

Natural Language Processing
Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture-23
Syntax – CKY, PCFGS

So, welcome back for the third lecture of this week. In the last lecture, we had discussed top-down and bottom-up parsing approach and then finally, we came up with the dynamic programming approach of using CKY algorithm; and we converted a grammar to Chomsky normal form. Now, once my grammar is converted to Chomsky normal form how does CKY algorithm work, so this is the idea.

(Refer Slide Time: 00:44)

CKY Algorithm

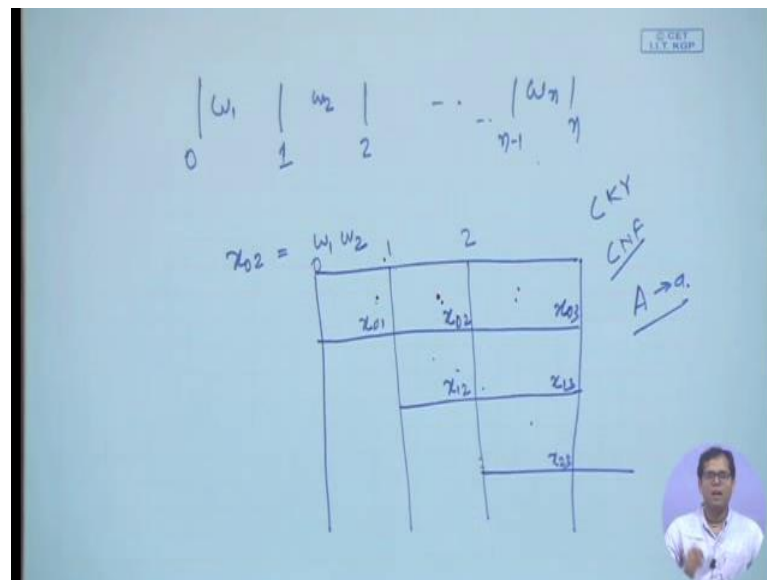
- Let n be the number of words in the input. Think about $n + 1$ lines separating them, numbered 0 to n .
- x_{ij} will denote the words between line i and j
- We build a table so that x_{ij} contains all the possible non-terminal spanning for words between line i and j .
- We build the Table bottom-up.

Home Exercise
Use CKY algorithm to find the parse tree for "Book the flight through Houston" using the CNF form shown in the previous slide.

Pawan Goyal (IIT Kharagpur) Syntax Week 5: Lecture 23

So, I will take a sentence; I have n words in my sentence. So, you will think about n plus one line that are separating them, is starting from 0 to 1. And so once you have done that any x_{ij} you will denote words between line i and j . So, you will build up a table such that any x_{ij} will contains all the possible non-terminal that can derive words between the lines i and j . And you do that bottom-up.

(Refer Slide Time: 01:23)



So, what I am saying, you have word 1, word 2 up to word n in your input stream. So, you assume some lines like 0, 1, 2, n minus 1, n . So, for example, x_{02} is words w_1, w_2 and so on. So, now what will you do? You build up a table and table will be some sort of triangular table. So, this can be 0, 1, 2; so this element will denote x_{01} , what are all the non-terminals that derive this word between 0 and 1. Similarly, here you will write x_{12} and so on. Suppose, they want a three words, this is x_{23} . So, we will write down all the non-terminals that derive this. You first fill this, then you will go next step, x_{02} , x_{13} . And once you fill this, you will use x_{03} – the final one.

Now, what is the, where have we using the fact that this grammar is in CNF - Chomsky normal form, I will make sure the fact that at any point, this can come from only two non-terminals or a single terminal. So, at this point, when I am seeing as an individual word, so 0 1 will always be a individual word, similarly for 1 and 2. So, this will come from a rule capital A goes to small a , a rule of this kind. So, I will find out all the non-terminals that derives this terminal that is how I will fill the diagonal elements, the first diagonal elements.

