WEBVTT

1

00:00:05.620 --> 00:00:14.439

Jisun An: Thanks for joining today's passcode is attention. Please mark your attendance. We have a few announcements today.

2

00:00:15.230 --> 00:00:22.750

Jisun An: So so, firstly, our 1st theoretical assignment will be released later tonight.

3

00:00:23.420 --> 00:00:25.969

Jisun An: You can check tomorrow because I will already like

4

00:00:26.200 --> 00:00:44.430

Jisun An: very late. But anyhow, so that will cover I mean, these are just multiple choice questions. It will be about 25 each. But I'm still finalizing it. So the number may not be really correct. This will cover everything that we have done so far till today.

5

00:00:45.270 --> 00:01:04.399

Jisun An: And I mean, so I mean, the questions are easy. It is just there for help you to review the contents that we have in doubt, and some of the major concepts that we have talked with and hopefully through this piece. You can understand better each points that we we talked about.

6

00:01:04.890 --> 00:01:17.170

Jisun An: the. So so these are really like for helping you to understand better. So you can submit multiple times so, and your finer submission will be your score.

7

00:01:17.450 --> 00:01:34.520

Jisun An: which which means that in case you want to resubmit it, just make sure that you copy everything, copy or the answer, so that you can just re-enter them like without much hassles. Otherwise you need to resolve them all. But anyhow, so they are relatively simple, and you will see them soon.

8

00:01:34.970 --> 00:01:55.750

Jisun An: The only thing, so that you will be next Sunday. You have plenty of time the late submission will have a window of 24 h, but you will receive 50% of credit. So, but obviously, if you have some emergency email, me, then I will try to take that into consideration. So no worries.

9

00:01:56.373 --> 00:02:03.640

Jisun An: But then, yeah, so that'll be something upcoming. So check again tomorrow. I will probably very late today.

10

00:02:05.160 --> 00:02:23.720

Jisun An: and another thing is about team formation. So if you go to in the canvas you will see the team roster link. Then I will share later. But basically, we will have about 10, I mean, until next Sunday. We will have some time to form the team for the final projects.

11

00:02:24.077 --> 00:02:49.469

Jisun An: So basically, each team should have one to 5 students, though I direct, I recommend forming team of like 3 to 5 members, so that you have like less working loads and also having some collaborative experience. And also there's usually more ideas coming up within more people. So so once you find a team member. Then you can update the spreadsheet that I shared there with your team name.

12

00:02:49.890 --> 00:03:05.550

Jisun An: But you can also like leave a note there so that maybe others can find you based on their interest. Or if you think that you cannot find any teammates, just leave Random, then I will try to like, assign you to some random group

13

00:03:06.026 --> 00:03:14.489

Jisun An: and also you need to find the team name, and you can pick any unique name. But I mean, you will basically use this name for the rest of the project. So just

14

00:03:14.570 --> 00:03:42.130

Jisun An: think it wisely, and the due is also 17.th So the team roster, if you go to canvas. So this is the home page. If you scroll down a little bit. There's this small section project which I just edit, and here is the team roster. If you click them, you will gonna see this spreadsheet. So these are all the students that currently enrolled in this course. So here, basically, you can just write your team name

15

00:03:42.785 --> 00:03:51.580

Jisun An: here, once you form your team. If not, leave a note to them, we will try to. Randomly. Group you for a team.

16

00:03:56.110 --> 00:03:59.260

Jisun An: Any any questions about this information?

17

00:04:00.500 --> 00:04:08.480

Jisun An: Okay, today's passcode is attention. Please mark your attendance if you haven't done so, and

18

00:04:08.900 --> 00:04:25.739

Jisun An: most of the informations are here in the home page. So you can check. And one other thing is, I mean, it's not mandatory. But I'm also updating reading material here and there. I mean, most of them are basically the research papers that like

19

00:04:25.800 --> 00:04:50.459

Jisun An: most foundational nap papers that lead to the large language model. So they are definitely interesting. And if there is a particular topic that you want to dig deeper, then I would recommend reading them. But they are not mandatory, and it will not be part of the any of the exams. I mean them solely will be based on our slides, so you don't need to look for, but in case, especially for the last week, we have some

20

00:04:50.460 --> 00:04:59.759

Jisun An: links toward to a textbook. So if you need some fundamental understanding of each of these concepts. This textbook might be also helpful, and also I mean.

21

00:04:59.910 --> 00:05:05.770

Jisun An: so I mean, you may check them time to time. If you feel that you need to look in them

22

00:05:10.130 --> 00:05:19.000

Jisun An: all right. We have a lot to cover today, but it's a very exciting topic. Attention and transformer is something that we will talk today.

23

00:05:21.840 --> 00:05:32.030

Jisun An: so we were at the last topic that we discussed was the sequence to sequence model, and in particular we introduced

24

00:05:32.110 --> 00:05:55.889

Jisun An: 3 models, Rnn. Stm. And the convolution models, which are all. It's unfortunately not very actively used in the Nlp. And even though they are used in some other domains. But the Rnn. Is not very commonly used in the Nlp domain, but so eventually the transformer which you see the very popular architecture that is actively using used in the modern Rnms.

25

00:05:56.210 --> 00:06:08.489

Jisun An: Is also the efforts of trying to resolve some of the issues that Rnn. Had. So we will start to talk about what was the what's the limitations of the Rnn. And how the transformer is tackling them.

26

00:06:09.300 --> 00:06:32.859

Jisun An: So this is the visualization of the Rnn. And as once again. Rnn. Was the sequence to sequence model. And so they were overcoming the word based models, and they were now trying to understand better about longer. I mean the the words that is at least like the further away from a context window, or trying to understand the structures or the syntax of the tension

27

00:06:33.237 --> 00:06:40.792

Jisun An: but they think the core of the argument was that so? Basically, it encodes each of the token and

28

00:06:41.320 --> 00:07:10.229

Jisun An: And and then it has this sequential kind of processing. So the the output of the 1st hidden unit became the input of the next hidden unit. So you basically process token by token from left to the right to encode this sentence. And then the rest of the training is actually, I'm basically the same. So this would be like, next token, prediction, task, or particular classification task, and then, based on that, you can train the Rnn.

29

00:07:10.810 --> 00:07:20.901

Jisun An: But the key is that so? Rnn. Was the one of the 1st model that will try to encode the sentence and trying to understand the structure of the sentence by using the sequential modeling.

30

00:07:22.040 --> 00:07:23.010

Jisun An: then.

31

00:07:23.200 --> 00:07:51.650

Jisun An: obviously, the Rn has some limitations. And one of the biggest limitation is basically it reuses the output of the previous token as an input meaning that basically, it's not possible to do the parallel processing. So you need to process and compute one by one, because for the next hidden unit you cannot do any computation before you get the input from the previous node, right? So basically you need to sequentially process the computations.

32

00:07:51.670 --> 00:08:13.009

Jisun An: And so and and and this and because of this concert processing. The pro like training process is just very, very slow, and that prevents to like scale up to a larger data. Even though you have powerful computation tools because you need to go through one token by one. Basically, it was very, very slow.

33

00:08:13.290 --> 00:08:16.590

Jisun An: And also the parallel computing was not possible.

34

00:08:17.020 --> 00:08:42.609

Jisun An: And another significant limitation was that as because these are like sequentially processing, if the the networks getting deeper, meaning that if the input synchrony is getting longer. Then information from the earlier token basically disappeared. And they were like, were not really affecting to definer course. So it was very so basically 2 problems. So you are not able to find the good.

