Natural Language Processing

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

Lecture - 29
MST - Based Dependency Parsing

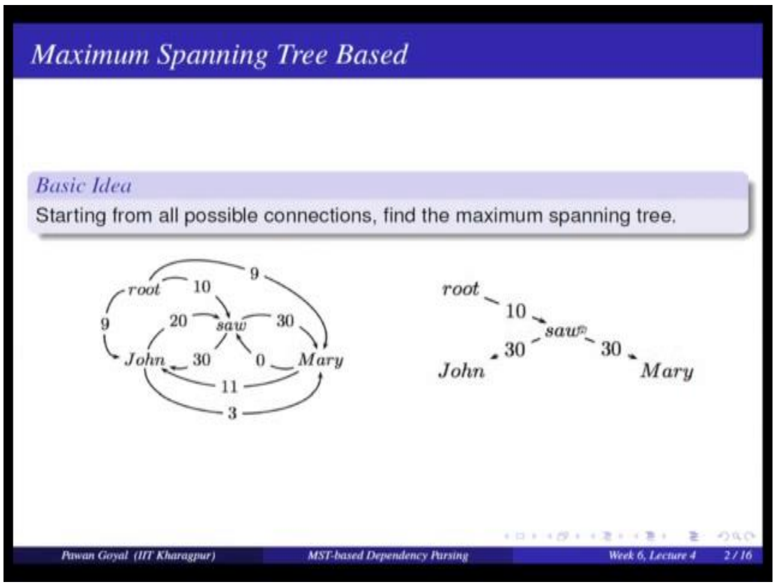
Welcome back for the fourth lecture of this week. So in the last 2 lectures we had discussed a particular (Refer Time: 00:26) dependency parsing that was a transition based parsing method. And we had discussed how we can formulate parsing as a particular problem where we starting with an initial configuration and moving from one configuration rather by using various transitions. And we modeled or learned these transitions using some training data that we already had. So this was a data driven approach.

Now, in this lecture we will talk about another data driven approach here. So the one main difference from the previous one is that we do not assume that projectivity constraint over the dependency graph. So in the last method that we had covered we were assuming that the dependency graph that we want should be projective although later on there are some variations proposed where this constraint is not required. So you might if you want to look into those.

So coming to what we are going to discuss in today's lecture and in the next lecture. So this is the approach using where we are formulating dependency parsing as a problem of finding maximum spanning tree.

Now, what is the basic idea? So we are starting with the sentence that is given to me. So this is the sentence for which I want to find the dependency graph. And how do we formulate the problem. Firstly, think about the sentence as a set of nodes these are the words and assume that all the possible nodes are connected to each other. We will also include additional node called root so that we can find out what is the main head of this sentence now. Once we start by assuming all possible connections my problem is to find out what is the maximum spanning tree from this all the connections that I can build. And this I would assume corresponds to my dependency graph. And the weights etcetera that will define for this graph will be learned by using some training data that will be available to me.

(Refer Slide Time: 02:35)



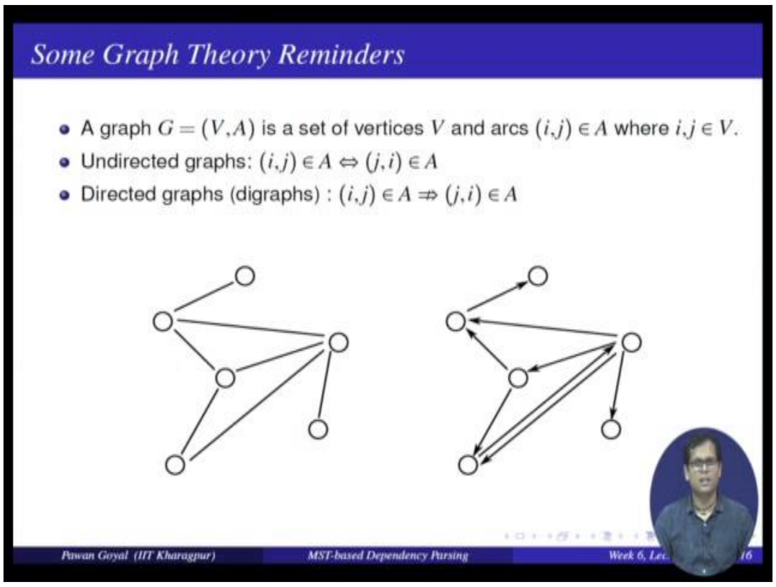
Maximum Spanning Tree Based

Basic Idea

Starting from all possible connections, find the maximum spanning tree.

So the main idea here is I am given a dependency graph. So I start with a sentence John saw Mary. And I assume an additional node like root. Assume that initially everything is connected. So from root there are all possible connections to all that node. And each node has an incoming and an outgoing edge with respect to every other node. How do we learn these edge weights? This is something that we will discuss, but assume that the weights are already given to you. Now your problem would be how do I find out what is the maximum spanning tree from this graph. And this suppose this is the maximum spanning tree then this is what I will assume corresponds to my dependency graph. So my problem is starting from here to come up with the maximum spanning tree that might be dependency graph.

(Refer Slide Time: 03:24)



Some Graph Theory Reminders

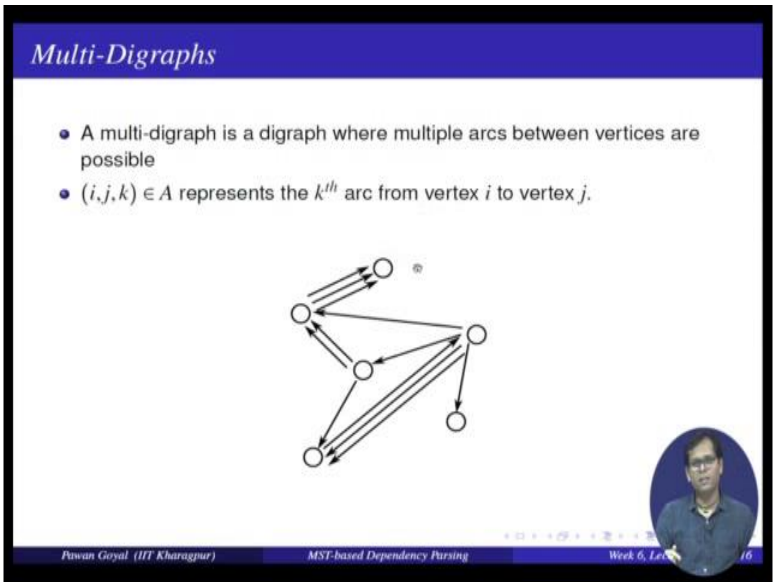
- A graph $G = (V, A)$ is a set of vertices V and arcs $(i,j) \in A$ where $i,j \in V$.
- Undirected graphs: $(i,j) \in A \Leftrightarrow (j,i) \in A$
- Directed graphs (digraphs) : $(i,j) \in A \nRightarrow (j,i) \in A$