Next time, this cannot come from a single terminal, because they are two words. So, this has to come from two non-terminals. So, I will find out if there is a rule that gives me these two non-terminals and that is why I am using Chomsky normal form. So, finally,

when I am here, I will see if the sentences generate the whole sentence. So, I am catching all the possible intermediate steps here, so that is I have to do certain again and again.

So, this is the simple home exercise, so in the in the last lecture, we had given you a grammar in Chomsky normal form. Now, take the sentence, “Book the flight through Houston” and use the CKY algorithm to find the parse tree for that. So, but what I will do? I will do an example in today’s class also so that it becomes more much more comfortable with using CKY algorithm.

(Refer Slide Time: 04:32)

CKY for CFG					
a	pilot	likes	flying	planes	
1	2	3	4	5	
DT	NP	-	-	S S	
	NN	-	-	-	
		VBZ	-	VP	
				VP	
			JJ	NP	
			VBG	VP	
				NNS	

$S \rightarrow NP VP$
 $VP \rightarrow VBG NNS$
 $VP \rightarrow VBZ VP$
 $VP \rightarrow VBZ NP$
 $NP \rightarrow DT NN$
 $NP \rightarrow JJ NNS$
 $DT \rightarrow a$
 $NN \rightarrow pilot$
 $VBZ \rightarrow likes$
 $VBG \rightarrow flying$
 $JJ \rightarrow flying$
 $NNS \rightarrow planes$

So, let us take this example, a pilot likes flying planes and the grammar is given to you.

(Refer Slide Time: 04:46)

The image shows a handwritten dynamic programming table for parsing the sentence "a pilot likes flying planes". The table is a 6x6 grid with indices 0 to 5. The words are: 0: a, 1: pilot, 2: likes, 3: flying, 4: planes, 5: (empty). The table contains non-terminal symbols and their corresponding word indices. To the left of the table, there are handwritten calculations for the DP values: $x_{03} = x_{01} x_{13}$, $x_{02} = x_{01} x_{12}$, $x_{14} = x_{12} x_{24}$, $x_{13} = x_{12} x_{23}$, $x_{04} = x_{01} x_{14}$, $x_{02} = x_{01} x_{12}$, $x_{03} = x_{01} x_{13}$. To the right, there are more calculations: $x_{25} = x_{23} x_{35}$, $x_{24} = x_{23} x_{34}$, $x_{25} = x_{23} x_{35}$, $x_{24} = x_{23} x_{34}$, $x_{25} = x_{23} x_{35}$, $x_{24} = x_{23} x_{34}$. At the bottom, there are calculations for the final DP values: $x_{05} = x_{01} x_{15}$, $x_{02} = x_{01} x_{12}$, $x_{03} = x_{01} x_{13}$, $x_{04} = x_{01} x_{14}$.

	0	1	2	3	4	5
0	DT	NP	—	—	—	S → NP VP
1	—	NN	—	—	—	—
2	—	—	VBZ	—	—	—
3	—	—	—	VBG	—	—
4	—	—	—	—	VP	—
5	—	—	—	—	—	NNS

So, let do that in the way I had explained, so 0, 1, 2, 3, 4, 5, there are five words. And this is the line. And now you should understand how do we denote these elements. So, this is x_{01} , this is denote on we the first word, this is x_{12} , x_{23} , x_{34} , x_{45} . This will be x_{02} – words between 0 to 2. x_{03} , x_{04} , x_{05} , x_{13} , x_{14} , x_{15} , x_{24} , x_{25} and x_{35} . Now, I have to fill here, what are different non-terminals that filled each of the individual elements, x_{01} .

