WEBVTT

1

00:00:06.370 --> 00:00:21.810

Jisun An: Right. Thanks for joining for today's lecture. So to mark your attendance, the passcode today will be word. So please mark your attendance. We have some seats inside over there and the front seats.

2

00:00:30.350 --> 00:00:31.170

Jisun An: Yeah.

3

00:00:32.140 --> 00:00:43.019

Jisun An: it's a freaking cold. Thanks for joining in this cold weather. So for those who just arrived. The passcode for today is word.

4

00:00:43.270 --> 00:00:45.430

Jisun An: please mark your attendance.

5

00:01:09.090 --> 00:01:23.930

Jisun An: Alright. So we have a lot to cover today. But I I think maybe most of you. Some of the concepts are familiar with you. So how many of you have taken machine learning course?

6

00:01:25.730 --> 00:01:44.379

Jisun An: So like half okay and a half, you're not familiar with. Okay, okay, I see. So so for some of you, especially if we go to like a neural network stuff for those who already taken machine learning courses, it'll be too easy. But just take it as a reminder. But for those who haven't been taken, this will be still very, very

7

00:01:44.460 --> 00:01:56.311

Jisun An: more piece of machine learning. So take it as a 1st step, and if you're interested in more on that, you probably need to take it by yourself to have a better understanding.

8

00:01:56.810 --> 00:02:03.029

Jisun An: but I will try to find a good balance between them. So just a reminder so we

9

00:02:03.820 --> 00:02:12.359

Jisun An: so I don't have. I can have some other announcement. But today just passcode. So the today's passcode is world.

10

00:02:12.550 --> 00:02:18.420

Jisun An: Wrl, mark your attendance, and let me get to the lecture directly.

11

00:02:19.460 --> 00:02:22.869

Jisun An: So so we started from

12

00:02:22.900 --> 00:02:41.009

Jisun An: how we can build a Nlp system right? And the example was, we were building a system where where the input is the text like a review from like a movie. And the output is the sentiment either positive negative or the neutral. And there are many different ways, that how you can build this system. And

13

00:02:41.030 --> 00:02:57.810

Jisun An: if you see that how the system has been developed over the years and times, then you can see that? Why, we ended up with this transformer architecture, and eventually the large language models. So we are kind of in that path to let you know that how these things has been

14

00:02:57.920 --> 00:03:13.350

Jisun An: developed. So we started from actually the rule based system, where basically, we can have a list of lexicons which are like positive or the negative, and you can extract a simple features like counter of the positive lexicons versus

15

00:03:13.350 --> 00:03:34.299

Jisun An: number of the negative words. So you consider them as a feature, and we had a weight vectors, and by multiplying by these weight vectors to the feature, will result in a some kind of the score, and then we had a decision function given that score value. Now you're assigning one of the label given that score right? So that was like the rule based

16

00:03:34.834 --> 00:03:55.015

Jisun An: system. The next one came up was tobacco bird. So now, instead of so the rule based system had some limited a lot of big limitations. But I guess the largest limitation would be, I mean, basically, you need to add all those electric that is required to classify your input. To a particular

17

00:03:55.580 --> 00:04:20.230

Jisun An: sentiment then, but that is not really possible. So now we are applying a machine learning algorithm to learn those features. So instead of we define the features. Now, we are using machine learning to learn what is the best feature to build this classifier. And in building those machine learning algorithm, one of the 1st choice is using the backwards. So the backwards. In another way.

18

00:04:20.540 --> 00:04:25.830

Jisun An: it's very simple. We basically consider all individual award as a

19

00:04:26.140 --> 00:04:54.036

Jisun An: kind of each individual kind of feature. And we are using all of them. And we have similar similar to this rule based method. We also had a weight factor where each of these value also meant or represents, how important each of these words. And we represent each of these words as a factor. And that's the way that how we let computer understand our language right? But the back of word model is basically it has

20

00:04:54.400 --> 00:05:11.739

Jisun An: the feature vector itself is the word identity itself. So we give just one to the identity of that word. So the same word. And then for all the other columns where it is basically all the other words we give 0. So the vector, themselves is represent the word identity.

21

00:05:12.050 --> 00:05:21.780

Jisun An: And given that we have these weights. And similarly to the rule based system. We compute the score and based on that score. You also decide, like which label to do.

22

00:05:21.990 --> 00:05:43.050

Jisun An: And then this backwards had several also limitations, and one of them was basically it cannot be really handling the conjugations or the compound words, or it would not be able to look at the similarity between the words. So even though we know that the love and the adore are exactly similar words, and those.

23

00:05:43.380 --> 00:05:45.439

Jisun An: if you see if

24

00:05:45.810 --> 00:06:05.940

Jisun An: if basically adore the word. It was not existing in your training data, then your model will never know that the adore is actually relating to the positive review, even though, as a human. We know that love and the adore are the similar value, so they should be considered similarly, and those that the second sentence also can be

25

00:06:06.090 --> 00:06:24.450

Jisun An: classified as a positive value reviews. But these, these things cannot be handled in the bow model, and also they cannot handle the combination of the features. So they, basically, this model cannot distinguish between. I love this movie versus, I don't love this movie.

26

00:06:24.680 --> 00:06:39.350

Jisun An: And lastly, handling the sentence structure, so maybe started from something more positive and then ending up with the negative. So these kind of structure characteristic may not be reserved in this bow value.

27

00:06:39.350 --> 00:07:00.420

Jisun An: and those there are a couple of different techniques that try to tackle each of these problems. So the 1st one was able to handled by the subord models which was also called as a tokenization, and that was something that we were talking about last lecture. And then the order, similarity, or

28

00:07:00.580 --> 00:07:14.179

Jisun An: representing these words based on more dense factors, so that from the vector we can compute to which words are more similar to each other, these can be handled by the word embedding, which we will go through today, and also the combination feature. Now.

29

00:07:14.180 --> 00:07:31.999

Jisun An: while each word is represented individually, if you have another layers of the features or the weights that can tell the combination of these 2 words actually mean for something, then that'll be solving this combination feature problem. And then neural network actually addressed

30

00:07:32.000 --> 00:07:35.829

Jisun An: to handle those that pick that particular issue.

31

00:07:35.940 --> 00:07:51.850

Jisun An: And that's also something. I hope, that we can go through today. But let's see how it goes. And then the sentence structure. Now you need to. You need another model to handle the sequence of these models, which will be built in the next lecture.

32

00:07:52.620 --> 00:08:17.999

Jisun An: So the sub word model. We have 3 common models which you see by pair encoding, and the word piece and the unigram language model tokenization. And basically all any submodel usually have 2 parts where, firstly, it need to learn how to tokenize the thing, and then it has the segmental where basically given an input, and given a list of the vocabulary. Now you need to segment your inputs.

33

00:08:19.137 --> 00:08:43.739

Jisun An: How? Our. So we're not going to go through everything. But then, the by pair encoding was we, starting from the very like minimum set of vocabulary, which is like character level. And then we were adding more and more vocabulary. Given the the frequency of the pairs of the characters in our corpus. Right? So if you remember this, then the the largest number, the pair

34

00:08:43.740 --> 00:09:07.929

Jisun An: with the highest frequency, was the enr. So basically you, you look for from your training corpus, you are looking for those pairs with the highest frequency, and you are adding those pairs to your vocabulary, and you are adding this vocabulary until until a certain threshold. And once you have this vocabulary, then now you are using that vocabulary in a greedy way. You are segmenting your

35

00:09:08.830 --> 00:09:32.430

Jisun An: input. So that was the Bpe that. And then the word piece is very similar to the Bpe. But instead of selecting the most frequent pair in. Now also consider how important each of those characters or the word is. And so basically they use this score to select those vocabulary to add so if

36

00:09:32.430 --> 00:09:43.620

Jisun An: combining the 2 is actually doesn't is hurting the model, then basically just a scheme and leave it as the element is. And if so, basically they are checking, whether

37

00:09:43.790 --> 00:09:47.650

Jisun An: merging the 2 symbols are really worthy enough or not. Yes.