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

So now starting with some simple reminders from graph theory on what is a maximum spanning tree. So you are now familiar with what is a graph. So when I say graph this is a set of V that is set of vertices and A set of arcs. So arcs generally connect to different nodes in the graph. So arc is connecting to nodes i and j both i and j are in the set V. Now if I talk about undirected graph like the figure on the left. So if I say that i and j are in the set of arcs then j and i are also there is no directions in this in this set of arcs.

But if I see that my graph is directed. Then if i and j are in the set of arcs then j and i may not be in the set of arcs. Like here I have a connection from this node to this node, but no connection from this node to this node.

(Refer Slide Time: 04:26)



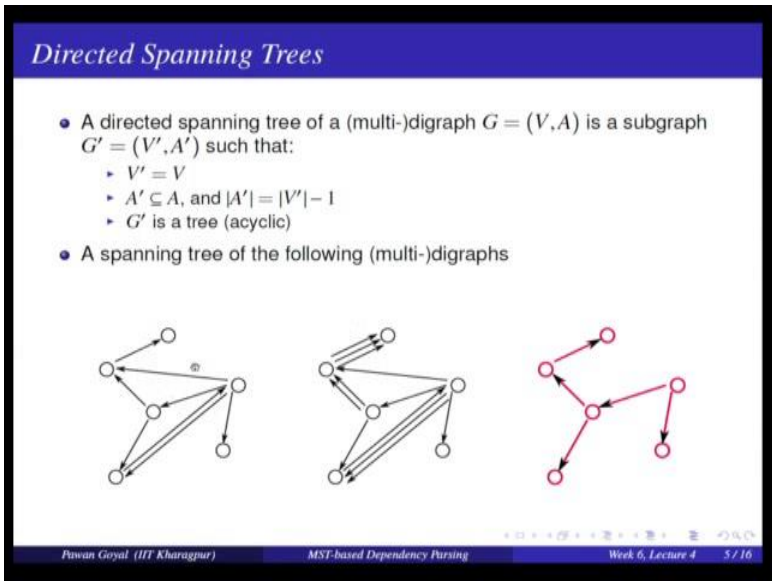
Multi-Digraphs

- A multi-digraph is a digraph where multiple arcs between vertices are possible
- $(i,j,k) \in A$ represents the k^{th} arc from vertex i to vertex j.

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lec

So what is a simple graph and what is a directed graph? Directed graph can also be written as a digraph. Now we can also see what is a multi-digraph. A multi digraph what would happen between a set of vertices? There might be more than one possible arc. Like here you are seeing between this node and this node there are 3 different arcs. So that is why you will have an additional index saying i j and k between the nodes i and j what is the kth arc. So it is similar to the previous one except that between 2 vertices you can have more than one arc.

(Refer Slide Time: 05:12)



Directed Spanning Trees

- A directed spanning tree of a (multi-)digraph $G = (V, A)$ is a subgraph $G' = (V', A')$ such that:
  - $V' = V$
  - $A' \subseteq A$, and $|A'| = |V'| - 1$
  - G' is a tree (acyclic)
- A spanning tree of the following (multi-)digraphs

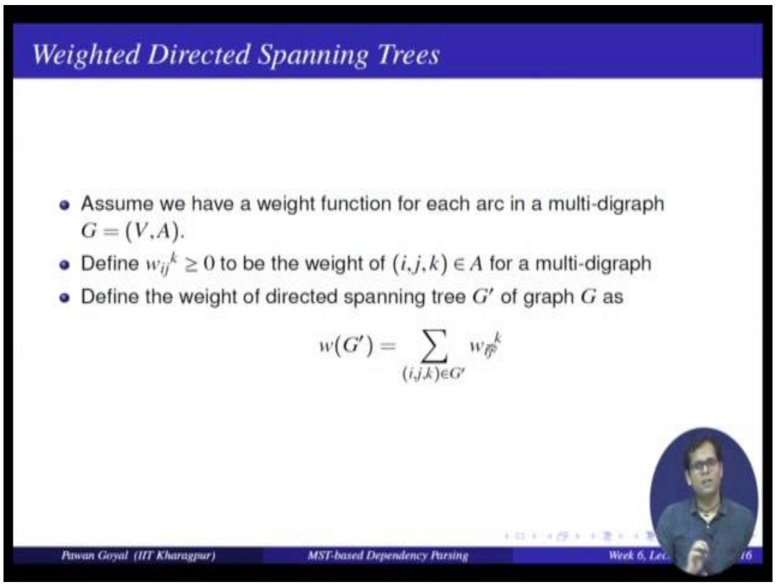
Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lecture 4 5 / 16

Now, once we know what is a multi-digraph, let me define the notion of what is the directed spanning tree of the multi digraph. So once I start with the multi digraph that is set of nodes V and there is a sub graph G prime that is some V prime a prime is called a directed spanning tree if the following conditions are followed. What are the conditions? In the directed spanning tree, the number of the set of nodes are the same as the set of nodes in the original graph.

So I will have to take the all the nodes from the original graph. So now, what do I do with the edges? So the set of arcs that I take in G prime is the subset of the original set of arcs with the constraint that the number of edges that I have now is equivalent to number of nodes minus 1 and the third condition is the graph G prime is a tree. It is there is no cycle in the graph. So let us take example.