What is x_{01} ? My word is a, so a pilot likes flying planes. So, is there a non-terminal that derives the word a. And if you see the grammar, DT derives a; so you can fill in DT here; DT derives a. Pilot, NN derives pilot. Likes, VBZ derives likes. Flying, so you there are two non-terminals VBG JJ they derive this; 'and' planes – NNS. So, filling the diagonal element is very easy, the first diagonal elements. Then you go the next step, x_{02} . Now, x_{02} can come from x_{01} and x_{12} , as a break up of these two points. So, is there an non-terminal where or is there a rule in my grammar when the right hand side, I have DT followed by NN. And if you see your grammar, yes, NP gives me DT followed by NN, so I can fill in NP here. x_{13} comes from x_{12} and x_{23} , yes x_{12} pilot likes, so it comes from pilot and likes, so any non-terminal that gives me NN followed by VBZ. And if you see, there is no non-terminal. So, I will fill in empty here.

Similarly, and if VBZ followed by VBG, no; and if anything from VBZ followed by JJ, x_{35} , VBG followed by NNS, yes, VP, and JJ followed by NNS – NP. So, there are two

possibilities. So, fine this row is done, this diagonal is done. Now, we go to next step. Now, how do I derive x_0^3 - a pilot likes. Now, that is why I am using the Chomsky normal form. I cannot derive it at a sequence of three non-terminals, because each individual non-terminal can give me at most, not at most exactly two non-terminals.

So, what are the two places, from which it can come? So, one possibilities I can break x_0^3 as x_0^1 and x_1^3 ; one word into two words or x_0^2 , x_2^3 , there are two possibilities. So, I have to check individually each of these possibilities, $x_0^1 x_1^3$; x_0^1 is DT followed by null. So, this is already gone; this is no non-terminal. x_0^2 is NP followed by, x_2^3 is VBZ, so NP followed by VBZ. Is there a non-terminal, when the right hand side I have NP followed by VBZ. And if you see your grammar, there is nothing. So, this is also null. So, here it is null, there is no non-terminal let me derive, a pilot likes.

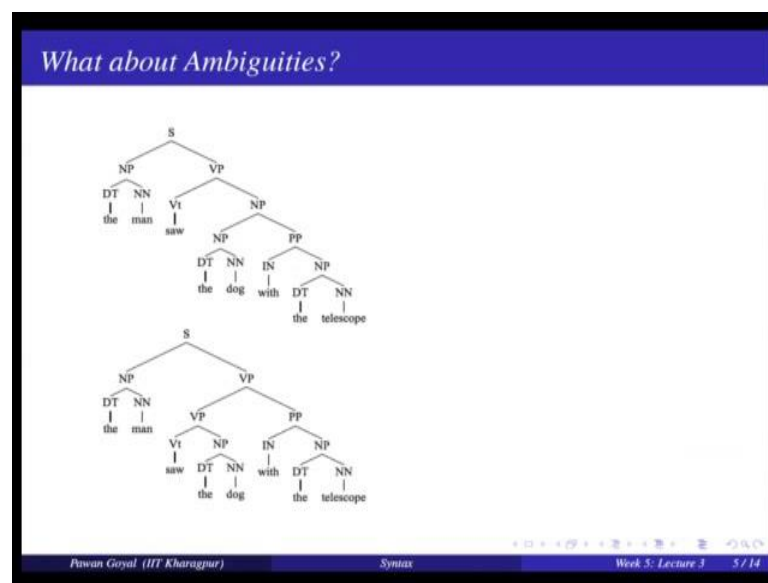
Now, pilot likes flying, 1 to 4. So, again 1 to 4 will be 1 to 2, 2 4, or 1 3, 3 4. So, 1 2, 2 4, 1 2 is NN, 2 4 is null – so this part is null. 1 3, 3 4, 1 3 is null, so this is becomes 1 also. 2 5 will be 2 3, 3 5, and 2 4, 4 5. So, 2 3 here is VBZ, and 3 5 is VP. Is there something that VBZ and VP, yes, VP this means VBZ and VP. 2 4, 4 5; 2 4 is null; so this is my VP. Similarly, now you will go to x_0^4 . Now, what are difference which you can break x_0^4 . Now, x_0^4 , again you to break in a sequence of two non-terminals, so it can be 0 1 followed by 1 4 or 0 2 followed by 2 4 or 0 3 followed by 3 4, there are three ways.

