

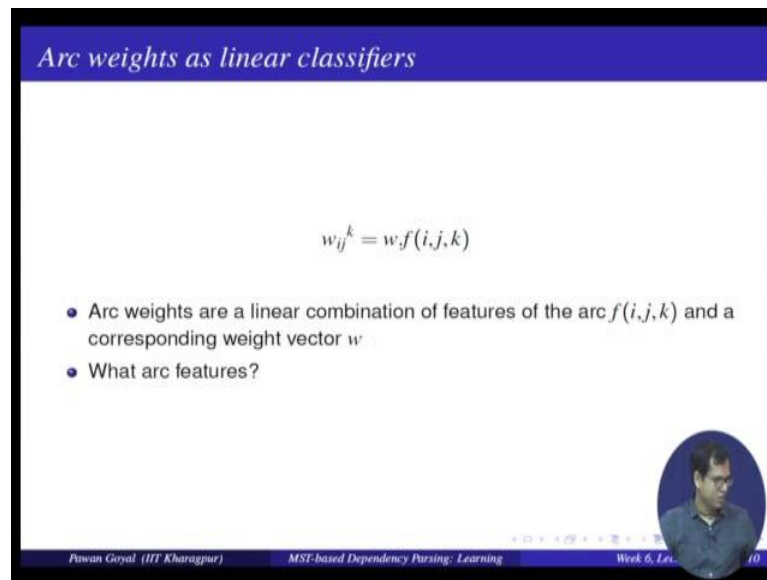
**Natural Language Processing**  
**Prof. Pawan Goyal**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 30**  
**MST - Based Dependency Parsing: Learning**

Welcome to the final lecture of this week. So, in the last lecture, we had talked about what is the approach of using maximum spanning tree for dependency parsing. So, there we only covered that if we have a sentence and we have some way of constructing the original graph multi diagram where there all possible connection we can usually (Refer Time: 00:39) algorithm to find out the MST, in that is what we covered, now important point is how do we find out the aspects and that is what we will be covering here. So, how do we treated as a learning problem where we given some data where we know these are sentences and they are corresponding dependency graph, how do we use that to establish the aspects for a new sentence and you understand now that once you can do that part, rest part is already defined by the algorithm. So, your main task is how do you effectively construct the aspects?

Now, just one thing what do I mean by the aspect? Aspect is something that will depend on the 2 nodes that you are connecting and what is the relation with which we are connecting them? So, this has to be some sort of a feature representation vector over the nodes and the label and you have to define what are your features and once you define for any possible connection you can find out this vector; find this vector is a separate task, it does not require any learning, the way we will pick a learning involved, each features will also have a weight assigned that will help you to determine the aspects. So, feature vectors when you multiple the weight, you will get the aspects and this weight vector for the feature is something you can learn from your data. So, that is what weight vector with this feature set help me to choose the optimal or the best a spanning tree that corresponds to dependency graph that I already have in my data if I have some weights that are not allowing me to choose the best graph I should a bit my weight and this is the idea that we will see in this lecture.

(Refer Slide Time: 02:34)



*Arc weights as linear classifiers*

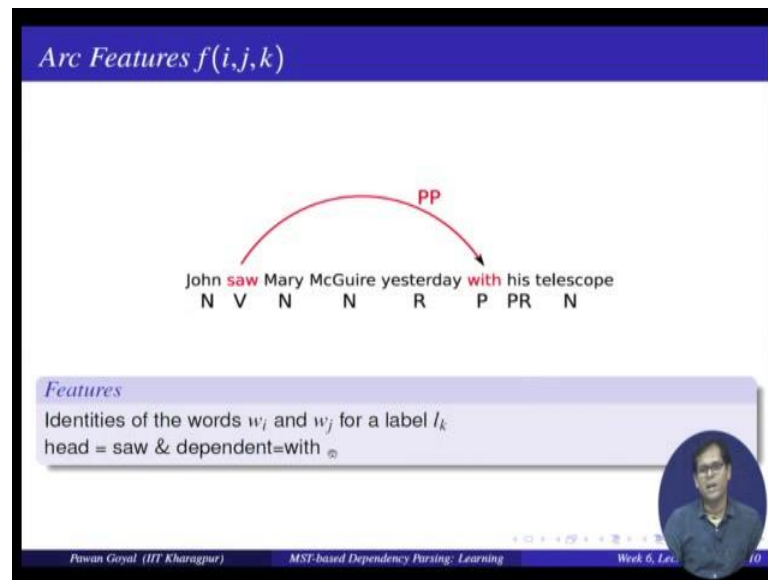
$$w_{ij}^k = w \cdot f(i, j, k)$$

- Arc weights are a linear combination of features of the arc  $f(i, j, k)$  and a corresponding weight vector  $w$
- What arc features?