So I have 2 different multi digraphs that are given in the first and second figure. And you are being shown a directed spanning tree for this multi digraphs. So what do you see here? So it has the same number of nodes as were there in the original multi digraph and what about the number of connections. So you see there are 1 2 3 4 and 5 connections and number of nodes are 1 2 3 4 5 6. So number of connections are exactly number of nodes minus 1. And you can take any possible any possible 5 connections in among these nodes such that there is no cycle and we called a directed spanning tree.

(Refer Slide Time: 07:14)



Weighted Directed Spanning Trees

- Assume we have a weight function for each arc in a multi-digraph $G = (V, A)$.
- Define $w_{ij}^{k} \geq 0$ to be the weight of $(i, j, k) \in A$ for a multi-digraph
- Define the weight of directed spanning tree G' of graph G as

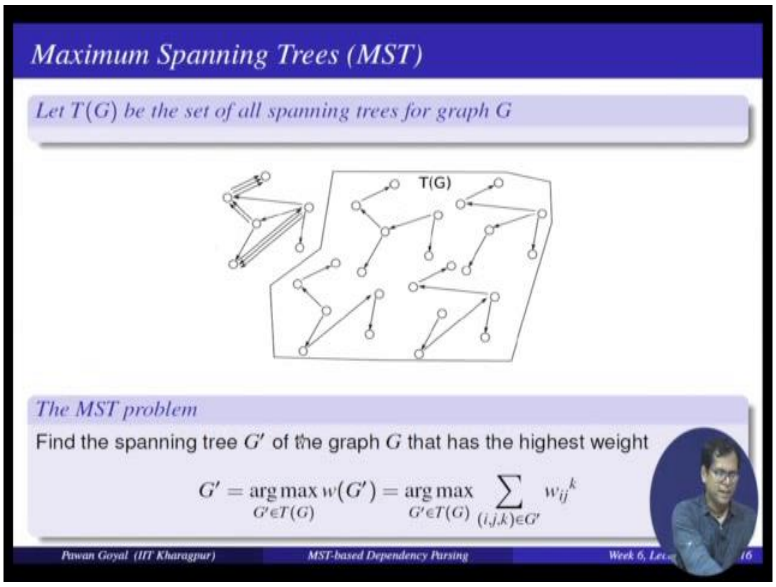
$$w(G') = \sum_{(i,j,k) \in G'} w_{ij}^{k}$$

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lec

Now, once we know what is directed spanning tree we can also define what is a weighted directed spanning tree. So here what we will do? In my graph with all the arcs I will also associate a weight. So each arc is now labeled or weighted by some numerical value and this I will call at wijk for the i between the vertices i and j what is the weight for the kth arc. So in w ij superscript k is the weight of the edge i j k. Now I will also define what is the weight of my directed spanning tree. So this will be simply summation over the weights of all the edges in my directed spanning tree. So from my multi digraph i find a directed spanning tree definition was already covered in the previous slide.

Now, whatever edges I am obtaining I will find out the weights for each of these and sum over them. And that will define the weight of my directed spanning tree. Now the idea here is for a given starting from a given multi digraph you can obtain a number of different directed spanning trees. So your problem is to find out which of these has the maximum weight. So now, I by this concept of weighted spanning tree I have also define what is the weight of the directed spanning tree. So now, among all the possibilities I will find out the one that is having the maximum weight.

(Refer Slide Time: 08:47)



Maximum Spanning Trees (MST)

Let $T(G)$ be the set of all spanning trees for graph G

The MST problem

Find the spanning tree G' of the graph G that has the highest weight

$$G' = \arg\max_{G' \in T(G)} w(G') = \arg\max_{G' \in T(G)} \sum_{(i,j,k) \in G'} w_{ij}{}^k$$

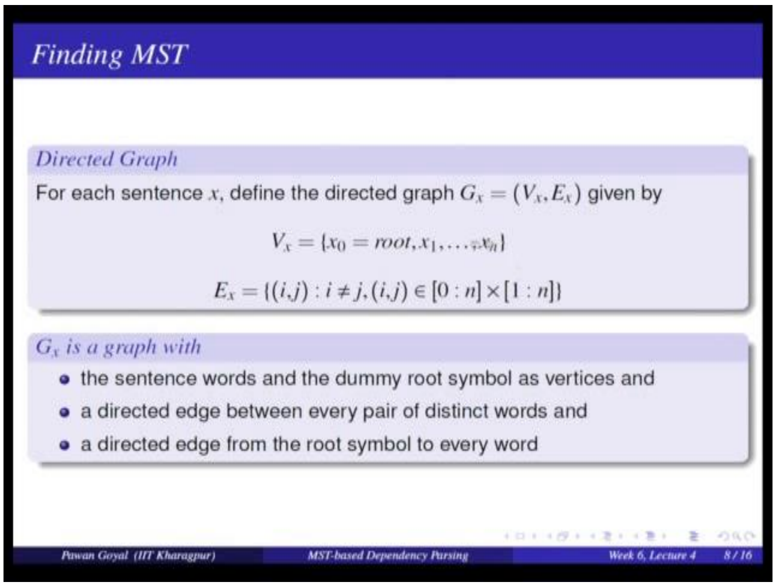
Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lec 16

So it is starting from the graph G let us say T G is the set of all the possible spanning trees that I can obtained. So like this is this is an example this multi digraph is given and there are these 4 possible spanning trees. So these are all directed spanning trees. Now I have to find out which of these has the maximum weight. So this is my MST problem.

Finding the spanning tree G prime of the graph G that has the maximum weight and the weight is written simply as summation over the weights of all the edge of that graph. So I find out all the edges of that graph, sum over the weights and that will give me the weight of the final graph.

So now my problem is from my sentence, whenever I construct a multi digraph that is all the possible connections, a fix set of directed spanning trees are possible. Now among those which is having the highest weight and that is the one that I will say as my dependency graph. So now, there are many questions like how do I convert a sentence to an initial configuration. Analogous to what we did in the previous method. Then the important problem here is how do we define the weights of my edges. And once I define the weights how do I choose the maximum spanning tree. So these are 3 different problems and then that is what we will be studying in this lecture and in the next.

