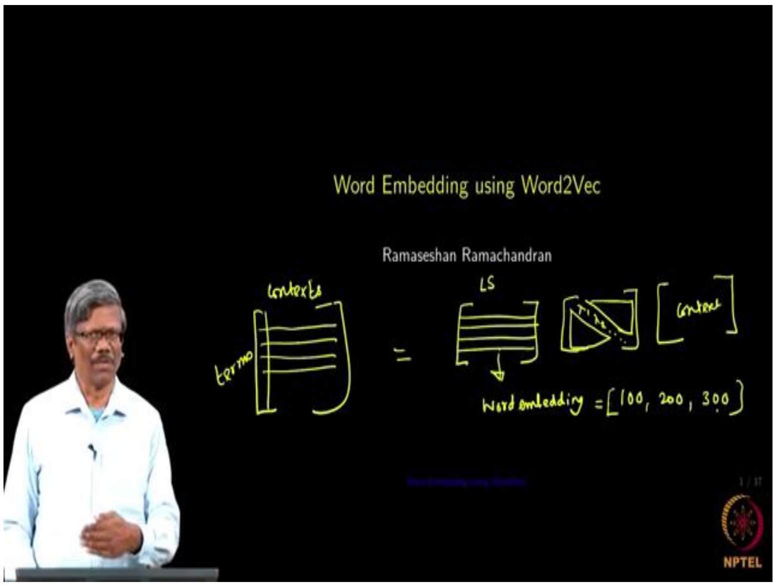
Applied Natural Language Processing

Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute

Lecture – 39
Why Word2Vec?

(Refer Slide Time: 00:15)



Word Embedding using Word2Vec

Ramaseshan Ramachandran

So, this class will be looking at the word embedding using Word2Vec of I will be taking you through the concepts applied in Word2Vec this was developed by Google in early 2013 or 14. And I will be taking you through the code as well as through the techniques and also through the mathematical equations on how word embeddings are created using neural network using two different concepts. Before we get into that I am sure you are familiar with the word embedding that we discussed earlier in the latent semantic indexing or latent semantic analysis, where we actually decompose a matrix into three components I hope you remember that right.

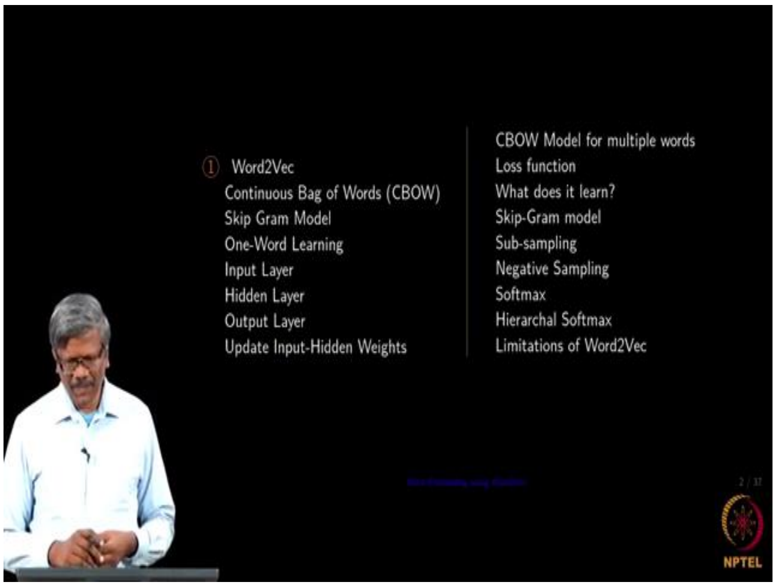
When you decompose the matrix that consists of term document, it gives you three different mattresses. One is the left singular matrix other one is the right singular matrix and then the middle one we call it as the singular matrix right. And then if you look at the dimensionality of the matrix which is driven by the singular values in the diagonal elements of the singular value metrics, we would be able to reduce the size of the total mattresses by curtailing the size of the singular values, I hope you remember that as well right.

So, in the same fashion we are going to be looking at the neural network to create a similar concept where we can create word embeddings. In the case of LSI the left singular matrix if you are using term context matrix as the input and then use the SED to decompose that into left singular right singular and singular matrix. The left singular would contain the word embeddings and then the right singular matrix would contain the context embedding and so, on right.