So, let us see one-by-one. 0 1 is DT, 1 4 is null, so this part is gone. 0 2 is empty and 2 4 is null, this part is also gone, 0 3 is null already, so this is null. 1 5, 1 5 can be 12 25, 13 35, 14 45. 12 and 25, yes, NN followed by a VP, so is this something in my grammar, NN followed by VP, no, so this is gone. Similarly, 13 followed by 35; 13 is null, and 13 is null, so this is gone. 14 followed by 45; 14 is null, so this is also gone, so this is also null. Now, the only thing remain is x_0^5 .

Now, how can I fill x_0^5 , what are different ways. So, that let me write down here. 05 can be 01 followed by 15; 02 followed by 25; 03 followed by 35; 04 followed by 45. 01 followed by 15; 01 is there, 15 is null, this is gone. 02 followed by 25; 02 is there and 25 is also there; this is NP followed by VP. And this is a sentence in my grammar. So, S gives me NP VP, so this is one possibility already, so; that means, this sentence is grammatical at least. There is one S that derives this, but are there any further S.

So, for that we have to look at other possibilities 03 35, this is null; and 04 45 is null, so fine. So, if you see there, I think we made one mistake here; there should have been another VP in this case. So, there is VP 1, VP 2. And then there will be S NP VP 1 and N P VP 2. So, I will search that look back into this calculation we did and see where we made a mistake. But everything else is the same that we did here. So, there are two different S in which this sentence can be parse tree, two different ways. So, now once you know this, I will say that use the previous example so that is Book the flight through Houston from the other grammar and try to get its parse tree.

(Refer Slide Time: 14:09)



So, now in this example, what we saw, there are two possibilities. So, can you think of the possibilities, why there are two possibilities in this case, a pilot likes flying planes. So, whether he likes to fly the plane or whether he likes to see flying planes something like that, so there are interpretation, that is why there are two different parse edge of this sentence. So, each individual parse will denote one particular interpretation. Now, by using this context by this CKY algorithm, we denote all the possible parse trees using my grammar. But I have no way of saying which parse is more probable than the other parse. I cannot assign some probabilities to them. So, something if I have so this is the sentence, the man saw the dog with the telescope, it has two different interpretation in terms of two different parse edges. Whatever I have covered till now, it cannot tell me which parse is more probable than other.

(Refer Slide Time: 15:15)

Probabilistic Context-free grammars (PCFGs)

PCFG: $G = (T, N, S, R, P)$

- T : set of terminals
- N : set of non-terminals
 - For NLP, we distinguish out a set $P \subset N$ of pre-terminals, which always rewrite as terminals
- S : start symbol
- R : Rules/productions of the form $X \rightarrow \gamma$, $X \in N$ and $\gamma \in (T \cup N)^*$
- $P(R)$ gives the probability of each rule.

$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

Pawan Goyal (IIT Kharagpur) Syntax Week 5: L

So, now we would like to have a way in which we can assign them the probabilities that this parse edge is more probable than the other. And for that what we use is called probabilistic context free grammars. So, this is simple extension of context free grammar, where in addition to whatever we have seen in context free grammar, each rule is also signed some probability. So, as you see in the formulation, it is T , N , S and R . They are exactly same as what we had seen in the context free grammar plus there is something called P .

So, I am assigning a probability distribution over the rules, and only the constraint here is that from a given non-terminal the lateral side the probability generating anything should add up to 1. So, if there are 5 possibilities, if I add all the 5 possibilities, the rule for all the 5 possibilities, they should add up to 1. So, this is the constraint. So, probability in the rule gives the gives the probability of each rule $P R$, so the constraint is for all X in non-terminals, probability of X to γ for all the possible γ s should add up to 1.