(Refer Slide Time: 10:25)



Finding MST

Directed Graph

For each sentence x, define the directed graph $G_x = (V_x, E_x)$ given by

$$V_x = \{x_0 = root, x_1, \ldots, x_n\}$$

$$E_x = \{(i,j) : i \neq j, (i,j) \in [0:n] \times [1:n]\}$$

G_x is a graph with

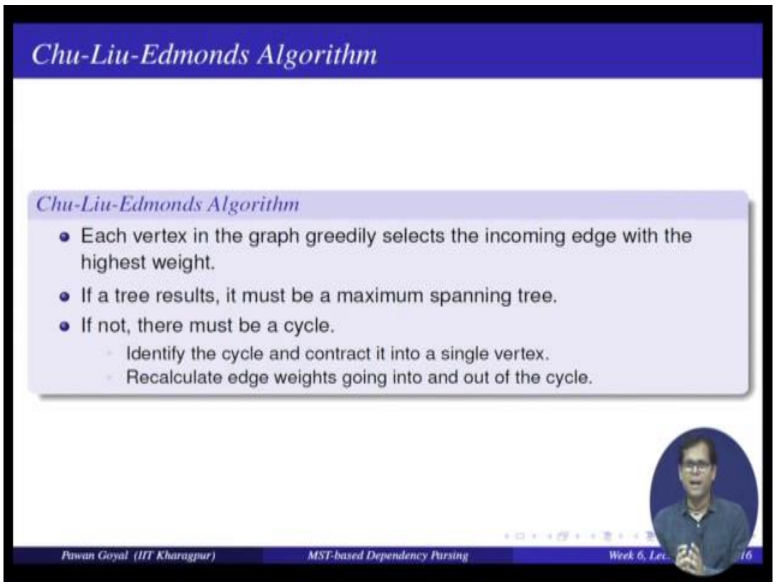
- the sentence words and the dummy root symbol as vertices and
- a directed edge between every pair of distinct words and
- a directed edge from the root symbol to every word

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lecture 4 8 / 16

So now for the sentence how do I find the maximum spanning tree? So let us see this is the first problem how do I convert the sentence to the initial configuration from where I can start, so what is that? For each sentence x you define the directed graph Gx as Vx Ex that is given by Vx contains and additional load like root and all the all the words that occur in my sentence x. And what are my edges I have all possible edges except for the same node I do not have any self-edge from the node to itself.

But I have edges from every node to every other node except that the word the root node will not have any incoming edges. That is why it is written here the edges are from the node 0 to n to one to n 0 denotes the root node from root there are only outgoing edges and for every other node they have both incoming edges and outgoing edges. So that is my Gx is a graph, where the sentence words and the dummy root symbol are the vertices. This is my set of vertices and there is a directed edge between every pair of distinct words yes that is what we have seen here. And a directed edge from the root symbol to every word from 0 to 1 to n this is my initial graph.

(Refer Slide Time: 11:54)



Chu-Liu-Edmonds Algorithm

Chu-Liu-Edmonds Algorithm

- Each vertex in the graph greedily selects the incoming edge with the highest weight.
- If a tree results, it must be a maximum spanning tree.
- If not, there must be a cycle.
  - Identify the cycle and contract it into a single vertex.
  - Recalculate edge weights going into and out of the cycle.

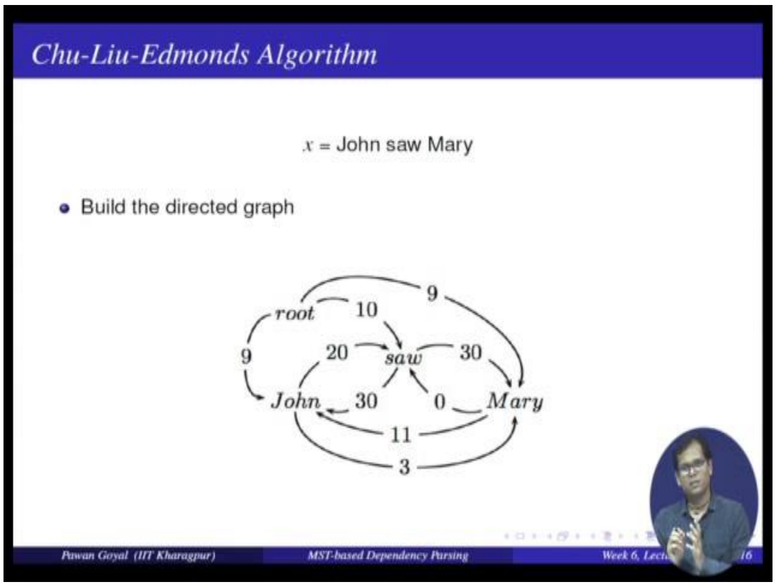
Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lec. 16

So now from a sentence I can construct this initial graph. And assume that I have some method of finding out what are the weights of my edges. We will discuss this this problem in the next lecture how do I find the weights, but for now let us assume that we have some method of finding the weights of my edges. So I obtain a multi digraph now my problem is how do I find the MST of that multi digraph. And for that we use a very famous algorithm called Chu Liu Edmonds algorithm. And that is the very simple algorithm as well.

So let us see what this algorithm does. So very simple steps each vertex in the graph greedily selects the incoming edge with the highest weight. So for each vertex in the graph I have incoming edges from every other node. So what we will do in the first, what I will do in the first step? For each node I will select the incoming edge with the highest weight. So now, I will now consider only those edges that have been selected. I will see whatever graph is resulting is it a tree or not. If it is a tree it has to be a maximum spanning tree. Because each node as related the maximum incoming edge. Now the problem is if whatever you obtain is not a tree. If this is not a tree; that means, there has to be a cycle.