So, if you really see how that happened to understand whatever I mentioned I will draw this in the matrix form this is your term and this is the context right. When you decompose this vec three different mattresses like this and then here this is your singular matrix where you will have lambda 1 lambda 2 and so, on, this is your left singular matrix which is giving you the word embedding; this one is your, and this is the context right context matrix.

So, the metric sizes are defined by the number of elements in the left in this singular values. So, I can reduce this to a smaller number. So, that my size of the left singular value reduces and the context also reduces to match the size of the singular matrix ok. So, in that way I can just have the word embedding to be off sized. Let us say 100 elements or 200 elements or 300 elements and depending on what is the kind of accuracy you want to have you can increase the size of the elements in the word embedding. So, we are going to be doing a very similar approach using the neural networks, and this course this session is all about this.

(Refer Slide Time: 04:14)

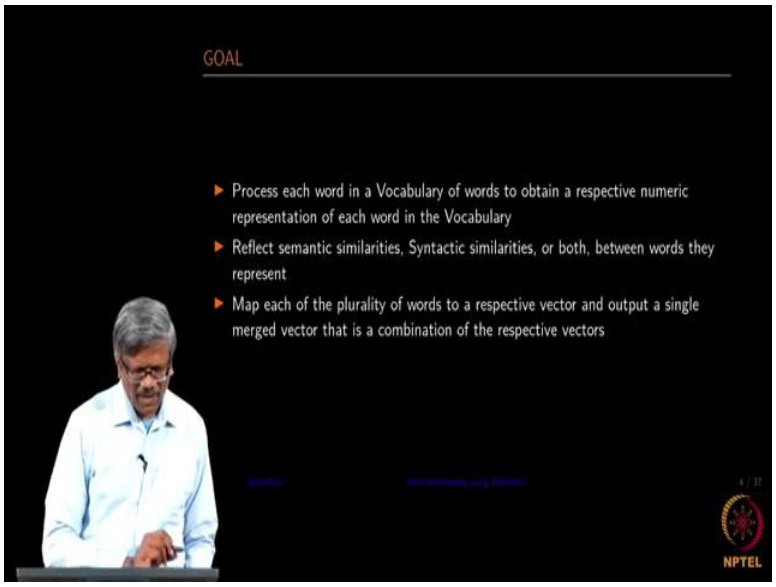


1. Word2Vec
   - Continuous Bag of Words (CBOW)
   - Skip Gram Model
   - One-Word Learning
   - Input Layer
   - Hidden Layer
   - Output Layer
   - Update Input-Hidden Weights
   - CBOW Model for multiple words
   - Loss function
   - What does it learn?
   - Skip-Gram model
   - Sub-sampling
   - Negative Sampling
   - Softmax
   - Hierarchal Softmax
   - Limitations of Word2Vec

So, in this session, I will be talking about what is a continuous bag of words and then what is this skip-gram model, and then what is one-word learning, how what is the input layer that we are going to be using and then what is the hidden layer, what is this size, how is it connected to the input layer and then there is an output layer and then how the output layer is connected to the hidden layer. And later we will also see how those input hidden weights and the output hidden and the hidden output weights are modified based on the last function.

And then we will see how and what is learned during the process of the training, later we will see why this model is extremely difficult to implement with respect to this size and then how we can reduce the computation complexity is using subsampling negative sampling and then hierarchical softmax and so, on and finally, will talk about the limitations of the Word2Vec right.

(Refer Slide Time: 05:27)



GOAL

- Process each word in a Vocabulary of words to obtain a respective numeric representation of each word in the Vocabulary
- Reflect semantic similarities, Syntactic similarities, or both, between words they represent
- Map each of the plurality of words to a respective vector and output a single merged vector that is a combination of the respective vectors

So, let us jump into the Word2Vec model. See the goal here is again very similar to what we saw in the LSI, we want to process each word in the vocabulary of words to obtain respect to numerical representation right. So, instead of just having it as a one-hot vector, I want to represent the word in terms of the vector which contains about 100 elements or 200 elements 300 elements and so, on and let us see how we can get to that level.

And then we want to be able to reflect the semantic similarities when we trade when we finished training the network, we want to be able to capture this syntactic similarity when we finished the training of the network or can we capture both in one shot. We will try to map the cruelty of each word to the respective vectors and then try to see how we can merge the vectors so, that its a combination of the respective vector. That is we are trying to combine multiple words that are similar and then push them as one vector ok. So, that is what we mean by the plurality here.