

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

**Lecture - 34**  
**Word Embeddings - Part I**

(Refer Slide Time: 00:36)

*Word Vectors*

- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or 'one-hot') encoding every element in the vector is associated with a word in the vocabulary.
- The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

*One-hot representation*

motel [0 0 0 0 0 0 0 0 0 1 0 0 0] AND  
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0] =

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 19

Welcome back. So, we are talking about distributions semantics. So, in this fourth lecture and the fifth lecture of this week, we will talk about another very interesting idea that is word embeddings that come from distribution semantics. So, what are word embeddings? So, when I am talking about word embeddings, so they are like word vectors. So, the word vectors are nothing but simple vectors of vectors of weight. So, we talked about these factors. So, you are having some dimensions and they are representation words in this set of dimensions and their various weights.

So, now we also talked about this one-hot encoding that is among the n-dimensions only one dimension is 1, everything else as 0; and these dimension might correspond to the index of the particular word that I wanted to be set. Now, and we saw that if I if we are using one hot encoding I cannot compute the similarity among words. So, if I motel and hotel, there will have one at different places, and if I compute similarity, if I do an AND, it will be 0. So, this is not conducive to for semantic similarity between words.

(Refer Slide Time: 01:37)

### Word Vectors - One-hot Encoding

- Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child.
- We could encode the word 'Queen' as:

The diagram illustrates the one-hot encoding for the word 'Queen'. It shows a 5-dimensional vector represented as a row of five boxes containing the values 0, 1, 0, 0, and 0. Handwritten labels in blue and green connect these values to the words in the vocabulary: 'King' (0), 'Queen' (1), 'Man' (0), 'Woman' (0), and 'Child' (0). The text '1-of-N encoding' is written in purple to the right of the vector. A small circular inset shows a man speaking.

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 19

So, now let us take a simple example. So, have a vocabulary that contains only five words king, queen, man, woman and child. So, I have vectors in five dimension. So, I can encode queen like this. So, queen has a weight of 1 in the second dimension that correspond to the index of queen and it has weight 0 and other dimensions. And I cannot compute the how similar king and queen are, how similar queen and child are by using this method.

(Refer Slide Time: 02:12)

### Limitations of One-hot encoding

**Word vectors are not comparable**

Using such an encoding, there is no meaningful comparison we can make between word vectors other than equality testing.

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 4 / 19

So, we cannot make any meaningful comparison. We can only find if these two words are same, so that is not very interesting.

(Refer Slide Time: 02:21)

The slide is titled "Word2Vec – A distributed representation". It contains a section titled "Distributional representation – word embedding?" which states: "Any word  $w_i$  in the corpus is given a distributional representation by an embedding". Below this, it shows the mathematical representation  $w_i \in \mathbb{R}^d$  and explains that it is "i.e., a  $d$ -dimensional vector, which is mostly learnt!". An example is provided for the word "linguistics", which is equated to a column vector of 8 values: 0.286, 0.792, -0.177, -0.107, 0.109, -0.542, 0.349, and 0.271. The slide footer includes "Pawan Goyal (IIT Kharagpur)", "Word Embeddings - Part I", "Week 7, Lecture 4", and "5 / 19".

**Word2Vec – A distributed representation**

*Distributional representation – word embedding?*

Any word  $w_i$  in the corpus is given a distributional representation by an embedding

$$w_i \in \mathbb{R}^d$$

i.e., a  $d$ -dimensional vector, which is mostly learnt!

