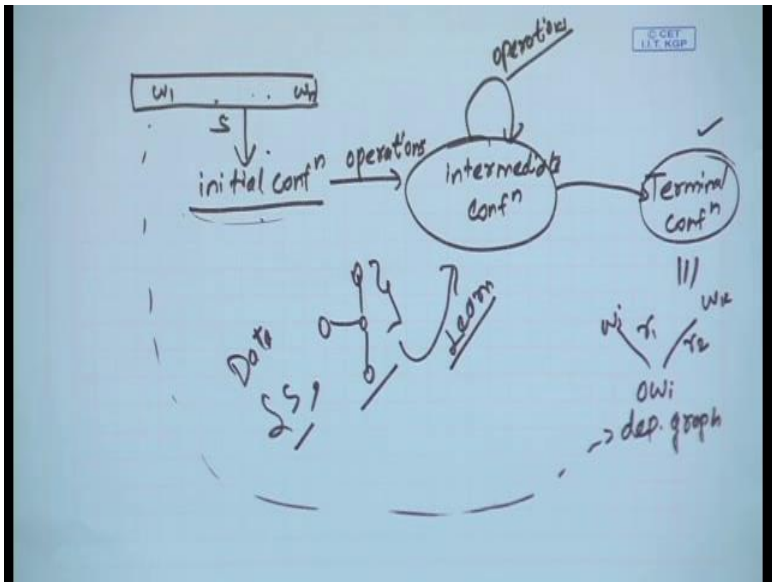
Natural Language Processing

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

Lecture - 28
Transition Based Parsing: Learning

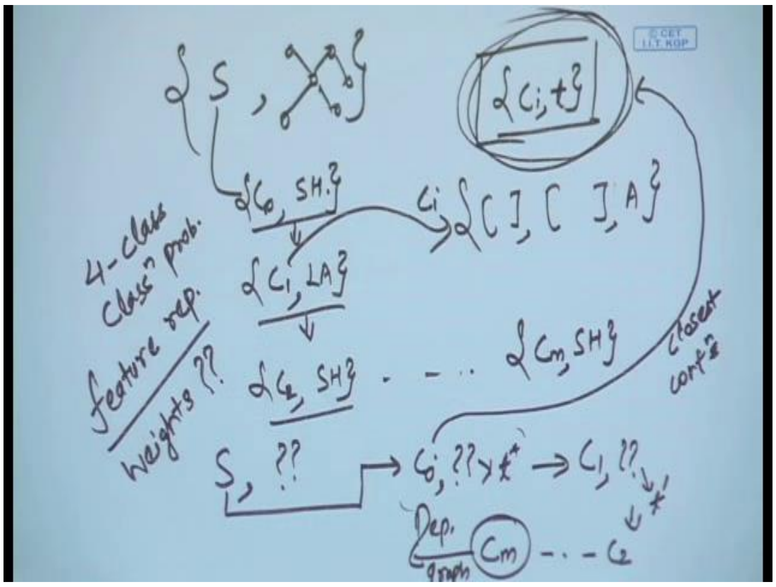
Welcome back for the third lecture of this week. So, in the last lecture, we had defined what is the notion of our transition based parsing. And we saw what are the configuration that I should have what are transitions I can take and how do I come up with a final dependency graph. And we took an example and showed what are the transitions you can take and how you should be taking the transitions. And then we ended with saying that how I will be using that for getting parse for a new sentence. So, this is something that we had initially asked.

(Refer Slide Time: 00:58)



So, I have some sentence S, S I can find out what initial configuration is why the function. I keep on taking some transitions go to some intermediate configuration until I obtain terminal configuration. And I said with the data I will have some sentences and their corresponding dependency parsing. And from there I will try to learn what are the operations of transitions I am where to take. Now, in this lecture we will see for a particular problem how we will be doing the learning part.

(Refer Slide Time: 01:48)



So, again let me start by giving you the basic intuition. So, what will be the basic intuition? So, in the last lecture, you had learned that for a sentence if you know the dependency graph so this is my labeled data, if you know that, you can run through the steps very, very easily, the steps of transitions. So, you can say that this is my initial configuration. And you should find out what is the transition I should take, say it is shift. So, you can store somewhere, this my initial configuration C 0 and this is the transition. Then by taking this transition, you will go to some other configuration C one, what is the transition it may be left-arc again some C 2, it can be shift and so on, some C m, it can be shift or something.

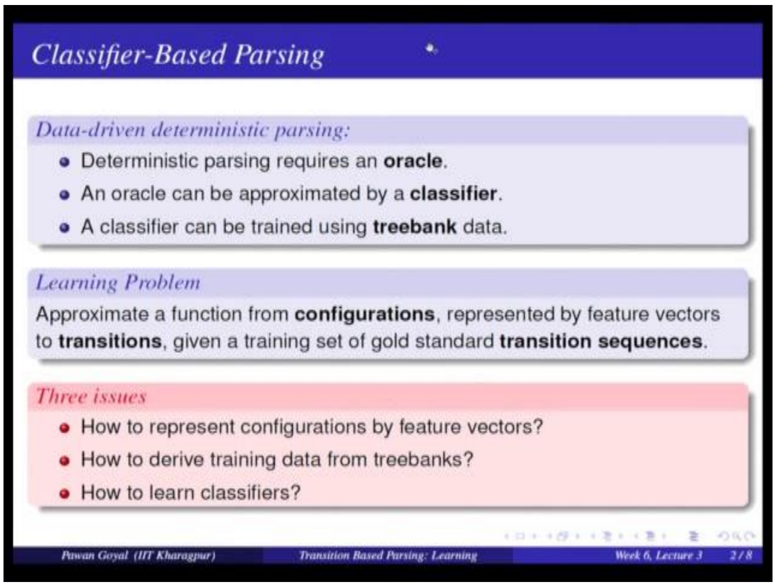
So, this given a sentence, you can easily run through these steps and find out. And what is this C i? C i is nothing but a some words in stack, some words in buffer and something in my arc that is my C i. So, that means, suppose I am given a set of sentences and their dependency graph, I can store all the possible set of configuration and all the possible by all the possible I mean whatever I obtaining from these sentences and their corresponding transitions. And this I can have a last set. So, I will have a set of all possible C i and that optimal transition that I should be taking and C i's of this form something in stack and something in buffer something in arc.