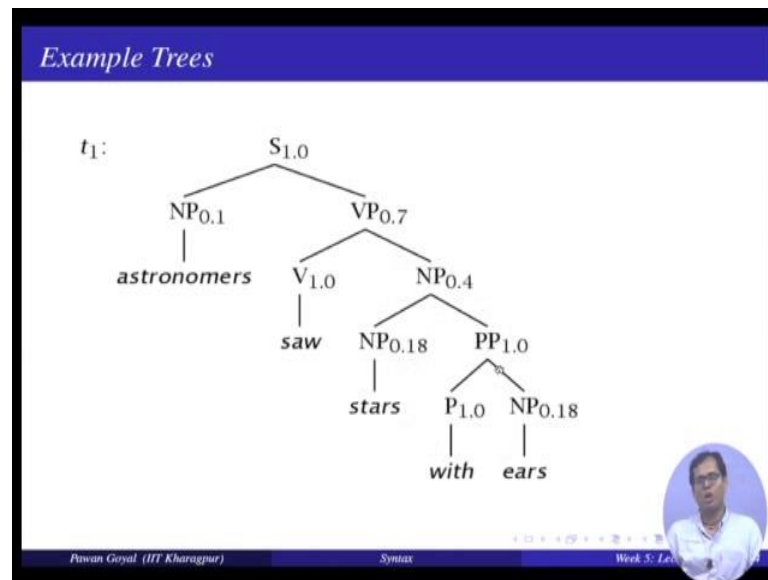
(Refer Slide Time: 16:31)

A Simple PCFG (in CNF)				
S	→	NP VP	1.0	
VP	→	V NP	0.7	
VP	→	VP PP	0.3	
PP	→	P NP	1.0	
P	→	with	1.0	
V	→	saw	1.0	
NP	→	NP PP	0.4	
NP	→	astronomers	0.1	
NP	→	ears	0.18	
NP	→	saw	0.04	
NP	→	stars	0.18	
NP	→	telescope	0.1	

So, let me give you an example. So, this is one simple CNF, sorry simple PCFG in Chomsky normal form. So, what do you see here? From S, there is only one rule, S goes to NP VP. So, because there is only one rule, it has a possibility of 1. From VP, there are two rules; VP can give, V followed by NP or VP followed by PP. So, the constraint is that for the possibilities of these two rules should add up to 1, so that is what is happening here. The first rule has a possibility of 0.7; second rule has a possibility of 0.3, these two add up to 1. Now, PP gives me P NP, only one rule with PP on left hand side, so this is the possibility of 1. P gives me only with again possibility 1; V gives me saw, this is the possibility 1.

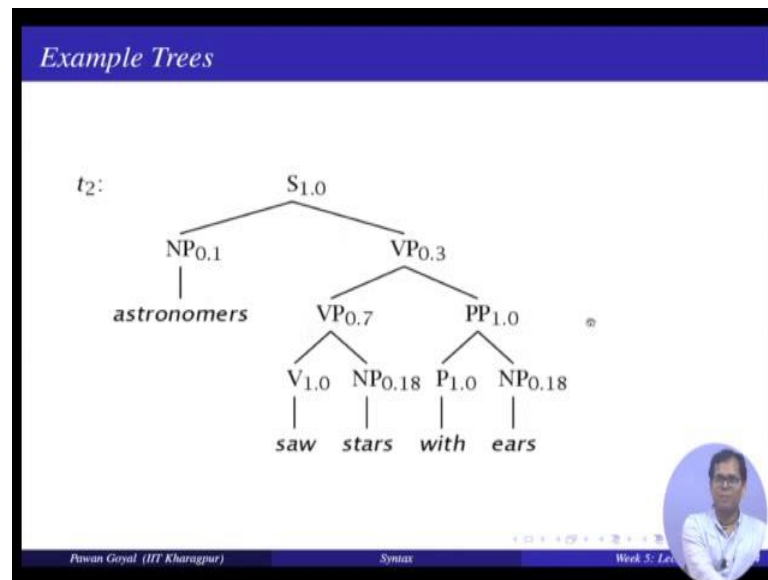
Now, all these rules in the right hand sides are starting from the NP. NP giving to me NP PP, all the words let us call as like astronomers, ears and so on. So, all these should add up to 1. And you can see that this actually happen 0.4 plus 0.1 – 0.5, 0.68, 0.72, 0.9 and 1. So, all these possibilities are adding up to 1. So, this is the constraint that is being followed. The rules have the same format as in context free grammar, but you shall have possibilities. Now, how does it help? It helps in that I can now assign possibilities to each individual parse tree.

