Computational Linguistics

9

CSC 2501 / 485
Fall 2015

9. Statistical parsing

Frank Rudzicz
Toronto Rehabilitation Institute-UHN,
Department of Computer Science, University of Toronto

Reading: Jurafsky & Martin: 5.2–5.5.2, 5.6, 12.4, 14.0–1, 14.3–4, 14.6–7. Bird et al: 8.6.

Statistical parsing 1

General idea:

- Assign **probabilities** to **rules** in a context-free grammar.
  - Use a likelihood model.
- Combine probabilities of rules in a **tree**.
  - Yields likelihood of a parse.
- The best parse is the *most likely* one.

Statistical parsing 2

Motivations:

- Uniform process for attachment decisions.
- Use **lexical preferences** in all decisions.

Two general approaches

1. Assign a probability to each rule of grammar, including lexical rules.
   - Parse string of input words with probabilistic rules.
     The can will rust.
2. Assign probabilities *only* to *non-lexical* rules.
   - Probabilistically tag input words with syntactic categories using a **part-of-speech tagger**.
   - Consider the syntactic categories to be terminals, parse that string with probabilistic rules.
     Det N Modal Verb.

Part-of-speech tagging 1

Part-of-speech (PoS) tagging:
Given a sequence of words $w_1 \dots w_n$ (from well-formed text), determine the syntactic category (PoS) C_i of each word.

I.e, the best category sequence $C_1 \dots C_n$ to assign to the word sequence $w_1 \dots w_n$.

Most likely

Part-of-speech tagging 2

Example:

The	*can*	*will*	*rust*
det	modal verb	**modal verb**	noun
	noun	noun	**verb**
	verb	verb	

Example from Charniak 1997

Part-of-speech tagging 3

$$P(C_1 \ldots C_n | w_1 \ldots w_n) = \frac{P(C_1 \ldots C_n \wedge w_1 \ldots w_n)}{P(w_1 \ldots w_n)}$$

We cannot get this probability directly.

We should estimate it (through counts).

Hey, let's approximate it first
(by modifying the formula).

Counts: Need representative corpus.

POS tagging: Unigram MLE 1

Look at individual words (***unigrams***):

$$P(C|w) = \frac{P(C \wedge w)}{P(w)}$$

Maximum likelihood estimator (MLE):

$$P(C|w) = \frac{\boxed{c(w \text{ } is \text{ } C)}}{\boxed{c(w)}}$$

Count in corpus

POS tagging: Unigram MLE 2

Problems of MLE:

- Sparse data.
- Extreme cases:
  a. Undefined if w is not in the corpus.
  b. 0 if w does not appear in a particular category.

POS tagging: Unigram MLE 3

Smoothing of formula, e.g.,:

$$P(\mathrm{C}|w) \approx \frac{c(w\ is\ \mathrm{C}) + \epsilon}{c(w) + \epsilon}$$

Give small (non-zero) probability value to **unseen events**, taken from seen events by *discounting* them.

Various methods to ensure we still have valid probability distribution.

POS tagging: Unigram MLE 4

Just choosing the most frequent PoS for each word yields 90% accuracy in PoS tagging.

But:

- Not uniform across words.
- Accuracy is low (0.9^n) when multiplied over *n* words.
- No context: *The fly* vs. *I will fly*.

Need better approximations for

$$P(C_1 \ldots C_n | w_1 \ldots w_n)$$

POS tagging: Bayesian method

Use Bayes's rule to rewrite:

$$P(C_1 \ldots C_n | w_1 \ldots w_n) = \frac{\overset{1}{\boxed{P(C_1 \ldots C_n)}} \times \overset{2}{\boxed{P(w_1 \ldots w_n \mid C_1 \ldots C_n)}}}{P(w_1 \ldots w_n)}$$

For a given word string, we want to maximize this, find most likely $C_1 \ldots C_n$:

$$\underset{C_1 \ldots C_n}{\operatorname{argmax}} P(C_1 \ldots C_n \mid w_1 \ldots w_n)$$

So just need to maximize the numerator.

Approximating probabilities 1

Approximate ❶ $P(C_1 \ldots C_n)$ by predicting each category from previous $N - 1$ categories: an **N-gram model.**

Warning: Not the same *n*!!

Bigram (2-gram) model:

$$P(C_1 \ldots C_n) \approx \prod_{i=1}^{N} P(C_i | C_{i-1})$$

Posit pseudo-categories START at C_0 , and END as C_n. Example:

$$P(\mathrm{A\ N\ V\ N}) \approx P(\mathrm{A}|\mathrm{START}) \cdot P(\mathrm{N}|\mathrm{A}) \cdot P(\mathrm{V}|\mathrm{N}) \cdot P(\mathrm{N}|\mathrm{V}) \cdot P(\mathrm{END}|\mathrm{N})$$

Approximating probabilities 2

Approximate ❷ $P(w_1 \ldots\ w_n | C_1 \ldots\ C_n)$ by assuming that the probability of a word appearing in a category is **independent** of the words around it.

$$P(w_1 \ldots w_n | C_1 \ldots C_n) \approx \prod_{i=1}^{N} P(w_i | C_i)$$

Lexical generation probabilities

Approximating probabilities 3

Why is $P(w|C)$ better than $P(C|w)$?

- $P(C|w)$ is clearly *not* independent of surrounding categories.
  - E.g., those octopodes never stop [redacted]/VBG vs. look at those [redacted]/ADJ octopodes
- The probabilities of **lexical generation** are *more* independent.
- Complete formula for PoS includes bigrams, and so it does capture some context.