Now, what is my problem at run time? At run time I am given a sentence S, I do not know its dependency parse. So, I start my transition, I converted to some initial configuration C 0 that is easy we have the function of converting to initial configuration. There I have to find out what is the transition I should take. Now, what will be the idea? I will try to use this set of data that I have. I know for what configurations, what transitions word taken in my gold standard or in my set of labeled data set. So, I will try to find out. So, before going into what this is the machine learning approach, we will use, so what will be intuitive idea try to find out what are the closest configurations in the set; And for those closest configurations, what transition was taken and I will try to use that t star from there.

Suppose I find out the t star is the transition that was taken to the closest configuration here. So, I will use t star. And once I know this t star, I can transit it to next one C 1; again the question will come what is the transition I should take. So, again go back to this and choose t dash, take this go to C 2 keep on doing that until you will come up with the final configuration C m, and that is why you say this is the dependency graph for S. And this is the intuitive idea. In the last lecture, we have seen how do come up with this set. And today we will see, how do I approach this problem, so that I can come up with this function that what is the closest configuration from here to here, what is the transition that I should be taking.

So, one important idea that we had already discussed earlier in this course is that for example, how do you find out the closest configuration, and what is the transition that we are taking. This you have to use by using some sort of classifier that is you are trying to classify for a given configuration among all the four transitions which transitions will be taken. So, you are treating it as a four class classification problem; at each point you are trying to classify among one of the four classes. And how are you going to classify? You have to convert your input data in some representation. So, this will be using some feature representation. So, you will have to convert your configuration into some sort features, feature vector. And for those features, you have to learn the weights. And this weight you have to learn by the training examples again. And once you have learn the optimal weights, you can find out what is the transition at any given point and this is what we will be discussing in this lecture.

(Refer Slide Time: 07:12)



Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle**.
- An oracle can be approximated by a **classifier**.
- A classifier can be trained using **treebank** data.

Learning Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.

Three issues

- How to represent configurations by feature vectors?
- How to derive training data from treebanks?
- How to learn classifiers?

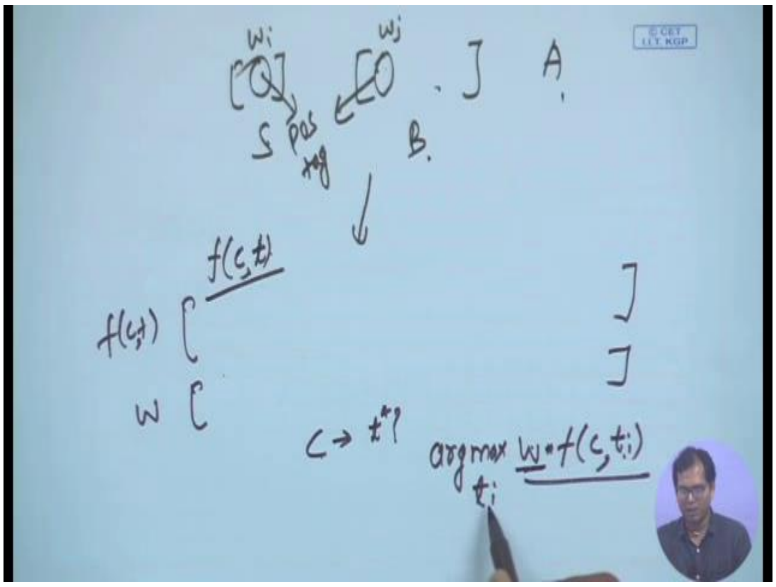
Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, Lecture 3 2 / 8

So, let us see. So, now we are talking about the data driven deterministic parsing. So, I have written here certain things like deterministic parsing requires in oracle. What do I mean by oracle? So, oracle is nothing but the set of configurations, and the set of transition that I took. So, this you have already seen in the last lecture, how do you find out configuration and transitions. Now, what we are going to do? We want to approximate it by a classifier. So, we will be learning classifier from there, and it will be trained using the treebank data. So, whatever data we have in the gold standard label data, we will use that to train our classifier also. So, you will use that label data for two different tasks; first for building the oracle configuration, transition, configuration, transition; second to learn the weights of my classifier.

Now, what is the learning problem? Now, as we said we will be given a configuration as an input and we want to find out what is the transition I should be taking at a particular configuration. So, ideally I want to approximate the function that takes a configuration which is represented by a feature vectors, two transitions - so configuration to transitions. Here configuration is nothing but a feature vector form, because otherwise how do you compare between two configurations, so that is why you will take give it a very abstract representation in terms of some feature vector form. And you will learn a function from feature vector to the optimal transition, and this will be your classifier. And how will you learn that you will be given a training set of gold standard transition sequences that we already have seen.