35

00:08:42.610 --> 00:09:00.010

Jisun An: how the words in the long distance are interact with each other. And also how this the prior, I mean, the only earlier tokens are impacting the goal or the loss function of all the objective of the function of the modem. So basically they were not very good at handling, like the long sequences.

36

00:09:01.030 --> 00:09:05.940

Jisun An: and they both lead to overall like performance degrades in the language model.

37

00:09:06.310 --> 00:09:27.210

Jisun An: And if you wanted to add more and deeper layers to improve the performance. Then, basically, your like, your network gets deeper and deeper, and this will result in like the gradient vanishing that we talked about. So I mean, you can. You can think it as the layer gets longer. Basically, the layers in the initial

38

00:09:27.360 --> 00:09:47.049

Jisun An: layers. Their influence gets just ignored, and that leads. And also, if you think about how we compute the gradients, the gradients was essentially the multiply, the products of the other functions. Right? So if you have like a longer layer, then your gradient will have

39

00:09:47.050 --> 00:09:58.890

Jisun An: go through multiple multiplication, and that will lead to like smaller and smaller value, so especially for the all the layers, their influence will be become really, really minimal, which is the gradient vanishing.

40

00:09:58.890 --> 00:10:23.810

Jisun An: But then, maybe what solution of this gradient vanishing. If the value of getting smaller is the problem, then you can actually change the range and then just make them to a larger value for each of the gradients. But then that will lead to gradient exploding now because you were multiplying all these values, so if each of the layer value has a big like large value, then the multiplication of them will be now too large, and that will lead to 2 gradient

41

00:10:23.810 --> 00:10:29.889

Jisun An: exploding. So these are the 2 common problems in the like net neural network.

42

00:10:30.502 --> 00:10:38.720

Jisun An: Modeling. But but so, but so this was like unfortunately inherited in the Rnn. And that was not avoidable.

43

00:10:39.990 --> 00:11:08.409

Jisun An: So so transformer is came to rescue for some of these problems, and many of them. And the core innovation was the self attention which we will really dig deeper today. But basically the idea is basically it replaced the sequential input processing into something else so that it can be computation in parallel. And also it can be efficiently computed, so that the execution is very fast.

44

00:11:09.170 --> 00:11:25.350

Jisun An: And and the the key idea of this self-attention is, they calculate the relationship between words and towards just their representation. This may not be very, very understandable at the moment, but we will go in that bit later. But that was the core innovation.

45

00:11:25.930 --> 00:11:28.322

Jisun An: And so basically,

46

00:11:29.300 --> 00:11:51.199

Jisun An: transformer made it possible to be scalable. So I mean, once again, before knowing the transformer, this may not really make much sense to you, but it became it. Let the model to be scalable. So now, even though you have multiple blocks which can in compass like diverse information and of the language.

47

00:11:51.200 --> 00:12:01.830

Jisun An: The model didn't have this problem of like gradient distance or the vanishing or the exploding. So it basically made it possible to train with the much larger data

48

00:12:02.420 --> 00:12:07.202

Jisun An: and also more efficient. So the now it's not using the sequential

49

00:12:08.170 --> 00:12:18.940

Jisun An: input, so the parallel computation maybe was possible to do, and that really shortened the training time. In other words, you can basically train with larger data in the same time.

50

00:12:19.390 --> 00:12:23.510

Jisun An: So being more efficient. And lastly, it also handled the Long inputs.

51

00:12:23.950 --> 00:12:42.040

Jisun An: So even though with the long sequences, it maintained the performance. So so basically, those were the 3 limitations of the Rnn and transformer basically tackle each of them so that it really became it enables the large language model. So now the model can be trained with the larger data

52

00:12:42.040 --> 00:12:56.859

Jisun An: and also transformer has been the backbone of the modern era. Once again, I mean this, even though it doesn't make much sense. It's okay because you don't know what the transformer is right. But at the end of this class I hope that this slide, if you come back to here, then it all makes sense

53

00:12:58.170 --> 00:13:11.319

Jisun An: right. So the transformer in particular, was introduced in this very famous paper, attention is on your needs, which is from the Google publicity in 2017

54

00:13:11.838 --> 00:13:27.310

Jisun An: and probably I mean, you probably seen this picture somewhere. It's a very famous architecture, and you also probably know some parts of it. But we will go through each of these, the core architecture and the core component of this transformer

55

00:13:27.520 --> 00:13:32.439

Jisun An: architecture. And hopefully, you can understand what this figure means

56

00:13:32.868 --> 00:13:52.860

Jisun An: so initially, this was, introduced to as a sequence to sequence model based on the attention. And here, they tackle the in particular. In Ap test the machine translation, and that's the reason that they have. So so in this figure, the left side is the in this architecture. The left one is the encoder where they

57

00:13:52.860 --> 00:14:16.699

Jisun An: process the inputs and the the right side is the decoder where they process the output so the fact that it has the input and output. It means that they are not the auto regressive model, but they are like task specific model, right? So the particular this particular paper was introduced to tackle the machine translation. So they do have an input and the output. And they, the architecture, were supporting for that

58

00:14:17.951 --> 00:14:35.478

Jisun An: and they basically was very, very fast, because they if I mean later. Once again, if we go through each of the these modules, you will know that all these computations are simply multiple. I mean matrix multiplications. And there's no other fancy kind of

59

00:14:36.463 --> 00:14:48.960

Jisun An: computations. And that made it fast. So it's so from this moment from the attention. This model were really enabled to train it with the larger model, larger, larger data.

60

00:14:51.320 --> 00:15:12.899

Jisun An: So and also the I mean. So you need. So the center one, the encoder decoder model. Is this something that was introduced in the this attention paper? But there was some variance of this architecture. So there are some models that are only using the encoders, and there are also models only using the decoders. And once again, I mean, if

61

00:15:12.900 --> 00:15:24.819

Jisun An: you will, probably it will probably look the same, and you may not be really figuring what's the differences. But after after today's class, I hope that you can understand. What's the differences between these 3?

62

00:15:24.930 --> 00:15:42.200

Jisun An: so I will come back to these 3 types of the transformer. But today, for the rest of the course, I will use a decoder only model, which is the this is the architecture that is popularly used by the modern Rrms. Like Gpt and Dillama, and then many other things.

63

00:15:42.350 --> 00:16:01.829

Jisun An: So I will just use that diagram to go with, but essentially like, if from the decoder and the encoder decoder model. If you remove this encoder part and the the part where the inputs from the encoder coming into the decoder part. If you just remove them, then you will get this decoder model.

64

00:16:03.750 --> 00:16:24.499

Jisun An: So so this decoder model has a few components. So, and these are the core concepts that we will go through position encoding attention masks to attention, register and the layer normalization and defeat for the layer. So once you understand, each of these modules just combining them together became the transformer.

65

00:16:25.580 --> 00:16:28.529

Jisun An: So let's start with the position or encoding.

66

00:16:31.620 --> 00:16:32.720

Jisun An: So

67

00:16:32.820 --> 00:16:41.369

Jisun An: so before the positional encoding. If you look at the the 1st part of this architecture is, I mean, basically, we are having given an input.

68

00:16:41.880 --> 00:17:06.000

Jisun An: we need to embed this. Input, so I hope that you know what is the input embeddings. Now, basically, I mean, most popularly, these inputs are split, based on the subword models, and like the byte pair unigram or the word piece, these are some of the common sub models that was used in across, like the different logic models.