Putting it all together

$$
\begin{aligned}
P(C_1 \ldots C_n \,|\, w_1 \ldots w_n) \\
&= \frac{P(C_1 \ldots C_n \wedge w_1 \ldots w_n)}{P(w_1 \ldots w_n)} \\
&= \frac{P(C_1 \ldots C_n) \times P(w_1 \ldots w_n \,|\, C_1 \ldots C_n)}{P(w_1 \ldots w_n)} \\
&\propto P(C_1 \ldots C_n) \times P(w_1 \ldots w_n \,|\, C_1 \ldots C_n) \\
&\approx \prod_{i=1}^{n} P(C_i \,|\, C_{i-1}) \times P(w_i \,|\, C_i) \quad ❸ \\
&= \boxed{\prod_{i=1}^{n} \frac{c(C_{i-1} C_i)}{c(C_{i-1})} \times \boxed{\frac{c(w_i \text{ is } C_i)}{c(C_i)}}}
\end{aligned}
$$

MLE for categories not the same as for words; *cf* slide 8

Really should use smoothed MLE; *cf* slide 10

Finding max 1

Want to find the argmax (most probable)

$C_1 \ \dots \ Cn$.

<u>*Brute force method:*</u> Find **all possible** sequences of categories and compute *P*.

That's unnecessary and stupid: Our approximations assume lots of independence:

- *Category bigrams:* C_i depends only on C_{i-1}.
  Lexical generation: w_i depends only on C_i.
- Hence we do *not* need to enumerate all sequences independently.

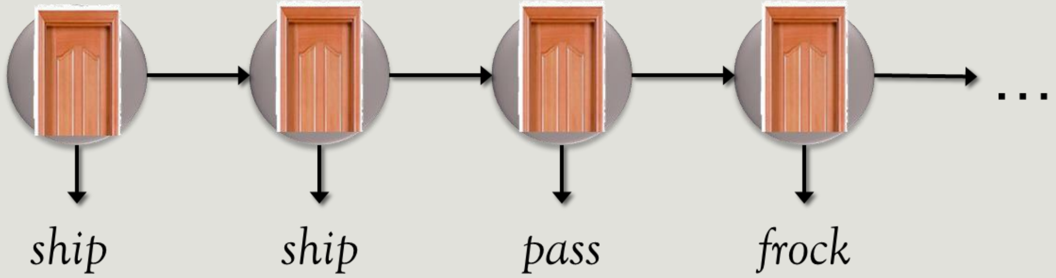
Finding max 2

Bigrams:
Markov model.

- *States* are categories and *transitions* represent bigrams.

Lexical generation probabilities:
Hidden Markov model.

- Words are outputs (with given probability) of states.
- A word could be the output of more than one state.
- Current state is unknown ("hidden").



Example

Artificial corpus of PoS-tagged 300 sentences using only Det, N, V, P.

- *The flower flowers like a bird.*
  Some birds like a flower with fruit beetles.
  Like flies like flies.
  …

Based on an example in section 7.3 of: Allen, James. *Natural Language Understanding* (2nd ed), 1995, Benjamin Cummings.

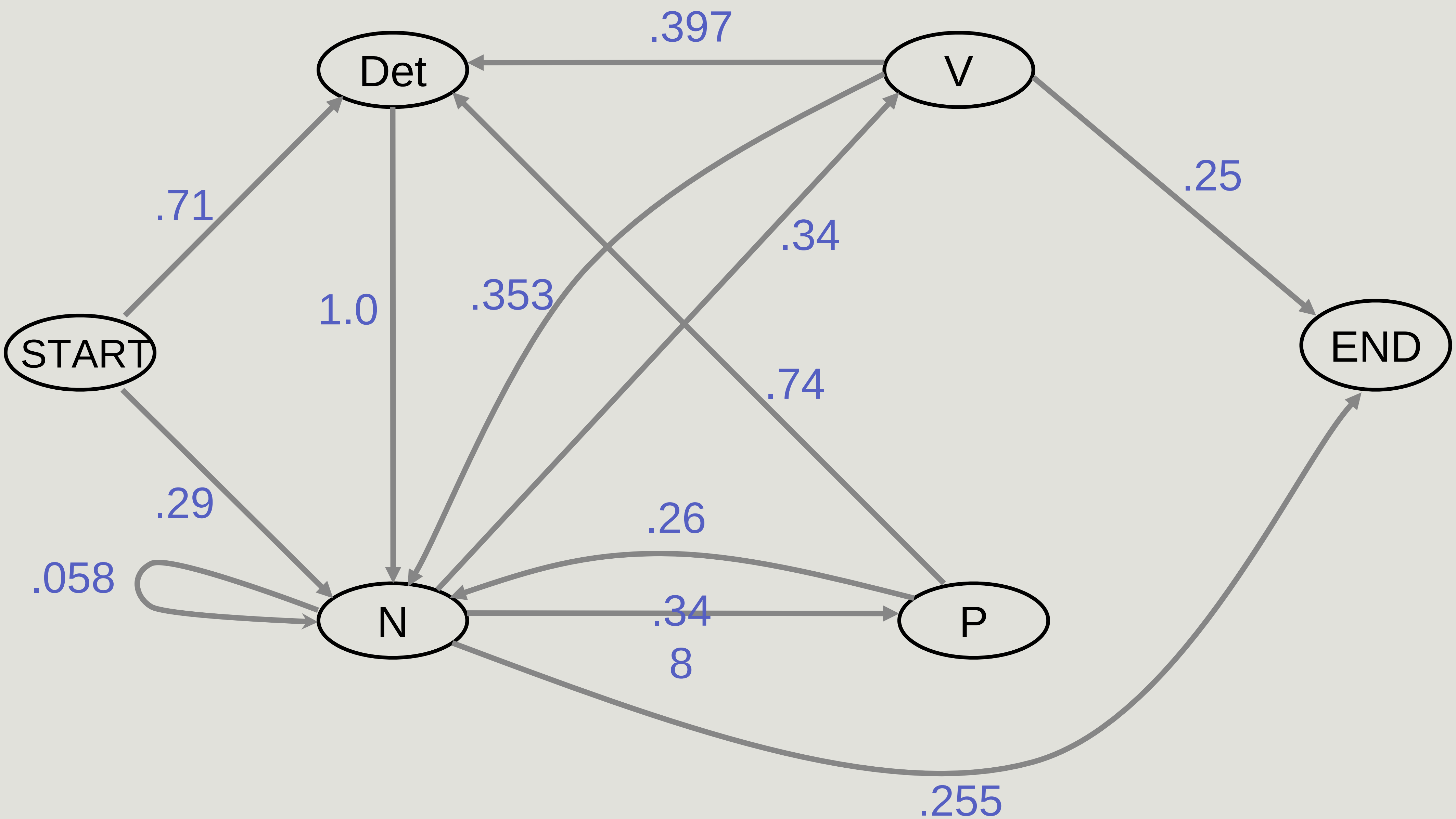
Some lexical generation probabilities:

