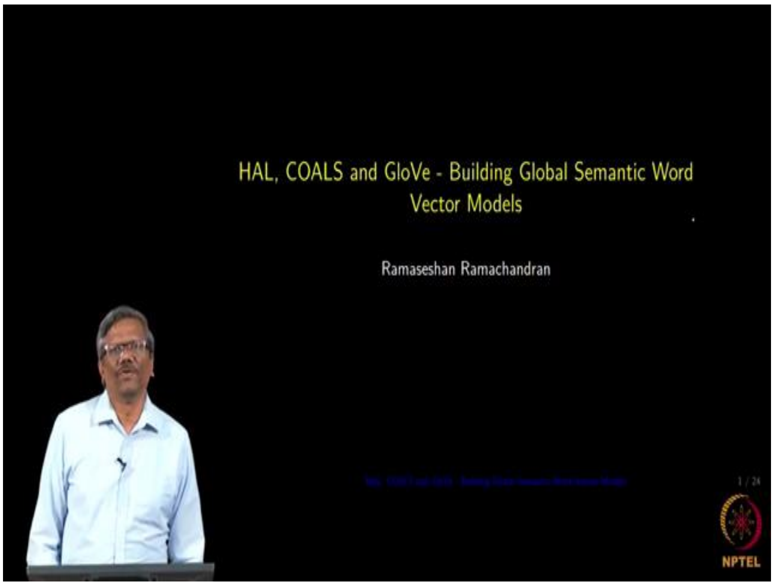
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture - 89
Hyperspace Analogue to Language

(Refer Slide Time: 00:15)



By now you know very well that word vectors are the most important and the fundamental one to natural language processing especially in the machine learning applications right. An especially downstream application that used natural language processing to recover some information or do some kind of a translation and all would require word vectors correct. If we have a very good set of word vectors for the vocabulary that we have, the downstream application is going to be doing very well. The application is going to be outputting some good and useful output for us correctly.

So, in order to have a good output we need to start from the good fundamental at the bottom of the pyramid, which I am calling it as the word vector space right. So, that is what we input into the application either into the neural network or into any other standard non-neural based applications. We also know that natural language applications have not reached the stage of maturity. We are still improving, we are not at a stage where we can say that NLP has reached the stage of 100 percent with respect to the performance, with respect to the usability and. so on. If you know well that in the translation we are still at 32 percent right.

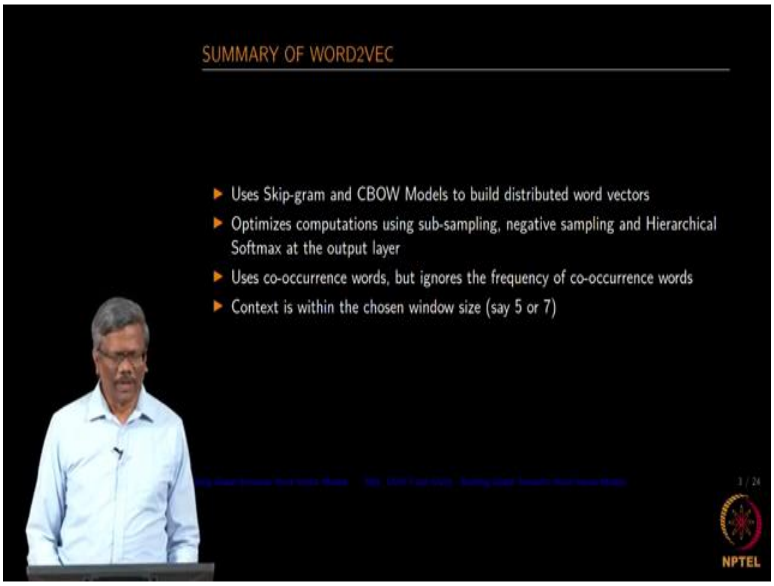
So, for all this we require a good set of word vectors, I keep emphasizing that right from the beginning. So, we have been using various mechanisms to capture the word vectors, the first we know you remember we tried with the SVD where we try to capture the term documents. And, then using L S A we tried to decomposed into three different matrices, and then we try to capture the word embedding from the L S A application right. And, then later we moved on to the learning module using neural networks where we try to input one heart vector as the input for the neural net layer.

And, then later captured the word embedding for that word using the context, that was available either through the CBOW model or through this Skip-gram model and so on. So, we have been using those word vectors in all the application so, far; the reason why I am again bringing this up is in order to improve the efficiency of the applications, downstream applications we require a good set of word vectors and the research is still going on in this subject. There are two ways of doing this, one is using a global model another one is using the local model.