69

00:17:06.569 --> 00:17:26.060

Jisun An: And then you assume that there are some embeddings, and you looked up, and for that input, you just get some embedding. I mean to be honest. This can be, can be, one hard encoding or something that which are tunable as well. But I think at the moment that may not be very important. But given an input, basically, you'd

70

00:17:26.800 --> 00:17:31.790

Jisun An: represent them based on the vector representation so that it can go into the rest of the model.

71

00:17:32.840 --> 00:17:43.833

Jisun An: And then the positional encoding is this something that is coming into the network? With this the the plus sign is literally means that they will be additive

72

00:17:44.500 --> 00:17:49.675

Jisun An: and the reason that we have the positioner encoding is

73

00:17:50.500 --> 00:17:55.600

Jisun An: Unlike the Rnns, the transformer will process all inputs simultaneously.

74

00:17:56.000 --> 00:18:13.169

Jisun An: And you can assume that now, you have like cpus, and you may be able to just call multiple function at the same time doing parallel processing. So you you should assume that you can really process this inputs at all of them individually at the same time.

75

00:18:13.800 --> 00:18:31.600

Jisun An: But then, if you are doing that, then you will lose the order information right? Because you are like processing them at the same time. But then order is basically critical in text, right? So the position encoding wanted to keep that information. So where the words were positioned in that sentence, the idea is that we simple? Right?

76

00:18:31.770 --> 00:18:32.820

Jisun An: And

77

00:18:33.150 --> 00:18:48.319

Jisun An: so if you are just using the embeddings, basically, there's no way that that you can distinguish between these 2 identical words like a big dog and a big cat. We basically got big the word in terms of the word embedding or their representation, it would be basically the same.

78

00:18:48.900 --> 00:19:00.670

Jisun An: So so the idea is assuming that you have the embedding for that word in particular. But you are adding the positioner embeddings for that word given, I mean, based on their position information.

79

00:19:01.000 --> 00:19:08.680

Jisun An: So the key point is that, how do we know, I mean, what, what, how to create the positional encodings or positional embeddings?

80

00:19:09.790 --> 00:19:18.649

Jisun An: So the initial idea, especially in the transformer paper. They use this sinusoidal encoding.

81

00:19:19.605 --> 00:19:32.029

Jisun An: So you don't need to understand this equation. So for now you are just given a position of your token. Given a sentence, then you just use this formula to compute the value

82

00:19:32.110 --> 00:19:54.060

Jisun An: and then use it as a position encoding. So I mean, that's enough. But then the key idea is that, and also at the on. The this graph shows that the positional encoding is computed, based on the 2, value the position themselves and the dimension themselves. So the on the right side. That is the example. Figure how this this, the value. The

83

00:19:54.990 --> 00:20:01.859

Jisun An: these encoding values are changing for each of the dimension and given given its position. So

84

00:20:02.640 --> 00:20:17.389

Jisun An: the the whole idea is that even though you don't know the exact. The whole idea is that you just created a function where, if you are nearby, then the the multiplication of the the top product of the embeddings should be just

85

00:20:17.770 --> 00:20:24.480

Jisun An: larger, so that you know that that, given the embedding, the 2 positions are close by. So that's the whole idea. Yes.

86

00:20:27.950 --> 00:20:30.409

Jisun An: dimensions, dimensions of the embeddings.

87

00:20:30.630 --> 00:20:33.869

Jisun An: Yeah. Token embeddings. Yep, okay.

88

00:20:33.950 --> 00:21:01.929

Jisun An: So I mean, because it's a single side. It goes like downs and ups and downs and ups right. So if you are in the the 1st and second, basically, the 2 value from the one dimension will be slightly different. But it will still similar, right, because the sine function and cosine function is kind of going like decreasing or increasing. If the word became now further away, then basically, the value will one value in one dimension become like different, and then that will lead to like the different value there.

89

00:21:02.090 --> 00:21:07.210

Jisun An: So that is the the idea of this synocide encoding any question.

90

00:21:09.230 --> 00:21:16.929

Jisun An: But once again, I mean this occasion itself. You don't need to understand. The key, and the key idea is just.

91

00:21:17.830 --> 00:21:20.640

Jisun An: If given the position of the token.

92

00:21:21.280 --> 00:21:29.439

Jisun An: If you just give that information to this function, it will return some value, and that value will be your position or encoding.

93

00:21:29.750 --> 00:21:31.056

Jisun An: and then

94

00:21:35.820 --> 00:21:51.929

Jisun An: I'm sorry. This the dimension is not the token embedding, but it's the positional embedding. Yeah, so sorry. I think I won't miss that. So I mean, if here, if we assume that the positional encodings are 6 dimensions, then these are the 6 dimensions. But it can be any dimensions. Yeah, yeah.

95

00:21:57.380 --> 00:21:58.455

Jisun An: so

96

00:22:01.660 --> 00:22:20.649

Jisun An: But then and there could be some another idea. I mean, basically, the position embedding can be also learned right? So you don't need to really think I mean. So this one is in a way that you just need to know the position of the token, then it will compute the exact value. So every 1st token of the sentence will have the same position, or encoding

97

00:22:22.290 --> 00:22:28.090

Jisun An: once again the 1st token of a sentence they will have the same positional encoding.

98

00:22:28.430 --> 00:22:41.220

Jisun An: because these are just one occasion, and then there's nothing more. But then, I mean, you can also let it learn from the model. So, depending on the positions. I mean, once again using the neural networks magic.

99

00:22:41.220 --> 00:23:06.459

Jisun An: you can learn like the Embeddings themselves. But once, basically, they did also try this, learn the encoding for the positions, but and then it became. It can be quite flexible, because it is not just, I mean, based not on the one particular equation. But then they found that it's impossible to exploit to the longer sequences, meaning that. So, assuming in the training data, you had certain tokens to maybe like.

100

00:23:06.760 --> 00:23:16.710

Jisun An: I don't know, like the length of like 50. But if in your test, if you have like a longer sequence, then basically, their value will not be encoded properly.

101

00:23:17.220 --> 00:23:27.930

Jisun An: because in the because learnable learnable coding means that you are limited by your training data. So if you in the test data, if you have longer sequence. Basically, this will not work.

102

00:23:28.330 --> 00:23:32.489

Jisun An: So it's basically the learned encodings are not really used in their states.

103

00:23:32.790 --> 00:23:47.609

Jisun An: And the the previous 2 encoding methods that I just introduced. They are the absolute encoding meaning that the absolute position will position encodings will add an encoding to the inputs in the hope that the

104

00:23:47.890 --> 00:23:54.770

Jisun An: related position will be captured. So the absolute position mean encodings once again means that

105

00:23:54.970 --> 00:24:03.250

Jisun An: if you, even for the learnable encodings, once you learn those embeddings. Given the position of the token, you will have the same embeddings.

106

00:24:03.410 --> 00:24:10.808

Jisun An: So for the single size. So the leader sorry, and the second one, the loan number encodings both.