$P(\textit{the}\mid\text{Det}) = .54$	$P(\textit{like}\mid\text{N}) = .012$	$P(\textit{flower}\mid\text{N}) = .063$	$P(\textit{birds}\mid\text{N}) = .076$
$P(\textit{a}\mid\text{Det}) = .36$	$P(\textit{like}\mid\text{V}) = .1$	$P(\textit{flower}\mid\text{V}) = .050$	$P(\textit{flies}\mid\text{V}) = .076$
$P(\textit{a}\mid\text{N}) = .001$	$P(\textit{like}\mid\text{P}) = .068$	$P(\textit{flowers}\mid\text{N}) = .050$	$P(\textit{flies}\mid\text{N}) = .025$
$\vdots$	$\vdots$	$P(\textit{flowers}\mid\text{V}) = .053$	$\vdots$
		$\vdots$	

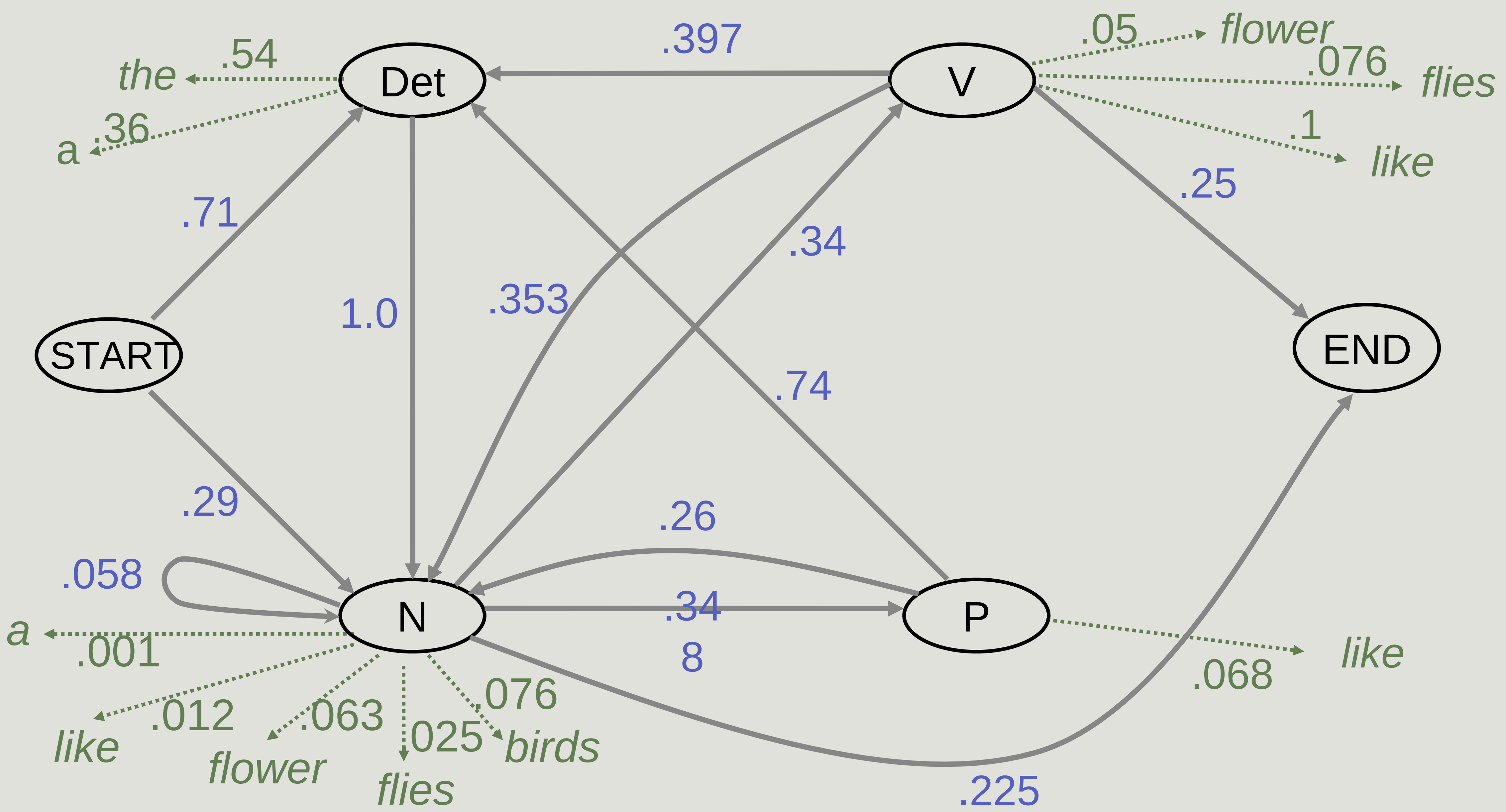
Markov model: bigram table

Bigram C_{i-1}, C_i	Count C_{i-1}	Count C_{i-1},C_i	$P(C_i\|C_{i-1})$	Estimate
START, Det	300	213	*P*(Det| START)	0.710
START, N	300	87	*P*(N| START)	0.290
Det, N	558	558	*P*(N|Det)	1.000
N, V	883	300	*P*(V|N)	0.340
N, N	883	51	*P*(N|N)	0.058
N, P	883	307	*P*(P|N)	0.348
N, END	883	225	*P*(END|N)	0.255
V, N	300	106	*P*(N|V)	0.353
V, Det	300	119	*P*(Det|N)	0.397
V, END	300	75	*P*(END|V)	0.250
P, Det	307	226	*P*(Det|P)	0.740
P, N	307	81	*P*(N|P)	0.260

Markov model: transitions



HMM: lexical generation



$P(\mathit{the} | \mathrm{Det}) = .54$ $P(\mathit{like} | \mathrm{N}) = .012$ $P(\mathit{flower} | \mathrm{N}) = .063$ $P(\mathit{birds} | \mathrm{N}) = .076$

⋮ ⋮ ⋮ ⋮

Hidden Markov models 1

Given the observed output, we want to find the most likely path *through* the model.

