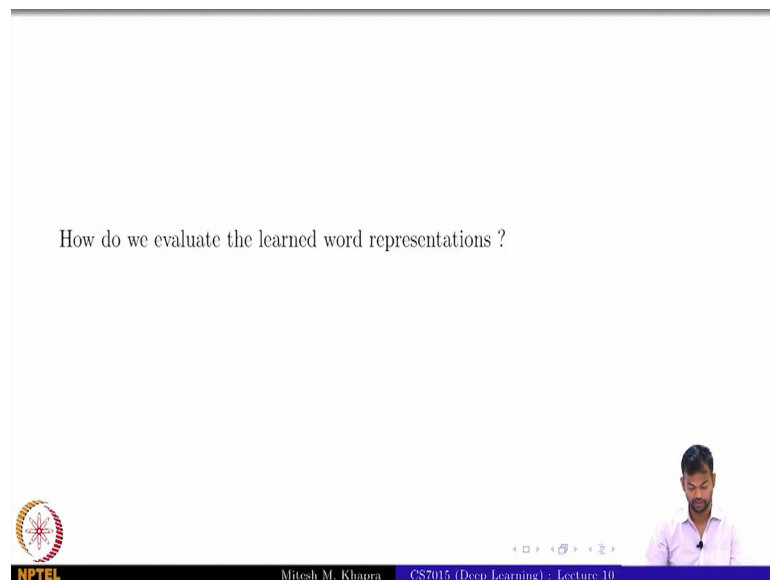


Deep Learning
Prof. Mitesh M Khapra
Department of Computer Science Engineering
Indian Institute of Technology, Madras

Module – 10.9
Lecture – 10
Evaluating word representations

So, now we come to this important part about how do you evaluate word representations right.

(Refer Slide Time: 00:16)



So, there are different tasks that are set up, I hope some of you have read that paper and I can see that none of you have read that paper. So, semantic relatedness is one way of evaluating word representations.

(Refer Slide Time: 00:30)

Semantic Relatedness

- Ask humans to judge the relatedness between a pair of words
- Compute the cosine similarity between the corresponding word vectors learned by the model

$$S_{human}(cat, dog) = 0.8$$
$$S_{model}(cat, dog) = \frac{v_{cat}^T v_{dog}}{\|v_{cat}\| \|v_{dog}\|} = 0.7$$

Handwritten annotations: A bracket groups w_1, w_2 and v_1, v_2 . Two vertical ovals labeled 'H' and 'M' are drawn. A presenter is visible in the bottom right corner.

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

So, ask humans to just the relatedness between a pair of words. So, I construct some pairs of words, and I show them to a human. And ask them how related do you think they are on a scale of one 0 to 1 right. So, it is likely for cat and dog someone would say 0.8 or at least you would expect values greater than 0.6 right?

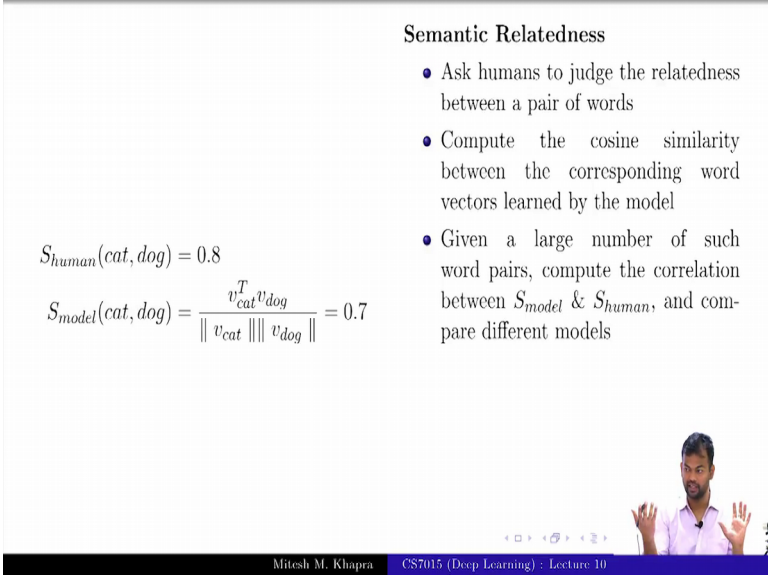
Now you have learned the representations using your model. It could be any of the models that we have seen so far, continuous bag of words, skip gram or glove vectors. So, these are the three things right continuous bag of words, skip gram which is known as word to wick and the glove representations. And within them you could have this hierarchical softmax and other things and so on.

So, you could if I asked you what is the similarity between cat and dog, according to your word representations; You could just use the cosine similarity and tell me that this is the representation right. So now, I will have many search words $w_1 w_2$ for which I have the human judgment and I have the model judgment right. So, I will have $w_1 w_2$ then $w_2 w_1$ oh sorry, sorry $w_1 w_2$ and so on. I will have many such word pairs for each of these word pairs I would have the human judgments and I would have the model judgments right. How close do the humans think they are and how close do the think they are?