Now, if this is a cycle then, there is a step that says you contract the vertices that are involved in the cycle in a single vertex. Now recalculate the incoming and outgoing edges for all those, for this new vertex that I have constructed, rerun the algorithm that is again for each vertex choose the edge with the highest incoming edge. Again see if the if whatever you have finding is a tree or not. If it is a tree you stop otherwise, you continue. And you can see that you will converge at some point. Why because at every step I either finding a maximum spanning tree. So I mean stopping or I am finding a cycle. If I am finding a cycle I will contract the vertices so; that means, it will reduce the number of vertices that I have in my tree. So it at some point I will have only one vertex. So that that is the point I will have to stop anyway. So this will converge. So now, again here are many questions like how do I compute the weight for the incoming arcs and outgoing arcs.

(Refer Slide Time: 14:28)



Chu-Liu-Edmonds Algorithm

x = John saw Mary

- Build the directed graph

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lect 16

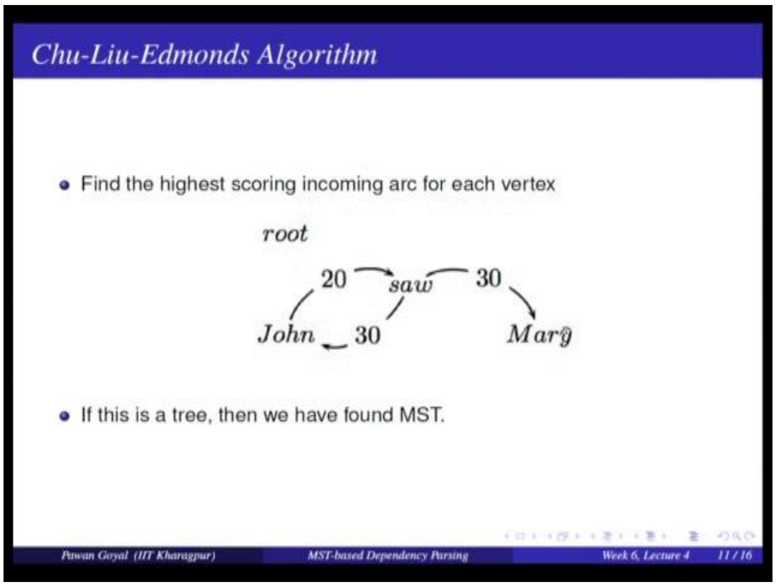
So let us see by an example. So I take the same sentence that we discussed in the previous method also the simple sentence John saw Mary. And I also know what is the dependency graph that I want to obtain. Now how do I start mounds algorithm. So let us see the first thing, would be I start by taking all these 3 words as my nodes plus root as an additional node. So here we have root John saw and Mary, has 4 different nodes. Next you make an edge between every 2 nodes in the graph.

So from root you will have only outgoing edges from root you have an edge to saw to John and to Mary and for every other pair of words you have an incoming and outgoing edges. So from saw to Mary and Mary, to saw, saw to John, John to saw and so on. And there are some edge weights and we are not bothering right now and how do we achieve these edge weights. So let us see I am given the sentence, I have some way of finding the edge weights and I complete I construct this initial multi digraph.

Now, once I have this this digraph what is the next step of my algorithm? The next or the very first step says that each node in the graph chooses the edge with the maximum. So chooses the incoming edge with the maximum weight simple. So there are only 3 nodes here that are having incoming edges. So let us see what is the edge that I will choose. So saw has incoming edge of weight 10 20 and 0. So it will have to choose there is with weight 20. Similarly, John has incoming edge of weight 9 30 and 11. So it will choose the one with weight 30 and Mary has 30 - 9 and 3, so it will choose 30 and then I will remove all the other edges, so I have, I will have only 3 edges in my graph.

Is this is this does the qualifier for the maximum spanning tree or the directed spanning tree. It satisfies the first tree first 2 conditions, number of nodes will be same as the original graph and number of edges will be one less than the number of original nodes in the graph. So I will now have 3 edges, one for each node as the incoming edge, but that condition is there should not be any cycle. So let us see do we get a cycle or do we get a tree.

(Refer Slide Time: 16:55)



Chu-Liu-Edmonds Algorithm

- Find the highest scoring incoming arc for each vertex

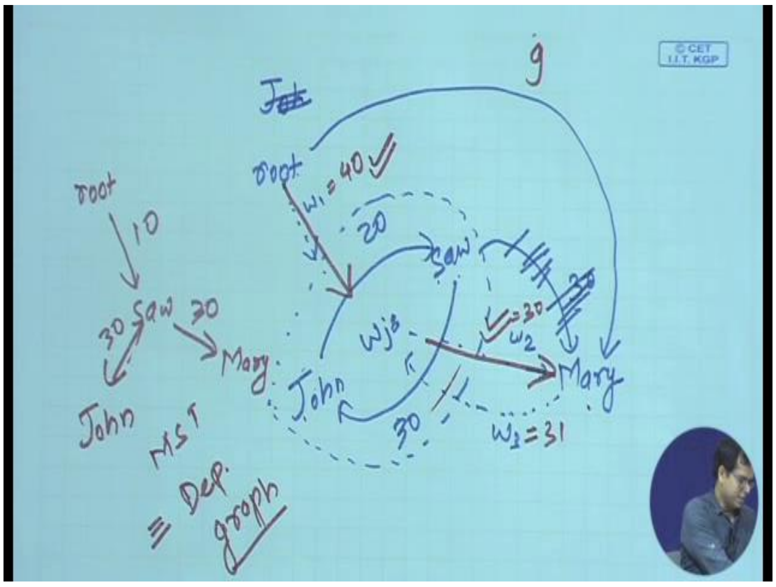
root
20
saw
30
John
30
Mary

- If this is a tree, then we have found MST.

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lecture 4 11 / 16

So if I do that I will get a graph like this. Because saw selects this edge, John select this edge and Mary selects this edge. So what are you seeing here if there is a cycle? So I find a cycle here from John to saw and saw to John so; that means, so I am not done yet I have to now continue my algorithm. So what was the next step? Whenever I find a cycle I contract that I make a single vertex.