So, now to completely solve this problem or to completely understand this problem there are three issues that you need to understand. First is how do I represent configuration by a feature vector. Second, how do I derive training data from my treebanks, and third is how do I learn my classifiers. So, let us try to answer each of these three questions. So, how do I represent configuration by feature vectors, and this is something that we had done earlier in the class that how do I convert a given state or represent to a feature vector.

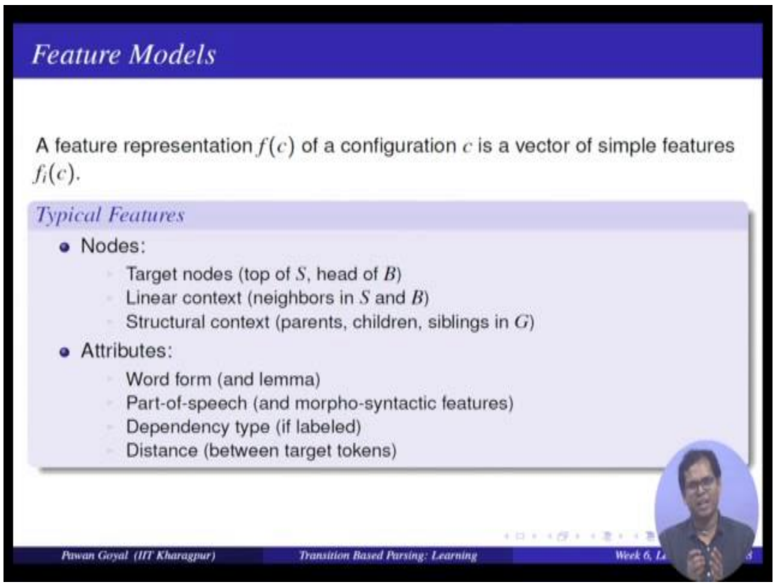
(Refer Slide Time: 09:58)



So, what is my configuration? My configuration is nothing but a stack, buffer and arcs. And I want to convert that to some feature vector. And feature vector again here we will take it a very simple form like f c, t, it is a function over the configuration and the transition. And we will try to define features independent of transition, so each feature that we define can have four copies four, four different transitions, so f c t 1, f c t 2, f c t 3, f c t 4 like that. So, what are the different things that I can use in feature? So, I can use things like what is the word at top of this stack, what is the word at top of buffer, they are very important, what is the word here.

Then I may want to use what is the part of speech tag of these words, because sometimes some relations would might directly dependent on whether it is a verb and it is a noun then there might be a relation otherwise not. So, I might use the part of speech tag. I might go to the lemma; I might want to use what is the distance between this word and this word in the actual sentence. I might also want to use what are neighbors here; I might want to use what are relations that I have already been established with this word or this word, so that means I will define certain conditions over stack, buffer and arcs and that will I would like to take my features. And again these can be some binary questions that I am asking. So, that is the distance between these two words between 2 to 5, yes or no. So, like that these can be my condition that is my features, and I will define it for all the four transitions.

(Refer Slide Time: 12:01)



Feature Models

A feature representation $f(c)$ of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - Structural context (parents, children, siblings in G)
- Attributes:
  - Word form (and lemma)
  - Part-of-speech (and morpho-syntactic features)
  - Dependency type (if labeled)
  - Distance (between target tokens)

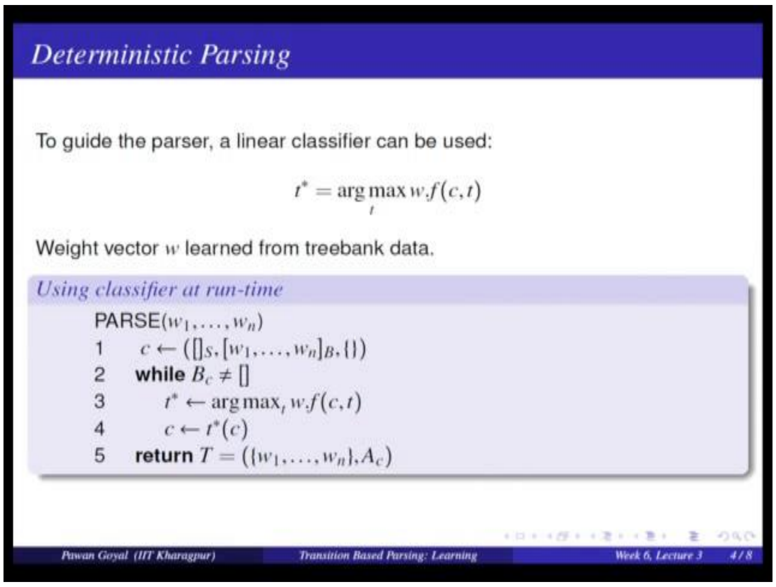
Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, L

So, let us look at what are the different feature models we can take in general. So, I am representing configuration c by a vector of simple features. So, like I can use the nodes. So, what is the top of this stack, what is the head of buffer, I can use the linear context that is what are the neighbors in s stack of the top word, what are the neighbors in buffer of the top word. Then I might also use what are the parents, children, siblings depending on what relations are already been established in my set of arcs.

Then I can go to some other attributes like I can use the word form, I can use also its lemma. And we can use them up part of speech tag and various other features for example, is the word on top of stack ends with ed or ing things like that. And I might be able to use the dependency type if I am handling a labeled dependency problem. So, just a word what do I mean by labeled dependency problem that means, I also want to find out for two words what is the dependency relation label. And if I am solving unlabeled problem that means, I want to just want to establish a relation, but I am not concern with the actual dependency type.