The local model is something that we had seen especially with the C BOW and Skip-gram model, where we try to capture 7 words or 5 words at a time and then try to predict the central word or predict the context word and so on. Right, I am sure you remember all this, and there is another one which is called the global model. It tries to find the co-occurrences of the words and then uses the counts of the co-occurrences to find out the semantics of these words or the meaning of the word or try to find out the similarity of words and so on.

So, now, what we are going to be doing is we are going to be looking at again some of the methods which are called as global methods, and then try to see how word vectors can be built. And, find out whether those word vectors are better than the C BOW model or Skip-gram model.

(Refer Slide Time: 04:09)

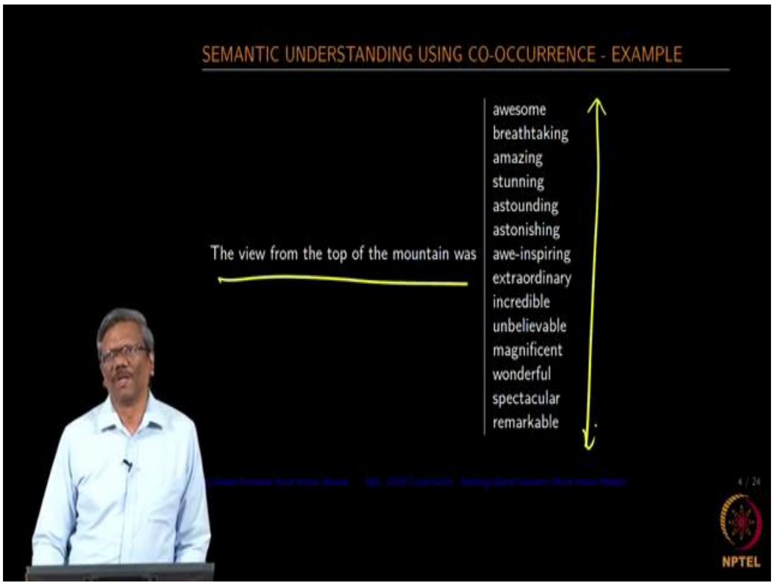


So, I think here you know well right the word vector that we have seen earlier in week 5 or 6, where we try to use the context words as the input especially for the C BOW model and then try to predict what was the central word right. Or, in the Skip-gram model we try to input the central word and then try to predict the context words at the output.

So, we used these models rather than using subsampling negative sampling and hierarchical softmax to improve the efficiency of identifying the word vectors. I am sure you also know this part, and then we know that it uses the co-occurrences, but it ignores the frequency of co-occurrences of words so, which is a very important right. So, we want to really identify how many times that particular co-occurrence had occurred in that corpus that gives the importance of the context words and the central word that we have been talking about in this CBOW model and then Skip-gram model.

So, those two models did not utilize the frequency of co-occurrences of the word, and then we utilize the small window size which is 5 or 7. So, that is what we call the local model. So, it is localized right. So, it does not know anything beyond that 5 words or 7 words. So, every time it sees as a similar set up of or rather the same set of 7 words it still retrains that. So, maybe we can find mechanisms to eliminate the duplicates, but there is no concept of knowing whether the particular set has been trained or not in the C BOW model it continuously takes those 7 words and then keeps training the model ok.

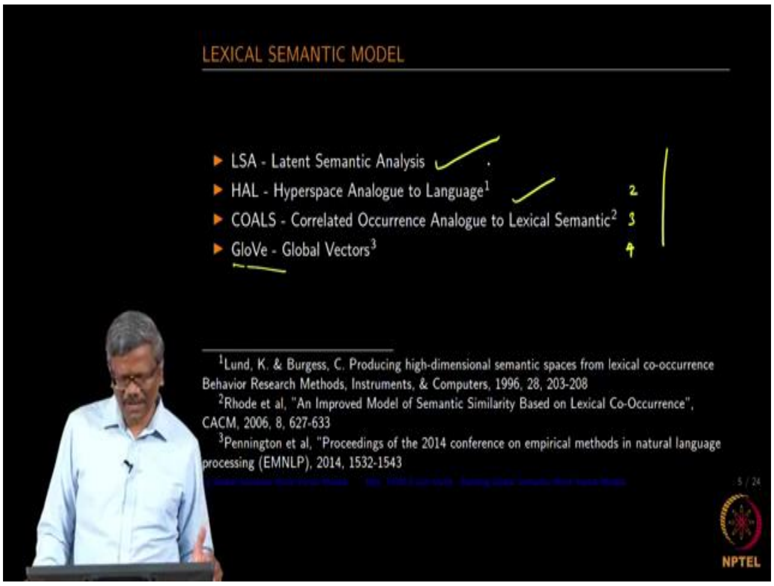
(Refer Slide Time: 06:06)



So, now what we want to do is to find out if this contextual information that we find in the corpus could be used using the count of the co-occurrences of words you know this very well, I think I use this slide at the beginning of the lecture series where the context was given as this right. And, then we have several similar words that could be the last word for this sentence.

