

6.864 (Fall 2006): Lecture 3
Smoothed Estimation, and Language Modeling

1

The Language Modeling Problem

- We have some vocabulary,
say $\mathcal{V} = \{\text{the, a, man, telescope, Beckham, two, \dots}\}$

- We have an (infinite) set of strings, \mathcal{V}^*

the
a
the fan
the fan saw Beckham
the fan saw saw
...
the fan saw Beckham play for Real Madrid
...

3

Overview

- The language modeling problem
- Smoothed “n-gram” estimates

2

The Language Modeling Problem (Continued)

- We have a *training sample* of example sentences in English
- We need to “learn” a probability distribution \hat{P}
i.e., \hat{P} is a function that satisfies

$$\sum_{x \in \mathcal{V}^*} \hat{P}(x) = 1, \quad \hat{P}(x) \geq 0 \text{ for all } x \in \mathcal{V}^*$$

$\hat{P}(\text{the}) = 10^{-12}$
 $\hat{P}(\text{the fan}) = 10^{-8}$
 $\hat{P}(\text{the fan saw Beckham}) = 2 \times 10^{-8}$
 $\hat{P}(\text{the fan saw saw}) = 10^{-15}$
...
 $\hat{P}(\text{the fan saw Beckham play for Real Madrid}) = 2 \times 10^{-9}$
...

- Usual assumption: training sample is drawn from some underlying distribution P , we want \hat{P} to be “as close” to P as possible.

4

Why on earth would we want to do this?!

- **Speech recognition** was the original motivation.
(Related problems are optical character recognition, handwriting recognition.)
- The estimation techniques developed for this problem will be **VERY** useful for other problems in NLP

5

Deriving a Trigram Probability Model

Step 2: Make Markov independence assumptions:

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &= P(w_1 \mid \text{START}) \\ &\quad \times P(w_2 \mid \text{START}, w_1) \\ &\quad \times P(w_3 \mid w_1, w_2) \\ &\quad \dots \\ &\quad \times P(w_n \mid w_{n-2}, w_{n-1}) \\ &\quad \times P(\text{STOP} \mid w_{n-1}, w_n) \end{aligned}$$

General assumption:

$$P(w_i \mid \text{START}, w_1, w_2, \dots, w_{i-2}, w_{i-1}) = P(w_i \mid w_{i-2}, w_{i-1})$$

For Example

$$\begin{aligned} P(\text{the, dog, laughs}) &= P(\text{the} \mid \text{START}) \\ &\quad \times P(\text{dog} \mid \text{START, the}) \\ &\quad \times P(\text{laughs} \mid \text{the, dog}) \\ &\quad \times P(\text{STOP} \mid \text{dog, laughs}) \end{aligned}$$

7

Deriving a Trigram Probability Model

Step 1: Expand using the chain rule:

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &= P(w_1 \mid \text{START}) \\ &\quad \times P(w_2 \mid \text{START}, w_1) \\ &\quad \times P(w_3 \mid \text{START}, w_1, w_2) \\ &\quad \times P(w_4 \mid \text{START}, w_1, w_2, w_3) \\ &\quad \dots \\ &\quad \times P(w_n \mid \text{START}, w_1, w_2, \dots, w_{n-1}) \\ &\quad \times P(\text{STOP} \mid \text{START}, w_1, w_2, \dots, w_{n-1}, w_n) \end{aligned}$$

For Example

$$\begin{aligned} P(\text{the, dog, laughs}) &= P(\text{the} \mid \text{START}) \\ &\quad \times P(\text{dog} \mid \text{START, the}) \\ &\quad \times P(\text{laughs} \mid \text{START, the, dog}) \\ &\quad \times P(\text{STOP} \mid \text{START, the, dog, laughs}) \end{aligned}$$

6

The Trigram Estimation Problem

Remaining estimation problem:

$$P(w_i \mid w_{i-2}, w_{i-1})$$

For example:

$$P(\text{laughs} \mid \text{the, dog})$$

A natural estimate (the “maximum likelihood estimate”):

$$\begin{aligned} P_{ML}(w_i \mid w_{i-2}, w_{i-1}) &= \frac{\text{Count}(w_i, w_{i-2}, w_{i-1})}{\text{Count}(w_{i-2}, w_{i-1})} \\ P_{ML}(\text{laughs} \mid \text{the, dog}) &= \frac{\text{Count}(\text{the, dog, laughs})}{\text{Count}(\text{the, dog})} \end{aligned}$$

8

Evaluating a Language Model

- We have some test data, n sentences

$$S_1, S_2, S_3, \dots, S_n$$

- We could look at the probability under our model $\prod_{i=1}^n P(S_i)$.
Or more conveniently, the *log probability*

$$\log \prod_{i=1}^n P(S_i) = \sum_{i=1}^n \log P(S_i)$$

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-x} \quad \text{where} \quad x = \frac{1}{W} \sum_{i=1}^n \log P(S_i)$$

and W is the total number of words in the test data.