(Refer Slide Time: 18:03)



So, suppose this is one parse tree. Astronomers for the sentence astronomers saw stars with ears. How do I find the probability of this parse tree? This is the parse. So, I know S gives me NP and VP, yes this is the first rule that I am applying. Now, as my PCFG, the probability of this rule is one. S giving NP, VP is 1, so I have this one here. NP giving astronomers, deriving astronomers is probability 0.1. VP deriving V and P is probability 0.7; V giving saw is 1. NP giving NP, V P is 0.4; NP giving stars is 0.18 and so on. These are the rule probability as per my grammar as my PCFG. So, what I will do? I will just multiply all these probabilities 1 times 0.1 times 0.7 times 0.1 times 0.4 times 0.18 times 0.18, this is my probabilities of this parse trees.

(Refer Slide Time: 19:03)



And if I get a second parse tree, where instead of VP giving me V and NP, VP giving VP followed by PP, I can again compute its probability by multiplying its corresponding rule probabilities. And I can find the probabilities of both the parse trees individually.

(Refer Slide Time: 19:21)

Probability of trees and strings

- $P(\phi)$: The probability of tree is the product of the probabilities of the rules used to generate it
- $P(w_{1:n})$: The probability of the string is the sum of the probabilities of the trees which have that string as their yield


Pawan Goyal (IIT Kharagpur) Syntax Week 5: Lecture 3 10 / 14

So, now how do I use that to compute the probability of the tree that is simply the product of the probabilities of all the rules that I used to generate this? And probability of assignments and the probability of sentences is nothing, but find out all the parse trees and the probabilities of the individual parse trees and just sum them up. So, probability

of this sentence is the sum of the probabilities of the trees that have this as their yield that is another way of saying that those parse trees are used to generate this particular strings.

(Refer Slide Time: 20:05)

Tree and String probabilities

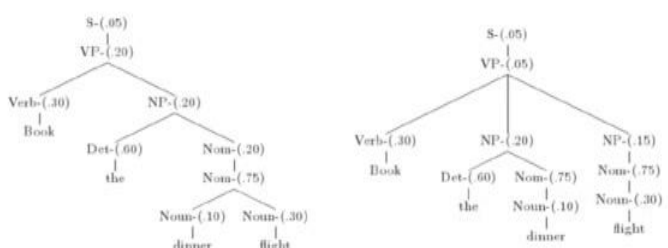
$$\begin{aligned}
 w_{15} &= \text{astronomers saw stars with ears} \\
 P(t_1) &= 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 \\
 &\quad * 1.0 * 1.0 * 0.18 \\
 &= 0.0009072 \\
 P(t_2) &= 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 \\
 &\quad * 1.0 * 1.0 * 0.18 \\
 &= 0.0006804 \\
 P(w_{15}) &= P(t_1) + P(t_2) \\
 &= 0.0009072 + 0.0006804 \\
 &= 0.0015876
 \end{aligned}$$


Pawan Goyal (IIT Kharagpur) Syntax Week 5: Lecture 3

So, in the first case, I had this sentence, and they were to parse trees, I can compute the probabilities of individual one. So, $P(t_1)$ and $P(t_2)$, I can find out and to find the probability of the whole sentence, I just add up these two probabilities $P(t_1)$ plus $P(t_2)$ and that gives me the probability of this whole sequence.