107

00:24:11.540 --> 00:24:23.400

Jisun An: given. The only. The only matter is the position of the token in the sentence so basically, they will have the absolute value for the positional encoding, and So

108

00:24:23.620 --> 00:24:28.900

Jisun An: in the synosotio encoding example. So even though

109

00:24:28.900 --> 00:24:54.199

Jisun An: they didn't, I mean, they were kind of designed in a way. But even though they are using the absolute values, basically, if they are close by, then the the embedding stock products will be also higher, which means that some similarity between the 2 embeddings will be also higher, so they will capture this relative distance between the words. So this, even though they are using the absolute positional encoding. They know that the the

110

00:24:54.420 --> 00:25:00.260

Jisun An: tokens that are positioned closely need to be also have higher similarity in the embedding space.

111

00:25:01.070 --> 00:25:19.019

Jisun An: But then, now, instead of using the absolute value. You may also need to exploit relative positions of the tokens. And this relative positional encodings explicitly encode relative position. So let's assume that you have these 2 different examples. The black cat eats

112

00:25:19.020 --> 00:25:40.699

Jisun An: food and drinks water. Yesterday morning I saw a black cat, so here both sentences has a black cat, and if we are using the absolute positional encoding, then the black cat in the 1st sentence, and the black cat in the later sentence will have different positional encoding. But what is most important thing is that black and cat are like, basically they are

113

00:25:40.890 --> 00:25:55.909

Jisun An: having. They are next to each other. Right so, but but in the absolute positional encoding these 2 will have different positional encoding because they are positioned in second and 3.rd The later they will, in the second sentence they will have 6. They they are in the 6th and 7.th

114

00:25:56.500 --> 00:26:01.850

Jisun An: So now they developed some proposed the relative positional in coins.

115

00:26:02.190 --> 00:26:07.680

Jisun An: and one of the popular method that is used is the rotary positional encodings.

116

00:26:07.740 --> 00:26:33.599

Jisun An: And the fundamental idea is basically they want to. It's similar. So they want to have some occasions, so that the dot product of the 2 positional encoding will be the higher for a for those tokens that are next to each other once again. You I also not 100% sure how we compute how and use these equations. So I wouldn't let me ask you to all understand all of them.

117

00:26:33.840 --> 00:26:42.129

Jisun An: But here's the the intrusion of this auto PE methods. So assuming that we have black cat, and these 2 are now

118

00:26:42.990 --> 00:27:09.809

Jisun An: represented in a vector space right? And then the in in the basically, this method will add or move this vector by by certain degree, based on their position. So we know that in the 1st set in the first, st this sentence the black cat was positioned in second and 3.rd So we are rotating this vector by 2 theta, and for the cat like rotate by the 3, set theta.

119

00:27:09.940 --> 00:27:26.320

Jisun An: But then for the. So this is like the the how the rope will work, but then in. So if we are applying this method for the second method, so because now black cats are the 7th and the 6th and 7th tokens. So we will rotate it for 6,

120

00:27:26.550 --> 00:27:30.629

Jisun An: theta and the 7, th theta, 7, th theta for the cat.

121

00:27:31.030 --> 00:28:00.770

Jisun An: But so, even though the vector will kind of change. We still know that the dot product between the 2 will be exactly the same for the black cat in this 2 example, in the same, even though they are in different position in a sentence. So that's the how they encoded the relative positioner informations in through the embeddings. And once again, if you come back to now, this equation, they created this equation to enable that intuition that I just showed

122

00:28:04.210 --> 00:28:08.540

Jisun An: make sense any questions.

123

00:28:10.380 --> 00:28:13.120

Jisun An: Yes, very short.

124

00:28:14.180 --> 00:28:26.210

Jisun An: So now your inputs token by tokens are feeding into the network. So firstly, it changes it to the word embedding

125

00:28:26.602 --> 00:28:44.870

Jisun An: and then each of the word will add it to the positional embedding. And so that will be the resulting embedding that this is visualizing, so you can see that as a word embedding at plus positional encoding position embeddings. And this will be resulting into that. The

126

00:28:44.980 --> 00:28:59.730

Jisun An: yeah, I mean, so rotate rotate means, yeah. So some yeah, rotate could. Yeah, change the values in in multiple. I mean in some kind of summations. Yeah.

127

00:28:59.880 --> 00:29:03.660

Jisun An: And that function that I showed you before enables to do that.

128

00:29:03.940 --> 00:29:21.340

Jisun An: But so the simple information is that I mean simple facts that you need to understand the token. Each token will have their own embedding that presents some kind of semantics. And then on top of that embedding the position of embedding will be added, and that will be the input to the rest of the model.

129

00:29:21.850 --> 00:29:34.819

Jisun An: So the the changes of this vector that you show is the what we are going to happen initially, their token embedding. And they will change it to that after incorporated with the position embeddings.

130

00:29:40.710 --> 00:29:44.489

Jisun An: Alright! So that's the positioner encoding part.

131

00:29:44.860 --> 00:29:48.799

Jisun An: Now I will move on to the attention. So the

132

00:29:49.090 --> 00:29:59.249

Jisun An: so you you probably heard attention many times, but then attention is essentially try to mimic, like our how we process the context in the sentence.

133

00:29:59.890 --> 00:30:06.579

Jisun An: so consider this sentence. That we are. I must or the other world. Then what would this bank mean?

134

00:30:11.060 --> 00:30:17.610

Jisun An: My bank means different things right? It can be financial institution, or it can be the land near a river.

135

00:30:18.338 --> 00:30:31.569

Jisun An: And without context, basically, we don't know to determine what it actually means. But what if we have? He said, by the river bank. And now we know that bank refers to a land nearby near a river.

136

00:30:31.810 --> 00:30:40.310

Jisun An: and we know that because of the surrounding word right? So human don't interpret word in isolation. But we drive the meaning from the surrounding words

137

00:30:41.430 --> 00:31:07.069

Jisun An: and the attention mechanism. The basic idea is basically they try to determine. So given the given the fact that the surrounding surrounding words determine the meaning of the word, the attention mechanism, try to compute, to determine which word to attend to accurately interpret a word within its context. So that's their pure goal of the attention.

138

00:31:07.760 --> 00:31:16.849

Jisun An: So in our example, he said, by the river bank. So we want to know and accurately interpret a word bank.

139

00:31:17.372 --> 00:31:22.020

Jisun An: and then we want to now determine how these all the other surrounding words.

140

00:31:22.484 --> 00:31:27.089

Jisun An: are more important or less important to determine the meaning of the bank.

141

00:31:27.330 --> 00:31:35.039

Jisun An: So the basically we want to compute where which surrounding words to attend, give attention to

142

00:31:35.677 --> 00:31:46.159

Jisun An: to understand better of this world bank. So that's the what attention mechanism is trying to do. And in our second example, I mean, so in in this example. And here the

143

00:31:47.090 --> 00:31:53.769

Jisun An: the river basically has probably the most strong relation with the bank. So I highlighted with a thick

144

00:31:53.930 --> 00:32:08.749

Jisun An: color. I'm yeah. And then maybe the set and by also could be relating to the bank. So these are the 3 surrounding words that that the model should learn that. Okay, these 3 words is something that I need to attend to understand better about this bank.

145

00:32:09.130 --> 00:32:17.899

Jisun An: But then, even for the same word, if they are in a different context, then, where they need to attend to is can be changed.

146

00:32:18.710 --> 00:32:31.030

Jisun An: So here she deposited money in the bank. So in this example, to understand better. What is this bank means? You actually need to give attention or focus on the other words, like deposited all the money.