9

Some History

- Shannon conducted experiments on entropy of English
i.e., how good are people at the perplexity game?

C. Shannon. Prediction and entropy of printed English. Bell Systems Technical Journal, 30:50–64, 1951.

11

Some Intuition about Perplexity

- Say we have a vocabulary \mathcal{V} , of size $N = |\mathcal{V}|$
and model that predicts

$$P(w) = \frac{1}{N}$$

for all $w \in \mathcal{V}$.

- Easy to calculate the perplexity in this case:

$$\text{Perplexity} = 2^{-x} \quad \text{where} \quad x = \log \frac{1}{N}$$

\Rightarrow

$$\text{Perplexity} = N$$

Perplexity is a measure of effective “branching factor”

10

Some History

- Chomsky (in *Syntactic Structures* (1957)):

Second, the notion “grammatical” cannot be identified with “meaningful” or “significant” in any semantic sense. Sentences (1) and (2) are equally nonsensical, but any speaker of English will recognize that only the former is grammatical.

(1) Colorless green ideas sleep furiously.

(2) Furiously sleep ideas green colorless.

...

... Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order of statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticality, these sentences will be ruled out on identical grounds as equally ‘remote’ from English. Yet (1), though nonsensical, is grammatical, while (2) is not. ...

(my emphasis)

12

Sparse Data Problems

A natural estimate (the “maximum likelihood estimate”):

$$P_{ML}(w_i | w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$P_{ML}(\text{laughs} | \text{the, dog}) = \frac{\text{Count}(\text{the, dog, laughs})}{\text{Count}(\text{the, dog})}$$

Say our vocabulary size is $N = |\mathcal{V}|$,
then there are N^3 parameters in the model.

e.g., $N = 20,000 \Rightarrow 20,000^3 = 8 \times 10^{12}$ parameters

13

Linear Interpolation

- Take our estimate $\hat{P}(w_i | w_{i-2}, w_{i-1})$ to be

$$\begin{aligned}\hat{P}(w_i | w_{i-2}, w_{i-1}) = & \lambda_1 \times P_{ML}(w_i | w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times P_{ML}(w_i | w_{i-1}) \\ & + \lambda_3 \times P_{ML}(w_i)\end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i .

15

The Bias-Variance Trade-Off

- (Unsmoothed) trigram estimate

$$P_{ML}(w_i | w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

- (Unsmoothed) bigram estimate

$$P_{ML}(w_i | w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

- (Unsmoothed) unigram estimate

$$P_{ML}(w_i) = \frac{\text{Count}(w_i)}{\text{Count}()}$$

How close are these different estimates to the “true” probability
 $P(w_i | w_{i-2}, w_{i-1})$?

14

- Our estimate correctly defines a distribution:

$$\begin{aligned}\sum_{w \in \mathcal{V}} \hat{P}(w | w_{i-2}, w_{i-1}) &= \sum_{w \in \mathcal{V}} [\lambda_1 \times P_{ML}(w | w_{i-2}, w_{i-1}) + \lambda_2 \times P_{ML}(w | w_{i-1}) + \lambda_3 \times P_{ML}(w)] \\ &= \lambda_1 \sum_w P_{ML}(w | w_{i-2}, w_{i-1}) + \lambda_2 \sum_w P_{ML}(w | w_{i-1}) + \lambda_3 \sum_w P_{ML}(w) \\ &= \lambda_1 + \lambda_2 + \lambda_3 \\ &= 1\end{aligned}$$

(Can show also that $\hat{P}(w | w_{i-2}, w_{i-1}) \geq 0$ for all $w \in \mathcal{V}$)

16

How to estimate the λ values?