38

00:09:50.198 --> 00:09:51.930

Jisun An: know, the score is high enough.

39

00:09:52.570 --> 00:09:59.770

Jisun An: high enough. Oh, oh, no. So they will basically select the one with the highest score out out of your courses in this case. Yeah.

40

00:10:01.000 --> 00:10:09.379

Jisun An: thanks for the question and then and and also when you also in the

41

00:10:09.380 --> 00:10:32.279

Jisun An: Ppe case, when you segment. Now, your input, basically, you are taking the greedy kind of methods. You. You have the vocabulary in order, and you will merge, as you see in in the next vocabulary. When in the word piece you would basically define the longest subword, and then do the merge, first, st etc. So that would be like the slight difference between the ppe and the word piece.

42

00:10:33.380 --> 00:10:56.829

Jisun An: Okay? So I think that was the way we were at. And now we are moving to the unigram model. And this one is slightly different from the other 2 approach and I mean this one is also very popular, used in like Alberta, t. 5, and the excel net, which are most all the popular language models. So once again, Bpe and the work piece they start from like very small

43

00:10:56.830 --> 00:11:07.910

Jisun An: vocabulary, and then they are finding those vocabulary to add to merge, and then they basically find the like merge rules right? But then the unigram mode actually start from the big vocabulary.

44

00:11:07.910 --> 00:11:18.119

Jisun An: and then they remove the tokens, and then they find the optimal number of the vocabulary that is working working best. So now and the start.

45

00:11:18.520 --> 00:11:45.429

Jisun An: because I don't have an example here. Big vocabulary means that you can imagine. If you have a sentence, then basically, you can add audio, and then you can also add all the like diagrams, trigrams, or the words themselves. Right? So so it can be really starting from the biggest vocabulary that you can find from your training corpus, and then you are removing those tokens that is like basically low frequency or not really helpful in your training. Corpus

46

00:11:46.483 --> 00:11:52.146

Jisun An: so, and then they also assume I mean in the wind. So now they need to

47

00:11:53.360 --> 00:12:14.039

Jisun An: So what is the selection? So how do you know that which vocabulary is really helpful or not? That will be determined, based on this unigram model. So unigram model is, I mean, we will probably talk more about it next lecture. But you can now assume that unigram model assumes that the words just exist independently, without any

48

00:12:14.040 --> 00:12:36.229

Jisun An: dependency with any other words, even though that's I mean. So this unicorn model will be probably the simplest model that you can find. So, in a very, very simple term, unigar model will simply count how many, how the frequency of each of the word, and then compute the probability of each of the words, and then you will assume that the word will be appear based on those frequencies. So

49

00:12:36.576 --> 00:12:51.843

Jisun An: the problem of the unigram model, I mean, you will probably see all the like, the articles the most right. So if you are generating the text. Given the unigram model, you will probably start from the it will like end like that right?

50

00:12:52.210 --> 00:13:21.280

Jisun An: so the unigram model is terrible at many of the system, but but this one, and but then but then, when you have the unigram model the only thing they borrow the unique. Some borrow things from this model is that you can compute the likelihood of the sentence. So now, so imagine each of the words can be turning into a probability given your given your corpus right? And now you're and then here the word

51

00:13:21.670 --> 00:13:27.319

Jisun An: I probably mentioned token, and the word as a

52

00:13:27.680 --> 00:13:52.019

Jisun An: interchangeably. So whenever I'm talking word, I also meant for the token as well. So sorry about that. So now that if you have an input and assuming that this sentence was split into word or the tokens, then each of the word or tokens now can be turned into the probability right? And then the multiplication, because unigram model assumes the independence of the word, you can compute

53

00:13:52.020 --> 00:13:58.960

Jisun An: the probability of a existence of that sentence by multiplying the probability of the individual word.

54

00:13:59.060 --> 00:14:16.939

Jisun An: So that's see how unigram model usually works. If this is not very clear, we will go back to this next next lecture. So don't don't worry. But so basically, you can compute for each of this sentence what is the likelihood that you will see this sentence

55

00:14:18.100 --> 00:14:29.320

Jisun An: using the unigram model once again, unigram model is just a simple frequency of the word or the probability of each of the words. And now so we, because we had this big vocabulary, where for the same

56

00:14:29.320 --> 00:14:57.509

Jisun An: word, you can tokenize it into like alphabet, individual alphabets, or you can tokenize it as a word in itself, or you can tokenize it as like the morphemes, are kind of separated, so like entertain y like separately them into the 2 different tokens. Now you have a different way to tokenize this sentence, and each token has a different probability. So if you multiply them all together, then even for the same sentence.

57

00:14:57.800 --> 00:15:02.440

Jisun An: given the way that how you tokenize this input, you will have different probability. Right?

58

00:15:04.480 --> 00:15:24.879

Jisun An: If you think about if if you are tokenizing this sentence based on the alphabets, then individual alphabet may have may have like very low frequency, so it may, and also it will, have more multiplications, and those it may lead to very low probability to observe, and then etc, etc.

59

00:15:25.260 --> 00:15:26.315

Jisun An: So

60

00:15:27.750 --> 00:15:43.819

Jisun An: So the low likelihood of the corpus here, so the peak, a vocabulary that maximize the log likelihood of the corpus, given a fixed vocabulary size, meaning that now the likelihood of the corpus is the the multiplication of this probability.

61

00:15:43.840 --> 00:16:03.010

Jisun An: and then the reason that we are using the log is because if you take the logo to the product, then you can sum these values up right. And that's the only reason that we are taking the log likelihood, and the log likelihood means that what is likely that you observe this sentence from your purpose?

62

00:16:03.630 --> 00:16:29.210

Jisun An: So we just want to take the vocabulary or the tokens that are increasing the low likelihood of the corpus. So, in other words, we are just. You are just using those tokens that are useful, or maybe appearing more frequent, but it would not be exactly the most frequent words, because it also depends on how many you appear, and in which other tokens you are kind of

63

00:16:29.400 --> 00:16:34.159

Jisun An: coming together and etc. But that's the how it kind of works.

64

00:16:34.863 --> 00:16:40.889

Jisun An: Yeah, I'm sorry that if this is not very clear, but

65

00:16:41.190 --> 00:17:01.920

Jisun An: at this moment, and maybe I should prepare some examples next time. But at this time just think it as differences of this unigram model versus the Bpe. So Bpe start from something very small vocabulary, which is like the characters, and then they are adding more vocabularies, by looking at, which are the most frequent

66

00:17:01.920 --> 00:17:23.600

Jisun An: pairs that is, appear in the copies, but in the unigram mode that you start from the big vocabulary, and then you are. You now have different because you have a big vocabulary. You have many different way to tokenize your same sentence. Right? So you actually examine all different tokenization strategies. And then you compute the log likelihood

67

00:17:23.599 --> 00:17:40.050

Jisun An: using this unigram model. And then you are selecting the set, the tokenization that gives the highest look likelihood, and for each of the example you are combining them, and those will be your tokenized, the vocabulary set. And then you are just repeating this over and over.

68

00:17:41.760 --> 00:17:50.330

Jisun An: I hope. Yeah. The example could have been, make this much easier. But for now, just focus on the differences between these 2 models.