147

00:32:31.725 --> 00:32:40.850

Jisun An: So these 2 words would be something that you need to find, and you need to give more attention to correctly interpret the words meaning of the bank

148

00:32:43.826 --> 00:32:58.530

Jisun An: and then so once. So first, st the goal of the attention is, firstly, you determine which word to attend. So you are kind of doing some computation to get that. And then, once you know which word is important, then you use the information to reinterpret the bank.

149

00:32:59.180 --> 00:33:10.739

Jisun An: So so given in the 1st example, maybe, the the some embedding that were representing this bank was these values because they are represented as embedding of the n-dimensional

150

00:33:11.770 --> 00:33:23.899

Jisun An: space. So even even though these are the 2 same words because they are having different context. Basically, their embedding may update it or change it.

151

00:33:24.224 --> 00:33:43.340

Jisun An: And so that's the what we say, as a reinterpret, the word based on this context. So from the model point of view, how they interpret a word is basically is representing the embeddings. Right. So we reinterpret the words based on this context means that they will update for just their embeddings for each of the words.

152

00:33:44.230 --> 00:33:58.019

Jisun An: So so in the transformer models, and most of the larger language models. Even for the same context. They are embedding values may change or different depending on their context. But that's another thing that you need to know

153

00:34:00.484 --> 00:34:17.849

Jisun An: and then so to achieve this goal, there are 2 steps. So one is they compute the relationship between words. So they want to know. Given a sentence given each of the token in a sentence, what other tokens that I should pay attention to, and you are kind of doing this computation, using curry and key

154

00:34:17.860 --> 00:34:32.329

Jisun An: and some weights, and also to reinterpret words based on the context. This will be basically the weights that you learn from the 1st step will be also multiplied as a weight to the value, so that the value will also update their value.

155

00:34:36.440 --> 00:35:01.734

Jisun An: And before describing more of this key value. Query thing, 2 concepts to know. So the attention there are cross attention and the self attention and the attention decisions. Once again the same. They want to know which of the words they need to attend to. Interpret the word. So the cross intention. Basically, you are computing the attention across 2 different sentences.

156

00:35:02.350 --> 00:35:16.449

Jisun An: and then the self-attention is basically you are looking at the same the the sentence themselves, and to understand better about the sentence itself. I mean, they just compute your attention, or given the one I mean the sentence itself.

157

00:35:20.470 --> 00:35:24.009

Jisun An: So the query and key and value is the core of the attention.

158

00:35:24.130 --> 00:35:25.110

Jisun An: And

159

00:35:26.380 --> 00:35:34.189

Jisun An: these so start with the definition. So the query here is the I mean, we kind of call it as a search term.

160

00:35:34.230 --> 00:36:02.169

Jisun An: and the key is the features of the data used for the relevant matching and value is the actual information retrieved. So I mean, there are many metaphor that people have used that that. So these words are basically coming from the information retriever. So assuming that you are like Googling, and then so you are searching with a particular query right and then, now each documents that are waiting there that is matched with this query.

161

00:36:02.170 --> 00:36:06.519

Jisun An: so each document will have a particular key that will be matched with the query.

162

00:36:06.520 --> 00:36:21.559

Jisun An: and then, once they are matched, then the actual value of that document will be returned. So I mean. These are some metaphor that is coming from the information retrieval, but I don't know whether that is really the good one, but I guess it's just something that has been

163

00:36:22.073 --> 00:36:27.829

Jisun An: introduced. But let me give you a very clear example of this Corey key value

164

00:36:28.900 --> 00:36:52.870

Jisun An: so so ideally what we want to do, and using the previous example that we had. So this each of the tokens of the sentence are now represented with some kind of embeddings, and once again this embedding. Theoretically, these are the token embedding plus position encoding. So we already kind of go through that step. Given these values. Now we want to know

165

00:36:52.920 --> 00:37:02.830

Jisun An: in a sentence for each. So we are, and also talking about the self attention. So we will talk about for the same in one sentence, how they will compute the attention.

166

00:37:02.900 --> 00:37:07.260

Jisun An: And the how. The process we're gonna go through is for each of the token.

167

00:37:07.550 --> 00:37:24.240

Jisun An: We will actually do this the same steps. But for. Now let's assuming that we are focused on the bank. So the bank is what we are interested in and we want to. We want to find out which of the other surrounding words, or including themselves, need to be paid attention to. So our query is the bank.

168

00:37:24.250 --> 00:37:46.059

Jisun An: and then our key will be now each of the tokens in a sentence. And what we want to find is that okay? So we already know that the set and the river and the banks are the 3 the key keys that that are that need to be focused on. Given this query and the key. So that's the what we, the preferred outcome that we want to have.

169

00:37:47.390 --> 00:37:59.140

Jisun An: And if we now think about the set, a word set. So, assuming the set as a query, and for the same. If we want to find the surrounding words that to focus on.

170

00:37:59.140 --> 00:38:26.290

Jisun An: then we will go through the same. But then, and also the preferred outcome of the set will be he, and by in the bank. And once again, these are just an example, and I'm just giving you some illustration. So if you think about set, then maybe the bank would be something that is meaningful to interpret the set and also, maybe he, I don't know how much it would be relevant. But set is basically the verb that is followed by he or she. So

171

00:38:26.340 --> 00:38:28.611

Jisun An: a person or the object so

172

00:38:29.520 --> 00:38:42.130

Jisun An: that would be some useful information, and also by May also can have some synthetic kind of relationship. So you can see that these words relationships are something that we want to find through the attention computation.

173

00:38:43.030 --> 00:38:52.150

Jisun An: So so now, coming back to the bank example, so assuming that we are reflecting, we are evenly reflecting surrounding context. So we assume that all the

174

00:38:52.420 --> 00:39:00.999

Jisun An: tokens in a sentence have the same weight, then after so, and if we have that criteria, then our outcome will be basically all the tokens.

175

00:39:01.160 --> 00:39:23.747

Jisun An: And if we reflect surrounding context based on the distance, so the words that are nearby is more important. So if we are giving a weight based on the distance. Then here the value here is just some weights, so the bank itself is the itself, so it has the highest value, and the river, which is one word

176

00:39:24.320 --> 00:39:42.859

Jisun An: away, became having like a weight of 4 and 3 and 2, as the as the other words are getting further away, and assuming that we have some threshold of like 2, we just based on that rule, the our outcome could be just a river bank, which is not similar to what our preferred outcome, is

177

00:39:49.860 --> 00:39:59.470

Jisun An: it? It doesn't matter, because so so here these these weights are doesn't matter with the. It's it's not part of the positional encoding at all.

178

00:40:00.410 --> 00:40:11.139

Jisun An: So here the value that you are seeing the vector themselves is already go through went through the token embedding and the positioning encoding. So that's only done before they are coming into the attention layer.

179

00:40:11.590 --> 00:40:17.790

Jisun An: So so once again, these are just hypothetical situation. I'm just, I mean just giving you some illustration.

180

00:40:18.640 --> 00:40:20.390

Jisun An: Oh, so

181

00:40:20.730 --> 00:40:46.360

Jisun An: so I mean, there could be a different way. How you compute how the query and keys are related to each other, and these are some of the possible simple rules. Right? You can just give the same weight to all the keys, or you can. You can give the weight based on their distance. But actually, that's not something that we want to do. What we want to do is we want to find this relevant, calculated from the data itself and not determined by any of these fixed rules.

182

00:40:46.510 --> 00:40:57.200

Jisun An: So ideally, we want to compute some calculate, some relevance between key between the query and each of the key values and get the relevance scores.