So, I also mention that if we have this same context, but there are different kinds of words that we are using as the last word, we know that all these three words all these words are similar right. So, this is how we identified the semantics of the word trying to understand the meaning of the words and so on using the context that is available.

(Refer Slide Time: 07:02)

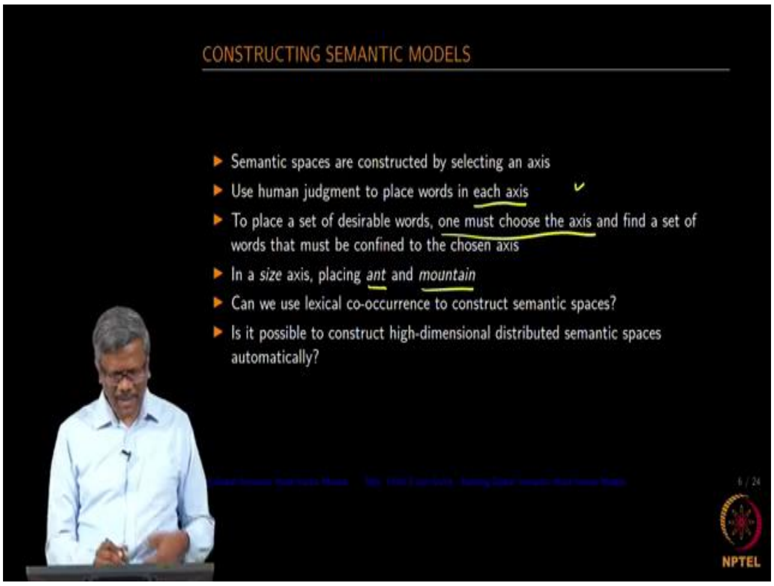


So, there are various approaches that are followed as I mentioned earlier LSA is one I am sure you know about this, I am not going to discussing this in detail. The second one is HAL it's Hyperspace Analogue to Language, and the third one is COALS its Correlated Occurrence Analogue to Lexical Semantic and then the last one which we will talk about is GloVe ok.

And, I am sure you would have heard about this several time during the lecture I was using some of the GloVe vectors for demonstrating the sentiment analysis where I used 50-word vectors and 100-word vectors; I am not sure which one I used there, but I utilized the glove vectors for identifying the sentiment right. So, we will talk about that as well and these are the model that utilized the local context as well as the count I especially these second, third and the fourth one ok.

So, the first one utilizes the term document rest of them are going to be using in the term-term document we will talk about how there was term-term matrix that can be formed and then how we can utilize the co-occurrence counts through that ok. So, we are going to be talking about this, in brief, all these 3 in the next few slides ok.

(Refer Slide Time: 08:33)



So, first before even starting that you know we need to figure out whether it is possible to that in a manual fashion right. So, only if you are able to do it in the manual fashion then it is possible for you to find out what type of rules that I have applied what should I do to automate this what can be automated, what cannot be automated all those things would be known clearly only when we try to do this in the manual fashion ok.

So, now, assume that you have been given the entire dictionary of words and then there are so, many different buckets let us say about 20000 buckets and you have been given about 100000 words. What you need to do it or what you are asked to do is take one word from the dictionary and then place it in one bucket ok. And, then take another one from the dictionary and then find out if it is similar to what we have just placed put this word in this same bucket.

So, in that way you would start filling all the buckets one after the other, and then at the end you will have various buckets that will contain similar words in the right. So, that is one way of classifying the words and then trying to semantically connect them using the similarity right. So, that is one way of doing it for the word vector. So, we can call that particular set of words as a word vector we still do not know how to really compute the numbers for each of those words, let us assume that we have some idea with respect to that and if you put all the words in the vector form that will constitute one-word vector.