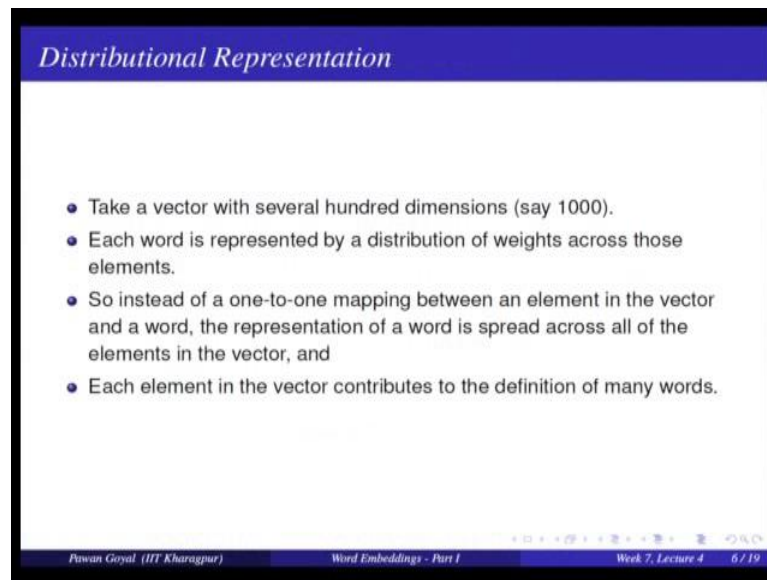
*linguistics* =

0.286
0.792
-0.177
-0.107
0.109
-0.542
0.349
0.271

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 5 / 19

So, what happens in the distributional representation or so we also called it word embeddings that any word  $w_i$  in the corpus is given a distributed representation by an embeddings. So, I have a fix dimension like  $d$ -dimensional vector, and I represent each word in these  $d$ -dimensions. And I will give them various weights and these weights are to be learnt by some method, and we will talk about what will be the method by which I will learn these weights. So, idea is that each word in my vocabulary, I will try to represent them in some fix dimensions  $d$ . So, this is one idea. So, like I have the word linguistics, and I can denote this word in some fix dimension here  $d$  is 8. So, there are 8 dimensions. So, it has different weights in different dimensions. And similarly all the words will be represented similarly they have different weights in these  $d$ -dimensions.

(Refer Slide Time: 03:26)



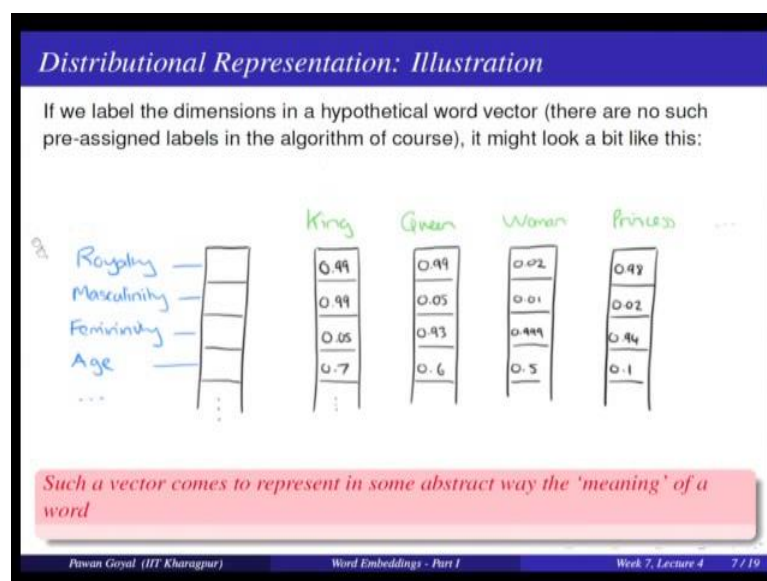
*Distributional Representation*

- Take a vector with several hundred dimensions (say 1000).
- Each word is represented by a distribution of weights across those elements.
- So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and
- Each element in the vector contributes to the definition of many words.

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 6 / 19

So, what is my distributional representation? So, take a vector with several hundred dimensions, so it can be 100, 50, 300, 1000. Now, each word is represented by a distribution of weights across these elements. So, each vector is nothing but a distribution of weights across these dimensions. So, what will happen? So, instead of a one-to-one mapping between an element in the vector and a word, you have a distributed representation of each word; and by using that you can capture similarity. So, whether two words are similar or not. If they have similar weights in many dimensions, then they will be similar and that is how I can capture similarity between words. So, we can say that each element in my vector or each dimension might contribute to the definition of multiple words.