So, I am not worried about putting the label on the arc, but just the structure of the tree. So, if I am solving label problem, I might also have to see what is the relation type that I am establishing. And we will also see the distance between different tokens as one of the features. So, these are some typical examples, but it does not mean that you are limited only to using this set of features. And as I keep on seeing for your particular task, you might have to think what might be some interesting features that you can use. So, by using all these features, I am putting my binary questions, I can represent my configuration as in terms of a feature representation. So, this is my first question. Now, what was the second question?

(Refer Slide Time: 14:08)



Deterministic Parsing

To guide the parser, a linear classifier can be used:

$$t^* = \arg\max_t w.f(c,t)$$

Weight vector w learned from treebank data.

Using classifier at run-time

```
PARSE($w_1, \ldots, w_n$)
1   $c \leftarrow ([]_S, [w_1, \ldots, w_n]_B, \{\})$
2   while $B_c \neq []$
3       $t^* \leftarrow \arg\max_t w.f(c,t)$
4       $c \leftarrow t^*(c)$
5   return $T = (\{w_1, \ldots, w_n\}, A_c)$
```

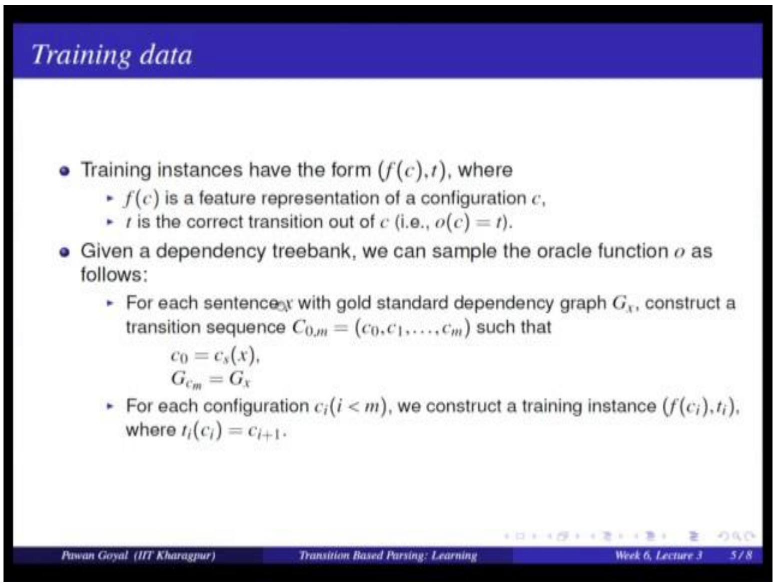
Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, Lecture 3 4 / 8

Now, how do I use this at run time, but actually find the parse of the sentence; idea would be something like that. So, let me start from here. At run time, you are given a sentence w 1 to w n. And you can always go to the initial configuration, where the s stack is empty, buffer contains all the words and arc is empty. Now, this is your configuration and you are in a loop while the buffer is not empty, you keep on taking some transitions. Now, this is the task say run time, so this configuration c you know how to covert to the to a feature vector because you are defined your features. So, you can ask the questions at this point and find out your vector f c, t.

Now, what is my classifier? My classifier is simple. I have learned the weights of my features assume that I have learned, we will see how to learn the weights. So, once I have learned the weights of my features, I will multiply the weights with f c, t and find out what is the particular transition that is giving the maximum value that means, I know what is my f c, t feature vector, and I know this is my f c, t and this is my weights. So, now at run time, I am given a configuration c and I need to find out what is the optimal transition how do I do that. An idea is that I will multiply w with f c, t i for all the four transitions. So, t i is shift, left-arc, reduce and right-arc. And I will take which one gives the maximum value argmax over t i.

So, at run time, any configuration, I can convert to f c, t very easily. I will already have the weight vectors the only thing that I have to do multiply the weight vector with the feature vector find out four different values four different transitions choose the maximum or choose the transitions that has gives the maximum value that is the transition you will take. And then if you go back this is the transition you take and then apply this transition over this configuration to find the next configuration, and keep on going in this loop until your buffer is empty, and this is what your run time. Now, what is not clear to you right now is how do we learn these weights, everything else is clear.

(Refer Slide Time: 17:12)



Training data

- Training instances have the form $(f(c), t)$, where
  - $f(c)$ is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., $o(c) = t$).
- Given a dependency treebank, we can sample the oracle function o as follows:
  - For each sentence x with gold standard dependency graph G_x, construct a transition sequence $C_{0,m} = (c_0, c_1, \ldots, c_m)$ such that
    $$c_0 = c_s(x),$$
    $$G_{c_m} = G_x$$
  - For each configuration $c_i (i < m)$, we construct a training instance $(f(c_i), t_i)$, where $t_i(c_i) = c_{i+1}$.