Now, I can compute the correlation between these 2 decisions or these 2 random variables. And I would want that for a good model this correlation should be high. So,

whenever humans said that the 2 words are actually similar the models word vectors should also predict a high cosine similarity. And whenever humans said that the 2 words are not similar, the models word vector should also result in a low cosine similarity, how many forget this?

(Refer Slide Time: 02:07)



Semantic Relatedness

- Ask humans to judge the relatedness between a pair of words
- Compute the cosine similarity between the corresponding word vectors learned by the model
- Given a large number of such word pairs, compute the correlation between S_{model} & S_{human} , and compare different models

$$S_{human}(cat, dog) = 0.8$$
$$S_{model}(cat, dog) = \frac{v_{cat}^T v_{dog}}{\|v_{cat}\| \|v_{dog}\|} = 0.7$$

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

So, that is one way of evaluating how good your word representations are right. So, as I was saying earlier, how do you tune those parameters. So, you could have such a set, once you have learned some word representations and you want to see whether parameter k 1 was better than sorry, rather hyper parameter k 1 was better than hyper parameter k 2 you could just take those two word representations learned by these two different hyper parameter settings. Evaluate them on this corpus and whichever gives a higher correlation you can keep that hyper parameter; how many forget that?

(Refer Slide Time: 02:38)

Synonym Detection

- Given: a term and four candidate synonyms

Term : levied ✓₁

Candidates : {unposed, ✓₂
believed, requested, correlated} ✓₃ ✓₄ ✓₅

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 10

Other task is synonym detection. So, from a resource known as word net or from other dictionaries, you could get all the synonyms of a word. So, then people create a corpus where you give us in sin a word and give 4 candidates or some k candidates, out of which one of these is the correct synonym, the others are just distraction words right and distracting words. Now, what would you expect your word representations to do? You have word representations for all of these, what would you want? How would you pick up the synonym based on word representations?

Students: (Refer Time: 03:13).

The one which has the highest cosine similarity; so, again you will compute the cosine similarity, you will rank these and you will pick up the synonym right. And now again I gave you 100 such instances, I gave you a word for candidates and I gave you 100 such different word comic candidate pairs and you pick the synonym for everyone and see for 60 of them you got it right then your accuracy 60 percent. So, that tells you how good your word representations are.

Again if you are given two different hyper parameter settings one gives you 60 percent accurate, the other gives you 70 percent accurate you will probably go with the one which gives you 70 percent accurate. They are find how you can use this.

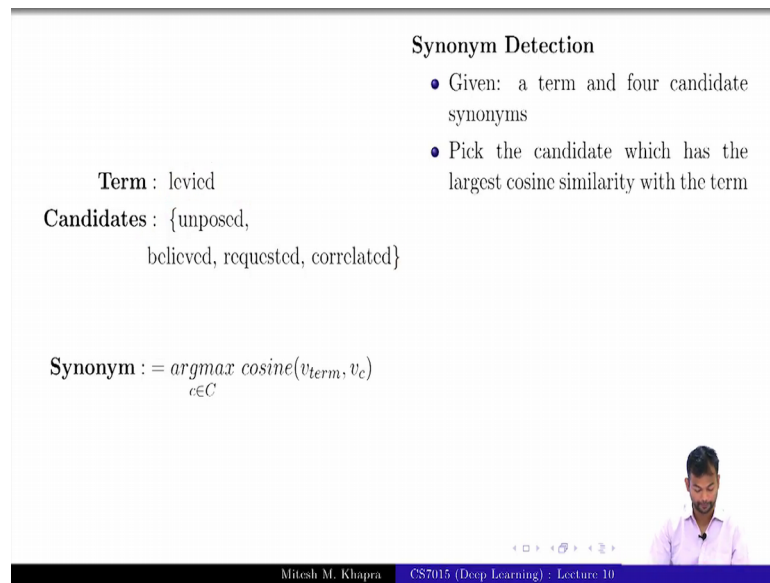
(Refer Slide Time: 03:49)

Synonym Detection

- Given: a term and four candidate synonyms
- Pick the candidate which has the largest cosine similarity with the term

Term : levied
Candidates : {unposed, believed, requested, correlated}

Synonym : $= \underset{c \in C}{\operatorname{argmax}} \operatorname{cosine}(v_{\text{term}}, v_c)$