(Refer Slide Time: 04:24)



So, what the dimensions might indicate this is just in illustration this is not a literal meaning, so that is what can you can help you to understand what are these dimensions. So, suppose all the dimensions in my distributional representation I can label by some hypothetical labels. So, my algorithm may not have some labels some very good labels like that it may not be possible to do that even manually, but this is simply for understanding that. Assume that so there are  $d$  dimensions assumed that you can assign a label to these some topic some concept like here my dimensions can be like royalty, masculine, feminine, age and so on, these are my dimensions. And assume that the weights that each for is have in this dimensions correspond to how much that word is closer to that concept.

So, for example, the word king word how much it is closer concept of royalty. So, this king is very close to concept of royalty it has a high weight in this dimension 0.99, it is very close to masculine, so it is has a high weight 0.99, but very small weight in feminine. Age suppose that a high weight means, it is elder, so it is 0.7. Similarly, for queen what will happen royalty will get a high weight yes, but masculine and feminine will be just opposite; and age can be again 3.6. Now, take women and princes women will have very low weight in the in the case of royalty high weight in the case of feminine. And princess will have a high weight in the case of royalty and high weight in the case of feminine and very low weight in the case of age. And this is simply for illustration.

So, idea is that I am trying to represent all my words instead of fixed dimensions. And these dimensions are latent, they met correspond to certain concepts a combination of concepts and so on that we may not know that the algorithm also does not assign. But we would assume that that these dimensions correspond to some concepts, and now each word can be written as a distribution among these concepts. And then I can measure two words based on how much similar they are on various concepts, this is the idea. The main question is how do I capture this representation that how do I represent different words in this fixed dimensions. So, now we can see that so such a vector is representing the meaning in some abstract manor.

(Refer Slide Time: 07:07)

*Word Embeddings*

- $d$  typically in the range 50 to 1000
- Similar words should have similar embeddings

*SVD can also be thought of as an embedding method*

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 3, Lecture 4 8 / 19

So, now what is my dimension size? So, my dimension- $d$  can be if we starting from 50 up to 1000. And the by focus is that the word that are similar in meaning, they should have similar embeddings or similar representation. Now, if you know about SVD singularly de decomposition that is also some sort of embedding method that converts the vectors in some low dimensions. So, I will encourage that you read about singular that valid decomposition, also another name for that is latest symmetry indexing, but I will just give you very briefly what is the idea of SVD.

(Refer Slide Time: 07:50)

$$A = \begin{bmatrix} w_1 & \dots & \dots & \dots & \dots \end{bmatrix} \begin{matrix} 500k \\ \text{high-dim} \end{matrix}$$
$$A = U \Sigma V^T$$

$\Sigma$  - diagonal matrix  
Singular values

$A_k \approx U_k \Sigma_k V_k^T$

$k$ -rank app.

So, we talked about this co occurrence matrices. So, this is my matrix A. So, these are the target words, these are context words. And this matrix has certain entries. Now, what is one problem is matrix this might be very high-dimensional. So, each word might have a say 500 k-dimensional representation. So, what this is co occurrence with different words and these words can be age number of words can be age 500 k and even more in some cases. So, now I want to give a low dimension representation a distribution representation. So, what will the idea? So, I will use the theorem that any matrix say can be written as U sigma V transport, and there are certain properties of these U sigma and V, but this sigma is a diagonal matrix, and the entities are singular values. So, you can get about it what is this matrix particular decomposition for singular values.

Now, once you have this matrix in this format, this is same matrix. So, idea is that you take a k rank approximation so that means, you have singular values they are arranged in the decreasing order and you take only the top case singular values, and you do that for all this U, sigma, and V. So, we have taking only top k entries. So, only the first k entries of U corresponding to top k singular values that is U k, sigma k, V k transports and this is my k rank approximation of my matrix A. So, now this U will denote my low dimensional representation. So, earlier U might be in the same dimensions, U can be again in 500 k dimension 5,00,000, but suppose you are trying to pick only 100 you will take only first hundred dimensions from here, and this will be a low dimensional

representation for U, V and so on. So, SVD is also one sort of embedding. So, each word you can embed some k-dimensions by taking the top case singular values only.