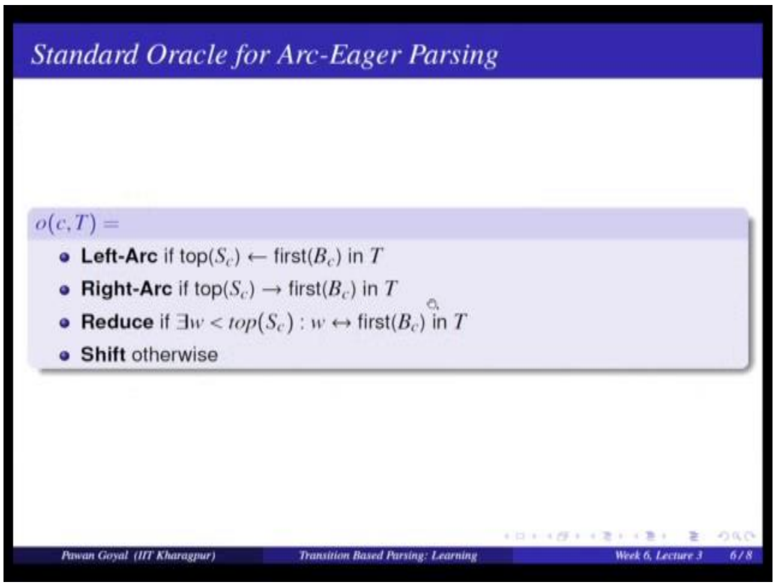
Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, Lecture 3 5 / 8

So, let us see. So, learning weights we will have to use the labeled data that we have is the training data. Now, let us see what are the steps that we need to do over the training data. Now, training data, I will have the instances of this form f c and t is this clear entering data we will have configurations and transitions, yes. And configuration I know what is the feature representation, so I convert the feature vector. So, I can have f c and t f c is nothing, but the feature representation of the configuration c and t is the correct transition out of it starting from c. And this I can for obtain from my oracle. Remember in oracle I had my configuration and the optimal transition, so I know this configuration what is the transition it should take. So, from there, I go to next step f c and t.

Now, this is something that we have done in the last class, but let me try to repeat that again that how do we sample this oracle function from the set of labeled sentences labeled of dependency graph. So, for each sentence x with the gold standard dependency graph g x, you have to construct a transition sequence right like we did in the last class for the example he sent her a letter, such that c 0 is something that will obtained by applying the initialization function on x and this is the final dependency graph.

Now, for each intermediate configuration we will construct a training instance. So, right we will have c i t i c i t i and c i will go to f c i and this is the condition for how are am I moving in from one configuration to another configuration. So, this is the same thing that that I have discussed earlier in today's lecture that what is the idea is starting from the gold standard sentence and dependency graph find out this sequence c i t i c i t i. Now, the only addition here is I covert each c i to it is feature representation, so that is why I what I have I have f c i t i f c i t i.

(Refer Slide Time: 19:17)



Standard Oracle for Arc-Eager Parsing

$o(c, T) =$

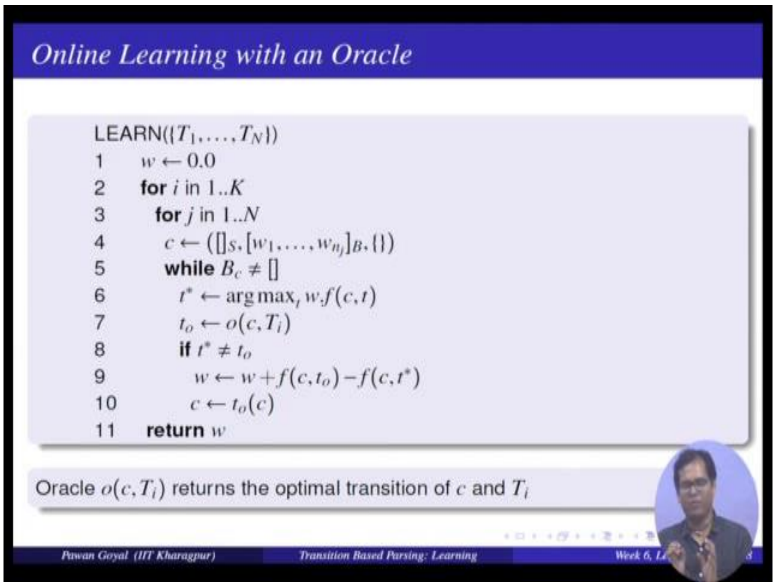
- **Left-Arc** if top(S_c) $\leftarrow$ first(B_c) in T
- **Right-Arc** if top(S_c) $\rightarrow$ first(B_c) in T
- **Reduce** if $\exists w < top(S_c) : w \leftrightarrow$ first(B_c) in T
- **Shift** otherwise

Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, Lecture 3 6 / 8

And how do you sample the transitions in oracle this is something again that we did in the last class. So, if you see that in my dependency graph the current top of the word in the stack is connected to the top of the word in buffer. So, you will have a relation depending on the direction it will be left-arc or right-arc. So, here if the top of the word in buffer is the head and this is the dependent you make a left-arc transition. So, this is what you will store.

If top word in the in the stack is head then you will have a right-arc relation, yes. Then how do you choose between reduce and shift, if there is a word below that of the stack such that it is connect to the first word in the buffer, then you do reduce otherwise you shift. And remember this is rule of thumb that we discussed in the last lecture. If you are choose between reduce and shift this is the condition that you can use. So, I hope the idea is clear, you are starting with the sentence in training data, you have the gold standard dependency graph, you keep on going through your transitions and store it somewhere f c i t i f c i t i f c i t i.