The	*can*	*will*	*rust*
det	modal verb	**modal verb**	noun
	noun	noun	**verb**
	verb	verb	

Hidden Markov models 2

At any state in an HMM, how you got there is irrelevant to computing the next transition.

- So, just need to remember the best path and probability up to that point.
- Define $P_{C_{i-1}}$ as the probability of the best sequence up to state C_{i-1}.

Then find C_i that maximizes

$$P_{C_{i-1}} \times \boxed{P(C_i \mid C_{i-1}) \times P(w \mid C_i)}$$

❸ from slide 16

Viterbi algorithm

Given an HMM and an observation O of a *sequence* of its output, Viterbi finds the most probable sequence S of states that produced O.

- O = words in a sequence, S = PoS tags of sentence

Parameters of HMM based on large training corpus.

Statistical chart parsing 1

Consider tags as terminals

(*i.e.*, use a PoS tagger to pre-process input texts).

Det N Modal Verb.

For the probability of each grammar rule, use MLE.

Probabilities are derived from hand-parsed corpora (treebanks).

- Count frequency of use *c* of each rule $C \rightarrow \alpha$, for each non-terminal C and each different RHS α.

What are some problems with this approach?

Statistical chart parsing 2

MLE probability of rules:

- For each rule $C \rightarrow \alpha$:

$$P(C \rightarrow \alpha | C) = \frac{c(C \rightarrow \alpha)}{\sum_{\beta} c(C \rightarrow \beta)} = \frac{c(C \rightarrow \alpha)}{c(C)} \qquad ❹$$

Takes no account of the context of use of a rule: ***independence assumption***.

```
>>> import nltk
>>> nltk.parse.pchart.demo()

  1: I saw John with my telescope
     <Grammar with 17 productions>

  2: the boy saw Jack with Bob under the table with a telescope
     <Grammar with 23 productions>

Which demo (1-2)? 1

s: I saw John with my telescope
parser: <nltk.parse.pchart.InsideChartParser object at 0x7f61288f3290>
grammar: Grammar with 17 productions (start state = S)
    S -> NP VP [1.0]
    NP -> Det N [0.5]
    NP -> NP PP [0.25]
    NP -> 'John' [0.1]
    NP -> 'I' [0.15]
    Det -> 'the' [0.8]
    Det -> 'my' [0.2]
    N -> 'man' [0.5]
    N -> 'telescope' [0.5]
    VP -> VP PP [0.1]
    VP -> V NP [0.7]
    VP -> V [0.2]
    V -> 'ate' [0.35]
    V -> 'saw' [0.65]
    PP -> P NP [1.0]
    P -> 'with' [0.61]
    P -> 'under' [0.39]
```