But we are not talking about SVD here; we will talk about the word vector method for computing this representation. So, now before I discuss what are the different tasks you can do with these word vectors, oh sorry how do you compute these word vectors, let us see why they are found to be very interesting in this domain and what are the different tasks they have been used in.

(Refer Slide Time: 10:53)

**Reasoning with Word Vectors**

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

**Case of Singular-Plural Relations**

If we denote the vector for word  $i$  as  $x_i$ , and focus on the singular/plural relation, we observe that

$$x_{\text{apple}} - x_{\text{apples}} \approx x_{\text{car}} - x_{\text{cars}} \approx x_{\text{family}} - x_{\text{families}} \approx x_{\text{bag}} - x_{\text{bags}}$$

and so on.

Prasen Ghosal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 9 / 19

So, what we found is that these representations are capturing some meaningful syntactic and semantic regularities in a very, very simple manner. So, what is that? So that is we can use the vector offsets to talk about relation between words and how much two words are similar compared to the other pair of words. So, for example, I want to capture the singular plural relationship between words like car and cars, boy and boys, bag and bags and so on. And suppose I do not know what are singular and plural words. So, can I capture that using word representation?

So, what we found is that for each word  $i$ , you have vector  $x_i$ . So, we can take the vector offsets and they will be coming out to be similar for singular to plural pair, so that is if I try to compute  $x_{\text{apple}} - x_{\text{apples}}$  that is coming out to be very close to  $x_{\text{car}} - x_{\text{cars}}$ , similarly to  $x_{\text{family}} - x_{\text{families}}$  and so on. That is I compute the vectors of these and if I take the vector offset apple minus



apples has similar offsets as car minus cars as family minus families, and this is very, very interesting this is not something for which this vectors for trained for, but they are capturing this regularity very nicely.

(Refer Slide Time: 12:23)

*Reasoning with Word Vectors*

Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

*Good at answering analogy questions*

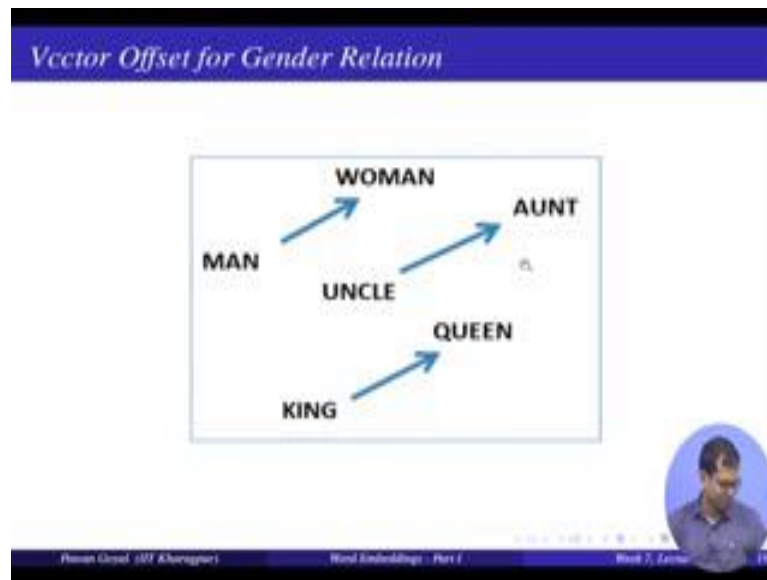
a is to b, as c is to ?  
man is to woman as uncle is to ? (aunt)