(Refer Slide Time: 20:40)



Online Learning with an Oracle

```
LEARN($\{T_1, \ldots, T_N\}$)
  $w \leftarrow 0.0$
  for $i$ in $1..K$
    for $j$ in $1..N$
      $c \leftarrow ([]_S, [w_1, \ldots, w_{n_j}]_B, \{\})$
      while $B_c \neq []$
        $t^* \leftarrow \arg\max_t w.f(c,t)$
        $t_o \leftarrow o(c, T_i)$
        if $t^* \neq t_o$
          $w \leftarrow w + f(c,t_o) - f(c,t^*)$
        $c \leftarrow t_o(c)$
  return $w$
```

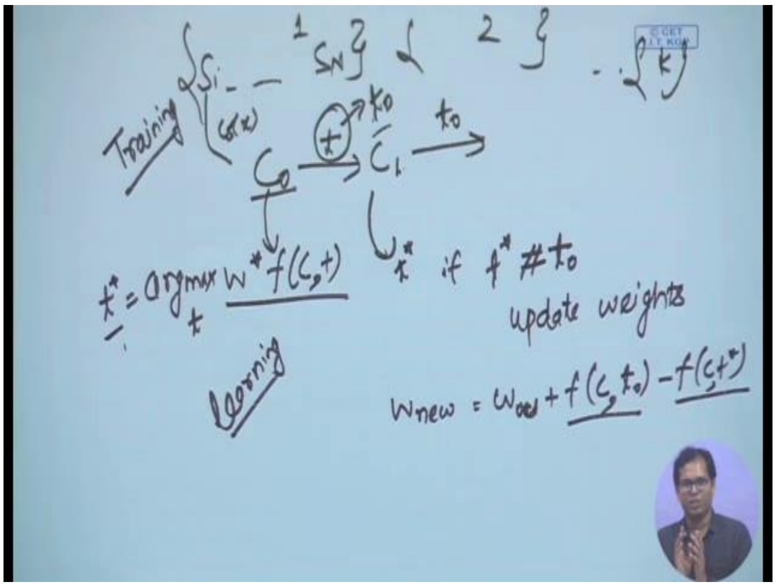
Oracle $o(c, T_i)$ returns the optimal transition of c and T_i

Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, L

Now, how do I use that to learn the weights? Now, and this is the idea of learning the weights. So, what you will do? You will start with some initial set of weights. So, here I have said all the weights can be initialized to 0, but probably not will 0, you can initialize with some other numbers say some initial random numbers or some uniform numbers, you initialize your weights. Now, what are you going to do? So, there are two loops here for i is 1 to K this is the number of iterations that you are doing; for j in 1 to N, 1 to N is the all the set of sentences that you have in your training data. So, you are doing multiple iterations over your training data.

In each iteration, what do you do, you take the sentence, yes, you get the initial configuration fine. Now while buffer is not empty, so what you are doing right now, you are again repeating the same stuff over each sentence in the training data. You start with the initial configuration and now try to find the transition as per your current weights, so that is where the idea of learning comes in.

(Refer Slide Time: 22:05)



$$t^* = \underset{t}{\operatorname{argmax}}\ w^* f(c,t)$$

t^* if $t^* \neq t_0$

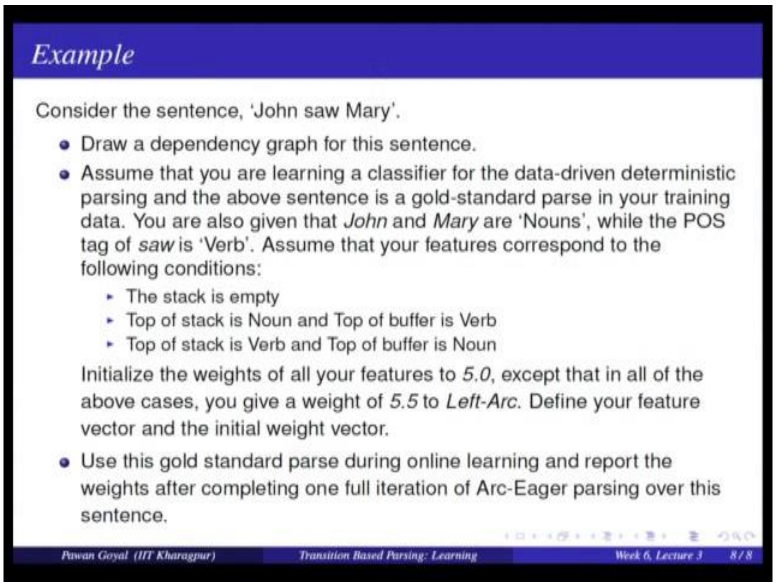
$$W_{new} = W_{old} + f(c,t_0) - f(c,t^*)$$

So, let me try to explain it here. So, you have a sentence S, now you can apply the c s x function or and you go to initial configuration C 0. Now, you take a transition to go to C. Now, this is your sentence in your training data that means, you know what is the transition you should take, yes, but I want to use this idea to learn and how do I do that. Now, suppose that you are actually at the run time at testing time, so then you do not know the transition. So, you will convert that to some feature vector f c, t. Now, at run time, how do you find out the optimal transition multiplied with the weights and take the argmax. And let us say this is t star, this you can do that even if it is in the training set and let us call it t naught - optimal transition.

Now, what is the idea? You are still in the learning stage. So, your weights will not be optimal. So, when you do this operation, you may not get the optimal transition, you may get something else and that is where you will try to adopt your weights. So, you will say if t star not equal to t 0, then you update your weights. And how will you update your weights such that you go in the direction of the actual transition, and away from the transition of the transition that you obtain at the current point. So, simple thing is w new would be w old plus f c, t 0 minus f c, t star. So, going in the direction of the optimal transition and away from the transition that you are currently predicting, and there can be some learning rate and all that we are not discussing right now. So, there will be some learning weights by which you will do this update. So, you will have now new weights.