- Hold out part of training set as “validation” data
- Define $\text{Count}_2(w_1, w_2, w_3)$ to be the number of times the trigram (w_1, w_2, w_3) is seen in validation set
- Choose $\lambda_1, \lambda_2, \lambda_3$ to maximize:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1, w_2, w_3 \in \mathcal{V}} \text{Count}_2(w_1, w_2, w_3) \log \hat{P}(w_3 | w_1, w_2)$$

such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i , and where

$$\begin{aligned} \hat{P}(w_i | w_{i-2}, w_{i-1}) = & \lambda_1 \times P_{ML}(w_i | w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times P_{ML}(w_i | w_{i-1}) \\ & + \lambda_3 \times P_{ML}(w_i) \end{aligned}$$

17

Allowing the λ 's to vary

- Take a function Φ that partitions histories
e.g.,

$$\Phi(w_{i-2}, w_{i-1}) = \begin{cases} 1 & \text{If } \text{Count}(w_{i-1}, w_{i-2}) = 0 \\ 2 & \text{If } 1 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 2 \\ 3 & \text{If } 3 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 5 \\ 4 & \text{Otherwise} \end{cases}$$

- Introduce a dependence of the λ 's on the partition:

$$\begin{aligned} \hat{P}(w_i | w_{i-2}, w_{i-1}) = & \lambda_1^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w_i | w_{i-2}, w_{i-1}) \\ & + \lambda_2^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w_i | w_{i-1}) \\ & + \lambda_3^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w_i) \end{aligned}$$

where $\lambda_1^{\Phi(w_{i-2}, w_{i-1})} + \lambda_2^{\Phi(w_{i-2}, w_{i-1})} + \lambda_3^{\Phi(w_{i-2}, w_{i-1})} = 1$, and $\lambda_i^{\Phi(w_{i-2}, w_{i-1})} \geq 0$ for all i .

19

An Iterative Method

Initialization: Pick arbitrary/random values for $\lambda_1, \lambda_2, \lambda_3$.

Step 1: Calculate the following quantities:

$$c_1 = \sum_{w_1, w_2, w_3 \in \mathcal{V}} \frac{\text{Count}_2(w_1, w_2, w_3) \lambda_1 P_{ML}(w_3 | w_1, w_2)}{\lambda_1 P_{ML}(w_3 | w_1, w_2) + \lambda_2 P_{ML}(w_3 | w_2) + \lambda_3 P_{ML}(w_3)}$$

$$c_2 = \sum_{w_1, w_2, w_3 \in \mathcal{V}} \frac{\text{Count}_2(w_1, w_2, w_3) \lambda_2 P_{ML}(w_3 | w_2)}{\lambda_1 P_{ML}(w_3 | w_1, w_2) + \lambda_2 P_{ML}(w_3 | w_2) + \lambda_3 P_{ML}(w_3)}$$

$$c_3 = \sum_{w_1, w_2, w_3 \in \mathcal{V}} \frac{\text{Count}_2(w_1, w_2, w_3) \lambda_3 P_{ML}(w_3)}{\lambda_1 P_{ML}(w_3 | w_1, w_2) + \lambda_2 P_{ML}(w_3 | w_2) + \lambda_3 P_{ML}(w_3)}$$

Step 2: Re-estimate λ_i 's as

$$\lambda_1 = \frac{c_1}{c_1 + c_2 + c_3}, \quad \lambda_2 = \frac{c_2}{c_1 + c_2 + c_3}, \quad \lambda_3 = \frac{c_3}{c_1 + c_2 + c_3}$$

Step 3: If λ_i 's have not converged, go to **Step 1**.

18

- Our estimate correctly defines a distribution:

$$\begin{aligned} & \sum_{w \in \mathcal{V}} \hat{P}(w | w_{i-2}, w_{i-1}) \\ & = \sum_{w \in \mathcal{V}} \left[\lambda_1^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w | w_{i-2}, w_{i-1}) \right. \\ & \quad \left. + \lambda_2^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w | w_{i-1}) \right. \\ & \quad \left. + \lambda_3^{\Phi(w_{i-2}, w_{i-1})} \times P_{ML}(w) \right] \\ & = \lambda_1^{\Phi(w_{i-2}, w_{i-1})} \sum_w P_{ML}(w | w_{i-2}, w_{i-1}) \\ & \quad + \lambda_2^{\Phi(w_{i-2}, w_{i-1})} \sum_w P_{ML}(w | w_{i-1}) \\ & \quad + \lambda_3^{\Phi(w_{i-2}, w_{i-1})} \sum_w P_{ML}(w) \\ & = \lambda_1^{\Phi(w_{i-2}, w_{i-1})} + \lambda_2^{\Phi(w_{i-2}, w_{i-1})} + \lambda_3^{\Phi(w_{i-2}, w_{i-1})} \\ & = 1 \end{aligned}$$