(Refer Slide Time: 20:27)

"Book the dinner flight"



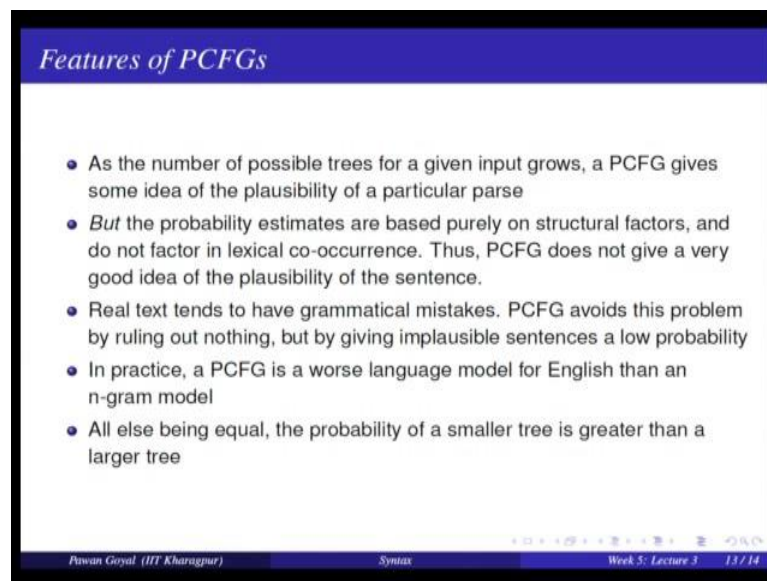
Probabilities

- Parse tree 1: $.05 \times .20 \times .30 \times .20 \times .60 \times .20 \times .75 \times .10 \times .30 = 1.62 \times 10^{-6}$
- Parse tree 2: $.05 \times .05 \times .30 \times .20 \times .60 \times .75 \times .10 \times .15 \times .75 \times .30 = 2.28 \times 10^{-7}$

Pawan Goyal (IIT Kharagpur) Syntax Week 5: Lecture 3 12 / 14

Similarly, if I have another sentence like book the dinner flight and as per the different grammar, I can compute the parse trees - the two parse trees. Compute the probabilities for the individual one. So, parse tree 1 here, book the dinner flight is 1.62×10^{-6} . And book the dinner flight will have a probability of 2.28×10^{-7} . So, one thing I can immediately see is that the first one parse is more likely interpretation than the second one.

(Refer Slide Time: 20:59)



Features of PCFGs

- As the number of possible trees for a given input grows, a PCFG gives some idea of the plausibility of a particular parse
- *But* the probability estimates are based purely on structural factors, and do not factor in lexical co-occurrence. Thus, PCFG does not give a very good idea of the plausibility of the sentence.
- Real text tends to have grammatical mistakes. PCFG avoids this problem by ruling out nothing, but by giving implausible sentences a low probability
- In practice, a PCFG is a worse language model for English than an n-gram model
- All else being equal, the probability of a smaller tree is greater than a larger tree