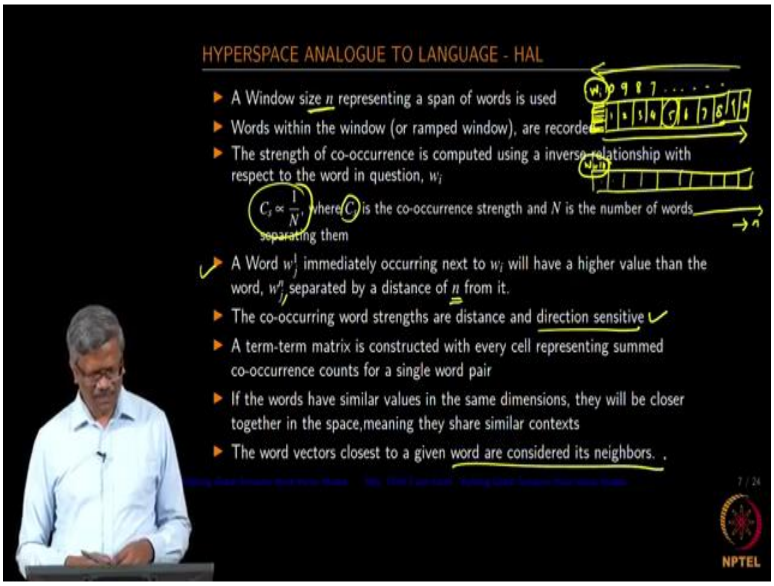
Supposing if we have about 10,000 buckets then we have about 10,000-word vectors for us, that is one way ok. For that we require human judgments right so, we can place those words in that fashion. Or another approach is supposing if you are given let us say the entire Wikipedia dump and then based on the word you have going to start in to fill all the buckets instead of the dictionary now replace that with the Wikipedia dump.

Every word you start again putting them in one of the buckets and then start adding similar words in each of those buckets ok. So, that way there is a human touch to figure out how to put those similar words in the same bucket ok. So, here what we are saying is instead of the bucket we are calling it an axis. So, in each axis you are going to be putting the desirable words. So, you must first choose the axis and find a set of words that must be confined to that chosen axis ok. So, here the bucket is our axis. So, for example, you want to have an axis that going to be describing the size right.

So, there is a bucket that is called size. So, what are all things that you will input into that bucket? So, for example, the ant is very small so, that will also go in there, the mountain also would go in there because it is big. So, with respect to that we are going to be moving those words into the respective axis. So, can we use lexical like co-occurrences to construct that semantic space? So, this space that contains those 10000 buckets or the 10000 axes we would call that as our semantic space.

So, is it possible to construct high dimensional distributed semantic space using this model? So, if you are able to do this in the manual fashion let us say using the Wikipedia dump, we should be able to do it with to extend using the automated fashion like that. So, for us what is going to be the helping hand is our context. So, based on the context we are going to be throwing one word after the other into the respective bucket. So, that similar words occupy all those ok.

(Refer Slide Time: 13:09)



So, let us see how that is achieved in HAL first ok. In the Hyperspace Analogue to Language or HAL what we are going to do is we are going to be taking a content there is a huge corpus and then start looking at 10 words at a time or n words at a time ok. So, we will define more than n means a little late [vocalized-noise. So, we are also going to be defining the window we know how we can create the words within a window right.

So, taking a window of size 3 for a trigram or a 5 window size for a 5 gram and so on right; so, we can slide that 3 gram or n-gram across the corpus and then whatever comes within that window we take it and then start putting it into the model right. So, that we can later figure out how many times certain words occurred together and then we can make a count of that. Where

$$C_s \ \alpha \ \frac{1}{N} \ w_i$$

So, in this case again we are going to be doing the same operation, but we are going to be having windows of size n we are going to be having 10 n neighborhood words right. So, if you create a window of this type, and then assuming that for this word I am going to be finding the neighborhood words let say about 10 of them ok. So, the word 1 is pretty close to the word here right what to us little far away and then 10 is far away from this.

So, can we say that words which are closer to w I should get a higher weightage than the one which are further away from here? So, this is the idea that is being followed in the

HAL model. So, that is what we call as the ramped window where the influence of the word that is further away from this word will be minimal when compared to the one which is closer to this.

So, how do we define that say very simple like the idea would be to have an inverse relationship of this type let us say that a co-occurrence strength is defined in terms of the inverse relationship ok? So, in this case what will happen is the word which is closer to the w_i will have let us say in this case a value of 10 and then 2 will have the value of 9 8 7 and so, on. So, in this way we are also saying that those who are closer to the word would be considered as more similar word in that context than the one which is further away.