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 10 10

We will be talking about the learning part here. So, the first thing that we should see here is the arc weights that we are defining can be written as some feature vector over  $i, j, k$  yes the node  $i$ , the node  $j$  and the relation  $k$  and a weight vector and I multiply by weighted feature I get the arc weight. Now we have to see how to define this feature weight features this is again analogous somehow to what we did in some of the previous classes and how do we learn this weight vector. So, here arc weights are nothing but linear combinations of features of the arc and a corresponding weight vector. Now what are different arc features you can take. So, it will be dependent on again what are the words I am connecting

(Refer Slide Time: 03:26)



Let us take a sentence like John saw Mary McGuire yesterday with his telescope and I want to find out the head weights between first saw and with the relation pp. So, this is my i, this is my j and this is the relation PP. So, here by the multi diagram arc means for k, you can have many variations you can have PP, you can have N subject, direct object, indirect object and non modifier etcetera, we have the various possible variations. So, what would this feature dependent on. So, feature would dependent on probably what are my words themselves they are some of the most important features if my word is saw and my word first word is saw, second word is with and the relation is whatever PP here. So, head is saw and dependent is with that is taken in my one primarily feature. So, every case I will have this feature I will have an answer 1 or 0.

(Refer Slide Time: 04:31)

*Arc Features  $f(i,j,k)$*

John saw Mary McGuire yesterday with his telescope  
N V N N R P PR N


*Features*  
Part-of-speech tags of the words  $w_i$  and  $w_j$  for a label  $l_k$   
head-pos = Verb & dependent-pos=Preposition

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lec. 10

Then I can have a feature composed from the part of speech tags of these words. So, I also know what are part of speech tags and. So, the feature can be like what is the part of speech for the head part. So, head part of speech is verb and what is the part of speech of dependent word that is preposition and that would be one of the most important features that I can use because many a times the relations are defined between various grammatical categories like for subject it will be between verb and a noun. So, if you know that the first the head word is verb and the dependent is noun, there is a good chance they may have a subject relation although there are other relation possible. So, part of speech is one very important feature.

(Refer Slide Time: 05:16)

**Arc Features  $f(i,j,k)$**



John<sup>s</sup> saw Mary McGuire yesterday with his telescope  
N V N N R P PR N

**Features**  
Part-of-speech of words surrounding and between  $w_i$  and  $w_j$

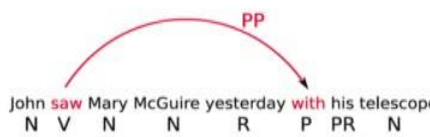
- inbetween-pos = Noun
- inbetween-pos = Adverb
- dependent-pos-right = Pronoun
- head-pos-left=Noun

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 5 5 / 10

Then what else? You can also use the part of speech of the words that are coming around, this head and dependent word in between like in between part of speech is noun; that means, there is a noun that occurs in between these 2 words yes in between part of speech is adverb there is adverb that is occurring dependent pos-right is pronoun for the dependent the right hand side the part of speech is pronoun and head pos left is noun to the head to the left of the head the part of speech is noun. So, like that you can have again have many different features on the neighboring words and the words in between. So, you can use the identity of the words or the part of speech.

(Refer Slide Time: 05:58)

**Arc Features  $f(i,j,k)$**



John saw Mary McGuire yesterday with his telescope  
N V N N R P PR N

**Features**  
Number of words between  $w_i$  and  $w_j$ , and their orientation

- arc-distance = 3
- arc-direction = right

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 5 6 / 10

And then you might also want to use the orientation whether the head is towards left or towards the right so that can be captured by arc direction this is to the right. So, arc direction is right you can also see the distance how many words are intervening between the head and dependent.

So, in this case there are 3 words that are intervening. So, like that you can take various different question or various different characteristic of head and dependent surrounding words and the labels and construct all your features. So, this gives you the set of arc features  $f(i, j, k)$  and now given any possible arc in your graph you can construct this feature vector yes it will be answer to all these questions is the arc distance 3 or not. So, answer will be one or 0. So, like that you can construct this feature vector, but what is your aspect that is the feature vector multiplied by the weight vector. So, next thing is how do we use this weight vector or how do we obtain this weight vector. So, as such you can use any features over the arc  $i, j, k$  and input  $x$ .