*A simple vector offset method based on cosine distance shows the relation.*

Thomas Gold (MIT & Carnegie) Word Embeddings - Part I Week 7, Lecture 4 18 / 18

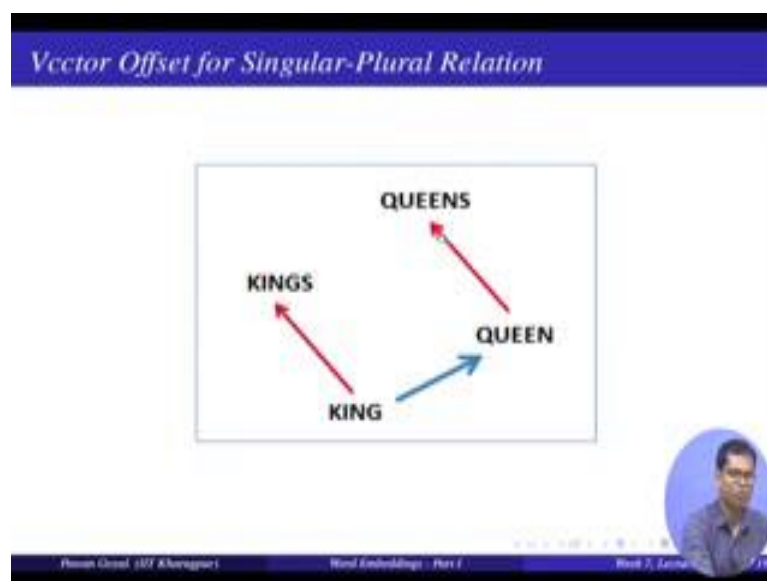
So, this would be like and because they capturing such regularities they can be used for various reasoning task like analogy task a is to b as c is to what. So, like man is to woman as uncle is to and you will answer aunt, but can my model answer that man I have given man woman pair, and for the next pair the first word is uncle, what be the next word, can you find out aunt. And that is what the word vectors have you found to be very useful in; they can predict these analogy sort of task. And how do we do that they just use the simple vector offset method.

(Refer Slide Time: 13:03)



So, let us see one example. So, here so this is the idea. So, I have my vectors denoted in a two-dimensional plane. So, how do you convert any say 100 dimensional vectors to two-dimensional representation, you can use principle component analysis PCA or some other methods to project them into some lower dimension, so that is been done here, two-dimensional representation. So, what you are seeing here? So, that outside would be woman and man that is similar to what has been observed in uncle and aunt, and king and queen and this is a very nice regularity.

(Refer Slide Time: 13:39)



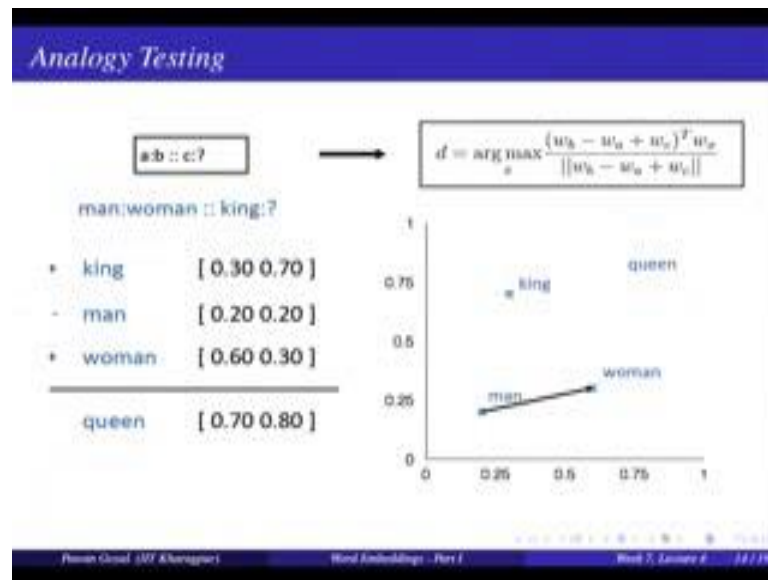
Similarly, here for singular plural king to kings and queen to queens, they are having similar offsets.

