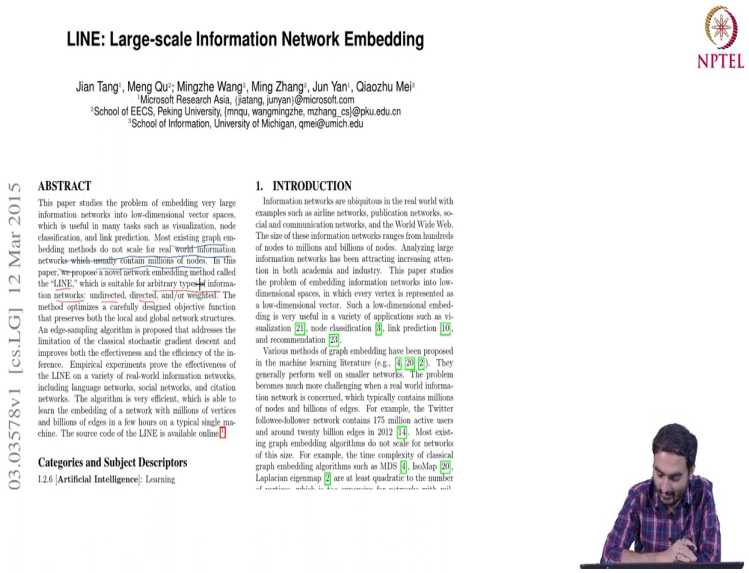
Social Network Analysis
Prof. Tanmoy Chakraborty
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Chapter - 09
Lecture - 07

Let us discuss another algorithm called a LINE, ok. This is another graph embedding algorithm. And LINE stands for Large Scale Information Network Embedding. So, the, so I am I am actually you know displaying the paper, the original paper of this method, ok. Just to show you how you know things have been written, and how we can also you know read a scientific paper.

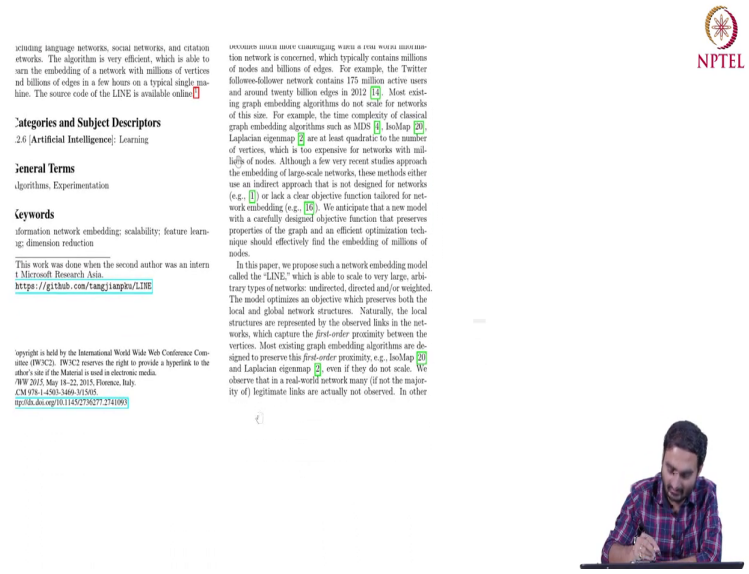
(Refer Slide Time: 00:48)



So, what is the claim here? The big claim is that most existing graph embedding methods do not scale for real world information network which usually contain millions of nodes, ok. So, this method is particularly designed for large network, ok. So, they proposed something called "LINE", which is suitable for arbitrary types of information network, be it undirected, directed, weighted, unweighted and so on and so forth, ok.