(Refer Slide Time: 07:16)

### Arc Features $f(i, j, k)$

John saw Mary McGuire yesterday with his telescope  
N V N N R P PR N

*Features*

- Combinations
  - head-pos=Verb & dependent-pos=Preposition & arc-label=PP
  - head-pos=Verb & dependent=with & arc-distance=3
- No limit : any feature over arc  $(i, j, k)$  or input  $x$

Puwan Goyal (IIT Kharagpur)    MST-based Dependency Parsing: Learning    Week 6, Lecture 5    7 / 10

(Refer Slide Time: 07:20)

*Learning the parameters*

- Re-write the inference problem

$$\begin{aligned} G &= \arg \max_{G \in T(G_s)} \sum_{(i,j,k) \in G} w_{ij}^k \\ &= \arg \max_{G \in T(G_s)} w \cdot \sum_{(i,j,k) \in G} f(i,j,k) \\ &= \arg \max_{G \in T(G_s)} w \cdot f(G)_{\otimes} \end{aligned}$$

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 5 8 / 10

Now, how do you obtain my weights? For that I have to define, what is my inference problem? So, this we have already seen for a graph the weight of a graph will be defined by summation over the weights of all its edges and the MST problem is find out the graph with the maximum weight among all the possible spanning trees that I can obtain from the starting graph take the one that is having the maximum weight now this  $w_{ij}^k$  can be written as  $w$  times the feature vector  $f_{ij}^k$  and  $w$  can go outside of this equation.

So, this will be in other words  $w$  times summation over all the feature vectors and take the arc max over that and suppose I call this as the feature of my graph the feature vector for my graph is nothing, but the summation over feature vectors of all the edges and this is my problem arc max  $w$  times  $f(G)$  over all the possible graphs or directed spanning tree  $G$  that I can construct from my initial graph now how do use this inference problem to learn my bits.

(Refer Slide Time: 08:47)

*Inference-based Learning*

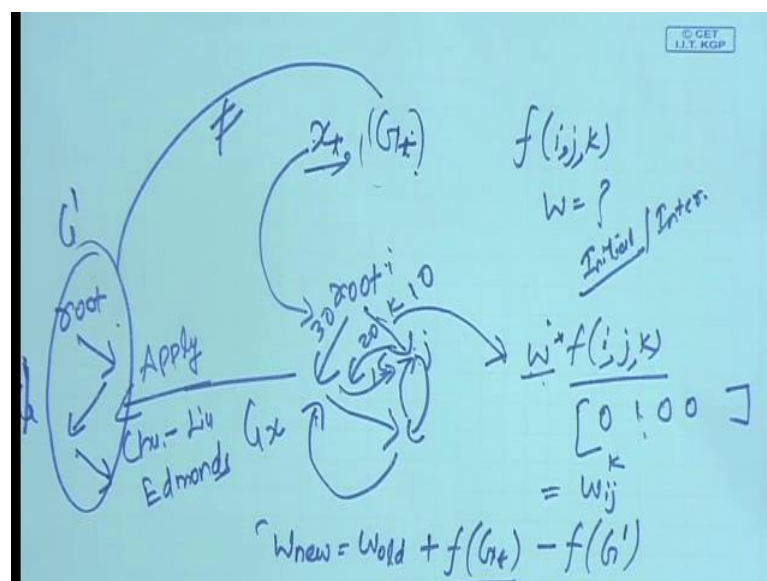
Training data:  $T = \{(x_t, G_t)\}_{t=1}^{|T|}$

1.  $w^{(0)} = 0; i = 0$
2. for  $n : 1..N$
3. for  $t : 1..|T|$
4. Let  $G' = \operatorname{argmax}_{G'} w^{(i)} f(G')$
5. if  $G' \neq G_t$
6.  $w^{(i+1)} = w^{(i)} + f(G_t) - f(G')$
7.  $i = i + 1$
8. return  $w^i$

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 10

This will be my learning criteria and this will be very similar to what we did in the previous example that the previous method also. So, what is the idea? You have a gold standard training data. So, you are given a training data of a sentence and its gold standard dependency graph. So, there are some capital T sentence that is (Refer Time: 09:12). Now start with some initial weights they can be 0 or they can be some arbitrary weights and let us start with the first iteration now I am iterating over all the sentences some capital n number of times like we did in the previous case in each iteration I go to each of the individual tree, so, 1 to up to capital T.