(Refer Slide Time: 13:48)

Encoding Other Dimensions of Similarity			
Analogy Testing			
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy - Rome	Japan - Tokyo	Florida - Tallahassee
big - bigger	small - larger	cold - colder	quick - quicker
Miami - Florida	Baltimore - Maryland	Dallas - Texas	Kona - Hawaii
Einstein - scientist	Messi - midfielder	Mozart - violinist	Picasso - painter
Sarkozy - France	Berlusconi - Italy	Merkel - Germany	Koizumi - Japan
copper - Cu	tin - Zn	gold - Au	uranium - plutonium
Berlusconi - Silvio	Sarkozy - Nicolas	Putin - Medvedev	Obama - Barack
Microsoft - Windows	Google - Android	IBM - Linux	Apple - iPhone
Microsoft - Ballmer	Google - Yahoo	IBM - McNulty	Apple - Jobs
Japan - sushi	Germany - beer/wurst	France - baguette	USA - pizza

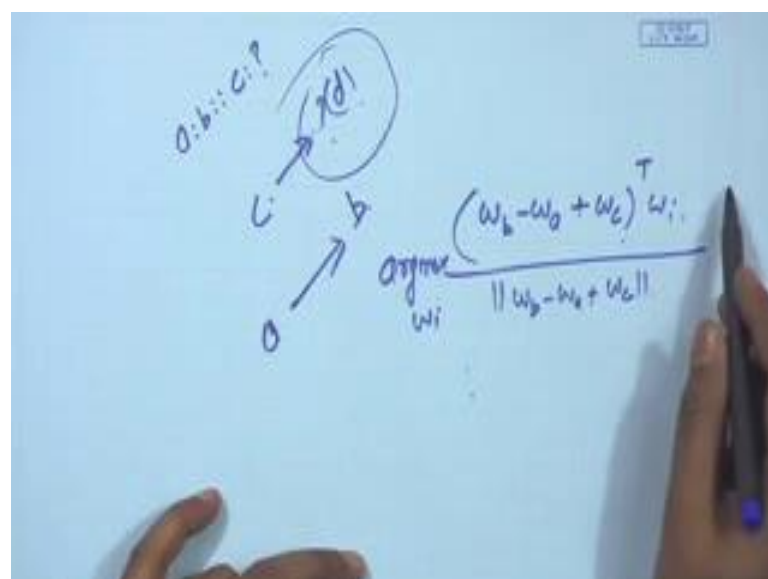
So, we can use that for analogy testing. So, what is the analogy testing task you are given a pair with a certain relation like France and Paris. What is the relation? So Paris is the capital of France. Now, we are given various examples and you have to find out which of this examples exhibit the same relation, so that is whether Italy, Rome has a relation of capital and country; Japan, Tokyo and Florida Tallahassee which of these has the same relation. So, can my vector representation capture this? So, a given one example can you find out the other example. And so similarly for big, bigger can you find out this small smaller, cold colder, quick quicker, and Miami Florida can you find out other examples so on.

(Refer Slide Time: 14:42)



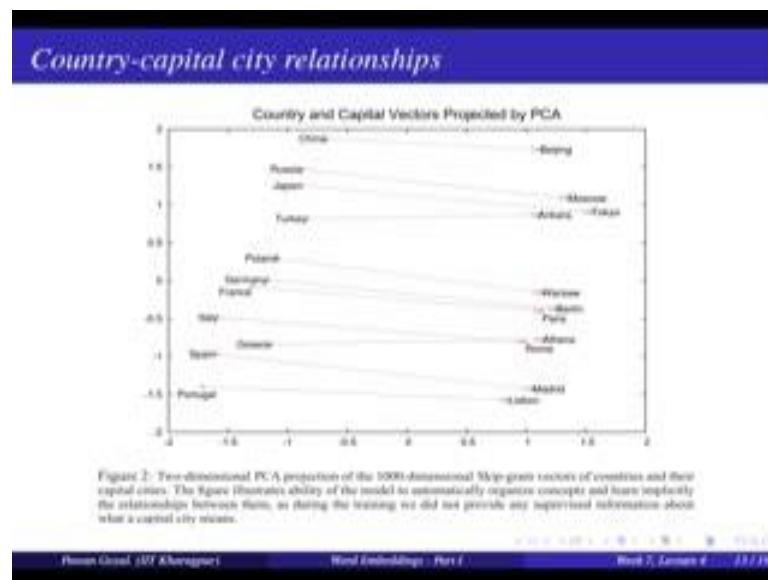
And how do they do that. So, I have this task a is to b as c to what. So, example is man is to woman as king is to what. And how will they do that. So, simple idea is take the vector offsets between woman and man, and add it to king. So, find out woman minus man, add it to king, but how do you find out the word queen, in general it may not be the exact match. So, how do you find out what words are coming closer.