69

00:17:52.940 --> 00:18:17.910

Jisun An: And then the sentence piece is actually not a new method. But this is actually the library. And basically, it's just very highly optimized library that you can. I mean, these are like the repetitive iteration kind of process. So it just requires some optimization. And this is the Python library that anyone can use for doing the tokenization, and they support for Vp

70

00:18:17.910 --> 00:18:23.739

Jisun An: and the unicor model. So this library themselves has been used a lot in different large language models.

71

00:18:26.060 --> 00:18:50.699

Jisun An: So here are some examples in which I will probably not go through very in depth, and I see that some of you, maybe this text might be too small. But but these are basically the examples of the researching tokenization. And take your time, and later, maybe at your own pace and see what kind of makes differences. Basically, these texts include, like lower cases, the capitals

72

00:18:50.700 --> 00:19:07.300

Jisun An: and emojis, and like other languages which is Chinese character and the numbers and the quotations and different signs. And and basically these, the different tokenization reserves in different tokenizing methods reserves in different tokenizing rizards.

73

00:19:07.680 --> 00:19:12.720

Jisun An: So you can have a quick look how how they are kind of ended up.

74

00:19:13.480 --> 00:19:27.159

Jisun An: And the only thing that I should empathize at this moment is that so different model use different tokenization. And when you start to actually do the coding with these models. Usually they are kind of came as a couple kind of

75

00:19:27.160 --> 00:19:45.550

Jisun An: you're using basically the same model name as a variable, and you're using it for the tokenizer and the model. So usually there's no mistakes. But make sure that you are actually using the same tokenizer and the tokenizer that is used for a train step for that model. So that is the only highlight that I I should give it to you

76

00:19:45.550 --> 00:19:46.609

Jisun An: at the moment.

77

00:19:47.530 --> 00:20:11.509

Jisun An: Yes, I I think I think I don't think so. Probably internally, when they when each of the model, when they built there for their own. They probably tested it, but I and there may be some information in the paper, but I don't recall at the moment, but but so

78

00:20:11.510 --> 00:20:22.496

Jisun An: so next lecture, I think we will go. We will talk about how you evaluate the language model, and there's a notion of like perplexity. The look like actually one of those

79

00:20:22.930 --> 00:20:45.809

Jisun An: method to evaluate whether your language model is good or bad. So I guess, using that, you can also evaluate which tokenization actually worked the best. But as you can see, this tokenization also depending on the training purpose, right? So I don't know whether this is depending on the models, but probably tokenization. More relate to what training data you have.

80

00:20:45.810 --> 00:20:53.279

Jisun An: and I assume that when each developers of these models, when they are building their own model. They probably did their internal test

81

00:20:53.320 --> 00:20:59.249

Jisun An: to see which one actually works the best. But yeah, but but that would be an interesting research question. Yeah.

82

00:20:59.520 --> 00:21:00.460

Jisun An: thank you

83

00:21:02.580 --> 00:21:03.580

Jisun An: And

84

00:21:07.090 --> 00:21:19.340

Jisun An: and so one of these. So one of the issues of this subord model is basically they cannot really handle well of the less common languages, because I mean, like

85

00:21:19.530 --> 00:21:28.039

Jisun An: both both method, or they are looking for those tokens that are like more frequently used.

86

00:21:28.680 --> 00:21:30.800

Jisun An: so they tend to

87

00:21:35.240 --> 00:21:57.109

Jisun An: If they don't see certain token many times, then usually they will be just segmented into individual characters. Right? So that will be what we will have especially the unigram model. Because, yeah, they will. They will prioritize those words that are more frequently appeared and then those that less common will be over segment.

88

00:21:57.470 --> 00:22:07.619

Jisun An: So for example, I mean, this is the tokenizer that's supported by the Gpt, and you can actually try it out under using this URL. So

89

00:22:08.180 --> 00:22:09.740

Jisun An: these are the

90

00:22:10.910 --> 00:22:17.200

Jisun An: I forgot the name of the the language from the these are from the Myanmar. Anyone knows the

91

00:22:17.520 --> 00:22:19.530

Jisun An: the the name of the language.

92

00:22:22.800 --> 00:22:49.659

Jisun An: Okay. Burmese. Yes, yes, thank you. Sorry. I've just yeah. So these are the Burmese. And then, as you see, the Gpt using the Bpe and then the Bpe, you can see that these are basically the the Burmese, the translation of the English. So these are the exact, the same sentence. And then here, the English basically resulting in 52 tokens, while the Burmese, resulting in

93

00:22:49.660 --> 00:22:56.909

Jisun An: 152 tokens, and the importance of the number of tokens. We will probably talk more later. But

94

00:22:56.990 --> 00:23:03.308

Jisun An: more tokens means that you just need a larger space to taken care of the same sentence. Right.

95

00:23:03.770 --> 00:23:13.380

Jisun An: because if you think about now, we will talk about the embeddings. So each word was representing on embedding. But now that will replace to the

96

00:23:13.380 --> 00:23:35.910

Jisun An: token level. Right? So if you have more tokens than to represent one sentence, basically, you need far larger space than the English, so this will be become far less efficient to represent. So this is something that you need to be. I mean, it's 1 issue that that has been largely discussed in the Nipra just wanted to mention.

97

00:23:38.550 --> 00:23:46.910

Jisun An: All right. So I will move on to the word embedding. So that was the sub model which is the tokenization. So

98

00:23:46.930 --> 00:24:12.059

Jisun An: the second issue that the bow model had was the they cannot really handle the word dissimilarity. And that can be reserved with the word vector so the idea is now, initially, we were representing each of the word as a 1 hot vector, where once again, you can imagine that you have the vector, n, vector where the n is the number of vocabulary that you have.

99

00:24:12.060 --> 00:24:21.769

Jisun An: And then you give one to the the words that you represent themselves and the all the 0 for the all the other word, and that was your one hot encoding.

100

00:24:22.140 --> 00:24:42.759

Jisun An: And even though I will keep using the word and showing the word as an example. Now, assuming that your model, using some one of the tokenization model, and this now will be each of the token, not the word, so it can be depending on what tokenization model that you're using. We will probably have like more tokens here.

101

00:24:43.260 --> 00:24:57.749

Jisun An: But instead of of this one hot representation, now we will try to come up with the dense representation. So now to represent each of the words rather than one hot representation, we will try to find the dense representation.

102

00:24:57.930 --> 00:25:27.490

Jisun An: So why do we need the dense factor? Because this dense vector may generalize better than the explicit count. So this one hot encoding can be used. One, or you can also give some weights like frequency or something. But the problem is that it just doesn't understand the relation between the words. But but we know that human languages are related to each other. So using this dense factor. Now, these values can represent some relationship among these words

103

00:25:28.110 --> 00:25:34.240

Jisun An: and those it can do better at capturing the synonyms. So as an example.

104

00:25:34.860 --> 00:25:44.845

Jisun An: So so with a word the feature is a word identity. So assume the feature. 5, that the the previous word was like a terrible, and then the

105

00:25:45.360 --> 00:26:11.830

Jisun An: basically, you require the exact same order to be in both the training and the test set. But then, with the embeddings, assuming that the previous word was factor something 35, 22, and 17. And now, if you know that in the test set, if you see a similar vector like 2421, 15, then you can kind of assume that these 2, even though this word was not.

106

00:26:11.950 --> 00:26:26.789

Jisun An: was in the training data. You can kind of assume that. Okay, these 2 are 2 vectors are similar to each other, so you can consider them similar kind of values. So, in other words, we can generalize this similarity to the unseen world.