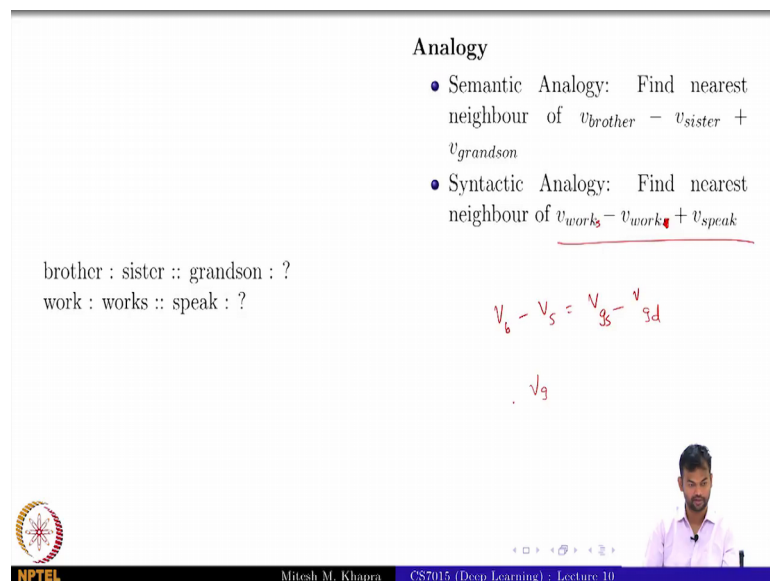
The third is analogy task.

(Refer Slide Time: 03:50)

Analogy

- Semantic Analogy: Find nearest neighbour of $v_{\text{brother}} - v_{\text{sister}} + v_{\text{grandson}}$
- Syntactic Analogy: Find nearest neighbour of $v_{\text{works}} - v_{\text{works}} + v_{\text{speak}}$

brother : sister :: grandson : ?
work : works :: speak : ?

$$v_b - v_s = v_g - v_d$$
$$v_g$$


Where you find the nearest neighbor of this operation what should it be (Refer Time: 03:58) this is this analogy teller brother is 2 sister as grandson is 2 something right. So now, the idea here is that if I mean it is like pretty weird right. So, if I take brother minus sister I get something.

Now, if I add grandson to that then I should get granddaughter. It is intuitive in a way right; I mean this is what you expect your word vectors to do right. So, that is how the

analogy task works. So, you could set up an analogy task, you could have and you could get several such an analogy tasks from online tests and so on. And you would want your word representations to exhibit this kind of a behavior right.

So, again you have these 100 analogy tasks, for each of these you know the true answer and from each of these you predict the answer from your word representations. And first see for how many of them you get it correct. Then you could also have a syntactic analogy. So, you can tell me what this would be right. In fact, here again it should be the other way round. We works minus we work, thus we speak would be we speaks right. So, that is the syntactic thing right.

So, you are getting a different form of the world. So, your word representations should also have this kind of properties, that is what you desire. So, just evaluating whether your word representations show this kind of a property or not. So, we have seen three tasks, one is semantic relatedness whether a pair of words how do humans rank it and how do the model how does the model rank it; Then the synonymous detection and the analogy tasks. In each of these you do something with the word representations in the first two you use the dot product, and this last one you use some arithmetic operation over the word representations.

So, you would want $v_{\text{brother}} - v_{\text{sister}}$ is equal to $v_{\text{grandson}} - v_{\text{granddaughter}}$ right. So, $v_{\text{granddaughter}}$ right; so, that is there is a plus minus error there.

(Refer Slide Time: 05:51)

- So which algorithm gives the best result ?
- Boroni et.al [2014] showed that predict models consistently outperform count models in all tasks.
- Levy et.al [2015] do a much more thorough analysis (IMO) and show that good old SVD does better than prediction based models on similarity tasks but not on analogy tasks.

So now which algorithm gives the best result right. So, whenever we see a bunch of algorithms same as we did with Adam and (Refer Time: 05:59) and so on we always want to answer this question, which of these gives the best result.

So, there was this study done by Boroni et. al in 2014, that show that the predict based models right. Which are either which are the predict based models actually; skip gram, continuous bag of words and even glove for that matter right because it is also a predict based model. These continuously are consistently outperform count based models that is what they said, but a year later there was a separate study done by someone. And in my opinion this was a more thorough analysis because the earlier study right, they did not really give SVD a chance to win in my opinion this is all on camera.

But the later the second set of guy right they gave SVD a chance to win. So, I will tell you one example of how they gave is really a chance to win. So, remember in word to wake you had this weird 3 by 4, which you are using to raise the probability right. Now what they did is they said even in the co-occurrence matrix actually these counts that you have, if here you are using them by 3 by 4 in the case of word 2 I can getting better results, why not do the same thing in the co-occurrence matrix also?

At the end of the day you are raising the count to 3 by 4 right. So, whatever counts you have here based on that you will compute ppmi or pmi or whatever, but first why not try to adjust these counts. So, why not have a parameter k such that you can raise the counts

to this parameter and then do all those computation. And that is fair because the word to wake has a parameter hyper parameter. So, why not give a similar hyper parameter to SVD.

Similarly, they did something to take care of the k negative samples which word to wake has why not give SVD also similar chance right. So, when they did these kind of adjustments, they found that after these modifications SVD does as well as or even better than word to wake models for the similarity tasks, but not for the analogy task. But the analogy task was the last task right, brothers to sister's grandsons to grandmother right

So, in most cases we care about similarity, and in very few cases we care about analogy if you are doing NLP application so; that means, in most cases SVD would just be fine. So, that is what I just said at the beginning.