Again you keep on doing it for c 1, c 1 you know what is the transition optimal transition, but you will find out argmax t star match with this; if they are not the same, you will again update your weights. So, you will keep on doing that for all the sentences S 1 to S N in your training set and you will do it 1, 2 some K times until the weights are converging. So, once the weights converged, we stop. So, this is what we have shown here. So, you start for each sentence you have some initial configuration, while buffer is not empty. So, you keep on doing the stuff. Find out what is the optimal transition as per your weights, find the optimal transition from the oracle; if they are not matching, update your weights, but you take the correct transition. So, the next time you are starting with correct configuration, keep on repeating it. And finally, you will end up with new set of weights.

(Refer Slide Time: 25:50)



Example

Consider the sentence, 'John saw Mary'.

- Draw a dependency graph for this sentence.
- Assume that you are learning a classifier for the data-driven deterministic parsing and the above sentence is a gold-standard parse in your training data. You are also given that *John* and *Mary* are 'Nouns', while the POS tag of *saw* is 'Verb'. Assume that your features correspond to the following conditions:
  - The stack is empty
  - Top of stack is Noun and Top of buffer is Verb
  - Top of stack is Verb and Top of buffer is Noun

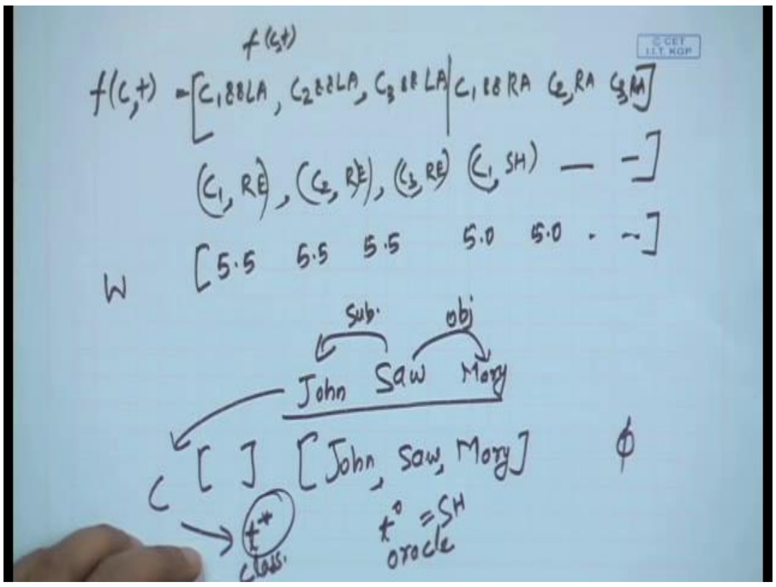
  Initialize the weights of all your features to *5.0*, except that in all of the above cases, you give a weight of *5.5* to *Left-Arc*. Define your feature vector and the initial weight vector.
- Use this gold standard parse during online learning and report the weights after completing one full iteration of Arc-Eager parsing over this sentence.

Pawan Goyal (IIT Kharagpur) Transition Based Parsing: Learning Week 6, Lecture 3 8 / 8

So, now to further understand that let us take an example and that will make it clear to you that how the learning or how the weight updation takes place. So, this is simple example. So, I have a sentence John saw Mary. So, what do you need to do? First question is draw a dependency graph for this sentence that is very easy, yes. Now, the next question says that you are using the data driven dependency parsing the same method that we have discussed in this lecture and in the last lecture. And you already have the gold standard parse in your training data and you have some other information like John and Mary are nouns; and saw is verb, and also given you features. And you are told that initialize your weights to 5 except that for left-arc the weights are 5.5 define your feature vector, and the initial weight vector.

So, let us try to do this. So, how many conditions are we seen, we are seeing three conditions over my configuration. The stack is empty, top of stack is noun and top of buffer is verb, top of stack is verb and top of buffer is noun, three conditions. Now, these three conditions I have to check for all the four different transitions. So, what is the size of my feature vector 3 into 4 - 12?

(Refer Slide Time: 27:15)



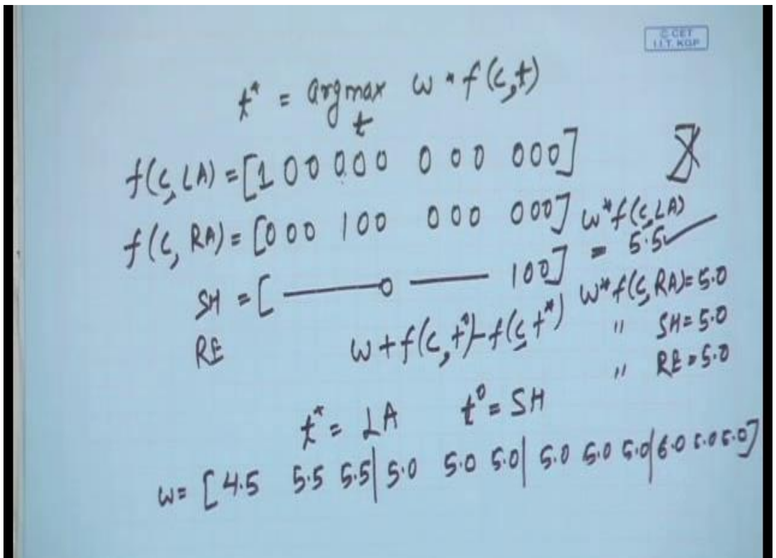
f(c,t)
f(c,t) = [C1 && LA, C2 && LA, C3 && LA | C1 && RA C2 RA C3 RA
(C1, RE), (C2, RE), (C3, RE) (C1, SH) — —]
W [5.5 5.5 5.5 5.0 5.0 . —]
Sub.
obj
John Saw Mary
C [ ] [John, Saw, Mary] ϕ
t*
class.
t° = SH
oracle