107

00:26:28.080 --> 00:26:50.290

Jisun An: And what do these vectors represent? So there's no guarantee, but we hope that these words are similar, but then similarity can be defined in very different ways. Right? It can be synthetically similar or semantically similar. Or they can be same language. But but those similar words are close in the vector, space

108

00:26:50.820 --> 00:27:15.749

Jisun An: and our weight vectors and each of the weight vector element could be representing some kind of feature, so they may meant for maybe they are meant for like, they can indicate whether the word is about anymore objects, any made objects or whether they are positive or not. So you can assume that now, the vector weights that we are learning that can now represent some kind of

109

00:27:15.750 --> 00:27:21.630

Jisun An: of the words, and that can give us some idea of that can help us to

110

00:27:21.630 --> 00:27:27.719

Jisun An: embeddings of the word are positioned in a more closer in this embedding space.

111

00:27:28.340 --> 00:27:44.650

Jisun An: So once again, and these are just the toy example. These are not the real kind of embedding visualization, but you can see that similarity can meant for different things, either synthetic relationship, semantic relationship, or the language and etc.

112

00:27:46.810 --> 00:28:13.153

Jisun An: And there are different methods to get these short stance vectors, and the 1 1 of the most popular one is the War 2 bag that was presented in, published in 2013, and they introduced, like the 2 different methods of Skipgram and the continuous Bagel bird. Both are kind of similar. But we will give you more details of this kickgram which has been most popularly used.

113

00:28:15.530 --> 00:28:17.336

Jisun An: and so

114

00:28:25.360 --> 00:28:27.389

Jisun An: so here's the idea of the word to pack.

115

00:28:29.320 --> 00:28:37.670

Jisun An: It's a really, I think when I 1st read this paper. I was really shocked because it was. It was so so inspiring. I would say.

116

00:28:38.347 --> 00:29:02.589

Jisun An: so before that, like I think representing world was really hard, I mean, now, I mean, that sounds like a old days kind of story. But I think this was one of the fundamental work that really lead to all the development. Now. So the idea here is that we train a classifier doing the binary prediction task. And that task is that East W.

117

00:29:03.101 --> 00:29:07.930

Jisun An: Some word like the context word likely to show up near the apricot.

118

00:29:08.190 --> 00:29:20.464

Jisun An: So I mean, we we could think like sentiment, analysis as a prediction task, or maybe topic classification as a classification, I mean prediction task. But now they were kind of

119

00:29:21.200 --> 00:29:33.960

Jisun An: creating a new kind of prediction task where the 2 words are more likely to be next to each other or not, and this was the just. They define it as a prediction task.

120

00:29:35.310 --> 00:29:50.310

Jisun An: But actually, we don't really care about this task. How well they are doing with this test corner. But we just take the learned classifiers weight and take and use it as a word embedding. So that was the entire idea of this word to vac

121

00:29:51.860 --> 00:30:16.710

Jisun An: here. This was the 1st time that brought this big idea of the self supervision. So if you think about prediction task, some of the classification that you've met so far usually requires a training data. Right? So you have a sentence like movie review, and you actually need to label them as positive or the negative etc, right? And then, based on these labels, you are trained this

122

00:30:16.710 --> 00:30:23.960

Jisun An: classifier. But then the self supervision doesn't require any of these labels. You can do

123

00:30:24.160 --> 00:30:38.739

Jisun An: formulate a task using just the data that text that you have. And then from that you can learn something from you can build the model that learning from the data themselves, and that's what is called as a self supervision.

124

00:30:39.270 --> 00:30:44.860

Jisun An: So in this world to back they assume that there's a word. See?

125

00:30:46.140 --> 00:31:06.810

Jisun An: that occurs near the upricut is treated as a like positive example, the correct answers in this provides learning, and then and then they also take the those words that doesn't appear near the upricut as a negative example, then they curate the prediction task. So this really doesn't need for any human label

126

00:31:07.410 --> 00:31:17.410

Jisun An: and still they were learning about the relationship between the words. So the key idea once again, is this self supervision. Where to learn these

127

00:31:17.550 --> 00:31:26.480

Jisun An: weight vectors for the word embeddings. You don't really require any human label. You are simply using the idea of those words that are

128

00:31:26.560 --> 00:31:52.409

Jisun An: likely to be appeared together, supposed to be close to each other, and those who are less likely to appear together in one sentence or 1 1 program, or the one documents, they are likely to be away to each other. So that was the key concept, and they, I think the idea was very simple, but they just formulated as a prediction task, and they trained the model and the the word embedding had so much different implications.

129

00:31:53.890 --> 00:31:59.358

Jisun An: The idea is similar. Right? I will. I will probably repeat this over and over. So so,

130

00:32:00.140 --> 00:32:23.989

Jisun An: so basically, given a target word T, you have neighboring context word C, as a positive example, and you and then you also need to select, like the negative examples right? And then there could be different ways, but here they simply randomly sample the other words within the corpus, but they they still use the free, the frequent word that didn't appear as my neighbor as a

131

00:32:23.990 --> 00:32:32.199

Jisun An: negative examples, and then they were simply using the logistic regression to train the classifier to distinguish between these 2 cases.

132

00:32:32.550 --> 00:32:35.804

Jisun An: and then the the weight vector that

133

00:32:36.660 --> 00:32:40.409

Jisun An: for this word, they just use it as the word embeddings.

134

00:32:42.230 --> 00:32:48.649

Jisun An: So the. So here's how this gigram classify your works?

135

00:32:50.650 --> 00:33:19.549

Jisun An: and so there's 1 parameter that you need to care is so, how how many like context you'd like to see so how many, how far your, how big your window should be. There will be one parameter. But for now we assume that our window size is just 2. So what it means that so if given a target world, I would say the word next to me up to 2 steps. 2 hopes away are my context word, and they are the ones that I'm supposed to have close.

136

00:33:19.550 --> 00:33:25.234

Jisun An: that I want to make them close to each other, and any other words are likely to be

137

00:33:25.760 --> 00:33:27.410

Jisun An: negative examples.

138

00:33:27.895 --> 00:33:42.819

Jisun An: So the goal is, we want to train a classifier that is given a candidate pair. So given a word. And the context pair we want them to. We want to know whether they are likely to be context word or not.

139

00:33:44.900 --> 00:33:54.790

Jisun An: So we basically, we want to assign a probability that the probability that, given a 2 word whether they are

140

00:33:54.910 --> 00:34:10.429

Jisun An: the true context, true kind of relation. So whether they actually appear in the within the context window, and the probability that these 2 words are not likely to be neighbor to each other. So these are the 2 probability that we want to compute?

141

00:34:11.270 --> 00:34:12.620

Jisun An: Does it make sense?

142

00:34:13.040 --> 00:34:27.099

Jisun An: And we want like so in this example, the apricot and the gem they are. They were in the same sentence, right? So we want the probability of these 2 words to be higher up, so

143

00:34:27.270 --> 00:34:52.419

Jisun An: p plus applicant to be higher, and then p minus given applicant, and out of it to be lower, to be also higher, because it is also the correct classification. So we want those words that are not neighbor to each other, to be to have the negative label, and those words to who are neighbor to each other, to have the positive label. So we want both

144

00:34:52.710 --> 00:34:57.610

Jisun An: them them as a probability.

145

00:34:58.180 --> 00:35:07.900

Jisun An: But and to to be like a probability like this, the sum of these 2 will be one. So there will be like a the actual probability.

146

00:35:10.060 --> 00:35:11.180

Jisun An: So

147

00:35:11.370 --> 00:35:24.559

Jisun An: yeah, it's it. It may not be very easy to understand. But the the entire intuition of this model is that we base this probability with the embedding similarity.

148

00:35:26.730 --> 00:35:40.883

Jisun An: and in the embedding so so we want to kind of map the similarity of in the embedding space and then using those value to compute this probability.