20

An Alternative Definition of the λ 's

- A small change: take our estimate $\hat{P}(w_i | w_{i-2}, w_{i-1})$ to be

$$\hat{P}(w_i | w_{i-2}, w_{i-1}) =$$

$$\lambda_1 \times P_{ML}(w_i | w_{i-2}, w_{i-1})$$

$$+ (1 - \lambda_1)[\lambda_2 \times P_{ML}(w_i | w_{i-1}) + (1 - \lambda_2) \times P_{ML}(w_i)]$$

where $0 \leq \lambda_1 \leq 1$, and $0 \leq \lambda_2 \leq 1$.

- Next, define

$$\lambda_1 = \frac{\text{Count}(w_{i-2}, w_{i-1})}{\alpha + \text{Count}(w_{i-2}, w_{i-1})} \quad \lambda_2 = \frac{\text{Count}(w_{i-1})}{\alpha + \text{Count}(w_{i-1})}$$

where α is a parameter chosen to optimize probability of a development set.

Discounting Methods

- Say we've seen the following counts:

x	Count(x)	$P_{ML}(w_i w_{i-1})$
the	48	
the, dog	15	15/48
the, woman	11	11/48
the, man	10	10/48
the, park	5	5/48
the, job	2	2/48
the, telescope	1	1/48
the, manual	1	1/48
the, afternoon	1	1/48
the, country	1	1/48
the, street	1	1/48

- The maximum-likelihood estimates are systematically high (particularly for low count items)

An Alternative Definition of the λ 's (continued)

- Define

$$U(w_{i-2}, w_{i-1}) = |\{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}|$$

$$U(w_{i-1}) = |\{w : \text{Count}(w_{i-1}, w) > 0\}|$$

- Next, define

$$\lambda_1 = \frac{\text{Count}(w_{i-2}, w_{i-1})}{\alpha U(w_{i-2}, w_{i-1}) + \text{Count}(w_{i-2}, w_{i-1})}$$

$$\lambda_2 = \frac{\text{Count}(w_{i-1})}{\alpha U(w_{i-1}) + \text{Count}(w_{i-1})}$$

where α is a parameter chosen to optimize probability of a development set.

Discounting Methods

- Now define "discounted" counts, for example (a first, simple definition):

$$\text{Count}^*(x) = \text{Count}(x) - 0.5$$

- New estimates:

x	Count(x)	Count*(x)	$\frac{\text{Count}^*(x)}{\text{Count}(x)}$
the	48		
the, dog	15	14.5	14.5/48
the, woman	11	10.5	10.5/48
the, man	10	9.5	9.5/48
the, park	5	4.5	4.5/48
the, job	2	1.5	1.5/48
the, telescope	1	0.5	0.5/48
the, manual	1	0.5	0.5/48
the, afternoon	1	0.5	0.5/48
the, country	1	0.5	0.5/48
the, street	1	0.5	0.5/48

- We now have some “missing probability mass”:

$$\alpha(w_{i-1}) = 1 - \sum_w \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

e.g., in our example, $\alpha(\text{the}) = 10 \times 0.5/48 = 5/48$

- Divide the remaining probability mass between words w for which $\text{Count}(w_{i-1}, w) = 0$.

25

Katz Back-Off Models (Trigrams)

- For a trigram model, first define two sets

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model is defined in terms of the bigram model:

$$P_{KATZ}(w_i | w_{i-2}, w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \frac{\alpha(w_{i-2}, w_{i-1}) P_{KATZ}(w_i | w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} P_{KATZ}(w | w_{i-1})} & \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

27

Katz Back-Off Models (Bigrams)

- For a bigram model, define two sets

$$\mathcal{A}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

- A bigram model

$$P_{KATZ}(w_i | w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{P_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} P_{ML}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

26

Good-Turing Discounting

- Invented during WWII by Alan Turing (and Good?), later published by Good. Frequency estimates were needed within the Enigma code-breaking effort.

- Define n_r = number of elements x for which $\text{Count}(x) = r$.