183

00:40:58.130 --> 00:41:21.939

Jisun An: So here, once again, we already kind of assume that the bank and back should have the highest relevance, because that's the word itself and the river and the set probably have some higher value. And maybe he and but I mean the basically nothing relevant to the bank. So we just give a very low relevance. So we want to do this through some automatic way. And that's the where the attention mechanism comes in.

184

00:41:23.504 --> 00:41:39.275

Jisun An: So so now the attention is simply we want to basically learn each of these key vectors. And the query vectors, and also their weights. So together the weights, and then they are vectors. So so now

185

00:41:40.010 --> 00:41:46.037

Jisun An: these all, how the relevance themselves is computed via a weighted of

186

00:41:47.240 --> 00:41:51.679

Jisun An: some which is basically like a 1 layer kind of neural network.

187

00:41:52.630 --> 00:41:56.200

Jisun An: we are just using the we are. We are just

188

00:41:56.420 --> 00:42:13.630

Jisun An: put the learnable parameters there, and by doing so the entire network will be able to learn all these weights and the key vectors, and etc. So that, given a particular context, they will determine what would be the relevant score.

189

00:42:13.630 --> 00:42:30.310

Jisun An: And in computing this relevant score once again, we are just having, instead of just having the query embeddings and the key embeddings. We have this weight metrics for each of the query and the key, and these are the weight matrixes will learn, like

190

00:42:30.650 --> 00:42:34.740

Jisun An: parameters, that how these should be computed each other. So

191

00:42:35.030 --> 00:42:37.800

Jisun An: the this is the architecture of the attention.

192

00:42:38.200 --> 00:42:54.290

Jisun An: and once the model starts to train. Then these wk, and each of the key and the query embeddings also will be updated, and that will enable you to compute the relevance between each of the tokens and each of the keywords

193

00:42:58.640 --> 00:43:11.570

Jisun An: and so the this part was computing the relationship among the words. So now basically, we know that which surrounding. So once we know that we just know that we computed this relevant score.

194

00:43:11.790 --> 00:43:29.570

Jisun An: And once we know this relevant score, we need to reinterpret our value, and that's the where this is coming. So we are combining the attention and the value themselves. So the value means that so each of the tokens already have their value embeddings. But we just give a different weight based on the relevant score.

195

00:43:29.600 --> 00:43:42.339

Jisun An: So we do radius some, and that will now resulting in I mean that will result in which are the most important the words, or that that this bank need to pay attention to interpret this bank.

196

00:43:42.440 --> 00:43:57.079

Jisun An: But instead of just using the value themselves. The value is, or value in embeddings will be also trained with the the weight metrics for the value. And that way metrics will be also part of. I mean.

197

00:43:57.220 --> 00:44:00.710

Jisun An: learnable parameters. So they will learn from the data themselves.

198

00:44:01.286 --> 00:44:13.770

Jisun An: So I mean, I mean once again, because these weights are like, now, high, dimensional, and so it will not be possible to exactly understand how they kind of work. But the key information is that

199

00:44:14.160 --> 00:44:42.729

Jisun An: to compute the attentions, meaning that the relationship among the words we are now using for each of the key each of the query we just compute. How important, how they are relevant to each of the key value. And then once we know that relevant score, then we update the values value embeddings based on them by the weighted sum, and each of these query and value and the key embeddings are.

200

00:44:43.321 --> 00:44:49.080

Jisun An: have this loanable parameter, so this value will be also learned through the training.

201

00:44:54.370 --> 00:44:58.515

Jisun An: And to to give you a bit more example.

202

00:44:59.030 --> 00:45:06.789

Jisun An: so so, assuming that I mean, even though I didn't like mention here. So he, assuming that we have these tokens support tokens.

203

00:45:07.358 --> 00:45:19.790

Jisun An: What they initially do is for each of the token they create the query embeddings, and then they compute their relevant score. By by like

204

00:45:20.340 --> 00:45:21.393

Jisun An: with the

205

00:45:23.400 --> 00:45:44.409

Jisun An: by like, that product will be the key value. So for each of the key value, basically, it measures the relevant score which will turn out to be these values. And then one additional thing that we're going to do is they will take the software mix here so that this value now turning into the probability that's just for the, I mean normalization purposes.

206

00:45:45.080 --> 00:46:11.210

Jisun An: So these are how the actual curry and key relationships are done. But after they get this soap to Max scores, then so now we know that the relevant scores are 0 point 2 0 point 1 0 point 5 and 0 point 2. And now this score will be also multiplied to the value. Vector so now you see that, assuming that we, for a particular token, the 1st token, all the

207

00:46:11.360 --> 00:46:35.879

Jisun An: all the other tokens had the same weight. But basically these vectors, now weighted after multiplied by this relevant score. And then after that, you just do the sum. And finally you get the jet one and jet one will be now the new embedding for the 1st token after going through them. So you can kind of see that the 1st token will now update their embeddings

208

00:46:35.880 --> 00:46:43.812

Jisun An: to do this attention mechanism, so the resulting embedding will be the size of the embedding, for each of the token will be like same

209

00:46:44.380 --> 00:46:50.990

Jisun An: but they but the value are just has updated, based on the relevance with their surrounding words.

210

00:46:55.300 --> 00:47:05.029

Jisun An: And basically you do for each of the token in a sentence. Then each of the token will now get the reinterpreted embeddings based on the attention mechanism.

211

00:47:08.450 --> 00:47:09.710

Jisun An: Any questions

212

00:47:12.710 --> 00:47:19.580

Jisun An: I I know it's not a like very easy concept to to understand.

213

00:47:27.590 --> 00:47:29.360

Jisun An: syncresusion?

214

00:47:30.540 --> 00:47:35.224

Jisun An: Oh, no. So the the query. And so the word plus

215

00:47:36.110 --> 00:47:40.880

Jisun An: the x, 1 is the word plus position encoding oh.

216

00:47:41.490 --> 00:47:55.519

Jisun An: and the the query and key. And the vectors are new, vector. New, just even kind of encodings that are solely used to compute the attentions only. So they are not part of the position or

217

00:47:55.830 --> 00:48:22.579

Jisun An: the initial embedding that they had. But thanks for asking. I think that that helps to clarify things. So yeah, once again, here the x 1 is the the one that we already go through the input encoding. And then the position encoding. And then that value came in. And for each of the token they will create a new query key and the value vector, for each of the token. So the each of token will have 3 additional embeddings

218

00:48:22.790 --> 00:48:29.740

Jisun An: solely used for computing the attention. So these are all like separate matrices and separate embeddings that is used for computation.

219

00:48:30.980 --> 00:48:35.960

Jisun An: And I mean, this would be very vague. But all these values will be initialized randomly

220

00:48:36.640 --> 00:48:41.777

Jisun An: together with their weight matrixes as well. Sorry that I think I missed that part.

221

00:48:44.150 --> 00:48:45.890

Jisun An: So once again, just

222

00:48:46.320 --> 00:49:07.840

Jisun An: with the with the one sentence for each of the token, they will come in. They will get the input embedding and then positional embedding and then feed into the attention. So starting from the attention mechanism for each of the token, they will create a new key vector key query value vector initialized randomly.

223

00:49:07.900 --> 00:49:26.740

Jisun An: and each of those factor, we also have these weight matrices and all these embedding value, together with the weight will be learned from the data based on their they will, I mean, supposed to learn the relationship among the words given the test in particular. I mean, they learned through the real model training.