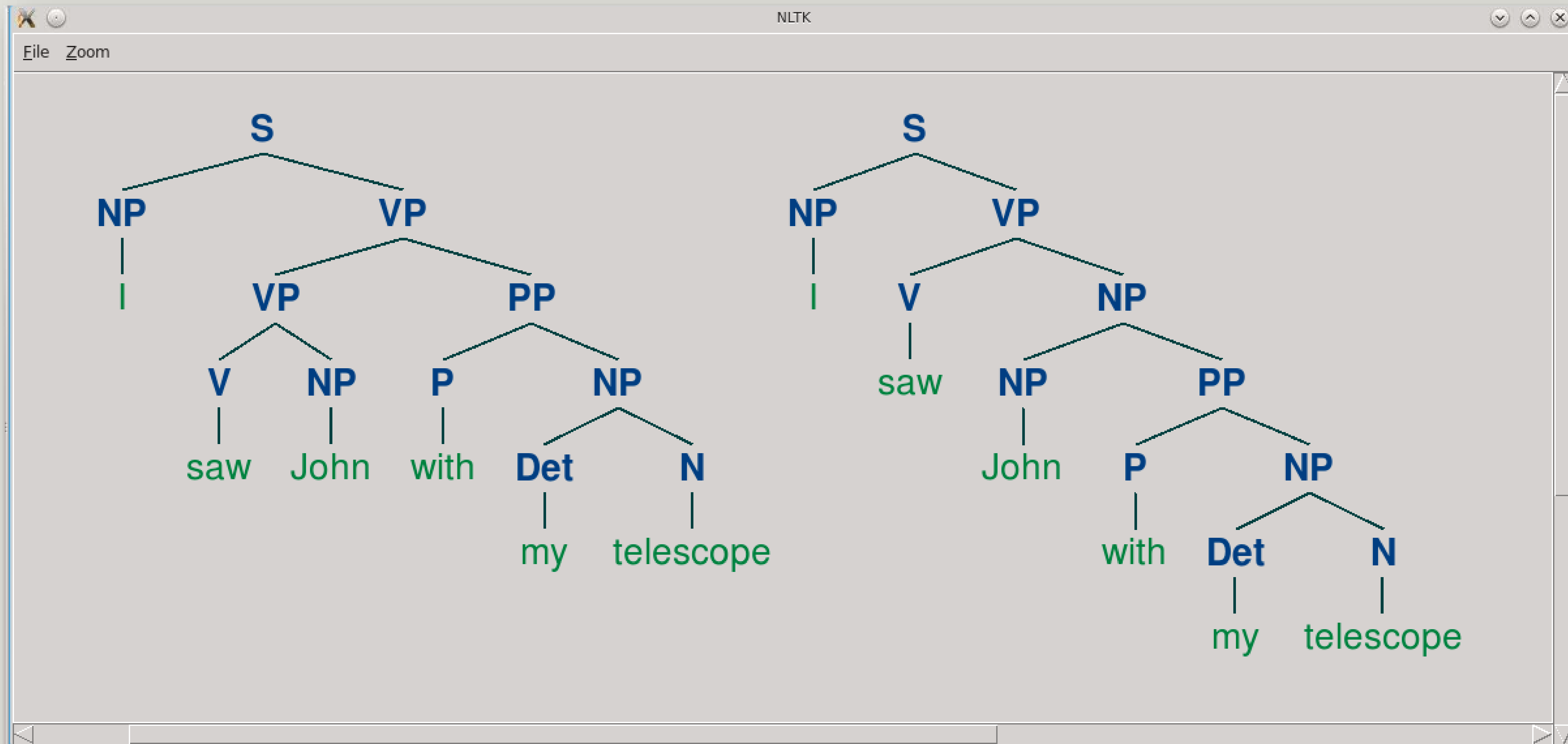
```
|[-] . . . . .| [0:1] 'I'                          [1.0]
|. [-] . . . .| [1:2] 'saw'                        [1.0]
|. . [-] . . .| [2:3] 'John'                       [1.0]
|. . . [-] . .| [3:4] 'with'                       [1.0]
|. . . . [-] .| [4:5] 'my'                         [1.0]
|. . . . . [-]| [5:6] 'telescope'                  [1.0]
|. . . . . [-]| [5:6] 'telescope'                  [1.0]
|. . . . [-] .| [4:5] 'my'                         [1.0]
|. . . [-] . .| [3:4] 'with'                       [1.0]
|. . [-] . . .| [2:3] 'John'                       [1.0]
|. [-] . . . .| [1:2] 'saw'                        [1.0]
|[-] . . . . .| [0:1] 'I'                          [1.0]
|. [-] . . . .| [1:2] V  -> 'saw' *                [0.65]
|. > . . . . .| [1:1] VP -> * V NP                 [0.7]
|. > . . . . .| [1:1] V  -> * 'saw'                [0.65]
|. . . [-] . .| [3:4] P  -> 'with' *               [0.61]
|. . . > . . .| [3:3] PP -> * P NP                 [1.0]
|. . . [-> . .| [3:4] PP -> P * NP                 [0.61]
|. . . > . . .| [3:3] P  -> * 'with'               [0.61]
|. . . . . [-]| [5:6] N  -> 'telescope' *          [0.5]
|. . . . . > .| [5:5] N  -> * 'telescope'          [0.5]
|. [-> . . . .| [1:2] VP -> V * NP                 [0.455]
|. > . . . . .| [1:1] VP -> * V                    [0.2]
|. . . . [-] .| [4:5] Det -> 'my' *                [0.2]
|. . . . > . .| [4:4] NP -> * Det N                [0.5]
|. . . . > . .| [4:4] Det -> * 'my'                [0.2]
```

⋮

:
:

```
|. . . . > . .| [4:4] S  -> * NP VP        [1.0]
|. . . . > . .| [4:4] NP -> * NP PP        [0.25]
|. . . . [--->| [4:6] S  -> NP * VP        [0.05]
|. [---] . . .| [1:3] VP -> V NP *         [0.0455]
|[-> . . . . .| [0:1] NP -> NP * PP        [0.0375]
|. . . [-----]| [3:6] PP -> P NP *         [0.0305]
|. . [-> . . .| [2:3] NP -> NP * PP        [0.025]
|[---] . . . .| [0:2] S  -> NP VP *        [0.0195]
|. [-> . . . .| [1:2] VP -> VP * PP        [0.013]
|. . . . [--->| [4:6] NP -> NP * PP        [0.0125]
|[-----] . . .| [0:3] S  -> NP VP *        [0.006825]
|. [---> . . .| [1:3] VP -> VP * PP        [0.00455]
|. . [-------]| [2:6] NP -> NP PP *        [0.0007625]
|. . [------->| [2:6] S  -> NP * VP        [0.0007625]
|. [---------]| [1:6] VP -> V NP *         [0.0003469375]
|. . [------->| [2:6] NP -> NP * PP        [0.000190625]
|. [---------]| [1:6] VP -> VP PP *        [0.000138775]
|[===========]| [0:6] S  -> NP VP *        [5.2040625e-05]
|. [--------->| [1:6] VP -> VP * PP        [3.469375e-05]
|[===========]| [0:6] S  -> NP VP *        [2.081625e-05]
|. [--------->| [1:6] VP -> VP * PP        [1.38775e-05]
```

```
Draw parses (y/n)? y
  please wait...
```



```
Print parses (y/n)? y
 (S
  (NP I)
  (VP
    (VP (V saw) (NP John))
    (PP (P with) (NP (Det my) (N telescope))))) [2.081625e-05]
(S
  (NP I)
  (VP
    (V saw)
    (NP
      (NP John)
      (PP (P with) (NP (Det my) (N telescope)))))) [5.2040625e-05]
```

Statistical chart parsing 3

Probability of chart entries — completed constituents:

$$\begin{aligned} P(e_0) &= P(C_0 \rightarrow C_1 \dots C_n \mid C_0) \times P(e_1) \times \cdots \times P(e_n) \\ &= P(C_0 \rightarrow C_1 \dots C_n \mid C_0) \times \prod_{i=1}^{n} P(e_i) \end{aligned}$$

❺

where e_0 is the entry for current constituent, of category C_0;
and e_1 ... e_n are the chart entries for C_1 ... Cn in the RHS of the rule.

NB: Unlike for PoS tagging above, the C_i are not necessarily lexical categories.