Pawan Goyal (IIT Kharagpur) Syntax Week 5: Lecture 3 13 / 14

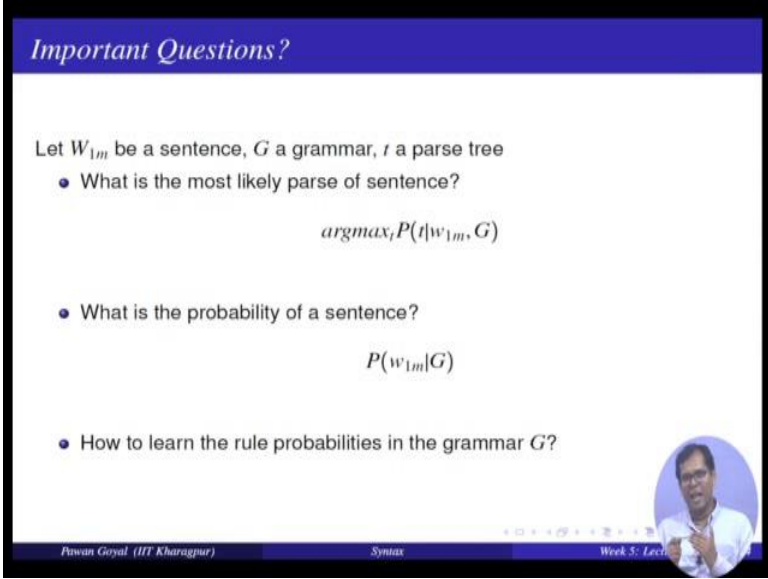
Let us look at some of the features of probabilistic context free grammars. So, why we started with this formulation? So we said that using the context free grammar, given a sentence we can find out all the possible parse trees, but you are not able to assign any probabilities to that. So, by using PCFG, if for a given string, the number of parse trees are increasing, I can also assign the probabilities for each individual parse tree, so that gives me some plausibility, which parse tree is more probable than the other one for the given string. So, this is important, but at the same time we should understand that although by using PCFG, I can compute what is the probability of the sentence by taking all the parse trees, taking the individual probabilities and adding them up, this is just not giving a very good plausibility of the sentence. This is only looking at the structural factors not some lexical co-occurrence.

So, in general, to find the probability of the sentence, you would prefer to use language model than a PCFG. PCFG is good only for finding the probability of a parse tree, which

parse is more probable than another one. Yes, if it helps in some cases like in real text you might find some grammatical mistakes, so PCFG will allow that, but will give you very, very low probability. So, in one case, you can also probably find out which sentence has some grammatical mistake. If the PCFG is giving it low probability. And yeah, this is something that I said earlier; so in practice, this is not good for modeling the probability of the sentence, language model is much better than PCFG.

So, why is that the case, so a simple example is if we have the same sentence, I have two different trees, one is smaller than the other, the smaller tree will always have a higher probability than the larger one, because all the probabilities are less than one. And for a larger tree, you are multiplying the probabilities. So, to take away here is that, you would use PCFG to find out what are all the probabilities for different parse trees for a sentence and try to choose the best one.

(Refer Slide Time: 23:24)



Important Questions?

Let w_{1m} be a sentence, G a grammar, t a parse tree

- What is the most likely parse of sentence?

$$\operatorname{argmax}_t P(t|w_{1m}, G)$$

- What is the probability of a sentence?

$$P(w_{1m}|G)$$

- How to learn the rule probabilities in the grammar G ?

Pravin Goyal (IIT Kharagpur) Syntax Week 5: Lecture 5

So, now once we have this formulation of PCFG, there are some interesting questions that we would like to explore using that. So, suppose I am given a sentence w_{1m} , I am given a grammar G , and there are various parse trees t is one of those. So, what is the most likely parse for this sentence given the grammar? So, which parse is tree t gives me maximum probabilities argmax of probability given the sentence and the grammar. Then what is the probability of the sentence? Probability of the sentence given by grammar,

and then finally, how do we learn the rule probabilities of my grammar G , which these are the three interesting questions that we would like to answer in the next lectures.

So, for example, how do I find the most likely parse of a sentence? So, one simple solution is find out all the possible trees and take the one with the highest probability, but is there any efficient method for doing that. What is the probability of a sentence? Again I can find out probabilities of the individual parse trees add them up, this gives me the probability of the sentence, but can I do a text enumeration all that and adding it or is there any some other methods. And finally, there is some interesting question that how do I learn the rule probabilities in the grammar G . And the answer to this will be similar to what we saw in the case of Head and Markov models.

So, there are they will be again two ways of running the parameter; one will be when the I am given the corpus and some labeled parse trees, I can use them to find the probabilities. Under the scenario, where I am given the corpus, but not the parse trees, I am only given the grammar, grammar in the sense CFG, not the rule probabilities, then how do I learn the parameter. So, this will be again very, very interesting topic and there you can use some ideas where that I had shown earlier for the (Refer Time: 25:33) algorithm. So, in the next lecture, we will start trying to answer some of these questions.

Thank you.