224

00:49:31.950 --> 00:49:33.290

Jisun An: Any other question.

225

00:49:39.030 --> 00:49:39.800

Jisun An: Okay?

226

00:49:41.360 --> 00:49:42.090

Jisun An: Oh.

227

00:49:46.266 --> 00:49:48.359

Jisun An: each single.

228

00:49:49.860 --> 00:49:50.790

Jisun An: Let's show.

229

00:49:52.586 --> 00:49:56.350

Jisun An: So so when you learn the model, you can just

230

00:49:56.530 --> 00:50:07.520

Jisun An: consider the method of metrics. But once you learn all of these parameters then given any, input, it will just give the for each of the token. It will just generate embedding.

231

00:50:08.070 --> 00:50:12.169

Jisun An: Yeah. But when it's learned, it's just the

232

00:50:12.440 --> 00:50:14.870

Jisun An: metrics is. And I think it's hard. Yeah.

233

00:50:17.920 --> 00:50:25.579

Jisun An: it's interesting that that we're in teaching language values.

234

00:50:26.355 --> 00:50:30.170

Jisun An: But just what I mean is like.

235

00:50:31.350 --> 00:50:33.930

Jisun An: so we're looking at like the like.

236

00:50:34.820 --> 00:50:35.519

Jisun An: I don't know.

237

00:50:35.790 --> 00:50:42.609

Jisun An: It's correct earlier. It's interesting to see this where that?

238

00:50:42.930 --> 00:50:48.040

Jisun An: Oh, like the yeah, it's not quite the same

239

00:50:48.300 --> 00:51:09.820

Jisun An: exactly. I I don't think it's it's very confusing to use this metaphor of the search engine, because it's not working in that way. But but then if you think about cross attention, then then maybe it make more sense. So so you have the inputs and the outputs and the output output sentences could be considered as our values.

240

00:51:10.060 --> 00:51:13.169

Jisun An: Then key and value will be the outputs

241

00:51:13.490 --> 00:51:24.470

Jisun An: and the query will be the inputs. So for input word for each of the token became the each of the query. And now they will compute how the each of the input query is

242

00:51:24.840 --> 00:51:27.330

Jisun An: associating with the output tokens.

243

00:51:27.490 --> 00:51:47.710

Jisun An: And once we've you found that which which of the output tokens are most important to interpreting the input token or solving any, the problem like machine translation. Then you update the the value of this query, using all the values of the output. So I guess that maybe may or may not make sense

244

00:51:53.430 --> 00:52:06.330

Jisun An: right. And the the additional thing that I want to mention is the so how do you compute the relevant score. I mean, simply they are using the dot product between the query and the key value embeddings.

245

00:52:06.330 --> 00:52:27.460

Jisun An: But then, instead of the dot products, they actually use this scale dot products to prevent the basically the in terms of like normalization purposes. So as the dimensions of the embedding gets larger, the dot products value is also getting larger, so they just divide it by the roots of the

246

00:52:27.460 --> 00:52:31.281

Jisun An: k, the dimensions of the key value, so that they just scale

247

00:52:31.730 --> 00:52:33.639

Jisun An: by the size of the vector.

248

00:52:34.590 --> 00:53:04.550

Jisun An: So I mean, so sometimes it's easier also to look at a little bit of code themselves. And so this is the part where they for a single attention block. They just create the weight matrixes and the key embedding key query, key value embeddings, and how they compute the attention. So the the final attention is computed based on the query vector multiplied by the key vector divided by the the size of the key embeddings. Taken this up to Max and then multiply by the value embeddings.

249

00:53:04.550 --> 00:53:12.649

Jisun An: and then we'll be resulting embedding, which is once again our reinterpreted embeddings of the target world

250

00:53:18.830 --> 00:53:20.100

Jisun An: any questions.

251

00:53:31.090 --> 00:53:38.459

Jisun An: Yeah. So in a way, the attention mechanism is trying to just create a giant matrixes where

252

00:53:38.600 --> 00:53:53.349

Jisun An: all these parameters are learning how each of the word in different sentences are correlated with each other or important to each other. I mean, that's the basic thing that the attentions is doing. And in doing so they just use.

253

00:53:53.490 --> 00:54:02.499

Jisun An: So once again, I the key Corey key value. Metaphor is not very intuitive, but for each, over the token you just

254

00:54:03.090 --> 00:54:15.280

Jisun An: compare with all the other surrounding words, to see which word is most important or to focus on. And once you know that weight which is the can be computed from this optimex.

255

00:54:15.860 --> 00:54:23.040

Jisun An: then you just give more weight to weighted sum to the the actual value.

256

00:54:24.010 --> 00:54:34.790

Jisun An: Actual value embeddings. So, assuming each of the world already has, like the value embeddings. You just want to know how much information that I want to take from each of the tokens.

257

00:54:35.250 --> 00:54:41.140

Jisun An: to to know my to know the token yourself.

258

00:54:41.330 --> 00:55:06.259

Jisun An: So that's something. And because this key and the value I mean no key, and the value also include the the query token themselves itself, so it itself will be also part of it. So to recompute or reinterpret the words of the query token, you are just using this key and the value information, and essentially the value themselves.

259

00:55:07.260 --> 00:55:19.940

Jisun An: The final value will come from these value embeddings. And you just want to know the weight for each of the value embedding, which is coming from the key and the query multiplication which are getting into the relevant score.

260

00:55:20.880 --> 00:55:22.990

Jisun An: Yeah, this is complex. Yes.

261

00:55:23.370 --> 00:55:28.419

Jisun An: So the goal is that, like the objective, is to connect.

262

00:55:28.530 --> 00:55:47.289

Jisun An: So. But so that's slightly different. So attention, mechanism itself is just the attention mechanism, and how that relevance is determined now can be related to what task we want to. Ask what test we are using to train the entire model.

263

00:55:47.440 --> 00:56:01.979

Jisun An: So yeah. So when the weights are adjusted, that's based on the task, the task and that task can be next token, prediction, or next sentence, prediction, or like masked token, prediction, and etc. So that has been the task that has been used.

264

00:56:02.100 --> 00:56:20.800

Jisun An: But these are like the one component of the transformer architecture that are different parts will be trained, and the attention will learn how different words are related to each other, or what other, how the other surrounding words are impacting me to be reinterpreted.

265

00:56:21.140 --> 00:56:24.950

Jisun An: So that's the while attention basically will compute.

266

00:56:27.040 --> 00:56:40.840

Jisun An: I hope this really clears up. Let me move on. So multi head attention is the the actual module name. And now and here was the I mean, it's actually mask multi head attention. So we will also talk a little bit about the masks.

267

00:56:41.430 --> 00:56:58.599

Jisun An: The intuition of the multi hat is now now like. If you want to define the relevance between the words, then there could be some many different ways to interpret something right? So if you have these 4 sentences, and if you want to reinterpret each of the wrong.

268

00:56:58.850 --> 00:57:15.000

Jisun An: and basically we know that the the meaning of the run is determined by the surrounding word. But then, maybe there could be some syntax relation. So the I and all coming before run mean something. But at the same time some other words may contain like semantic,

269

00:57:15.670 --> 00:57:35.689

Jisun An: semantic kind of context. So basically, you need, like the different head to learn each of this relationship. So the multi head is, instead of like one single attention mechanism, you will have just multiple of them. And you just assume that each of these attention will learn something useful.