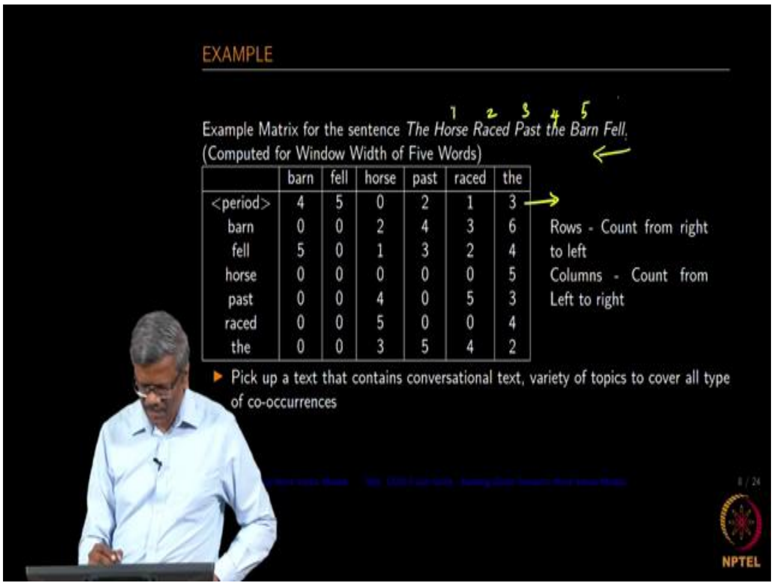
So that is what we say here in this particular bullet point. So, the word w_{j1} immediately occurring next to w_i will have a higher value than the word w_{jn} separated by a distance of n from it ok. The co-occurring word strengths are distance and direction sensitive. So, in this case what these authors have also done is instead of just taking the or finding the co-occurrence in this direction, for this word also finds the co-occurrence in the opposite direction. So, every word you will see there is a forward and a backward co-occurrence count ok.

So, that is what we call it as it is not only dependent on the distance it is also direction sensitive. The term-term matrix is constructed with every cell representing the summed co-occurrence count for a single word pair. So, when we do that you start counting how many times certain words occur in that context if this occurred in the forward as well as in the backward direction then you start summing that up ok. So, we will see through an example how that could be done if the words have similar values in the same dimensions they will be closer together in space meaning that they share similar contexts you got it right.

So, if you are if the words have similar values in the same direction ok. Supposing we have found some value for the word I and then we have done the same thing for another let us say a word i + 10 right and then if the values that are found for this w_1 + 10 and w_i if they are similar then we can say these two words are similar; that means, we are just looking at the counts and then say how similar of this counts for these two words.

So, we try to use a normalized distance measure to find the similarity in this case the word vectors closest to a given word are considered as its neighbor. This i think you know very well so, I do not have to explain this or else.

(Refer Slide Time: 19:00)

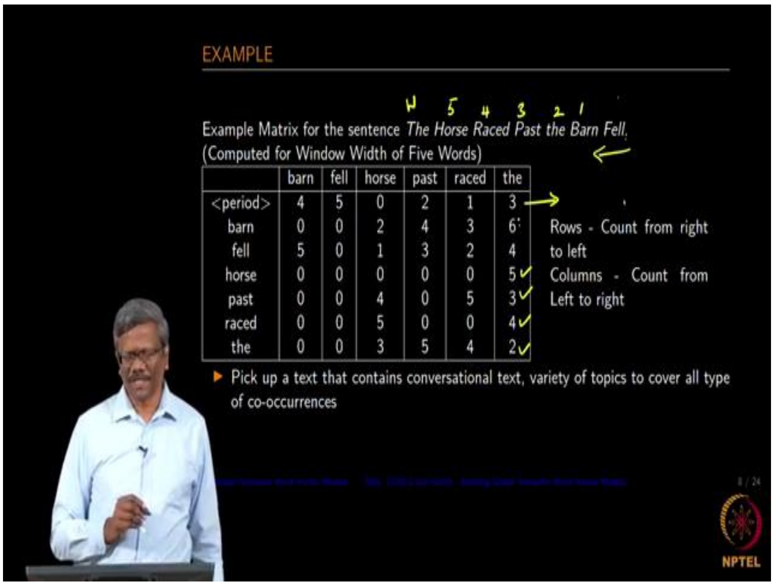


Let us take one small example and then see how this could be computed. So, the idea is to compute the count of the co-occurrences in both directions ok. So, we are going to be taking this sentence along with the period at the n and then start counting how many times or start counting the strength of the co-occurrences for each of the words. So, let us start with the word periods let us start from this direction ok. So, for this one, the window is going to be 5 ok. So, we have going to have five because which is closer to the period barn is little away 1.

So, if you look at the period and then start looking at the barn, the barn has the value of 4 right, and then if you look at the fell it is pretty close to this it is next word so, the value is 5. And, then the horse is far away because it is not within that window of 5 so, it gets the value 0 then past is in the past will get the strength 2 and racing will get the strength 1 right. And, then the will get three because it is the third word from the period ok. So, this is why if you look at this one, it is coming from the backward direction the rows are filled from for the words that are coming in the reverse direction. So, if you take a fell. So, if this is 5 4 3 2 and 1 you got it.