149

00:35:41.880 --> 00:36:00.980

Jisun An: because computing this probability may not be really easy, as intuitively, we are kind of connecting these 2 words and then compute this probability, using the embedding similarity. So if we think about a similarity on the embedding space. Now, the 2 words are represented in the vector. Space. And then

150

00:36:00.980 --> 00:36:11.663

Jisun An: how you compute the similarity between the 2 vector is basically you are using the dot product. So the multiplication of the 2 vector these are the simple algebra. But

151

00:36:12.020 --> 00:36:28.429

Jisun An: probably you know that the cosine similarities is something that you've heard to measure the similarity between the 2 vectors. The cosine similarity is basically the normalized dot product. So here we are. Instead of using the cosine similarity, we are just using the dot products between the 2. Vector

152

00:36:29.440 --> 00:36:38.250

Jisun An: and so so whether the similarity of these 2 vectors can be represented as a dot products of the 2 vectors.

153

00:36:38.690 --> 00:36:39.780

Jisun An: And then.

154

00:36:40.030 --> 00:37:04.179

Jisun An: if and then, because these are the similarity value which are the just, some scholar. So we want to turn these value into the probability, and in doing so, one of the function that can turn the scholar value into the probability is the sigmoid. So we are simply taking a sigmoid to these dot products of the 2, vector and that will now turn into the probability whether

155

00:37:04.558 --> 00:37:09.861

Jisun An: I mean can can be interpreted as a probability of this pair of the

156

00:37:10.920 --> 00:37:17.720

Jisun An: to a word is in the neighbor, in the, in, in neighboring each other or not.

157

00:37:20.200 --> 00:37:43.330

Jisun An: So on the right side. You just see the sigmoid function, which is sigmoid function itself is just using like this. And then the idea, I mean, at the moment you can simply assume that sigmoid is a given X value. The Y output will be basically range from 0 to one. So sigmoid will turn any values into the probability.

158

00:37:44.950 --> 00:37:46.080

Jisun An: So

159

00:37:48.410 --> 00:38:02.380

Jisun An: so the probability of 2 wars, whether they are neighboring to each other or not, will can be estimated, based on the sigmoid of the thought products.

160

00:38:02.380 --> 00:38:25.110

Jisun An: and then we also want them, these, and then the sum of the 2 events. So some of the all possible events, should be one in the probability space, so the probability that the 2 words are not in the neighboring will be simply one minus p plus given Wc. And that will be just resulting in this equation.

161

00:38:25.190 --> 00:38:38.840

Jisun An: So I mean, I will not ask you to understand every single equations here. I'm just showing you how these things has been derived, and then how it can lead to the actual learning and etc?

162

00:38:39.060 --> 00:38:41.000

Jisun An: but I think yep.

163

00:38:41.810 --> 00:38:43.970

Jisun An: Any questions up to here? Yes.

164

00:38:47.980 --> 00:39:06.239

Jisun An: oh, so initially, all the vectors will be randomly initialized, and then we will learn. We will update these these parameters and the values. And yeah, thanks, thanks for asking, and I will. I will go into I will show you the steps of the how the parameters are updated as well.

165

00:39:07.884 --> 00:39:28.769

Jisun An: So once again the the key back to the key. So here we want to classify. Given a 2 words, whether whether they are actually neighboring or not, which can be interpreted as a probability for 2 words to be neighboring or not. And these

166

00:39:29.420 --> 00:39:37.020

Jisun An: probability is basically based on the the embedding similarities. because

167

00:39:39.100 --> 00:40:06.650

Jisun An: and we just want to use the embedding similarity to to base the probability. So we compute this probability, using the characteristic of the embeddings. So when the 2 words are similar to each other, in the embedding space, it can be computed as adult products. And then we just turn that, using the sigmoid, we just turn that to the probability, and we hope that this probability represents the actual probability of the 2 words to be neighboring to each other.

168

00:40:07.270 --> 00:40:13.289

Jisun An: That's the how that's the how the skigram model kind of designed for

169

00:40:20.130 --> 00:40:49.360

Jisun An: and now, this one was just for the one context value. And if because now as we for each of the words, we had one positive word, and we have multiple negative examples. So we if you consider like the multi. If you have, like the multiple context word, then you can simply multiply them. And here we are assuming, like the independence so now, instead of just the one word, you will have L context word, and then you can just

170

00:40:49.900 --> 00:40:56.130

Jisun An: we represent them as a like product of all these functions.

171

00:40:56.530 --> 00:41:14.619

Jisun An: And then, as I mentioned you before, when the so these each of the value, if you think about it, they are eventually the. After taking the sigmoid. This value will be probability, so the value will be ranged from 0 to one, and if you are multiplying like

172

00:41:14.620 --> 00:41:33.400

Jisun An: value that less than one, then the product of those value will become really, really small right? And that's the reason when, instead of using the the product value themselves. We are taking the look so that instead of multiplying these values, they can be now just some

173

00:41:33.400 --> 00:41:50.350

Jisun An: sum through them, so the local probability would be much easier to deal with. And also there will be. This is also relating back to the problem of how compute computer actually compute or do this process. So

174

00:41:50.350 --> 00:42:09.709

Jisun An: if you have more smaller value, then you will basically require more bytes and bits to represent those values. And that will be a problem eventually to the computer. So basically for you, you've heard here and there why, we are using the log. But then, basically, the idea is here. If you multiplying the small values.

175

00:42:09.740 --> 00:42:18.450

Jisun An: many of them, then that value will become really, really small. So they want to take the look and just sum them up so that you have like larger values.

176

00:42:19.050 --> 00:42:28.910

Jisun An: So this will be now the probability that that your context words are neighboring to your target words

177

00:42:30.634 --> 00:42:55.309

Jisun An: and and usually the the actual Gram model will be trained based on based on these 2 sets of vectors. So they they tend. They have 2 different sets of words. So I mean, vector sets. So one vector is for the target word and the other vector for the context and the noise word. So just so this is the way that how the speaker model was model was trained. So so one thing. So you, if the vocabulary

178

00:42:55.310 --> 00:43:00.560

Jisun An: size is the V, then your total number of parameters will be about 2 V's. Yes?

179

00:43:01.510 --> 00:43:04.140

Jisun An: Oh, no. Yeah.

180

00:43:04.750 --> 00:43:10.520

Jisun An: The window words. Remember those. Those are all marked as true examples.

181

00:43:10.740 --> 00:43:12.000

Jisun An: Is that how that.

182

00:43:12.180 --> 00:43:16.409

Jisun An: So when it's predicting? If this is one of the

183

00:43:16.950 --> 00:43:27.589

Jisun An: context words, each of those 4 words is the one that would be considered like a positive.

184

00:43:29.660 --> 00:43:51.459

Jisun An: So I was actually thinking, similar thing, and I haven't checked the paper. So what was the exact training? But so if the window is 2, then, basically, you have 4 positive example for each of the positive example. You would also have K negative samples, and then you are updating them all together at one training

185

00:43:51.460 --> 00:44:04.860

Jisun An: and then because you have 4. Now, it can be either you can consider them as a 4 different examples, or there are also possibility. They add all the 4. But I wasn't sure what was the actual training steps.

186

00:44:05.150 --> 00:44:07.970

Jisun An: Yeah, but all the 4 will be included

187

00:44:08.090 --> 00:44:11.710

Jisun An: as a part of the exam. The pairs of the examples.

188

00:44:11.950 --> 00:44:14.179

Jisun An: Yeah, but I don't know whether the updates are

189

00:44:14.460 --> 00:44:19.220

Jisun An: get done altogether or separately. That was something that I was also a little confused.

190

00:44:22.270 --> 00:44:23.170

Jisun An: Thank you.

191

00:44:24.530 --> 00:44:26.750

Jisun An: Any questions up to here.