(Refer Slide Time: 09:55)





Each sentence instance now what do I do. So, I am in the learning part, I do not know, what are my optimal weights? So, how does learning work. So, I am given a sentence  $x$  and its dependency graph  $G$ , I am already given this I know what are my features yes I know what are my features I do not know what are my weights. So, how do I proceed?

Let us take a particular instance for my learning. So, what will happen? I am given this sentence  $x$ . So, I will apply my algorithm as if I am doing it at run time. So, I will make the root node, I will make all the possible connections, somehow, we can have any possible connection that is not matter. So, now, you start by making all the possible connections that we discussed, this is my initial graph. You can (Refer Time: 10:57) that graph  $G_x$  that has all the possible connections this also all the possible connections.

Now, at this point how will you proceed for proceeding you need to know, what are the edge weights here? So, now, how do you obtain the edge weights? So, this edge; it will be nothing but  $W_{ijk}$ ,  $i$  is the  $i$ th word here,  $j$  is the  $j$ th word here and  $k$  is the relation, now how do you obtain  $f_{ijk}$ ? This already given to you, so you know, what are the features? So, features will be some question like  $i$ th words is root and  $j$ th word is something like saw it will be either 1 or 0, yes. So, your feature vector will be some 0 some ones and this might be different for different edges.

Now, what else you do not know your weight, but you will have initial weights or some intermediate weights you will use those weights multiply by the feature vector and you will get some number and that is your  $w_{ijk}$  yes and you will put back this number it can be say 10 or 30, whatever suppose you get 10, 30, 20, 15 and so on, you will filling all the values using the same method. Now once you have filled in all these values what will be the next step. So, next step should be apply Chu Liu Edmond's algorithm yes and obtained a dependency graph and in other words obtained a MST. So, suppose you obtain a graph like root here. So, there are 3 words. So, something like this suppose let us take a simple case and like this, this a one of the dependent one way of writing dependency graph and suppose this is what you obtain from your algorithm.

Now, where is the learning here learning is if the dependency graph that you obtain is if this is not the same as your gold standard dependency graph; that means, your weights are not correct at this point you might have to modify your weights and how do you modify your weights very simple like you did in the previous method also you would say

$w_{new}$  or  $w_{old}$  plus you want to go towards the actual graphs and away from them. So, call it  $G_{prime}$  that you obtain from by your algorithm. So, call it  $G_{prime}$  that you obtain by your algorithm;  $G_t$  is the gold standard graph.

You want to go in this direction and away from here. So, it will be simply feature of  $G_t$  minus feature of  $G_{prime}$  and what is  $F(G_t)$ ? This is nothing but the summation over features of all the edges of  $G_t$ , similarly here summation over features of all the edges of my  $G_{prime}$  and that will give you your new weight vector of course, there can be some learning weights and all, but this is just to give you the intuition and that is all you will have a new weight. Again you will start with this new weight for a new sentence see whether you obtaining the  $G_t$  like show graph by this algorithm if not you will keep on updating your weights and this will sometime at some point converge and some of one of your iteration you will have your weights roughly stabilized and that is where you stop. So, now, let us so, hope this idea is clear, let us take a simple example to further see that how do you apply this algorithm for learning the weights.

Let me just quickly go through this algorithm again. So, I am repeating it for some  $n$  number of iterations for each tree let us find out  $g_{prime}$  as the Argmax over weight at the particular instance and  $f(g_{prime})$ . So, that is I apply my Chu Liu Edmond's algorithm find out the tree with the maximum weight, if  $G_{prime}$  is not the same as my gold standard graph; that means, I have to update my parameters and how do I do that at this  $i$  plus one th time my weights are initial weights plus  $F(G_t)$  minus  $F(G_{prime})$ , this I am doing only if I am not obtaining the correct graph, I continue further and at some point it will converge and that is where I will return the weights and I will assume at this point I have the ideal weights. So, that whenever you give me a new sentence I can use these weights to compute all the arc weights yes that arc weights are nothing, but weights multiplied by the feature vector and then I can I can apply Chu Liu Edmond's algorithm to compute the dependency graph.

(Refer Slide Time: 16:28)

*Example*

Suppose you are training MST Parser for dependency and the sentence, "John saw Mary" occurs in the training set. Also, for simplicity, assume that there is only one dependency relation, "rel". Thus, for every arc from word  $w_i$  to  $w_j$ , your features may be simplified to depend only on words  $w_i$  and  $w_j$  and not on the relation label.