(Refer Slide Time: 15:20)



So, this is the idea. So, I have a is to b as c to what. So, what will happen in my vector representation a, b and c, I want to find out what is the word d. So, what will I do, I have the vectors of each of the words. So, I will find out the offsets  $w_b$  minus  $w_a$  added to c. Now, I am trying to find out which vectors are similar to this. I write it this and where what are the vectors are similar to this one. So, I will just find out similarity of that to all the words  $w_i$  in my corpus and I can also normalize it  $w_b$  minus  $w_a$  plus  $w_c$ , so that is a normalize all this is not very important and we take the argmax over all  $w_i$  in my corpus. So, all words are vectors. So, I find out which words are coming closer in this space. So, what is the closest words to by when I add this offsets to this word and that is my answer, for example, if you do this here, so will find that the word queen comes up.

(Refer Slide Time: 16:45)



And this has we shown in many different cases like country and capital vectors, China-Beijing, Russia-Moscow, Japan-Tokyo, and the what you are seeing here the offsets between the vectors are very, very regular in all this cases. And this is just coming out from the word vectors.

(Refer Slide Time: 17:06)

More Analogy Questions			
Newspapers			
New York San Jose	New York Times San Jose Mercury News	Baltimore Cincinnati	Baltimore Sun Cincinnati Enquirer
NHL Teams			
Boston Phoenix	Boston Bruins Phoenix Coyotes	Montreal Nashville	Montreal Canadiens Nashville Predators
NBA Teams			
Detroit Oakland	Detroit Pistons Golden State Warriors	Toronto Memphis	Toronto Raptors Memphis Grizzlies
Airlines			
Austria Belgium	Austrian Airlines Brussels Airlines	Spain Greece	Spainair Aegean Airlines
Company executives			
Steve Ballmer Samuel J. Palmisano	Microsoft IBM	Larry Page Werner Vogels	Google Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 16 / 19

And more questions like newspapers. So, New York and New York Times (Refer Time: 17:12) Baltimore and Baltimore Sun, San Jose San Jose Mercury News and so on; various NBA teams Detroit Detroit pistons and so on; airlines - Austria Austrian airlines, Belgium Brussels airlines and company executives. So, what do you see it is capturing a generic relation also? In general, any relation that can hold between two words, so if you are finding out two words with one relation you can find out some many other words that I having the same relation by simple this vector offsets method. So, find out vector offset between pairs, is it similar to the vector offsets of my initial example.

So, this is the problem we also tackled in the case of structured models of distribution semantics that is we were making the pair pattern matrix and then capturing the co occurrences. In the case of word vectors, even though you did not start by pairs and patterns, even though you found out only with the word embeddings word vectors this helped further in doing this task also, and this was very one of the very interesting aspects.


(Refer Slide Time: 18:27)

### Element Wise Addition

We can also use element-wise addition of vector elements to ask questions such as 'German + airlines' and by looking at the closest tokens to the composite vector come up with impressive answers:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zoty	Viet Nam	flag carrier Lufthansa	igriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cécile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.