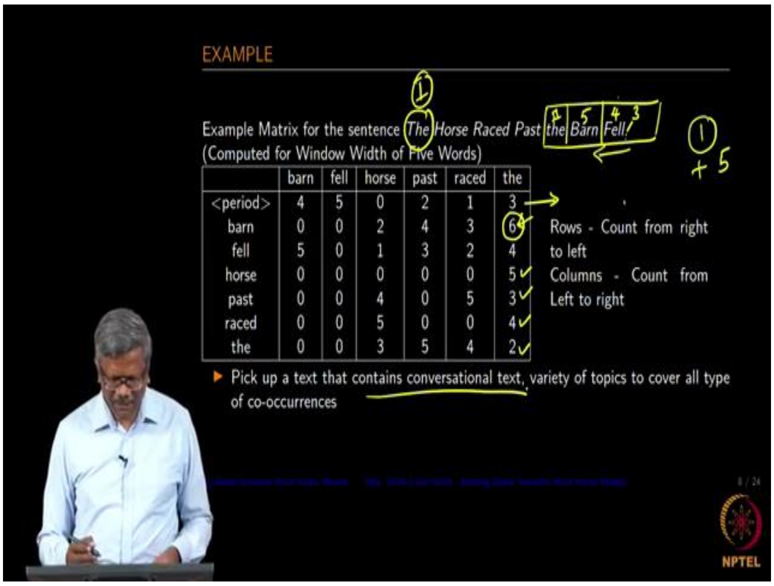
So, in this way you can fill this table from the left from the right to left for the rows, and then for the columns if you want to fill the columns what you do is you start filling it from left to right. So, what you do so, in this case now we start from for this word one ok.

(Refer Slide Time: 21:27)



So, let us look at this for the column. So, let us for the word we have to start looking at the horse is at 5 so, 5 here. And then raced is 4 past is 3 the is 2 and then barn is 6, how is this 6 coming the 6 is coming first whenever you do from the left to right we get early of one here right.

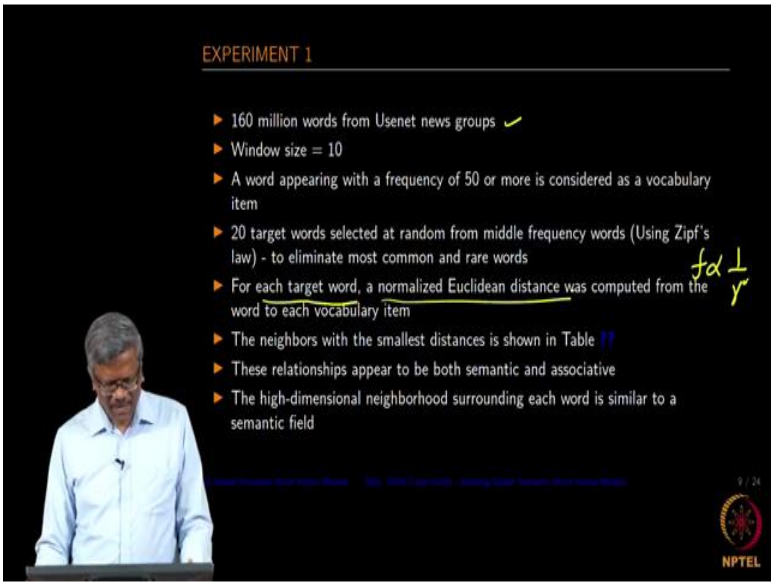
(Refer Slide Time: 22:22)



And then for the word, if you consider this one and then we have 5 4 and then for the period we have. So, add the count of 5 to that. So, it gets the value of six OKs. So, when you do from the left to right there is a value for it and then when you do from right to left you get some co-occurrence values. And, then you do from left to right you get the co-occurrence values and if you find a more than ones for example, the barn had occurred twice for the word the this is the first time and then this is the second time right. So, we are able to get a value of 6 in that fashion. So, in this way you fill the whole table and the table is not going to be asymmetric table

Right because the counts are different from both directions and then in order for us to do this you know the authors are advising that we need to consider a conversational text. So, only if you have this type of text it is possible for you to have all kinds of mixtures in terms of the context and so on, then it is possible to have a good set of word vectors they prove it by looking at various corpus and then proving that the one that they have gotten from the news net is used net is a lot better than anything else.

