{3} \]

NP → Det N
\[ \frac{2}{3} \quad \frac{4}{9} \]

NP → Det N
\[ \frac{1}{3} \quad \frac{2}{9} \]

N → ◦ cat
\[ \frac{1}{3} \quad \frac{1}{3} \]

P → ◦ near
\[ \frac{1}{3} \quad 1 \]

PP → P • NP
\[ \frac{2}{9} + \frac{1}{9} = \frac{2}{9} \]

NP → ◦ NP PP
\[ \frac{1}{6} \quad \frac{1}{3} \]

NP → NP • PP
\[ \frac{1}{3} \quad \frac{2}{9} \]

NP → NP • PP
\[ \frac{1}{3} \quad \frac{2}{9} \]

NP → NP • PP
\[ \frac{1}{3} \quad \frac{2}{9} \]

NP → NP • PP
\[ \frac{1}{3} \quad \frac{2}{9} \]

NP → Det • N
\[ \frac{1}{3} \quad \frac{2}{9} \]

Det → ◦ the
\[ \frac{1}{3} \quad 1 \]

N → ◦ dog
\[ \frac{2}{3} \quad \frac{2}{3} \]

P → ◦ near
\[ \frac{1}{3} \quad 1 \]

P → ◦ near
\[ \frac{1}{3} \quad 1 \]

ROOT → ◦ NP
\[ \frac{1}{3} \quad \frac{2}{3} \]

the
\[ \frac{1}{3} \quad \frac{2}{9} \]

dog
\[ \frac{1}{3} \quad \frac{2}{9} \]

near
\[ \frac{1}{3} \quad \frac{2}{9} \]

the
Efficient incremental parsing: the probabilistic Earley algorithm

Det → ◦ the
1 1

NP → NP ◦ PP
$\frac{1}{3} \quad \frac{2}{9}$

NP → Det N ◦
$\frac{2}{3} \quad \frac{4}{9}$

NP → ◦ Det N
$\frac{2}{3} + \frac{1}{3} \quad \frac{2}{3}$

NP → ◦ NP PP
$\frac{1}{2} \quad \frac{1}{3}$

N → ◦ cat
$\frac{1}{3} \quad \frac{1}{3}$

P → ◦ near
$\frac{1}{3} \quad 1$

PP → ◦ P NP
$\frac{1}{3} \quad \frac{2}{9}$

NP → ◦ NP PP
$\frac{1}{6} \quad \frac{1}{3}$

ROOT → ◦ NP
1 1

N → ◦ dog
$\frac{2}{3} \quad \frac{2}{3}$

NP → Det ◦ N
$\frac{1}{2} \quad \frac{2}{3}$

Det → ◦ the
1 1

N → ◦ dog
$\frac{2}{3} \quad \frac{2}{3}$

P → ◦ near
$\frac{1}{3} \quad 1$

Det → ◦ the
$\frac{1}{3} \quad 1$

NP → ◦ Det N
$\frac{2}{9} + \frac{1}{9} \quad \frac{2}{3}$

PP → P ◦ NP
$\frac{1}{3} \quad \frac{2}{9}$

Det → ◦ the
$\frac{1}{3} \quad 1$
From the *inside probabilities* of the states on the chart, the posterior distribution on (incremental) trees can be directly calculated.

This posterior distribution is *precisely* the correct result of the application of Bayes’ rule.

Hence, probabilistic Earley is also performing rational disambiguation.

Hale (2001) called this the “eager” property of an incremental parsing algorithm.
Probabilistic Earley algorithm: key ideas

- We want to use probabilistic grammars for both disambiguation and calculating probability distributions over upcoming events.
- Infinitely many trees can be constructed in polynomial time ($O(n^3)$) and space ($O(n^2)$).
- The *prefix probability* of the string is calculated in the process.
- By taking the log-ratio of two prefix probabilities, the surprisal of a word in its context can be calculated.
Probabilistic Earley algorithm: key ideas

- We want to use probabilistic grammars for both disambiguation and calculating probability distributions over upcoming events.
- Infinitely many trees can be constructed in polynomial time ($O(n^3)$) and space ($O(n^2)$).
- The *prefix probability* of the string is calculated in the process.
- By taking the log-ratio of two prefix probabilities, the surprisal of a word in its context can be calculated.
Probabilistic Earley algorithm: key ideas

- We want to use probabilistic grammars for both disambiguation and calculating probability distributions over upcoming events.
- Infinitely many trees can be constructed in polynomial time \(O(n^3)\) and space \(O(n^2)\).
- The *prefix probability* of the string is calculated in the process.
- By taking the log-ratio of two prefix probabilities, the surprisal of a word in its context can be calculated.
Other introductions

- You can read about the (non-probabilistic) Earley algorithm in (Jurafsky and Martin, 2000, Chapter 13)

- Prefix probabilities can also be calculated with an extension of the CKY algorithm due to Jelinek and Lafferty (1991)
Also... 

Applications of the idea of surprisal to comprehension and production


References II