Statistical chart parsing 4

Consider a complete parse tree rooted at S.

Recasting ❺, the S constituent will have the probability

$$P(S) = \prod_n P(rule(n) \mid cat(n)) \quad ❻$$

where n ranges over all nodes in the tree of S;
$rule(n)$ is the rule used for n;
$cat(n)$ is the category of n.

- "Bottoms out" at lexical categories.

Evaluation 1

Evaluation method:

- *Train* on part of a parsed corpus.
  (*I.e.*, gather rules and statistics.)
- Test on a different part of the corpus.

Evaluation 2

Evaluation: PARSEVAL measures compare parser output to known correct parse:

- Labelled **precision,** labelled **recall.**

Fraction of constituents in output that are correct.

Fraction of correct constituents in output.

- ***F*-measure** = harmonic mean of precision and recall = $2PR \ / \ (P \ + \ R)$

Evaluation 3

Evaluation: PARSEVAL measures compare parser output to 'known' 'correct' parse:

- Penalize for **cross-brackets** per sentence:
  Constituents in output that overlap two (or more) correct ones;
  e.g., [[A B] C] for [A [B C]].
- [[*Nadia*] [[*fondled*] [*the eggplant*]]]
  [[[*Nadia*] [*fondled*]] [*the eggplant*]]

Improving statistical parsing

Problem: Probabilities are based only on structures:

$$P(C \rightarrow \alpha | C) = \frac{c(C \rightarrow \alpha)}{\sum_{\beta} c(C \rightarrow \beta)} = \frac{c(C \rightarrow \alpha)}{c(C)} \quad \text{❹}$$

But *actual words* strongly condition which rule is used (*cf* Ratnaparkhi).

We can improve the results by conditioning on *more factors*, including words.

Lexicalized grammars 1

Head of a phrase: its central or key word.

- The noun of an NP, the preposition of a PP, etc.

Lexicalized grammar: Refine the grammar so that rules take *heads of phrases* into account — the actual words.

- BEFORE: Rule for NP.
  AFTER: Rules for NP-whose-head-is-*aardvark*, NP-whose-head-is-*abacus*, … , NP-whose-head-is-*zymurgy*.

And similarly for VP, PP, etc.

Lexicalized grammars 2

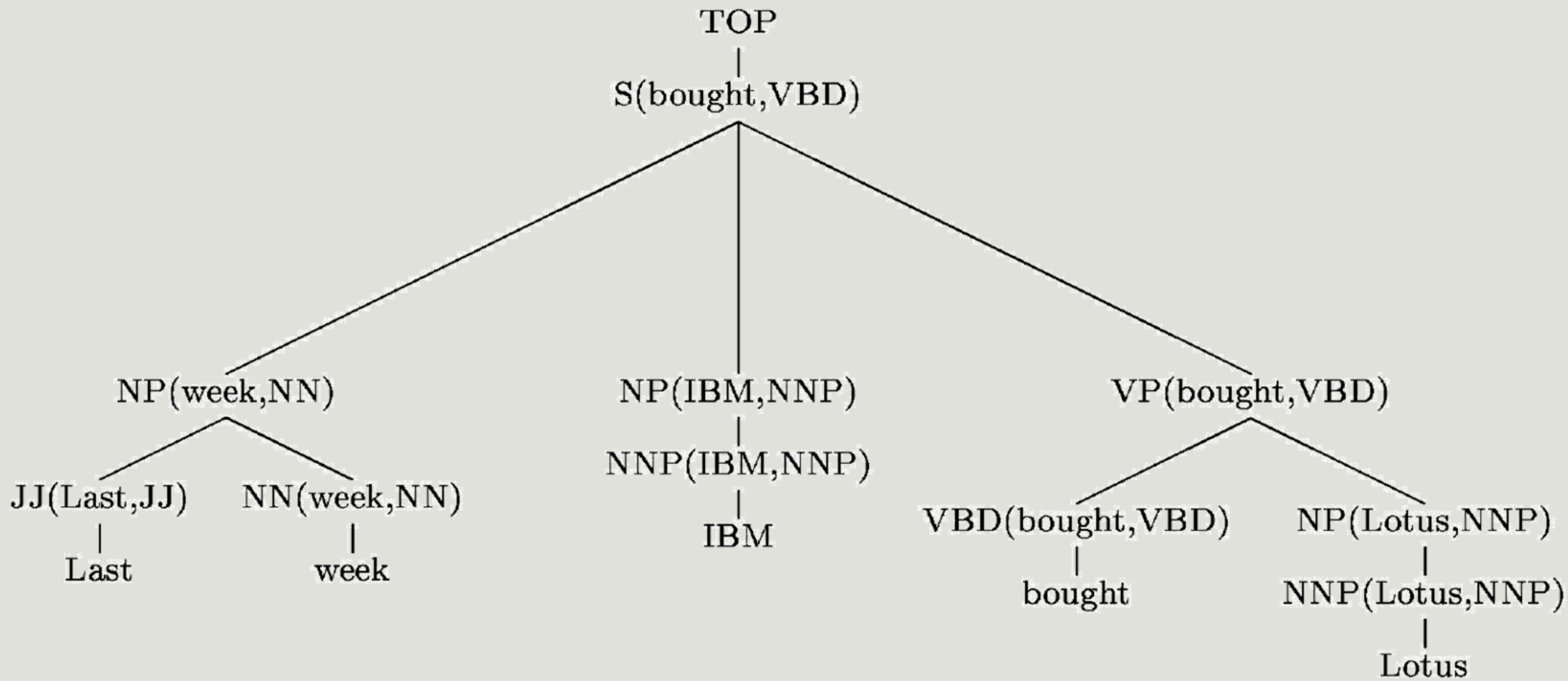
Notation: $cat(head, tag)$ for constituent category *cat* headed by *head* with part-of-speech *tag*.