(Refer Slide Time: 24:01)



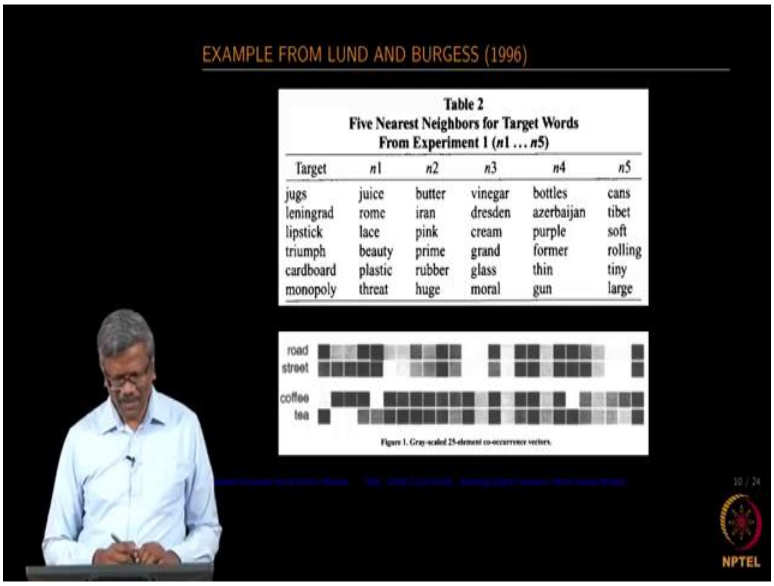
So, they have considered for an experiment 160 million words from Usenet newsgroups and then they have taken the window size of 10. So, earlier I have taken the window size of 5, in this case they have taken the window size of 10 and then the word appearing with the frequency of 52 or more is considered as the vocabulary otherwise he would discard them right.

So, you do not want to have a noisy or very large matrix to deal with. So, you discard some of the words from the mat. And then they have selected twenty target words at random from the frequency of word use in the zip law you know what this right.

I am sure you would remember this right. So, they are using a Zipf's law to eliminate the higher frequency ones and the low-frequency ones and then consider only the middle frequency words. And, then try to find out the word vectors to figure out whether they are really getting a good set of word vectors. So, what they are doing they take a target word and then using a normalized Euclidean distance. I am sure you remember this too they compute the distance between those two words; if they are small, then those two words are similar ok.

So, we will see that in the table in the next slide the relationships appear to be both semantic and associative, high dimensional neighborhood surrounding each word is similar to a semantic field.

(Refer Slide Time: 26:08)



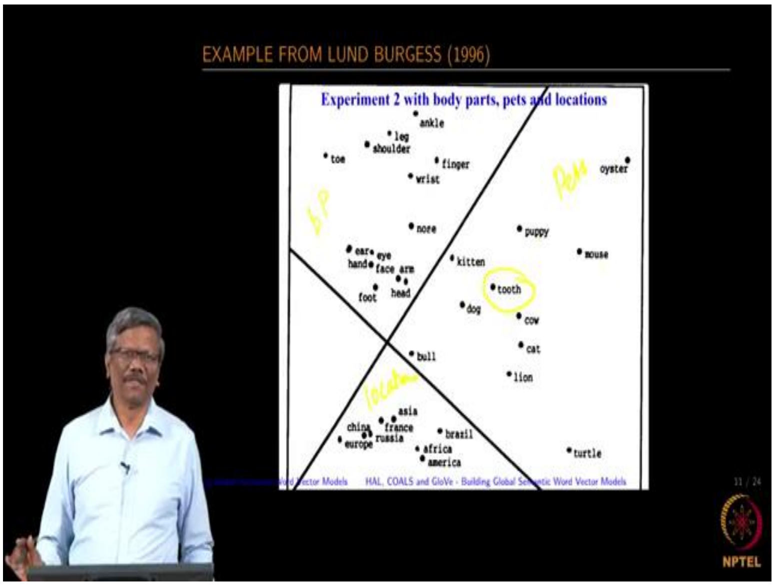
So, I think we spoke about that earlier as well ok. So, we taking an example you know like what we mentioned earlier there using the Zipf's law to pick up only those middle-frequency words to find out whether I was able to get the words that are very similar ok. So, let us look at the table here and then these are the target words and then they try to find out the five nearest neighbors for the target words ok. They found juice as one of the neighbors and then butter vinegar bottles and cans, somewhat ok.

I guess right and then if you look at the names of these series you have Leningrad Rome, there is a country here and then Dresden Azerbaijan and Tibet another country or on here then look at the lipstick the similar words are lace pink cream purple and soft and then let us look at this one cardboard here. Similar words are found to be plastic rubber glass thin and tiny ok.

So, without getting into the complexity of any of these C BOW models or Skip-gram model just using the count of the co-occurrence words in both directions. They are able to show that you know it is possible to build such a word vectors high dimensional word vectors using some of the simplest of the ideas ok. So, another way they want to demonstrate was to use some kind of grey level values for each of the word vectors and then show how similar they are. So, in this case, if you look at the road and street ok.