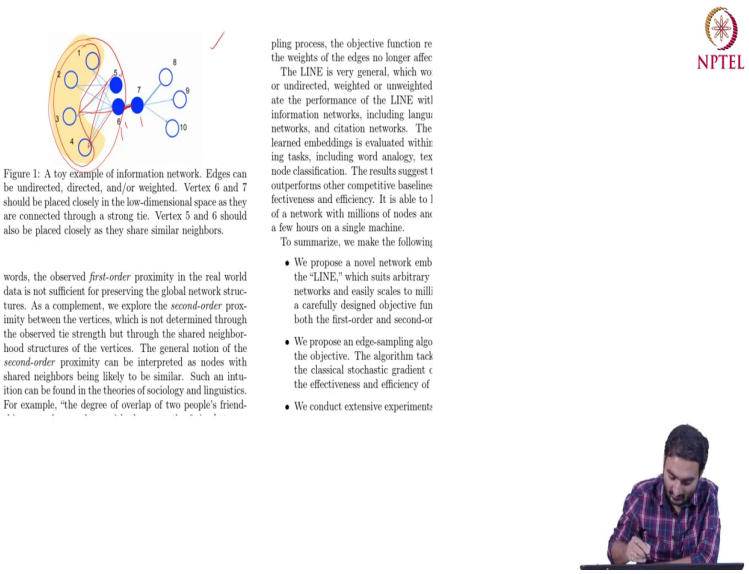
So, here the idea is also same. The idea is the idea is very same as the one which was used in hop.

(Refer Slide Time: 01:30)



The idea is that, it is not always the case that two nodes are closer only when they are directly connected, ok. It may also be the case that two nodes are closer, two nodes are similar if their neighbors are also similar, ok.

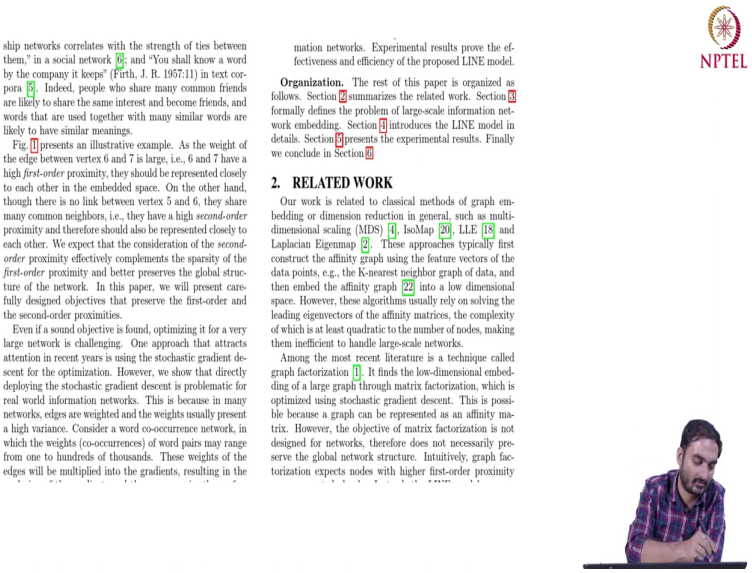
(Refer Slide Time: 02:04)



They kind of you know motivated this thing, motivated their algorithm by you know presenting kind of a dummy figure. This is the figure, ok. What they are saying is that node say 6 and node 7, they are of course close, they are of course similar, because they are directly connected, ok.

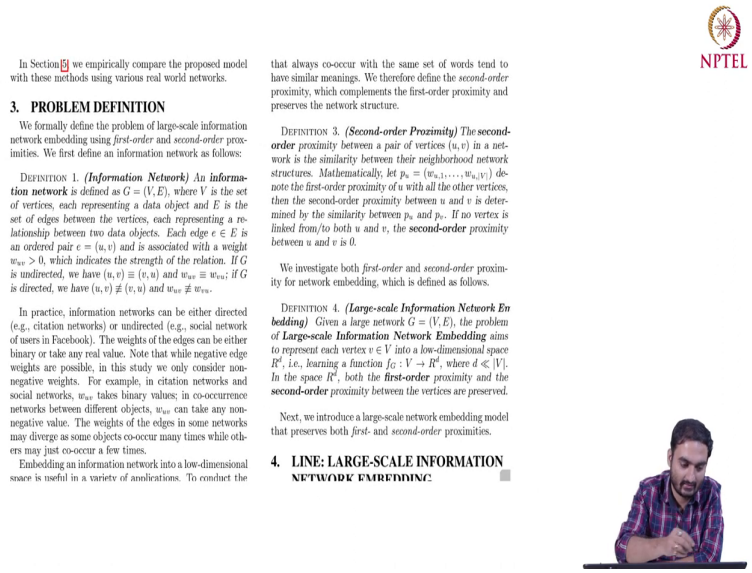
But node 5 and node 6, they are also similar although they are not directly connected. Why they are similar? Because they have common set of neighbors. 4 neighbors are shared by both 5 and 6, ok. So, this is called the first-order proximity. Because they are directly connected within one hop. And this is called the second-order proximity because 5 and 6 are similar with respect to two hops, right, with respect to two hops, right.