Below is the set of features

- $f_1$ :  $\text{pos}(w_i) = \text{Noun}$  and  $\text{pos}(w_j) = \text{Noun}$
- $f_2$ :  $\text{pos}(w_i) = \text{Verb}$  and  $\text{pos}(w_j) = \text{Noun}$
- $f_3$ :  $w_i = \text{Root}$  and  $\text{pos}(w_j) = \text{Verb}$
- $f_4$ :  $w_i = \text{Root}$  and  $\text{pos}(w_j) = \text{Noun}$
- $f_5$ :  $w_i = \text{Root}$  and  $w_j$  occurs at the end of sentence
- $f_6$ :  $w_i$  occurs before  $w_j$  in the sentence
- $f_7$ :  $\text{pos}(w_i) = \text{Noun}$  and  $\text{pos}(w_j) = \text{Verb}$

The feature weights before the start of the iteration are: {3, 20, 15, 12, 1, 10, 20}.

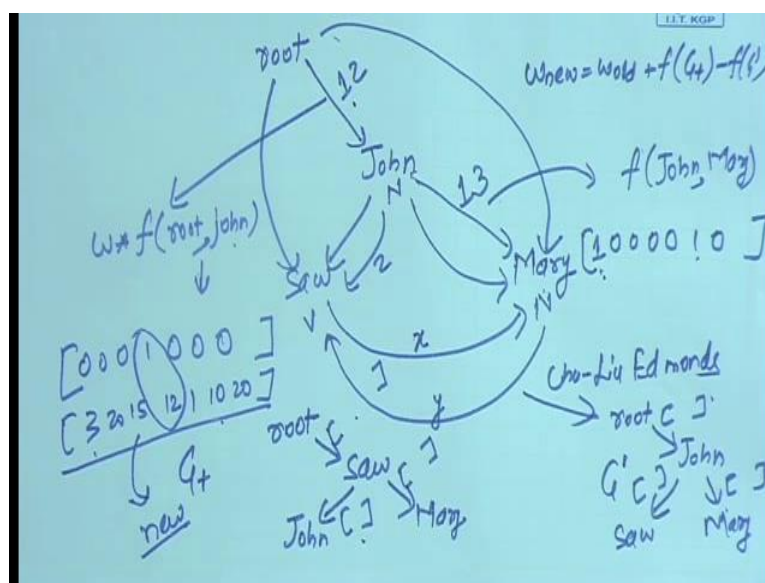
Determine the weights after an iteration over this example.

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing: Learning Week 6, Lecture 5 10 / 10

Let us see some example for learning the weight vectors. So, we are using the same sentence like the previous case. So, we are using the sentence John saw Mary and suppose that this occurs in the training set now for simplicity we are not worrying about what are the relations between 2 words I am saying that there is only one relation rare. So, my features will only dependent on the  $i$ th word and  $j$ th word that will make things easier. So, here what I am showing? Let us take some 7 different features and some weights that have been initialized or some intermediate weights and suppose you are un counting this sentence in your training data how will you do the weight update.

Here let us see what are my 7 features defined over any word  $i$  and  $j$  for a arc from  $w_i$  to  $w_j$ . So, features are the  $i$ th word has a part of speech of noun and  $j$ th word has a part of speech of noun,  $i$ th word is part of speech of verb,  $j$ th word is part of speech of noun and so on,  $i$ th word is root,  $j$ th word is verb and for this simple sentence you already know.

(Refer Slide Time: 17:39)



You will have root John, let me write it like that saw Mary you know this is a noun, this is a verb and this is a noun and this is root. So, now, your task is suppose you are given for these 7 features the weights are 3, 20, 15, 12, 1, 10, 20, before the start of the iteration, now you have to determine the weights after iteration over this example, now what do I mean by doing that same as I said in the last slide. So, what I will do? Now I will first find out all the possible edge weights, let us try to find some edge weights what is the edge weight from root to John. So, what is this edge weight? This is weight vector multiplied by the feature vector between this root and John this might be  $W_i$ , this might be  $W_j$ , now what is this  $F$  root to John? This will be a vector of size 7, each value to be 1 and 0 and this will depend on the answer of my question.