192

00:44:28.610 --> 00:44:43.870

Jisun An: and also for each of the word word, now has the d dimension or vector. So here, usually they using like 500 to 12 kind of vectors. So you can d can be any number. But normally, I think it was 512,

193

00:44:47.080 --> 00:45:10.889

Jisun An: all right. So getting into more of this key gram training. So once again, given this particular example, the apricot is our target word, and then, when the window size is 2, the table of gem a will be our context word. So these 4 will be our positive examples, and for each of the positive example. We also sample like the K negative example.

194

00:45:12.230 --> 00:45:18.725

Jisun An: Here we are just seeing just 2 slk so the

195

00:45:19.902 --> 00:45:30.330

Jisun An: we basically label, the the those positive example as a 1 and those negative example as a 0. And we just run kind of logistic regression. Here

196

00:45:31.430 --> 00:45:54.020

Jisun An: and and then then. Now, so how the how do we learn how these vectors are really learned is through this learning mechanism. So so what we want is basically, we want to maximize the similarity between the target word and the context words and the minimize the similarity between the negative pairs. Right? So that's the obvious

197

00:45:56.380 --> 00:45:57.420

Jisun An: And

198

00:45:57.780 --> 00:46:08.770

Jisun An: now, yeah, oh, my God! If so, I I feel like there are so many things that I'm kind of missing to describe, but I hope that somehow you can get the concepts. So

199

00:46:08.770 --> 00:46:33.810

Jisun An: you probably also heard of these loss functions, and I think I intentionally give all these equations, because I think this is easy enough for you to understand all this so, but once again, I will not ask you to memorize any of these equations, so I will never ask. But I think this was really easy enough to exercise and and really learn what this loss function usually means. So

200

00:46:34.330 --> 00:46:35.916

Jisun An: so here,

201

00:46:37.080 --> 00:46:51.290

Jisun An: this loss function, and this is loss function for one word with the one positive word and the K negative words. And if you look at this equation that it literally describe

202

00:46:51.290 --> 00:47:16.119

Jisun An: what we want to maximize or the minimize. So so the 1st negative sign here. So this one is, we just take the negative because we want to. I mean, this is very computer science thing that we want to just minimize something. So we take negative so that value can be negative. But so I don't think that means a lot. And the log is now just changing.

203

00:47:16.260 --> 00:47:34.139

Jisun An: We are taking the log. So the important part is just just here. Right? So what so? And these are also at the same time code as like objective function. So we we want to maximize the similarity of the target function. And we just want to minimize the

204

00:47:34.820 --> 00:47:40.940

Jisun An: the similarity between the target and the negative value. So but but in in other words,

205

00:47:41.300 --> 00:48:06.279

Jisun An: so in classification problem, this is the correct answer. Right? So when we see the target word and our context, words that we want their label to be positive. And when we see our target words and the negative words, we want their their label to be negative. So we actually want both probability to be maximized, but as a result it will have more similarity between

206

00:48:06.280 --> 00:48:21.360

Jisun An: positive examples and less a similarity between the target word and the negative example. But if you think about it at the probability level here, we want to maximize the both values, because this is the correct classification. Right?

207

00:48:22.080 --> 00:48:23.460

Jisun An: Does this make sense.

208

00:48:25.450 --> 00:48:30.930

Jisun An: So once again, if if this was not correct, like clear, let me go back to

209

00:48:31.150 --> 00:48:36.719

Jisun An: here. So this is the probability where the when the

210

00:48:37.270 --> 00:48:46.830

Jisun An: given a target word and the positive neighbor the positive word meaning that it's neighbor word to be

211

00:48:47.020 --> 00:49:09.240

Jisun An: correctly classified as how they are neighboring to each other. And this is the probability is that, given a target word and given this negative word, which is not the neighboring word to be classified as a negative meaning that they are not neighboring to each other. So these are the cases in your classification when when they are correctly classified.

212

00:49:11.230 --> 00:49:23.350

Jisun An: Right? So we want these, both the probability to be high and and those these are sometimes we also call this a like objective function. So we want these 2 probability to be higher.

213

00:49:23.800 --> 00:49:28.130

Jisun An: And that's the way that how we kind of define this loss function.

214

00:49:28.510 --> 00:49:45.230

Jisun An: and by taking the log for the product of the probability, and also by taking the negative value. Now, our objective function just turning into our loss function and our the goal of the our model is to minimize this loss function themselves.

215

00:49:46.300 --> 00:49:53.070

Jisun An: So minimize the loss of function means that these 2 probability will be higher and higher as our model is trained on.

216

00:49:54.460 --> 00:50:23.770

Jisun An: and the follow is just how you can now, because we know that this probability can be represented as a sigmoid of the dot products between the 2 vectors. So we just change that, and then you will kind of live into that. So that will be how the the our actual log loss function of the model, and then what the loss function will do is test. I mean, your weights will be updated to reduce this based on this low loss function.

217

00:50:25.130 --> 00:50:34.179

Jisun An: And then how this was derived is starting from this simple equation, where it literally represents what you want to do in this model.

218

00:50:34.520 --> 00:50:52.200

Jisun An: So you have once again 2 words, and if they are neighboring to each other, you want to predict that they are neighboring to each other, and if you have 2 words that are not neighboring to each other. Then you want to predict them as a non-neighbouring to each other. And that's literally what this loss function is represents.

219

00:50:54.310 --> 00:51:13.601

Jisun An: Then how how can we learn this classifier? And then, yeah, this is already like all the kind of method. But using this stochastic, radiant descent. So if you have taken the machine learning courses. Then probably you're already familiar with but I will just describe this shortly.

220

00:51:16.170 --> 00:51:17.290

Jisun An: and

221

00:51:17.620 --> 00:51:24.860

Jisun An: so what we want to do is for every steps of the gradient descent. We will look at one particular example like

222

00:51:24.860 --> 00:51:49.499

Jisun An: a pair of the positive examples and the a few set of the negative pairs. And then we just update these weight values to be closer to each other and like moving further things away. So we already saw that the apricot and the gems are the neighboring words. So we want these 2 vectors are closer to each other, and then the metrics or the tours toy are some random

223

00:51:49.500 --> 00:52:10.370

Jisun An: negative example. And we want to just move these 2 words apart. And and in other words, if you think about the dot products of the 2 vectors. You want them. These 2 vectors. The dot products of these 2 vectors are increasing and the dot product of these 2 vectors, the target and the negative word

224

00:52:10.680 --> 00:52:16.290

Jisun An: are decreasing, so that will be what will happen for each steps of the gradient descent.

225

00:52:18.090 --> 00:52:23.249

Jisun An: So reminder of the gradient descent. So the gradient descent is

226

00:52:23.830 --> 00:52:43.799

Jisun An: we want to find. So here, these are the our weights, and then the current weight. And this is the future weight. And we want to basically update the current weight by this amount. And the direction is basically is the the reverse direction of the gradient of the loss function.

227

00:52:43.970 --> 00:52:49.749

Jisun An: and the magnitude will be just learning like by this learning rate.

228

00:52:50.715 --> 00:52:51.570

Jisun An: So

229

00:52:51.830 --> 00:53:05.299

Jisun An: I think it's the easy way to understand the gradient descent is. Look looking at this graph. And so I think if if we because these weight factors are already like n dimensional, so it gets really

230

00:53:05.450 --> 00:53:31.519

Jisun An: hard to visualize. But think about if we have only one weight. So maybe you can go back to the rule based method. If we have only one weight, 2 updates, then how? And also these weights are actually working independently to each other. So you can think them as a separate. So what the gradient descent we're going to do is assuming that this each of the weights are somehow contribute to the loss function. Right?