- Modified count for any x with $\text{Count}(x) = r$ and $r > 0$:

$$(r + 1) \frac{n_{r+1}}{n_r}$$

- Leads to the following estimate of “missing mass”:

$$\frac{n_1}{N}$$

where N is the size of the sample. This is the estimate of the probability of seeing a new element x on the $(N + 1)$ 'th draw.

28

Summary

- Three steps in deriving the language model probabilities:

1. Expand $P(w_1, w_2 \dots w_n)$ using **Chain rule**.

2. Make **Markov Independence Assumptions**

$$P(w_i | w_1, w_2 \dots w_{i-2}, w_{i-1}) = P(w_i | w_{i-2}, w_{i-1})$$

3. **Smooth** the estimates using low order counts.

- Other methods used to improve language models:

- “Topic” or “long-range” features.
- Syntactic models.

It's generally hard to improve on trigram models, though!!

29

A Probabilistic Context-Free Grammar

S	⇒	NP VP	1.0
VP	⇒	Vi	0.4
VP	⇒	Vt NP	0.4
VP	⇒	VP PP	0.2
NP	⇒	DT NN	0.3
NP	⇒	NP PP	0.7
PP	⇒	P NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5

- Probability of a tree with rules $\alpha_i \rightarrow \beta_i$ is $\prod_i P(\alpha_i \rightarrow \beta_i | \alpha_i)$

31

Further Reading

See:

“An Empirical Study of Smoothing Techniques for Language Modeling”. Stanley Chen and Joshua Goodman. 1998. Harvard Computer Science Technical report TR-10-98.

(Gives a very thorough evaluation and description of a number of methods.)

“On the Convergence Rate of Good-Turing Estimators”. David McAllester and Robert E. Schapire. In Proceedings of COLT 2000. (A pretty technical paper, giving confidence-intervals on Good-Turing estimators. Theorems 1, 3 and 9 are useful in understanding the motivation for Good-Turing discounting.)

30

DERIVATION	RULES USED	PROBABILITY
S	S → NP VP	1.0
NP VP	NP → DT N	0.3
DT N VP	DT → the	1.0
the N VP	N → dog	0.1
the dog VP	VP → VB	0.4
the dog VB	VB → laughs	0.5
the dog laughs		

$$\text{TOTAL PROBABILITY} = 1.0 \times 0.3 \times 1.0 \times 0.1 \times 0.4 \times 0.5$$

32

Properties of PCFGs

- Assigns a probability to each *left-most derivation*, or parse-tree, allowed by the underlying CFG
- Say we have a sentence S , set of derivations for that sentence is $\mathcal{T}(S)$. Then a PCFG assigns a probability to each member of $\mathcal{T}(S)$. i.e., *we now have a ranking in order of probability*.
- The probability of a string S is

$$\sum_{T \in \mathcal{T}(S)} P(T, S)$$

33

PCFGs

Booth and Thompson (73) showed that a CFG with rule probabilities correctly defines a distribution over the set of derivations provided that:

1. The rule probabilities define conditional distributions over the different ways of rewriting each non-terminal.
2. A technical condition on the rule probabilities ensuring that the probability of the derivation terminating in a finite number of steps is 1. (This condition is not really a practical concern.)

35

Deriving a PCFG from a Corpus

- Given a set of example trees, the underlying CFG can simply be **all rules seen in the corpus**
- Maximum Likelihood estimates:

$$P_{ML}(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

where the counts are taken from a training set of example trees.

- **If the training data is generated by a PCFG**, then as the training data size goes to infinity, the maximum-likelihood PCFG will converge to the same distribution as the ‘true’ PCFG.

34

Algorithms for PCFGs

- Given a PCFG and a sentence S , define $\mathcal{T}(S)$ to be the set of trees with S as the yield.
- Given a PCFG and a sentence S , how do we find

$$\arg \max_{T \in \mathcal{T}(S)} P(T, S)$$

- Given a PCFG and a sentence S , how do we find

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T, S)$$

36

Chomsky Normal Form

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of two forms:
 - $X \rightarrow Y_1Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
 - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

37

A Dynamic Programming Algorithm

- Base case definition: for all $i = 1 \dots n$, for $k = 1 \dots K$

$$\pi[i, i, k] = P(N_k \rightarrow w_i | N_k)$$

(note: define $P(N_k \rightarrow w_i | N_k) = 0$ if $N_k \rightarrow w_i$ is not in the grammar)