Let us see what are my features word?  $W_i$  has a part of speech of noun now  $W_i$  is a root. So, it does not have any part of speech this will be 0  $w_i$  has part of speech of verb again 0,  $W_i$  is root, let's see the next 1  $w_j$  is verb. So, here  $w_j$  is John that is noun. So, this is also 0 root and  $j$  is noun this will be 1 root and  $w_j$  occurs at the end of the sentence know John occurs in the John occurs in the start of the sentence this is also 0,  $W_i$  occurs before  $w_j$  in the sentence roots does not occurs in the sentence. So, this will be 0 also  $W_i$  is the part of speech of noun. So, it is not true for root. So, we see out of 7 features only 4th one is one everything has a 0. So, the feature vector here is 0001000, now how do I get the weight the weight vector is already given to me that is 3, 20, 15,

12, 1, 10, 20. So, I multiply this weight vectors to the feature vector and I obtain the weight of this arc that is 12.

Similarly, I compute the weight of the arc of from root to saw. So, let us take some other case like John to Mary, how do I compute the arc weight? Again this is let us find out the feature for John to Mary. So, first 1, first question again  $W_i$  is the part of speech of noun  $w_j$  is noun, now John and Mary both are nouns. So, this should be 1,  $W_i$  is verb and  $w_j$  is noun should be 0,  $W_i$  root none of these. So, none of these features will be 1 because all the required (Refer Time: 21:12) to be root. So, the all 3 are 0. So, 2, 3, 4, 5 are 0 6,  $W_i$  occurs before  $w_j$ . So, John occurs before Mary in the sentence there should be 1 7th feature,  $W_i$  had part of speech of noun,  $w_j$  is verb that is 0. So, only 1 and 6 are 1, everything else are 0. So, it will be 1000010 and multiplied with the weight vector. So, similarly first element and 6th element; 3 plus 10, so 13, so here the weight will be 13.

Like that you have to find the feature vector for each edge multiplied by the weight vector and obtain these values you will said get some values here x y z etcetera now once you obtain all these values what you will do apply Chu Liu Edmond's. So, once you have all the weights, you can very easily obtain Chu Liu Edmond's and you will obtain what is your dependency graph now what will happen suppose you find a graph like this root John saw Mary suppose, this is your output after applying Chu Liu Edmond's, now what you will do? So, you will; this is your  $G$  prime by applying Chu Liu Edmonds. So, your first thing you will check is that what is your  $G_t$ ? So, your  $G_t$  is root remember you will first have connection from root to saw to John and Mary, this is my  $G_t$ , are they same? They are not the same. So, I will update my weights.

And how do I update my weights (Refer Time: 23:11) should be  $W_{new} = W_{old} + f(G_t) - f(G_{prime})$ , now what is  $f(G_t)$ ?  $f(G_t)$  is the feature of this graph that is this feature vector plus this feature vector plus this feature vector 3 feature vectors combined and  $f(G_{prime})$  is this feature vector plus this feature vector plus this feature vector and these are all some 1s and 0s. So, weights will be nothing, but these weights plus summation over all these 3 elements of features minus all these 3 elements and that will give you the new set of weights and then you will continue with this new set of weights for the next example if you have to do.

This is in a nutshell, what we will be doing, but again I will encourage you to go through these examples fully find out all the weight vectors yourself see whether you are getting the same tree as the gold standard if not how will you update your weights in this will be your next exercise. So, here we are not going to very much detail in how we are going to learn it by various learning rates and all just an intuition that how do you pose this as a learning problem? So, in the last module of using (Refer Time: 24:45) parsing or transition parsing and this module of using MST based parsing, we have seen that how we can solve a problem dependency parsing in a detailed different manner. So, we treated as some sort of algorithmic way I was starting from some initial configuration going to some final configuration that resembles a dependency graph.

And in the in between for example, how are we taking transitions. We will determine by using various training data or how are we taking the edge weights, we are learning by training data and this would help you to also think about in terms of some other problems that how can I pose them in machine learning way and I was also saying there are some other they are some other or I would say even many other methods of for doing dependency parsing. So, we will not be covering everything in this course, but now whatever we have covered will help you to take any new approach and understand that. So, this also ends our discussion on main discussion on syntax. So, we had talked about starting from word order information language models to part of speech and morphological information to various syntaxes in the sense of constituency parsing and dependency parsing.

From the next week, we will start our discussions on semantics. So, we will first see what is called different methods by which being capture the meaning; the semantics. So, we will have distribution semantics and (Refer Time: 26:19) semantics for that.

Thank you, I will see you in the next week.