231

00:53:31.800 --> 00:53:55.610

Jisun An: So here, the Y value is basically our loss. And this loss function is now given, depending on what is our weight value. Our loss value will be also changing. But once again, our goal of the model updates will be lowering the minimizing the loss. Right? So we want to move. So, assuming that our current weight value is here

232

00:53:56.340 --> 00:54:16.149

Jisun An: and given w. 1, we want to move our w 1 to certain side, either to the right, also increasing or the decreasing. So these are also the values of from 0 to, let's say, like 100. So we want to change this wave factor either to decrease it or decrease it. But we just need to determine

233

00:54:16.150 --> 00:54:29.539

Jisun An: whether to increase or determining right, increase or decrease, and to determine that you can use the gradient descent. So what you do is at that moment given the loss function. So what is the

234

00:54:29.690 --> 00:54:58.779

Jisun An: the gradient of the loss given this of the w. 1, and which, in other words, this is the slope of the loss at w. 1. And what we found is that here the slope is the negative. Then now we know that by increasing the W value our loss will be decreasing. Right. So now, given this, we know that which direction that we need to take for this one particular weight to minimize the loss.

235

00:54:58.920 --> 00:55:11.246

Jisun An: and that's the the one. And then so once we find out which direction that we need to take, then the learning weight will be the strength of how much we want to change this weight, and and because of that

236

00:55:11.720 --> 00:55:17.479

Jisun An: this weight will be updated to minimize this loss.

237

00:55:17.500 --> 00:55:44.389

Jisun An: And this is just the one weight. But then, now, you have like weight. It's a weight. Vector so you have like multiple weights. And each of the vector elements in the weight will update according to the loss function. So every of the every vector element in the weight factor will contributing to the loss, and they will using the gradient descent, meaning that they will find the slope so like.

238

00:55:44.710 --> 00:55:52.070

Jisun An: what is the best direction to go for to minimize the loss, and then they will just update their value based on that.

239

00:55:52.460 --> 00:56:16.181

Jisun An: And that's the what this gradient is to do. And if you are formally representing it, then it will be looking like this. So we have the weight. And so this term will be the derivative of the loss function here. L is the loss function. So what this will once again, will tell is so given this our weight. Vector

240

00:56:16.680 --> 00:56:19.829

Jisun An: what is the direction that we need to take?

241

00:56:19.920 --> 00:56:43.800

Jisun An: And then this learning rate will be the magnitude. So how much we want to change. So we basically update this. And then we do subtract because we want to go in the opposite direction. In other words. So here, if the gradient slope is the negative, then we want to increase the W vector and if we are here, and if our slope is the positive.

242

00:56:43.800 --> 00:56:52.579

Jisun An: then we want to decrease the W. Vector so if you find the slope, then we want to actually move the reverse way to the gradient.

243

00:56:56.520 --> 00:57:01.949

Jisun An: And that's the how I mean, just the gradient, decent in general, any questions here

244

00:57:07.990 --> 00:57:11.570

Jisun An: and then, now. These, so

245

00:57:15.420 --> 00:57:35.798

Jisun An: so here that that we had the the loss function that we applied. So once again, the loss function started from these probabilities right? And then we probability can be computed by the dot product of the vector. And then we take the sigmoid. And this is the that loss function. And we now have

246

00:57:36.220 --> 00:57:59.880

Jisun An: So for each run. We had, like the positive words and the negative words. And then we have the target vectors. So we need to get the derivative for each of these variables? And then, if you so, we're not going to go through how you actually compute these derivatives. But then, if you just take these

247

00:57:59.940 --> 00:58:25.419

Jisun An: equation, and if you for each of C plus and C negative and the w, and you will get this very nice, well designed derivatives for each of these parameters. And these are the parameters that we are updating. So for each of the example, we have the vector of the positive example vector of the negative example and the target vector right? So we are basically updating those 3 weight vectors

248

00:58:25.690 --> 00:58:27.770

Jisun An: that we've initially saw.

249

00:58:28.910 --> 00:58:43.250

Jisun An: And then just adding to this equation to the gradient descent. You will just see this one and then so we start with the random. So we. So we had these 2 vectors. The the context words and the target target words.

250

00:58:43.250 --> 00:58:59.481

Jisun An: Right? So we had the entire 2 V vectors and we initialize them randomly. And then now we have different pairs positive and the negative pairs, and then you can imagine that one example, and we compute the loss with that value. And then

251

00:59:00.090 --> 00:59:09.110

Jisun An: and then, given that loss. We compute the derivatives. And then we update these vectors. And we just repeat that to update our models.

252

00:59:13.120 --> 00:59:27.949

Jisun An: Yeah, so I mean these, these are all about this training, this particular model. And I know that maybe some things are not completely understandable here, but I hope that you got the concept. And

253

00:59:29.000 --> 00:59:31.490

Jisun An: once again, I think you can.

254

00:59:32.997 --> 00:59:43.072

Jisun An: These are like simple enough example. And if you have some knowledge of like doing the calculus, and I think you can actually follow all this.

255

00:59:46.900 --> 00:59:51.720

Jisun An: there's equations, and then and then just driving. So it should be.

256

00:59:53.580 --> 00:59:55.730

Jisun An: and I don't know why these are not.

257

01:00:09.840 --> 01:00:10.710

Jisun An: oh, okay.

258

01:00:15.150 --> 01:00:17.500

Jisun An: right? And then, so

259

01:00:17.620 --> 01:00:41.669

Jisun An: that okay, we we just went through the detour of all the order. But the the goal of that. What we want to explain was that, okay? So through this model, we can get the embeddings where it represents the word, where it can also show the similarity between the words. But then, in the in the Skipgram model, we actually have 2 different sets of the vector for the same word, right? One for the context words and the other for the target word.

260

01:00:41.680 --> 01:00:53.789

Jisun An: then, so what were, how do we actually get the embeddings? And so basically, I mean, this is not proven. But in the in the paper they were just, I mean, just adding the 2 vectors.

261

01:00:55.765 --> 01:01:04.060

Jisun An: and then this represent represent the word I as the vector Wi plus ci, was the usual way to represent each of these words.

262

01:01:05.260 --> 01:01:34.480

Jisun An: So these are like a summary of this, how the skigram input betting. So we start from debris, random d dimensional vectors as an initial embedding. And then we take the positive examples and the negative examples. And then we train the classifier using the logistic regression and then ignore the classifier, and we only use the learned weight vectors, which then will represent some very nice word embeddings. That was the key for this word today.

263

01:01:36.730 --> 01:01:37.970

Jisun An: Any questions

264

01:01:42.390 --> 01:02:06.460

Jisun An: and this rejecting, embedding actually had a really interesting properties. And you probably heard it before. These embeddings kept really well the analogical relation of the word. So yeah, I mean, you once probably heard about this. So they found that these resulting word embedding can be

265

01:02:06.530 --> 01:02:17.160

Jisun An: really kept well, these relationships that can be identified by this parallel parallelogram method. So these factor, which is

266

01:02:18.550 --> 01:02:30.419

Jisun An: this, vector which is the king minus man. And if we just sum this vector, to the moment, vector, then it is likely to PIN. So somehow, this kind of relationship

267

01:02:30.910 --> 01:02:47.293

Jisun An: are supposed to be found from the language. But before this world to bank model there was no method. They were actually keep this analog relationship. Then they found that this was I mean, with this word embedding this vector computation was

268

01:02:47.680 --> 01:03:01.359

Jisun An: possible. And this is another example from the glove and glove is just another word embedding method, and they also found that this kind of relationship is very well captured from the embedding themselves.

269

01:03:03.400 --> 01:03:10.149

Jisun An: But the I'm sorry that I I think I this connections are not.

270

01:03:14.210 --> 01:03:17.159

Jisun An: I hope that they are coming back.