- e.g., NP(*aardvark*,NN), PP(*without*,IN)

NP-whose-head-is-the-NN-*aardvark*

PP-whose-head-is-the-IN-*without*

A lexicalized grammar



TOP → S(*bought*,VBD)
S(*bought*,VBD) → NP(*week*,NN) NP(*IBM*,NNP) VP(*bought*,VBD)
NP(*week*,NN) → JJ(*Last*,JJ) NN(*week*,NN)
NP(*IBM*,NNP) → NNP(*IBM*,NNP)
VP(*bought*,VBD) → VBD(*bought*,VBD) NP(Lotus,NNP)
NP(*Lotus*,NNP) → NNP(*Lotus*,NNP)

Lexical Rules:
JJ(*Last*,JJ) → *Last*
NN(*week*,NN) → *week*
NNP(*IBM*,NNP) → *IBM*
VBD(*bought*,VBD) → *bought*
NNP(*Lotus*,NNP) → *Lotus*

Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4), 2003, 589–637.

Lexicalized grammars 3

Number of rules explodes given more heads, but no theoretical change in parsing (whether statistical or not).

But far too specific for practical use; each head is too rarely used to determine its probability.

Need something more than regular (unlexicalized) rules and less than complete lexicalization …

Lexicalized parsing 1

Starting from unlexicalized rules:

1. ***Lexicalization:*** Consider the head word of each node, not just its category:

$$P(S) = \prod_n P(rule(n) \mid head(n))$$

Replaces ❻ from slide 34

where $head(n)$ is the PoS-tagged head word of node n.

Needs finer-grained probabilities:

- e.g., probability that rule r is used, given we *are in* an NP whose head is the noun *guzzler*.

Lexicalized parsing 2

2. *Head and parent:* Condition on the head *and* the head of the *parent* node in the tree:

$$P(\text{Sentence, Tree}) = \prod_{n \in \text{Tree}} P(rule(n) \mid head(n)) \times P(head(n) \mid head(parent(n)))$$

e.g., probability of rule *r* given that head is the noun *guzzler*.

e.g., probability that head is the noun *guzzler*, given that parent phrase's head is the verb *flipped*.

Effects on parsing

Lexical information introduces *context* into CFG.

Grammar is larger.

Potential problems of sparse data.

- Possible solutions: Smoothing; back-off estimates.

If you don't have data for a fine-grained situation, use data from a coarser-grained situation in which it's contained.

Collins 2003

Can condition on *any* information available in generating the tree.

Basic idea: Avoid sparseness of lexicalization by decomposing rules.

- Make plausible independence assumptions.
- Break rules down into small steps (small number of parameters).
- Each rule parameterized with word+PoS-tag pair:

S(*bought*,VBD) → NP(*week*,NN) NP(*IBM*,NNP) VP(*bought*,VBD)

Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4), 2003, 589–637.

Collins' first model 1

Lexical Rules, with probability 1.0:
$tag(word, tag) \longrightarrow word$

Internal Rules, with treebank-based probabilities. Separate terminals to the left and right of the head; generate one at a time:

$$\mathrm{X} \rightarrow \mathrm{L}_n \mathrm{L}_{n-1} \ldots \mathrm{L}_1 \, \mathrm{H} \, \mathrm{R}_1 \ \ldots \mathrm{R}_{m-1} \, \mathrm{R}_m \quad (n, m \geq 0)$$

X, L_i, H, and R_i all have the form ***cat(head,tag)***.
Notation: Italic lowercase symbol for (*head,tag*):

$$\mathrm{X}(x) \rightarrow \mathrm{L}_n(l_n)\mathrm{L}_{n-1}(l_{n-1}) \ldots \mathrm{L}_1(l_1) \, \mathrm{H}(h) \, \mathrm{R}_1(r_1) \ \ldots \mathrm{R}_{m-1}(r_{m-1}) \, \mathrm{R}_m(r_m)$$

Collins' first model 2

Assume there are additional L_{n+1} and R_{m+1} representing phrase boundaries ("STOP").

Example:
S(*bought*,VBD) $\rightarrow$ NP(*week*,NN) NP(*IBM*,NNP) VP(*bought*,VBD)
$n = 2$, $m = 0$ (two constituents on the left of the head, zero on the right).
X = S, H = VP, L_1 = NP, L_2 = NP, L_3 = STOP, R_1 = STOP.
h = (*bought*,VBD), l_1 = (*IBM*,NNP), l_2 = (*week*,NN).

Distinguish probabilities of heads P_h, of left constituents P_l, and of right constituents P_r.

Probabilities of internal rules 1

$$
\begin{aligned}
& P(\mathrm{X}(h)) \\
&= P(\mathrm{L}_{n+1}(l_{n+1})\mathrm{L}_n(l_n) \ \ldots \mathrm{L}_1(l_1)\ \mathrm{H}(h)\ \mathrm{R}_1(r_1)\ \ldots\ \mathrm{R}_m(r_m)\ \mathrm{R}_{m+1}(r_{m+1}) \,|\, \mathrm{X}, h) \\
&= P_h(\mathrm{H} \,|\, \mathrm{X}, h) \\
&\quad \times \prod_{i=1}^{n+1} P_l(\mathrm{L}_i(l_i) \,|\, \mathrm{L}_1(l_1) \ldots \mathrm{L}_{i-1}(l_{i-1}), \mathrm{X}, h, \mathrm{H}) \\
&\quad \times \prod_{j=1}^{m+1} P_r(\mathrm{R}_j(r_j) \,|\, \mathrm{L}_1(l_1) \ldots \mathrm{L}_n(l_n), \mathrm{R}_1(r_1) \ldots \mathrm{R}_{j-1}(r_{j-1}), \mathrm{X}, h, \mathrm{H}) \\
&\approx P_h(\mathrm{H} \,|\, \mathrm{X}, h) \times \prod_{i=1}^{n+1} P_l(\mathrm{L}_i(l_i) \,|\, \mathrm{X}, h, \mathrm{H}) \times \prod_{j=1}^{m+1} P_r(\mathrm{R}_j(r_j) \,|\, \mathrm{X}, h, \mathrm{H})
\end{aligned}
$$