So, the word vectors that they have obtained for the road have been converted into grey levels and presented as squares here. So, if you see road and street they are somewhat similar I would not say they are 100 percent right, but they are definitely closer and then if you look at the words coffee and tea you can find the similarity again in this fashion ok.

(Refer Slide Time: 28:52)



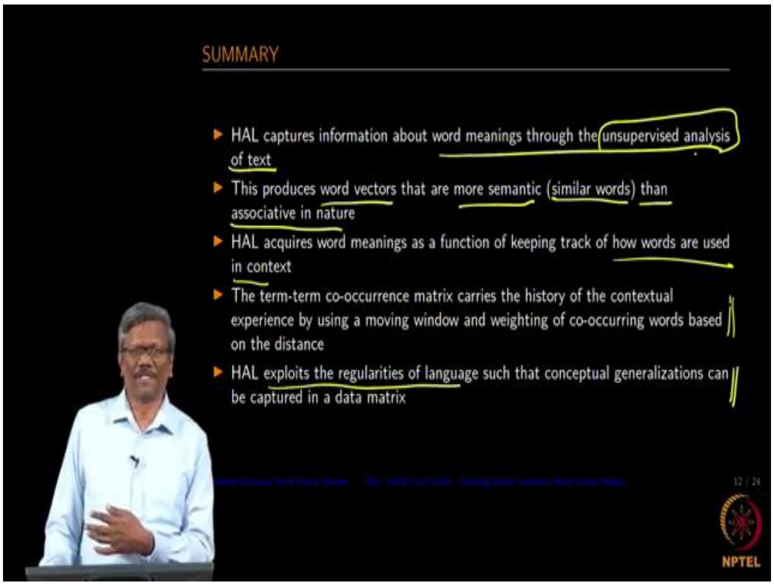
In another experiment what they did was they took somebody parts pets and locations as the keywords and then try to see if those words are within a certain axis or within certain boxes. So, they try to plot them in this fashion, if you look at this one here you have the pets and then you have the body parts around here and then locations around here. So, within that axis that they have defined they are able to bring in bull, turtle, lion, cat, cow, dog, tooth interesting kitten, puppy, mouse and oyster as part of that axis.

And then the body parts again are classified according to the similarities values ok. So, in my view I think this is a good example of how you can really utilize the co-occurrences values and build a word vector in the simplest of the way right. So, there is no complexity involved in this is a very simple automated way where you just start using a sliding window which is also ramped one. And, then start counting the number of co-occurrences of the co-occurrence, co-occurring words and then finally, create a word vector and so on ok.

So, in this case there are two kinds of word vector right. So, one is as you saw earlier. So, there is a word vector around here right. So, these are also word vectors ok. So, what I am going to do is I am not going to tell you which one they have considered. So, I am going to ask you to go on and read this paper and then find out. So, what is the word vector that they are talking about towards the end ok?

So, at the end of the day you need to have a word vector and which one they have taken or did they combine it in some fashion or so on. So, we need to go on then read this paper and then figure out what mechanism they have used and paper is very easy to read it there is no complexity involved in this paper ok. So, I want you to go on and read that thoroughly to find out the answer to this question ok.

(Refer Slide Time: 31:49)



So, if you look at the HAL right, it actually captures information about word meanings through unsupervised analysis of texts. We are not really trying the system to say that this word is closest should be closer to this and. So, on like what we have done in this C BOW model right. So, we are not reinforcing that it is just a very simple unsupervised analysis of contour the word vectors are really meaning full ok. So, they are showing similar words in some fashion or not ok. so, based the earlier right.

So, they initially thought that the vectors are going to be word vectors are going to be semantics as well as associative in nature and then based on the experiment they found that it is not they are more semantic than associative in nature HAL acquires word meanings as a function of keeping track of how words are used in the context correctly. So, based on the counting mechanism of the co-occurrences of words we are able to find out the meanings of the word. The term-term co-occurrence matrix carries the history of the contextual experience by using a moving window and weighting of the co-occurring words based on the distance. So, this is the mechanism used to obtain the strength of the co-occurrence.

Again like any other neural net model it exploits the regularity of the language so, that we are able to capture the word vectors in the right fashion. you do not require a very complex mechanism to really capture word vectors. So, mechanisms of this type would really do a good job in terms of capturing the word vectors, it is better than the C BOW model or Skip-gram model supervised models are definitely better. Because, we keep reinforcing to the network saying that a learned this because this is related to this.

So, we are making the system learned the relationship whereas, in this case we are not doing any of this it is just an unsupervised analysis. So, in terms of the performance this would be a little lower than the supervised model. So, can we make this better is it possible for you to get into another supervised or rather unsupervised model that can do better than this and pretty close to the Skip-gram model or C BOW model?