Prasen Ghosal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 19

Similarly, here we can do element-wise addition. So, suppose you have the embedding for a German, you have embedding for airlines and if you add these two embeddings, and find out words that are coming closer to the new vector now. And you find out some very interesting words coming up like here check plus currency, and you find out word that very close, Vietnam plus capital again words setup very close German plus airlines you find Lufthansa airline and Lufthansa a carrier so on, Russian plus river you find words like Moscow French plus actress and you find some French actress. And this was again some interesting aspect. So, you can have embedding of two words and you add these gather if find something that is some sort of composition of these term. So, it is not a generic method. So, this was coming out in some cases may not be true for all the cases, but even coming out in some cases for what in interesting observation.

(Refer Slide Time: 19:22)

*Learning Word Vectors*

*Basic Idea*

*Instead of capturing co-occurrence counts directly, predict (using) surrounding words of every word.*

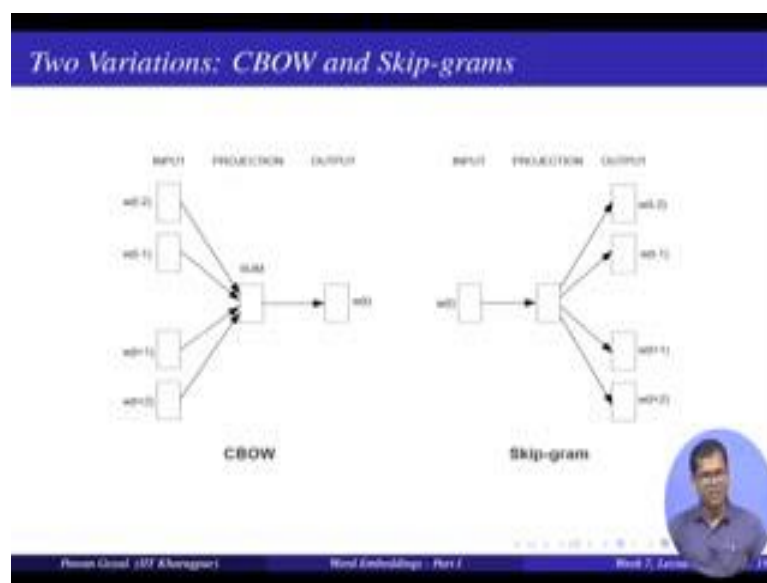
Code as well as word-vectors: <https://code.google.com/p/word2vec/>

Pawan Goyal (IIT Kharagpur) Word Embeddings - Part I Week 7, Lecture 4 18 / 19

So, now how do we capture these word vectors, how do we compute this word vectors. Now, basic idea is we will again go back to the co occurrences, we will trying to use co occurrences. But instead of counting the co occurrences from a corpus, what we will say I am given a sentence a word is there, and the contest is there, try to predict the contest from the word or the word from the contest. And if you not able to predict, try to update your weights, and this will be the idea. It starts with some initial vectors, using those vectors try to predict. From the contest, what to do the targets, after the target what will be the contest if it is not matching update your vectors. So, all the codes as well as the learned vectors are available here. So, you can actually download these and try to use them for many of your task also, but we will discuss how what are this word vectors in how to they come.



(Refer Slide Time: 20:26)



So, in general, there are two different variation of this models, so that tried to capture these word embeddings, and they are called CBOW for continues back of words model and skip-gram models. And let me just quickly explain what are these and we will discuss in details in the next lecture. So, in continuous back of words model, what you are doing, so, we are taking the contest. So, you are focusing on the current word  $w_t$ , you are taking the previous words. So, it can be actually any number of previous words and to next words and you are trying to use those to predict the center word. So, using the contest predict the target word or the center word.

In the skip-gram model, what you are doing you are using the center word and trying to predict the context words. So, there are two different ways of learning these embeddings. So, idea would be I will start with some initial vectors, now trying using the context vectors, I will try to predict what is my center vector. If I am not finding the match I will update my different vectors center as well as the context. And I will keep on doing that until I am able to predict this some high confidence or I am converging at certain point my vectors are not changing. And these vectors that I am learning by this method will be my word vectors and I do that slightly different in both continuous back of words model and skip-gram model.

So, in the next lecture, what we will do, we will discuss in detail how do we learn these vectors.

Thank you.