(Refer Slide Time: 03:06)



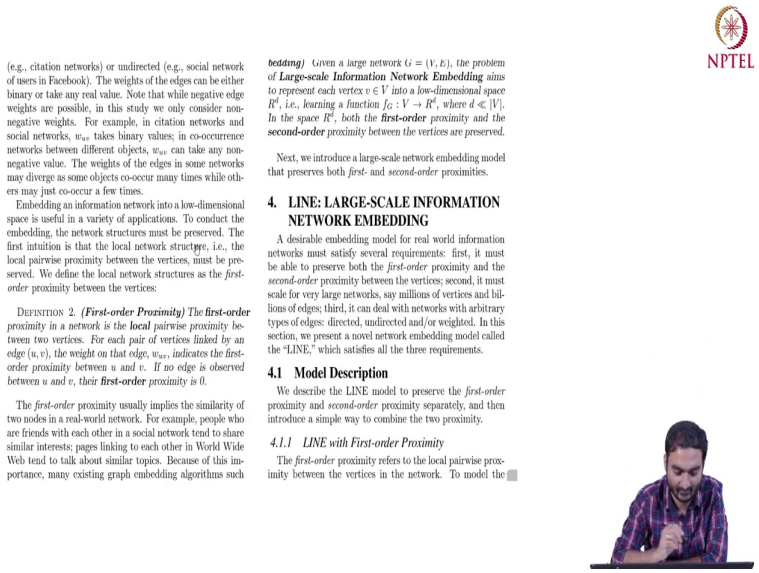
So, they carefully designed an optimization function which takes care of the first-order proximity and second-order proximity in a systematic manner, ok.

(Refer Slide Time: 03:18)



So, they started off by defining what is information network. Information network is basically a graph V comma E, V is a set of vertices, E is the set of edges. And every edge small e is an ordered pair u comma v, and is associated with an weight with a weight w, right, indicating the strength of the relation u and v. And if it is undirected, then this is symmetric. If this is directed, this is asymmetric, ok.

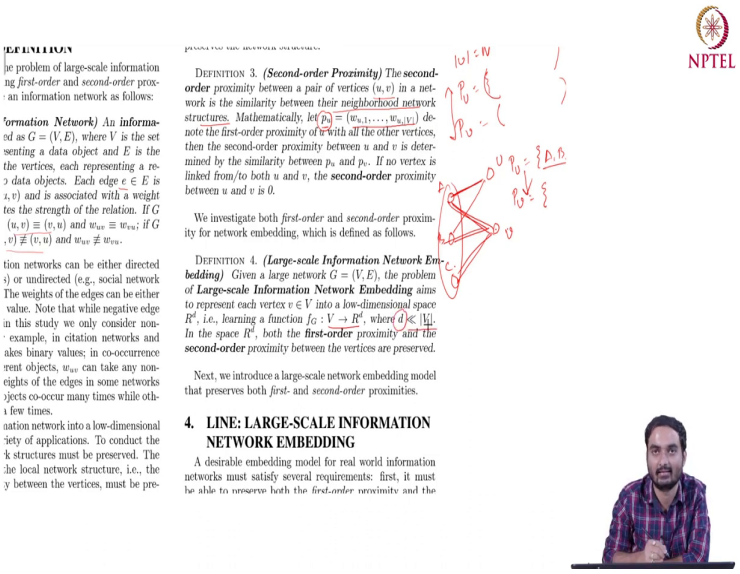
(Refer Slide Time: 03:54)



Then, they suggested something called first-order proximity. What is first-order proximity? The first-order proximity in a network is a local pairwise proximity between two vertices, ok. For each pair of vertices linked by an edge u, v, the weight on that edge w uvm indicates the first-order proximity between u and v, straightforward, ok. If no edge exists, then the first-order proximity is 0.

So, what is second-order proximity? Second-order proximity between a pair of vertices u comma v is a network in a network is the similarity between their neighborhood network structures, right.