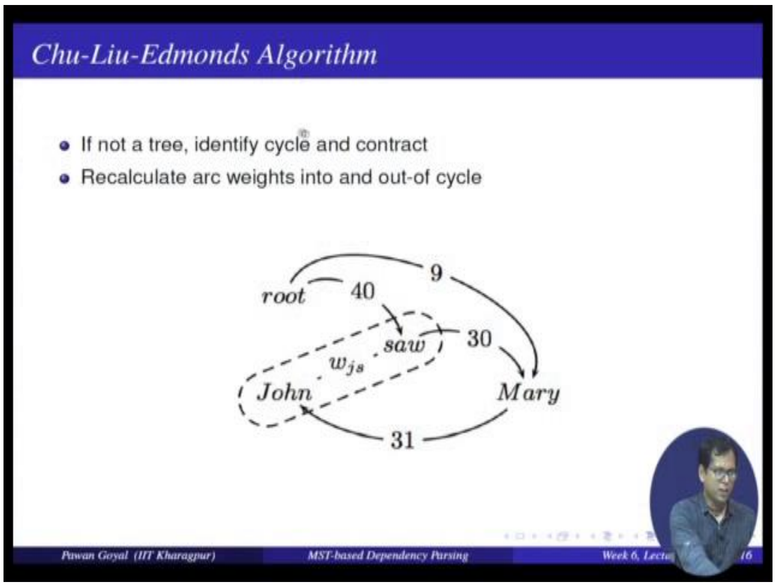
(Refer Slide Time: 17:29)



So now I have a root node, then John saw Mary. So what do I have found till, now 20 30 30. This is not a tree it is a cycle. So what is the next step? I will take it; I will contract it into a single vertex. So there is a word j and s let us let us call it wjs this is a single vertex now.

And I will have to repeat my algorithm. For repeating the algorithm, I should know to this vertex or to this new vertex w j s, what are the incoming and outgoing edges from all the nodes. So there are 2 nodes now, so root and Mary. So from root it will have an incoming edge from Mary it will have incoming edge as well as outgoing edge. So question is how do I compute the incoming and outgoing weights from this vertex and for that we have some very different tools. How do I compute the outgoing edge weights and how do I compute incoming edge weights? So let us see how many I have to compute, I have to compute this weight 1 I have to compute. So now, let us forgot about this edge I have to compute weight 2 and I have to compute weight 3 yes. And root to Mary will be already there that will not change, yes. So I have now 1 2 3 edges 3 vertices only and I have to compute these edge weights. So how do I do that?

(Refer Slide Time: 19:48)

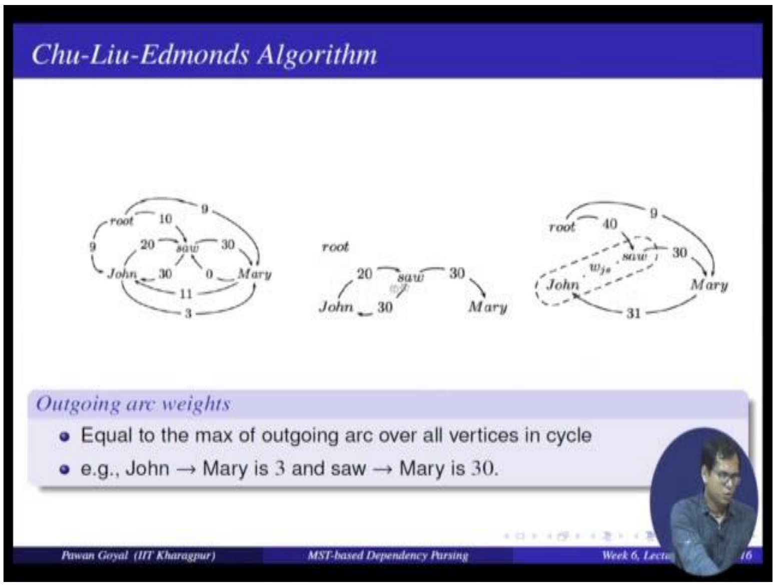


Chu-Liu-Edmonds Algorithm

- If not a tree, identify cycle and contract
- Recalculate arc weights into and out-of cycle

So if not a tree, you have to identify the cycle and contract that and you have to recalculate the arc weights into and out of the cycle. And what is the algorithm for doing that.

(Refer Slide Time: 19:58)



Chu-Liu-Edmonds Algorithm

Outgoing arc weights

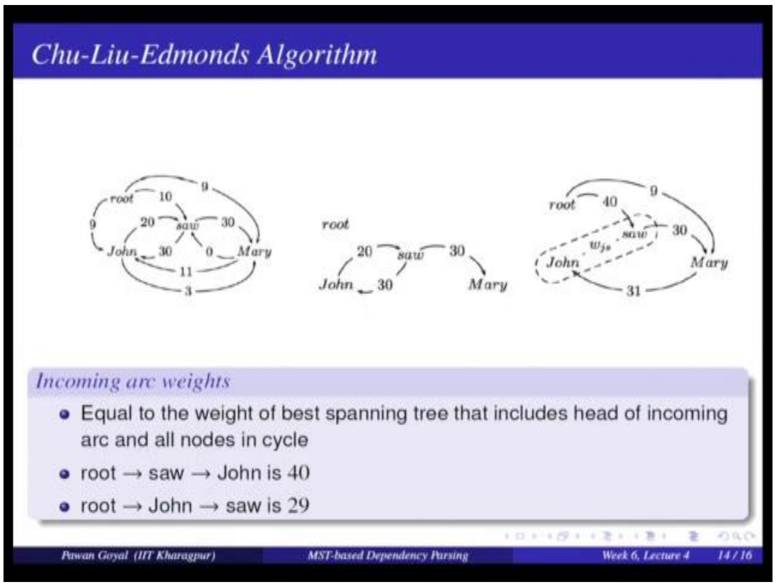
- Equal to the max of outgoing arc over all vertices in cycle
- e.g., John → Mary is 3 and saw → Mary is 30.

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lectu... 16

So algorithm is that for outgoing arc weights. You take the arc weights that are equal to max of outgoing arcs over all vertices in the cycle, what do I mean by this. So here is my contracted vertex John, and saw I want to compute the outgoing arc weight from this vertex to Mary yes. So this will be equal to the max of outgoing arcs from John to Mary, and saw to Mary. Which of these gives me the maximum will be final weight.