- Recursive definition: for all $i = 1 \dots n, j = (i + 1) \dots n, k = 1 \dots K$,

$$\pi[i, j, k] = \max_{\substack{i \leq s < j \\ 1 \leq l \leq K \\ 1 \leq m \leq K}} \{P(N_k \rightarrow N_l N_m | N_k) \times \pi[i, s, l] \times \pi[s + 1, j, m]\}$$

(note: define $P(N_k \rightarrow N_l N_m | N_k) = 0$ if $N_k \rightarrow N_l N_m$ is not in the grammar)

39

A Dynamic Programming Algorithm

- Given a PCFG and a sentence S , how do we find

$$\max_{T \in \mathcal{T}(S)} P(T, S)$$

- Notation:

n = number of words in the sentence
 N_k for $k = 1 \dots K$ is k 'th non-terminal
 w.l.g., $N_1 = S$ (the start symbol)

- Define a dynamic programming table

$\pi[i, j, k]$ = maximum probability of a constituent with non-terminal N_k
 spanning words $i \dots j$ inclusive

- Our goal is to calculate $\max_{T \in \mathcal{T}(S)} P(T, S) = \pi[1, n, 1]$

38

Initialization:

For $i = 1 \dots n, k = 1 \dots K$

$$\pi[i, i, k] = P(N_k \rightarrow w_i | N_k)$$

Main Loop:

For $length = 1 \dots (n - 1), i = 1 \dots (n - length), k = 1 \dots K$

$j \leftarrow i + length$

$max \leftarrow 0$

For $s = i \dots (j - 1)$,

For N_l, N_m such that $N_k \rightarrow N_l N_m$ is in the grammar

$prob \leftarrow P(N_k \rightarrow N_l N_m) \times \pi[i, s, l] \times \pi[s + 1, j, m]$

If $prob > max$

$max \leftarrow prob$

//Store backpointers which imply the best parse

$Split(i, j, k) = \{s, l, m\}$

$\pi[i, j, k] = max$

40

A Dynamic Programming Algorithm for the Sum

- Given a PCFG and a sentence S , how do we find

$$\sum_{T \in \mathcal{T}(S)} P(T, S)$$

- Notation:

n = number of words in the sentence
 N_k for $k = 1 \dots K$ is k 'th non-terminal
 w.l.g., $N_1 = S$ (the start symbol)

- Define a dynamic programming table

$\pi[i, j, k]$ = sum of probability of parses with root label N_k
 spanning words $i \dots j$ inclusive

- Our goal is to calculate $\sum_{T \in \mathcal{T}(S)} P(T, S) = \pi[1, n, 1]$

41

Initialization:

For $i = 1 \dots n, k = 1 \dots K$
 $\pi[i, i, k] = P(N_k \rightarrow w_i | N_k)$

Main Loop:

For $length = 1 \dots (n - 1), i = 1 \dots (n - length), k = 1 \dots K$
 $j \leftarrow i + length$
 $sum \leftarrow 0$
 For $s = i \dots (j - 1)$,
 For N_l, N_m such that $N_k \rightarrow N_l N_m$ is in the grammar
 $prob \leftarrow P(N_k \rightarrow N_l N_m) \times \pi[i, s, l] \times \pi[s + 1, j, m]$
 $sum \leftarrow sum + prob$
 $\pi[i, j, k] = sum$

43

A Dynamic Programming Algorithm for the Sum

- Base case definition: for all $i = 1 \dots n$, for $k = 1 \dots K$

$$\pi[i, i, k] = P(N_k \rightarrow w_i | N_k)$$

(note: define $P(N_k \rightarrow w_i | N_k) = 0$ if $N_k \rightarrow w_i$ is not in the grammar)

- Recursive definition: for all $i = 1 \dots n, j = (i + 1) \dots n, k = 1 \dots K$,

$$\pi[i, j, k] = \sum_{\substack{i \leq s < j \\ 1 \leq l \leq K \\ 1 \leq m \leq K}} \{P(N_k \rightarrow N_l N_m | N_k) \times \pi[i, s, l] \times \pi[s + 1, j, m]\}$$

(note: define $P(N_k \rightarrow N_l N_m | N_k) = 0$ if $N_k \rightarrow N_l N_m$ is not in the grammar)

42