(Refer Slide Time: 04:46)

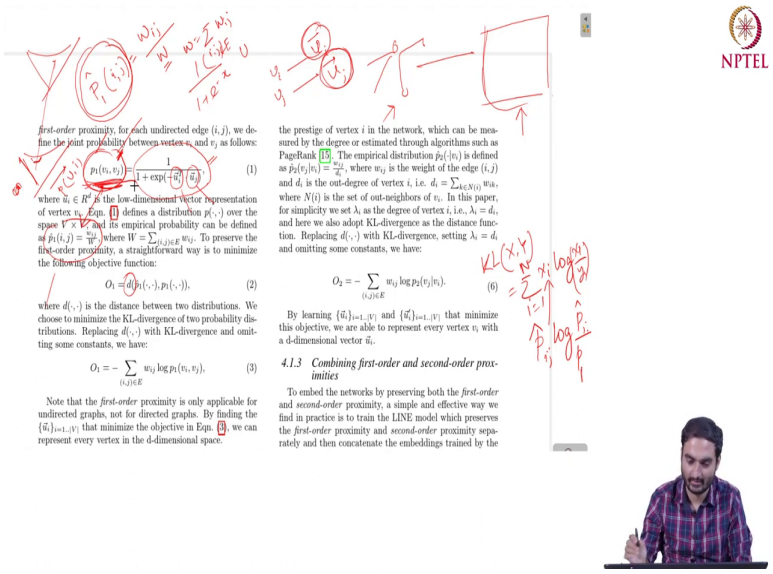


Mathematically, let p u be, p u denote the first-order proximity of u, right. You see that there are mod v number of, so mod v is essentially N, right. Let p u be the first-order proximity of u, right with the other vertices. Similarly, say for p u, you have a vector like this. For p v, you have also a vector like this, right.

So, the second-order proximity between u and v is determined by the similarity between these two, ok. This is essentially this one, right. This is essentially saying that this is a this is u and this is v, ok. And p u consist of say this is A, B, C; A, B, C with some ways, right. It may not be directly A, B, C, but say the weight between u A, u B and u C.

Similarly, for v you have p v which is the weight between A v, B v and C v, right. And then, the second-order proximity between u and v is the similarity between p u and p v. Meaning the similarity between the neighborhood structures, ok. So, and what is the goal? The goal is to come up with an embedding, right. Embedding of vertices, so that you map every node to a d dimensional space, where d is less than, less much much less than mod v, ok low-dimensional space.

(Refer Slide Time: 06:44)



Now, let us see how you how we, you know how we capture these two proximity measures, ok. So, right. So, what is the idea? The idea is that every vertex say you have u i v i and v j. We would try to come up with an embedding which is denoted by say u i, right vector, right. This is u j vector, right. This is these are the embedding of v i, v j respectively which we would try to come up with, right.

And the proximity the first-order proximity, so think about it, you have the graph, ok and you are mapping it, mapping all the nodes to an embedding space, right. So, you have first-order proximity with respect to the original graph, you have first-order proximity with respect to the embedding space, right. So, let us say these are the embeddings.

So, what is the first-order proximity with respect to the embedding space? The first-order proximity between v i and v j is defined in this way, right. This is essentially 1 by 1 plus e to the power minus x sigmoid, right. Where, x is nothing but the dot product of these two embeddings. So, higher the dot product higher the similarity, ok.

So, this is the first-order proximity on the embedding space, right. So, this is the; this is the embedding. This is the this proximity is something that we are trying to learn, right. Similarly, we have we can measure the proximity based on the graph structure, which would be the original empirical proximity, right which is given to us. And how do we measure this? So, this is this one.

So, with respect to graph the proximity between two nodes v i, v j is nothing, but first-order proximity, meaning two nodes are connected. The weight between i and j, v i, v j divided by the total weight, total weight of all the nodes, right. This is basically w equals to sum of all ij in E, right, w ij.