So here let us see. From John to Mary and saw to Mary; so John to Mary I have a weight of 3. And saw to Mary I have a weight of 30. So among these highest is 30. So I will pick the one with 30. And this is the final graph that is being shown. So from this vertex the outgoing edge to Mary will have a weight of 30. And that the only outgoing edge for this contracted vertex this is fine. Now how do I find out the weight for the incoming arcs? So incoming arcs are from root as well as from Mary yes, so from incoming arc the algorithm is different I do not choose the now. So I have to find out from root to this vertex. So I do not directly choose the one that is having the maximum. So what I do? I have to also consider this the edges inside the tree.

(Refer Slide Time: 21:44)


Chu-Liu-Edmonds Algorithm

Incoming arc weights

- Equal to the weight of best spanning tree that includes head of incoming arc and all nodes in cycle
- root → saw → John is 40
- root → John → saw is 29

Pawan Goyal (IIT Kharagpur) MST-based Dependency Parsing Week 6, Lecture 4 14 / 16

So what I will do? I will take the incoming arc weights as equal to the weights of the best spanning tree that includes the head of the incoming arc and all the nodes in my cycle. So what do I mean by that.

So let us take from root from root. There are 2 different ways of defining incoming arc weights, either it has to be root to saw and saw to John or root to john, John to saw. These are the 2 possible trees. So what I will do, I will find out the weight for individual trees and take which one is having the maximum weight. So let us say root to saw, saw to john. This is 10 plus 30 40 root to john, John to saw, this is 9 plus 20, 29. So I will take the weight as 40 and that is the edge weight from root to saw. This is having the weight of 40.

Now, can you compute the weight from, Mary to this vertex again I will see the 2 different ways Mary to saw, saw to john. So this is 0 plus 30. Choose this 30, second is Mary to John, John to saw and that is 11 and 20 31. So I have 2 ways 30 and 31. So I will choose one with 31. So it is starting from Mary to John and John to saw. So now, once I have computed all these weights I have now got the new tree on which I have to compute my algorithm. So here what did you find? This weight is equal to 40 this weight we found to 30 and this weight we found as 31 and this first (Refer Time: 23:38).

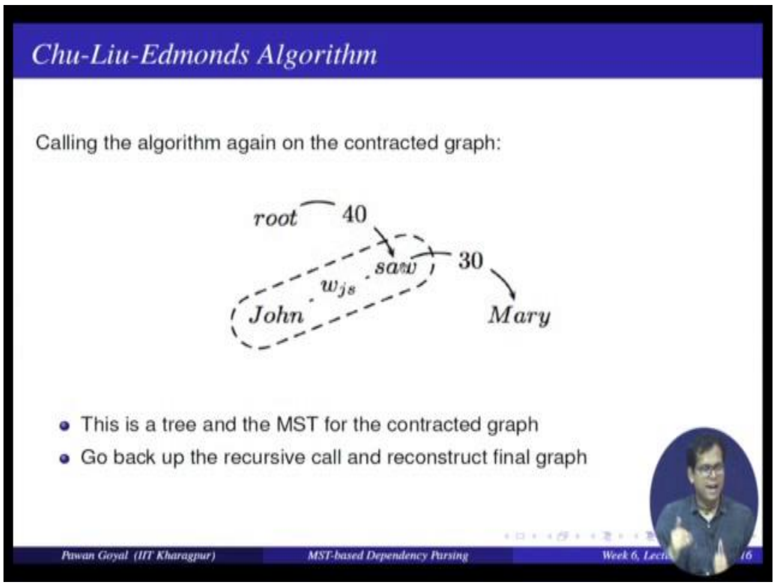
So now, once I have found these weights, what is the next step? I will have to see if each vertex now selects the incoming arc with the highest weight do we obtain a tree. So let us see, for this vertex incoming arcs are 40 and 31. So it will choose this arc for this vertex 9 and 30. So we choose this arc. So what is my, what is the graph that I am seeing. So let me color only the edges by this. So one edge is here and the second edge is here yes. So do you see it is a tree? It is a tree and this is directed spanning tree.

So am I done? So as one point I am done because there is no cycle. So I do not have to repeat my algorithm, but I still have not the optimal dependency graph. Why? Because I am now stuck with an edge from root to this contracted vertex and I do not know what happens inside. And I know there is an outgoing edge from this contracted vertex to Mary again I want to know what happened inside. So how do I construct my full graph and for that you have to go back to how you actually constructed these connections for this contracted vertex.

So what you will do? So the incoming edge from root to John and saw and the outgoing edge from John saw to Mary. So let us see how did we compute the outgoing edge. We said the outgoing edge should be the maximum of both the outgoing edges. So from John to Mary and from saw, saw to Mary and where did these come from this came from saw. So I will now see here this edge from saw to Mary that may write it 30. So this is done. So this outgoing edge is from saw not from john. That is how we found it out.

Now it is about the incoming edge. This is an incoming edge could have been from saw then John or John then saw. In our algorithm where did it come from? 40, so it came from root to saw and saw to John yes. So I have to construct now root to saw, saw to John. So root to saw and saw to John this was 10 and this was 30. And that is my final MST that is my dependency graph. So this is the algorithm.