Generate head constituent

Generate left modifiers (stop at STOP)

Generate right modifiers (stop at STOP)

By independence assumption

Probabilities of internal rules 2

Example:

$$P(\ \mathrm{S}(\textit{bought},\mathrm{VBD}) \rightarrow \mathrm{NP}(\textit{week},\mathrm{NN})\ \mathrm{NP}(\textit{IBM},\mathrm{NNP})\ \mathrm{VP}(\textit{bought},\mathrm{VBD})\ )$$

$$\approx P_h(\mathrm{VP} \mid \mathrm{S}, \textit{bought},\mathrm{VBD})$$

Generate head constituent

$$\times P_l(\mathrm{NP}(\textit{IBM},\mathrm{NNP}) \mid \mathrm{S}, \textit{bought},\mathrm{VBD}, \mathrm{VP})$$
$$\times P_l(\mathrm{NP}(\textit{week},\mathrm{NN}) \mid \mathrm{S}, \textit{bought},\mathrm{VBD}, \mathrm{VP})$$
$$\times P_l(\mathrm{STOP} \mid \mathrm{S}, \textit{bought},\mathrm{VBD}, \mathrm{VP})$$

Generate left modifiers

$$\times P_r(\mathrm{STOP} \mid \mathrm{S}, \textit{bought},\mathrm{VBD}, \mathrm{VP})$$

Generate right modifiers

Collins' first model 3

Backs off …

- to tag probability when no data for specific word;
- to complete non-lexicalization when necessary.

Collins' 2^{nd} and 3^{rd} models

Model 2: Add verb subcategorization and argument/adjunct distinction.

Model 3: Integrate gaps into model.

- Especially important with addition of subcategorization.

Results and conclusions 3

Model 2 outperforms Model 1.

Model 3: Similar performance, but identifies traces too.

Model 2 performs best overall:

- LP = 89.6, LR = 89.9 [sentences ≤ 100 words].
- LP = 90.1, LR = 90.4 [sentences ≤ 40 words].

Rich information improves parsing performance.

Results and conclusions 2

Strengths:

- Incorporation of lexical and other linguistic information.
- Competitive results.

Weaknesses:

- Supervised training.
- Performance tightly linked to particular type of corpus used.

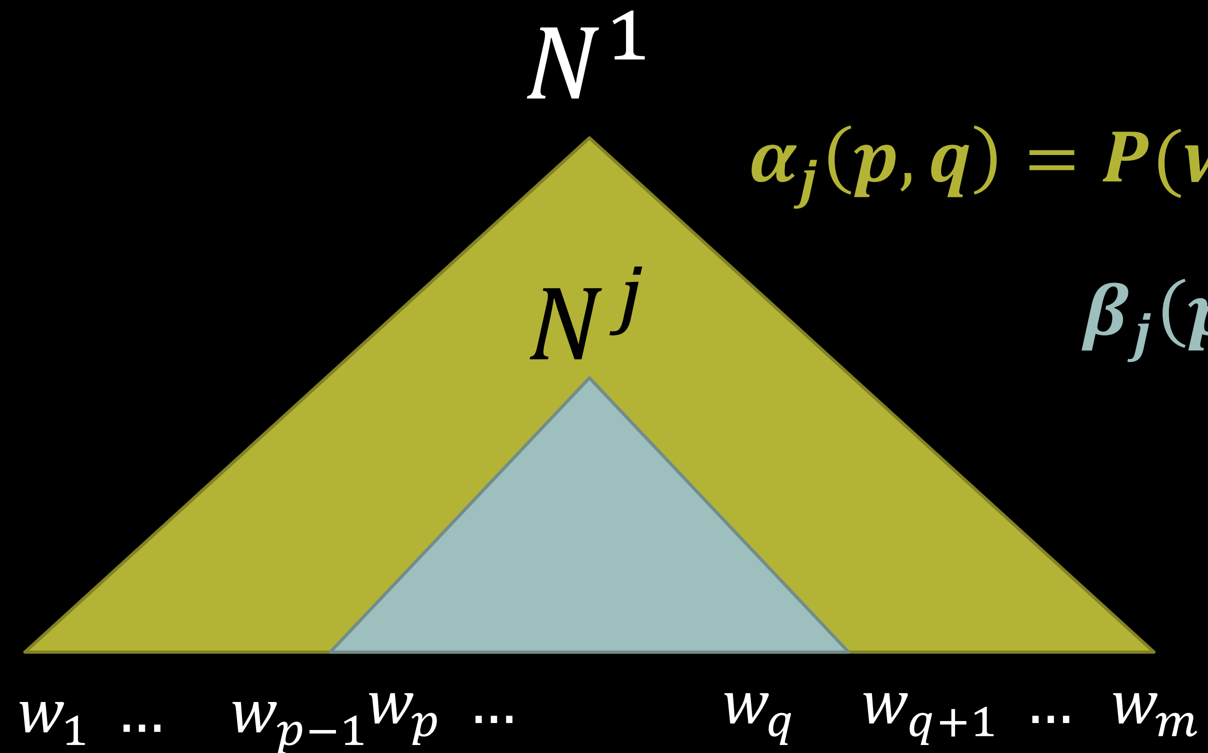
Results and conclusions 3

Importance to CL:

- High-performance parser showing benefits of lexicalization and linguistic information.
- Publicly available, widely used in research.

Unsupervised inside-outside

The **inside**-**outside** algorithm is a **glorious** generalization of the forward-backward algorithm that learns grammars ***without*** annotated data.



$$\boldsymbol{\alpha_j(p,q) = P(w_{1:p-1}, N^j_{pq}, w_{q+1:m} | G)}$$

$$\boldsymbol{\beta_j(p,q) = P(w_{p:q} | N^j_{pq}, G)}$$

Manning & Schütze: 11.3.1-11.3.5