271

01:04:05.600 --> 01:04:10.689

Jisun An: Okay, let me try with this some other connector but the

272

01:04:11.000 --> 01:04:17.980

Jisun An: but it it turns out these relationships are not. Oh, oh, let me try to switch this one. Thank you.

273

01:04:18.630 --> 01:04:22.319

Jisun An: Yeah, definitely. Yeah, I think it's probably the connector is the issue.

274

01:04:26.980 --> 01:04:40.542

Jisun An: All right. Thank you. Yeah. So it it found out that the I mean it doesn't work for all the relationship, but for those frequent word and certain relationship were the only ones that were

275

01:04:42.380 --> 01:05:07.350

Jisun An: found a good relationship from these embeddings, but not the others. And and so, like understanding, this analogy is still the open area of the research. But there were many really interesting work research that, using the property of this embedding method. So here they trained embeddings for the text corpus for each of the decade, and they tried to

276

01:05:07.350 --> 01:05:13.109

Jisun An: that. How the semantic of the words were changing over time.

277

01:05:13.270 --> 01:05:40.211

Jisun An: So here they were, using the 30 million books and train embeddings for different decades. And, for example, here the if you look at the figure B, then the broadcast. In 1850 they were actually the word that is relating to the agriculture. So they were used for, like broadcasting the seed, and then the 9,000. They now became more relating to broadcasting in the news media, and

278

01:05:41.170 --> 01:06:00.986

Jisun An: and so on. So it is kind of past because it it has this virtue of those words that are like semantically, since synthetically similar to each other's, are nearby. So by examining the words that are surrounding those words, you are kind of aiming, I mean, can understand better how

279

01:06:01.610 --> 01:06:05.190

Jisun An: These words semantics has been changes over time.

280

01:06:05.490 --> 01:06:30.459

Jisun An: And and also this analogy kind of method also turn out embedding also reflects the cultural bias. So there's this paper that, looking at how like the different occupations are relating more with the aligning with the gender, and they also found that when asked, like the Paris, France, and the Tokyo, to the Japan. And and if you ask, like the father, doctor.

281

01:06:30.460 --> 01:06:43.410

Jisun An: and mother, than X. Was like a nurse or men to the computer program, and they were more associating the woman to the homemakers, and, etc. So this kind of cultural bias was also examined, using this word embedding method

282

01:06:44.090 --> 01:07:13.230

Jisun An: and also going beyond this analogy. They were now kind of define a semantic axis, and then they were map like the different entities, and see whether they also, including the meaning of the class here. So in this case they were mapping different sports aligned to these dimensions of the rich and the poor, and you can see that, like golf, is more closer to the rich and the like. The camping and boxing are more to the poor. So

283

01:07:13.420 --> 01:07:42.240

Jisun An: I think there was. There was this kind of the work that are trying to map different items and objects. I mean, this was particularly focused on the class, but I think this was a really interesting work, and and also this for the same word, it could be mean different in different communities. So, for example, a word soft in the like subreddits named my little pony could be like a soft poise, but then, if

284

01:07:42.240 --> 01:07:56.249

Jisun An: the word in dissolved in the sports, then this would be actually the negative meaning. Right. So you can also exploit this. Differences of the word semantics in the different community can be also exploited. Using this method.

285

01:07:57.220 --> 01:08:21.119

Jisun An: and interestingly, word to back method themselves also can be applied to, not just looking at the understanding, the text themselves. It can be also applied in other domains, for example, building a recommendation system. So if you think that the history of all the purchases of the different products are a sentence, and each of the product is a word, then you can apply

286

01:08:21.120 --> 01:08:29.700

Jisun An: model to find out which products are more likely to be purchased together. So this can be turning into like a recommendation system.

287

01:08:30.390 --> 01:08:54.279

Jisun An: And also there has been work that are so, for these are information of how one person have migrated into different institutions. So for one person like and different institutions. So, starting from your high school to like your career, you will see the trajectory of these institutions that you are moving on, and consider them as a sentence and each institution as a word.

288

01:08:54.279 --> 01:09:11.350

Jisun An: you can. And also if you train your word to back model using this data, then, now, you will see the relationship among these institutions. And they found that I mean, obviously, there's a regional clustering right? I mean, if you I mean, you're more likely to move from one within.

289

01:09:11.350 --> 01:09:28.170

Jisun An: within the same country institution, in the same country, but then within, if you are looking now deeper for each of the region, then now you see that there's also regional clusters, and also there are like topical clusters, and also like press, more prestigious and less prestigious schools, and etc. Etc. So

290

01:09:28.340 --> 01:09:54.459

Jisun An: this kind of mapping is also possible. And and finally, these were not technically used to work to that. But the methodology was very same, and just wanted to let you know that other concepts, like belief also, can be represented in the embedding space. And all these were following the similar ideas of finding the positive examples and the negative examples and then represent them on the vector space using similar methods.

291

01:09:55.120 --> 01:09:56.250

Jisun An: And

292

01:09:56.710 --> 01:10:10.220

Jisun An: so that was the what i. So the next one will be the neural network, but I don't think I will be able to get to there, so I will probably stop here. But any question before wrapping up. Yes.

293

01:10:13.580 --> 01:10:14.906

Jisun An: semantic access.

294

01:10:15.960 --> 01:10:24.710

Jisun An: So here, I mean, there could be different method. But here, if you have 2 sides, then you have.

295

01:10:24.960 --> 01:10:35.750

Jisun An: You need like seed word to define the semantic. And then you take the average of those vectors and you assume that that's the center of that semantic.

296

01:10:35.830 --> 01:10:57.480

Jisun An: So that's that's the manual effort. Yeah, I mean, I mean using like a lexicon, I mean dictionaries and synonyms. Antonyms. Yeah, that's the manual efforts. But then, at the same time, if you are using like the word to back models, you may be able to find those words as well, so I think there could be, but but definitely. That's all manual efforts.

297

01:10:57.810 --> 01:11:26.020

Jisun An: Then once you find the seed words, then take the average and then take the subtract from one to the other. Then that's became the semantic access. And but these embedding space are trained for different corpus. So if you are training building a word embedding for my little pony community. You basically collect all the text from the my little pony subreddit. And you use that model to train the word to back model. And from that model you define the semantic axis.

298

01:11:26.630 --> 01:11:27.360

Jisun An: Yes.

299

01:11:29.290 --> 01:11:30.830

Jisun An: Any other question. Yeah.

300

01:11:32.550 --> 01:11:44.210

Jisun An: Oh, so that could be. All the vocabulary that you have in your training data?

301

01:11:44.970 --> 01:11:57.379

Jisun An: No, no. So these are like the random vectors initializing the random vectors. But the the vocabulary can be coming from the training purpose.

302

01:11:58.350 --> 01:11:59.859

Jisun An: You mean the vocabulary.

303

01:12:00.070 --> 01:12:05.120

Jisun An: The vocabulary will be the yeah in the entire vocabulary from the training corpus.

304

01:12:07.900 --> 01:12:30.130

Jisun An: Oh, so there are 2 sets of vectors. One is for the target words, and the other is for the context word. So even for the same same words, we'll have 2 different vectors, and they will be updated separately. But then, when you are using, you will just combine and then add and then use it.

305

01:12:30.390 --> 01:12:31.070

Jisun An: Okay.

306

01:12:42.580 --> 01:12:49.659

Jisun An: alright, yeah. Thank you so much for those who are a little late. Yeah. The

307

01:12:50.297 --> 01:12:57.279

Jisun An: yeah. The password today for is the word, I think. Mark your attendance. I will see you on Thursday. Thanks a lot.

308

01:13:02.300 --> 01:13:07.920

Jisun An: hey? Thank you. I think I may not have the June

309

01:13:08.400 --> 01:13:11.960

Jisun An: recordings. Oh, oh, they do have okay.