(Refer Slide Time: 26:57)



Chu-Liu-Edmonds Algorithm

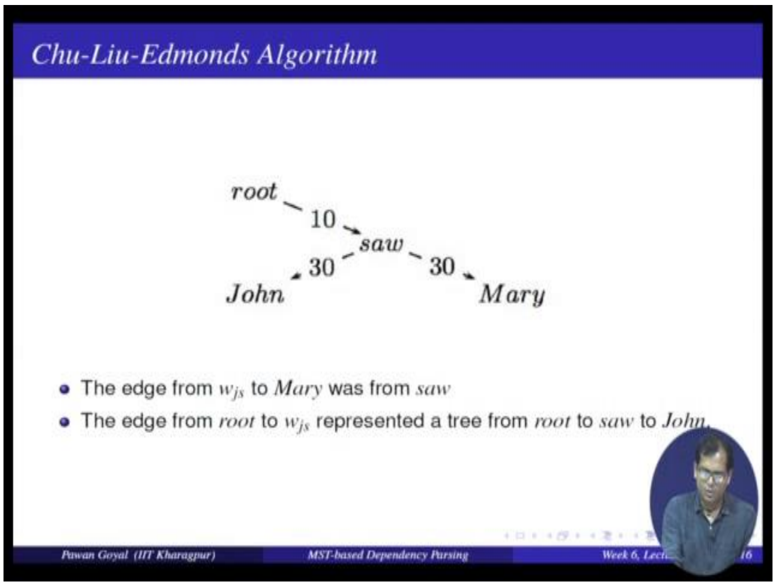
Calling the algorithm again on the contracted graph:

root
40
saw
30
w_{js}
John
Mary

- This is a tree and the MST for the contracted graph
- Go back up the recursive call and reconstruct final graph

So let us see again here. So you are at this step where you have this contracted vertex and you found out the maximum incoming edge for each vertex and you obtain a tree. So now, you have to go back after your recursive call and we construct the original graph. And there you have to find out where is this outgoing edge coming from is it coming from saw or John, and you found it is coming from saw incoming edge is it coming for this tree, or that is root to saw, saw to John or root to John, John to saw and that you can again find out where did you find this 40, it was from root to saw, saw to John.

(Refer Slide Time: 27:34)



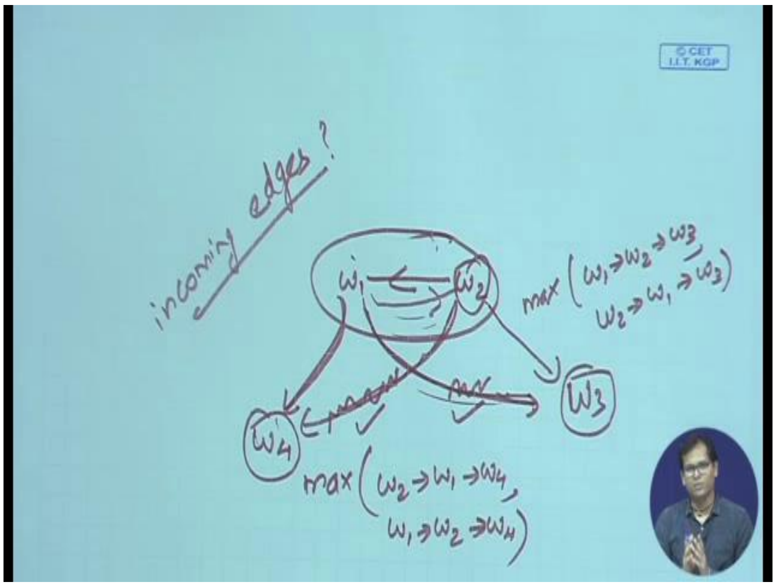
Chu-Liu-Edmonds Algorithm

root
10
saw
30
30
John
Mary

- The edge from w_{js} to *Mary* was from *saw*
- The edge from *root* to w_{js} represented a tree from *root* to *saw* to *John*

So yes the edge from wjs to Mary was from saw. So we construct that the edge from root to w j s, represents a tree from root to saw, saw to John and that is your dependency graph. So now, there might be a question in your mind that in my algorithm I have different ways of computing incoming arc weights and as well as outgoing arc weights they are not the same. So we need them to be different. So that we can also account for the weight for the connection inside the tree, but the question is can I reverse this, that is can I say that the outgoing weight should be the maximum of the directed spanning tree and the incoming weight should be the maximum among all the possible incoming weights. So I would like you to think about it, but I can give you basic intuition that why this is the case.

(Refer Slide Time: 28:40)



So let us see. So suppose I have a contracted vertex that has nodes word 1 and word 2. So right now what we are doing? We are saying the outgoing weight is max of to a vertex like w 3 it is a max of these 2. So we take the max here. Suppose there is another word x w 4. I will do the same the outgoing weight from here to w4 is the max of these 2.

Now when I have taken the max here and I am done my with my call and everything, each vertex will choose only one incoming edge weight. So w 3 will choose either this or this w4 will choose either this or this or anything else. So that means, I will know for sure whether this there is connection from w 2 to w 3 or w 1 to w 3 or there is no connection. And whatever I obtain even if I say that from w1 suppose my finally, I find out a situation like this that is from w 1 there are both the connections are there this is valid. From a because from a single node you can have 2 different outgoing edges, this is fine. Or if w 1 and w 2 that is also, in no case there will be a there will be 2 incoming edges for the same node this is not allowed by the algorithm itself.

Now, suppose I change my algorithm, and say that the outgoing edge is coming from the directed spanning tree. So that will be the situation where I will have to see. So this edge is the max of w 1 to w 2 to w 3 and the other one that is w 2 to w 1 to w3, yes. Similarly, this will be the max of w 2 to w 1 to w 4 and w 1 to w 2 to w 4. Now what might happened that in your final tree, that you obtain you find out this edge yes and this edge.

Now, you have to go back and construct your tree. So what you will see there, you will see that this edge is coming from by following this connection, and this is coming by following this connection. So you will end up by saying there is a cycle inside. And this will not terminate again you can also try to see what will happen in the case of. So this is only for the outgoing edges. So you see there might be a case where you can end up finding a cycle or may not be able to say which of these 2 is the correct sign between the edges. So this case might happen this will never arise in the way we have define the algorithm.

Similarly, you can try for the incoming edges. Would you end up with the situation where you are either violating some principle of the dependency graph or where you are getting the cycle and you are not able to converge? This is something I will say that you try to take both the cases and see whether you are finding why the algorithm proceeds in this way of for finding the incoming and outgoing edge weights.

So we discuss this algorithm of Chu Liu Edmonds and how do we use that for finding maximum spanning tree. So what we did not cover is how do we find the edge weights. And that is where the learning algorithm comes in that how do we find the edge weight from my dependency graph. And that is what we will see in the next lecture.

Thank you.