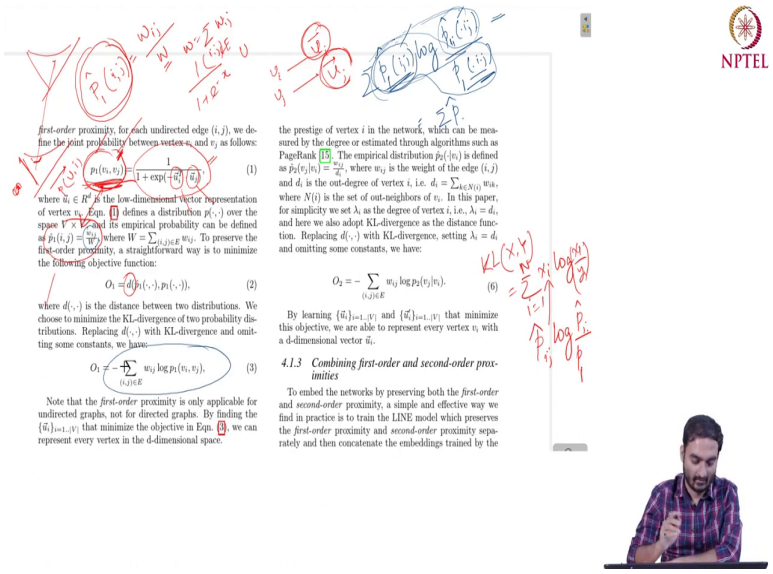
So, this is the empirical proximity which is known to us. Because the graph is known to us. Proximity, this proximity is not known to us. Why? Because these embeddings are not known to us, ok. So, what would be our target? Our target would be to come up with embeddings of i and j, such that these two proximities are closer, ok.

So, how do we measure the closeness? So, now, think about it. So, for every node pair you have the you have a value, for every ij pair you have value. So, we can get a distribution of this learned proximity, right. So, this would be p of v i, v j. And this is some sort of CDF or whatever PDF, right. Similarly, for empirical proximity, we have another distribution.

So, this two distribution should be close to each other. And how do we do that? How do we measure the closeness between two distributions? We can use KL divergence for example, right. So, we use the KL divergence, right between empirical proximity distribution and learn proximity distribution, right. And if you are aware of this, so KL diversion between two discrete as I said discrete distribution, right say X and Y, X and Y, log x i, x i, log x i y i, right.

Let me write it a fresh. So, KL divergence between this is discrete distribution i equals to 1 to N, x i log of x i by y i. In our case, x i is the empirical proximity, right p ij, right a log of p ij hat by p 1, v i v j, p 1 ij, right.

(Refer Slide Time: 11:47)

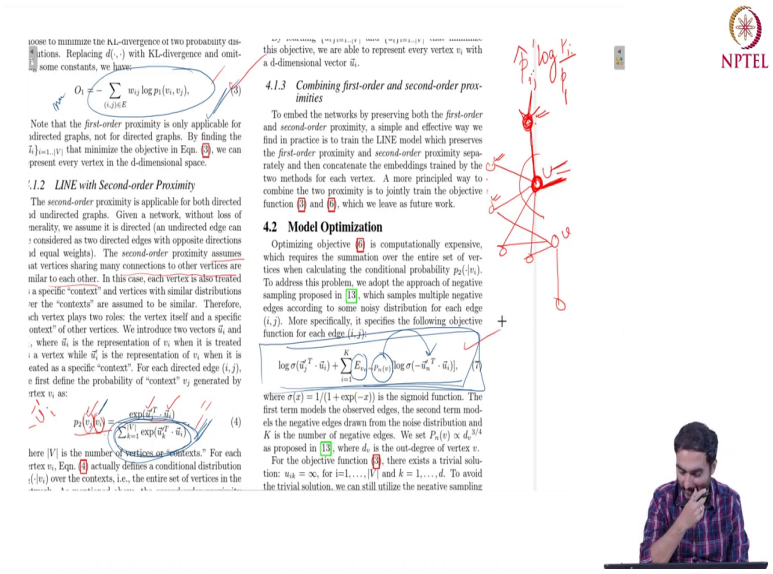


If you do the math, what you get here? If you do the math you will see that this is nothing, but summation of; right. So, what you have? p i p 1 i, j hat, right log of p 1 i, j hat by p 1 i, j, right.

The denominator is the learned one and the numerator is the empirical one. So, this is essentially log, so log x by y is log x minus log y. So, this should be this minus this, ok. Look at the first term. The first term all these numbers are constant, right because this part is nothing, but v ij by w which is constant. This part is also constant. So, there is no point in minimizing this. You can just ignore this part.