So, my feature vector, so it is of 12 dimension. And what are my features, so first feature let me write it simply condition one that is the stack is empty and same as starting with left-arc; transition is left-arc. Second feature can be condition 2 and left-arc; third condition three and left-arc, yes. This is my f c, t a condition over the configuration and transition. First three elements next three elements same c 1, but now transition will change still right-arc; c 2 right-arc, c 2 right-arc. And then the next elements c 1 reduce, c 2 reduce, c 3 reduce, and here c 1 shift, c 2 shift, c 3 shift. So, this is my feature vector twelve elements here. Now, what is my weight vector? Initial weight vector we said all the elements are 5 except the left arc is 5.5. So, the weights are 5.5, 5.5, 5.5 and everything else is 5.0, 5.0 and so on that is your initial weight vector and your task is now so this was the first question what is my dependency parse John saw Mary. So, saw is here, John and Mary this is subject and this is object.

Now let us see what the question says further. So, the next question says use this gold standard parse during online learning and report the weights after completing one full iteration of arc eager parsing. So, it says that now you have to learn the weights using the arc eager parsing or the transition parsing that we have seen.

So, now let us see how do we learn the weights. So, this is what we have defined right now. We have this features, we have this weights initially as per dependency graphs. So, how will I start learning? In learning, I will take this sentence I will put it to the initial configuration initial configuration is what this stack is empty, buffer contains John, saw and Mary; and arc is empty. So, now, at this configuration C, I have to choose what is the optimal transition as per my classifier. And I have to choose the optimal transition as per my oracle from oracle what will be t 0. Oracle I will be very easily saying that this would be shift I am doing a shift at this point I should shift here, but what is being predicted by my classifier. So, let us see what will be t star as per my classifier.

(Refer Slide Time: 30:55)



$$t^* = \underset{t}{\operatorname{argmax}}\ w * f(c,t)$$

$f(c, LA) = [1\,0\,0\ 0\,0\,0\ \ 0\,0\,0\ \ 0\,0\,0]$

$f(c, RA) = [0\,0\,0\ 1\,0\,0\ \ 0\,0\,0\ \ 0\,0\,0]$ $w * f(c, LA) = 5.5$ ✓

$SH = [$ —— 0 —— $1\,0\,0]$ $w * f(c, RA) = 5.0$

RE $w + f(c, t^0) - f(c, t^*)$ " $SH = 5.0$

" $RE = 5.0$

$t^* = LA$ $t^0 = SH$

$w = [4.5\ \ 5.5\ \ 5.5 \mid 5.0\ \ 5.0\ \ 5.0 \mid 5.0\ \ 5.0\ \ 5.0 \mid 6.0\ \ 5.0\ \ 5.0]$

So, for my classifier how do I obtain t star. So, t star would be argmax w times f c, t for all possible transitions. So, let us do one-by-one. So, what will be f c, LA what is the feature vector when the transition is LA? So, for that let us look at my feature vector definition. So, this will be a binary value at each point c 1 and LA. What is c 1 top of the stack is this stack is empty, so c 1 is 1, and transition is left-arc, so this will be 1. C 2 top of stack is noun, and top buffer is verb. Now, top of stack is empty it cannot contain a noun, so c 2 is 0; so already this will be 0. C 3 again say top of stack is verb and top of buffer is noun again this will be 0 that is why I fill in my feature vector

Now, let us go to this. Now, immediately as you move to some other transition RA this should be 0, yes because here your transitions is LA. So, everything else will be 0, 12 elements. Now, what is your weight vector? Weight vector is here, say if you multiply weight vector with f c, LA, what do you get? So, w times f c, LA is equal to 1. So, only one element is 1. So, I will multiply with that this will give you 5.5. Now, similarly now you can easily figure out what are the other features f c, RA. For RA, similarly only this element will be 1, everything else will be 0. So, what is w times f c, RA that will be 5. And similarly, if you keep on doing for all shift and reduce you will find this for shift is 5 and this for reduce is 5, yes.

So, now what is your t star argmax t w times f c, t that is will be left-arc. And what is your t 0 optimal is shift as per your oracle. And how do you learn your weights, if t star is not the same as t 0, you will update your weights. And what is the (Refer Time: 33:45) update w plus f c, t 0 minus f c, t star. So, what is f c, t 0? That is f c SH. So, what will be this function? So, suppose SH was at the end. So, it was 1 0 0 and everything else is 0 and this is my LA. So, what will be the new weight vector? So, I have the original weight vector that is this 1 plus this minus this. So, what will be the new weight vector? t will be I am subtracting one here, so 4.5, 5.5, 5.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0. And now I come to shift in shift I am adding that1, so it will be 6.0, 5.0, 5.0, and that is your new weight vector.

And now you will work with this weight vector for the next set of configuration. So, what will be the next configuration from here you will apply shift and next configuration will be John saw Mary, and phi. Again you will convert it to the feature vector see what is t star that you are getting what is t 0 if they are not matching update your weights and that you will continue until you arrive at the terminal configuration. And then you will have the final weight vectors. So, I will encourage all of you that you should try it this full example on your own, and see what is the final weight vector that you are getting. And even if you are trying to see that by using this weight vector, does that help in that now with the new weight vector if you try it on the old configuration you will be closer to the optimal configuration as per the oracle and not what your (Refer Time: 36:10) early updating.

So, this is the idea of how you can use the machine learning methods for this dependency parsing by taking this example of arc eager of transition based parsing. Now, in the next lecture, we will start discussion on a new method of dependency parsing, and we will see that again how we can use the label data for doing this.

Thank you.