So, only second part will be there which is minus of this log this, right, ok. So, this is the; this is the optimization, this is the objective function which we want to minimize, right with respect to the first-order proximity. Now, let us look at the second-order proximity, right. So, here also the idea is same. Second-order proximity, as I mentioned, it assumes that vertices sharing many connections to other vertices are similar to each other, right. So, what they do here? Think about it.

(Refer Slide Time: 13:47)



So, when I say that this is u and this is v, these are common neighbors and these are uncommon neighbors. So, when I look at u, this node, this node, this node, these nodes are context, these node has the context of node u, right. So, this is the central node and these are the context nodes.

When I look at this node, right this is the central node and this is the context node, right. So, here also, every node plays dual role. You know one time it acts as a context, another time it acts as a central node, right. So, the second-order proximity is this one.

So, given v i, v i is the central node. Given v i, what is the probability of encountering v j as a center as a context node? Given V, what is the probability of obtaining this as a context node? Ok. And how do we capture this? They basically use softmax kind of function; where.

So, as I mentioned already, so for every vertex v i, we have here we have two embeddings. So, u i is the embedding when you use vertex v i as a central node and u i dash is an embedding which you when you use v i as a context node. So, here you see that v i is the central node and v j is the context node. So, what is the similarity?

So, similarity would be that basically dot product, you take the dot product, dot product of v i as a central node and v j as a context node, u j dash, you see here dot product, right. And you pass it through a kind of a softmax, right. Therefore, e to the power, e to the power x by summation of e to the power x dash, x dash, all the remaining part, ok.

So, this is the proximity. This is a second-order proximity, remember based on the embedding. So, these things are not known to us. What is known to us is the empirical proximity. So, this is the empirical proximity.

Empirical proximity that v i is the central node and v j is the context node is simply the weight between i and j by, so remember this is conditional probability. So, it should not be normalized by the weight, right by the total weight. It should be normalized by the degree of v i, right. You are basically saying that I am fix, let us fix this one, let us let us look at all the weighted degree and what is the probability that this weight is chosen, right. So, w ij by d i.

Now, d i is the weighted degree remember this, ok. So, here also the same target, the target is to minimize the distance between empirical distribution and the learned distribution using KL divergence, ok. You see here.

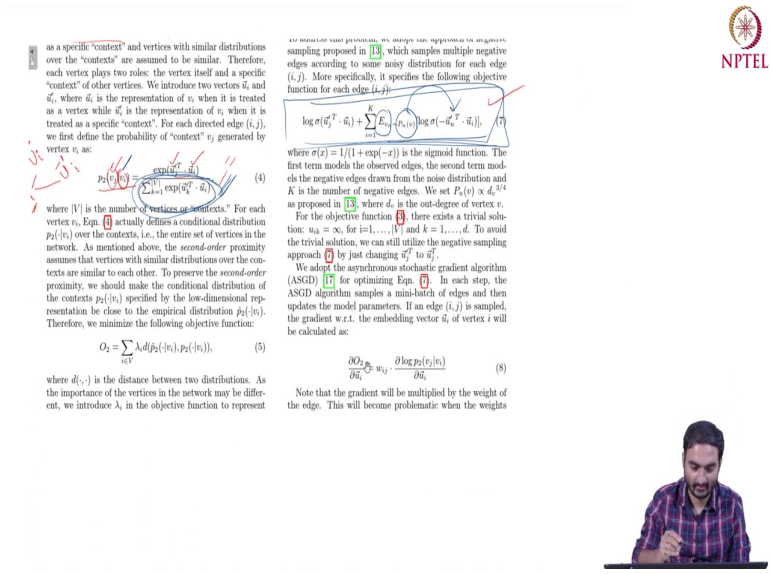
So, these are the two objective functions that we are now optimizing, right. So, the remaining part is very straightforward. So, what they are saying is that first-order proximity this one is easy to optimize, right for a large graph.

But for a large graph this is difficult to optimize. Why? Because the denominator as you see here this ranges over all the vertices. For every central node, you need to do this calculate this denominator, right across all the vertices, and this is quadratic. It will take a lot of time.

What is the remedy? The remedy is negative sampling the one that we discussed in the last lecture. So, what we do in the negative sampling is that we basically sample nodes from the distribution of vertices, right distribution of nodes. And this sample nodes would act as negative samples, would act as non-context samples, and then we feed it to this one.

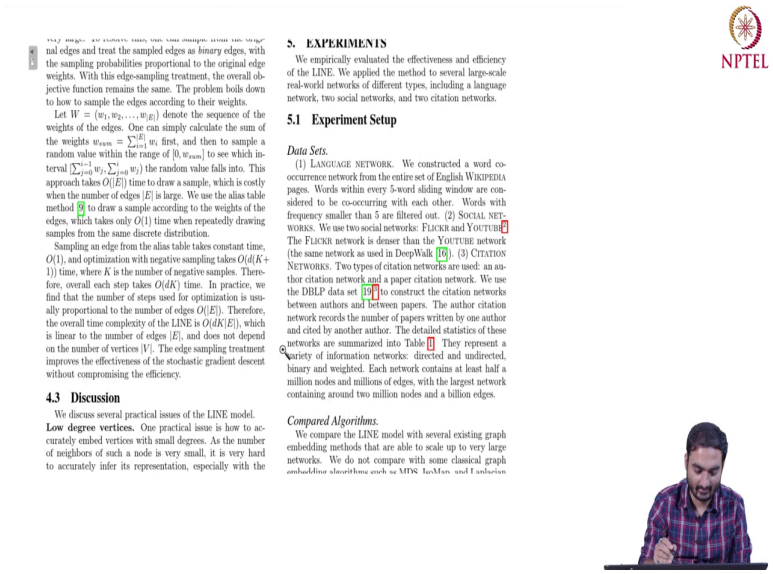
And this is very straightforward, we have seen, I think this equation multiple times if you are aware of say what to wake you know glove, these kind of methods in NLP and this is very straight forward, ok. This is well-known, right. And then the rest of the part is same.

(Refer Slide Time: 19:10)

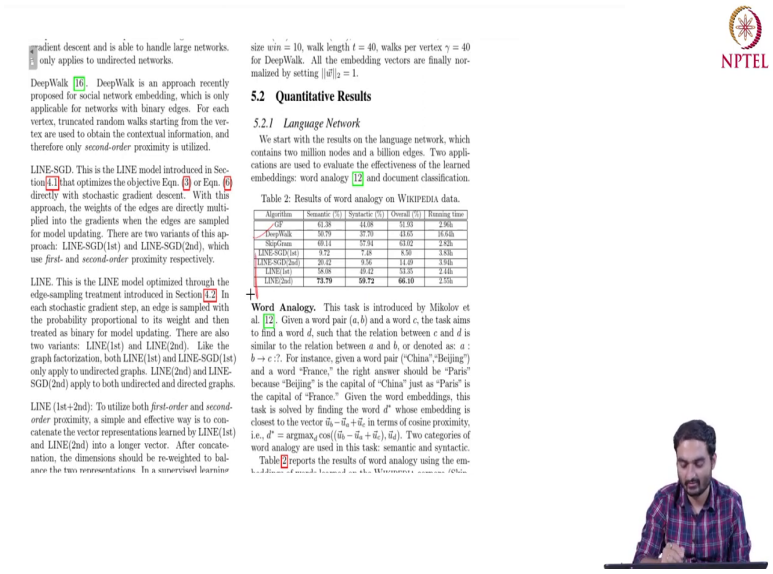


So, now, we have two objective functions O 1. So, this is O 1, this is O 1 and this is O 2 now. And then what we do? We take the gradient descent, right as usual with respect to u 1 and u 1, u i and u i dash, right. You update the, you update the this one you update the objective function, right.

(Refer Slide Time: 19:27)



(Refer Slide Time: 19:44)



And then, you basically keep on updating the embedding vectors and then you stop when you get the convergence, right, when you obtain the convergence. And they showed that I mean with respect to methods like, with respect to methods like deep walk. Deep walk is something that we will discuss in the next lecture, right a LINE with SGD, first-order SGD, second-order SGD perform significantly better, right across different networks.

So, this is another method, method which basically looks at higher order proximity, right. And all these methods hop, LINE, (Refer Time: 20:15) these are non-neural network based methods, right.

In the next lecture, we will start with random work based approaches, and then we will move to the neural network based approaches for graph embedding.

Thank you.