270

00:57:36.051 --> 00:57:49.409

Jisun An: It can be syntactic information. It can be semantic information. It can be long distance information, or it can be, whether, like the relation between the negations. So so now that if you have

271

00:57:49.850 --> 00:58:08.959

Jisun An: so each of these has will learn something about how the our words are related to each other. And you can. You can see that the power of the transformer is coming from this multi-head attention. Because if you have now multiple, multiple, many, many heads, if you learn more and more about the language themselves, or the word associations.

272

00:58:09.240 --> 00:58:15.349

Jisun An: and subtle kind of differences among the words, how the words are related to each other.

273

00:58:19.160 --> 00:58:44.986

Jisun An: and in in terms of the implementation is actually quite simple. So you are basically doing very similar attention computation, as we we've seen before. But if you have like 8, and then basically, you have 8 larger metrics. And if you have 8 head, then you are just dividing that metrics into 8 separate metrics, and then you will do the computation.

274

00:58:45.860 --> 00:59:03.080

Jisun An: so here the 3 layers you've seen is are the each of the head. So for each of heads, basically, you have the set of query key value embeddings and their weights that we've seen from the attention mechanism. So each head will have exactly the same thing. And

275

00:59:03.360 --> 00:59:16.086

Jisun An: once again, through the neural networks magics of training each you've had will learn something different to each other. So so at the so for each of the head you will have the

276

00:59:17.090 --> 00:59:19.298

Jisun An: jet values, and then

277

00:59:20.130 --> 00:59:31.079

Jisun An: which is the reinterpreted version of this word, and you also concatenate them to to get like the finer embeddings of each of the token.

278

00:59:32.300 --> 00:59:58.936

Jisun An: And once again, this is the code example. And here, the view is basically the way that you, you are changing the shape of the metrics. So what what here they do is they simply dividing this issue of weights and the embeddings based on the number of the heads? I mean, I mean, you don't need to understand entirely. But if you are familiar with the pytorch, I think like reading, this code will help you to understand how the multi head was

279

00:59:59.480 --> 01:00:00.710

Jisun An: implemented.

280

01:00:01.310 --> 01:00:23.159

Jisun An: And here, so what happens with the multi head is so these are one of the visualization that were presented in the transformer paper and and it shows that for a given query, which is the making, it shows how the attention goes to another tokens in this example. And the different color basically represents a different heads.

281

01:00:23.160 --> 01:00:36.429

Jisun An: So many of has actually learned like the the making and more difficult are basically associated and very relevant. But some other are having some information like 2,009 or laws.

282

01:00:37.060 --> 01:00:43.409

Jisun An: Yeah, so you can, you basically see that different has learned slightly different things for a token

283

01:00:48.250 --> 01:00:51.729

Jisun An: and then the masked attention is

284

01:00:52.810 --> 01:00:58.819

Jisun An: The reason that we have mass notation, and especially this, was used in the decoder part is because

285

01:00:59.140 --> 01:01:07.770

Jisun An: if we are using just the normal attention. We assume that to know the meaning of the word, we also know the

286

01:01:08.310 --> 01:01:11.939

Jisun An: all the other ones. But if you think about text generation.

287

01:01:12.120 --> 01:01:33.680

Jisun An: then it's usually pre like generating the the next token, and when you generate the next token, you only know the previous token that has been already generated. Right? So basically, it is not fair to know the future tokens when you actually learn these parameters. So once again, this is another attention mechanism that existing as a separate of all these

288

01:01:33.970 --> 01:01:59.809

Jisun An: parts. And this part is master attention is dedicated for like text generations. So it basically ensures that the model only attends to the past tokens, and they do so, using the triangular mask. So this the scale score is the something that the value before the software. Max. So these are the attention scores after getting the.

289

01:02:00.240 --> 01:02:07.389

Jisun An: So after the Korean key multiplication. And then so before moving forward, they basically adding this mask

290

01:02:07.924 --> 01:02:14.710

Jisun An: triangular mask, where basically they have some minus infinite value for the future tokens.

291

01:02:14.710 --> 01:02:40.229

Jisun An: So you can see, assume that this means that the 1st token know itself, but they just don't know what is the next 3 tokens, and the second token know itself and the previous one. But they don't know the 3rd and 4th and etc. So by using these masks masks, they basically hide all this master score. And once you have this minus infinite value. If you take this optmax, then this value will all became 0.

292

01:02:41.664 --> 01:03:08.560

Jisun An: So so basically, the mask self attention for a second query. They only assume that they know the 1st 2 queries, and they don't know the other 2 queries. So when they compute all these relevant scores, they don't compute for the future tokens they only compute with the previous tokens, and after that they take this off to Max, and after taking the softmax, this minus infinite value will be just 0. So the basically the model will assume that I haven't

293

01:03:08.590 --> 01:03:12.199

Jisun An: seen any future tokens and given that what would be the best

294

01:03:13.106 --> 01:03:19.340

Jisun An: best to talk best surrounding words that I need to attend to. So the mask attention will basically compute that values.

295

01:03:22.060 --> 01:03:24.989

Jisun An: So we'll stop here any questions.

296

01:03:27.160 --> 01:03:55.260

Jisun An: So I I hope this was really helping you to understand the attention mechanism. And if it's not very clear I mean. So these images is coming from Jay's Illustrated Gpt tool which has really good illustrations about the transformer. So you it'll be a good reading if you want to know more. But what I I guess what I try to give you as detail as possible, because it's an interesting concept. And also, once you understand, it's not a very difficult concept. It's just

297

01:03:55.320 --> 01:04:05.969

Jisun An: I don't know why, but the key point value was very confusing to me initially as well. But it's actually quite just weighted some. It's the metrics multiplication, that's all. Nothing complex.

298

01:04:07.640 --> 01:04:08.710

Jisun An: Oh.

299

01:04:13.430 --> 01:04:26.919

Jisun An: oh, I always confused by the end time I'm sorry, like, okay, give me 5 more minutes. I will. I will do 5 more minutes. Yeah, I thought, we are ending. So. We have 10 more minutes. I will. I will give you 5 more minutes.

300

01:04:27.040 --> 01:04:34.040

Jisun An: Let me carry on. I'm sorry. So the next one is the layer normalization and the register connection.

301

01:04:34.550 --> 01:04:36.109

Jisun An: So now we are

302

01:04:36.260 --> 01:04:46.529

Jisun An: so input, embeddings input, embeddings positional encodings, and then the after this master pension, we have some reinterpreted embeddings. It's all clear right.

303

01:04:46.530 --> 01:05:07.600

Jisun An: The next one is the the addition and the normalization. And so now I mean, think it as a very simple. So for each of the token you have now new embeddings computed, based on the attentions, and for that value. Vectors, you just want to do the normalization. And the reason that we are doing the normalization is simply

304

01:05:07.720 --> 01:05:10.585

Jisun An: if the different different

305

01:05:12.080 --> 01:05:22.400

Jisun An: embedding values have different scales. Then it could be a problem. Right? I mean the simple. I mean, everyone knows that. Why people do the normalization in the machine learning, right? So like

306

01:05:23.280 --> 01:05:52.930

Jisun An: basically, the range of the variable can be very different. Then you are unnecessarily give more weight to those feature that has, like higher values or higher range of the value. So we are doing like normalization. And the easiest way to normalize this value is by taking the jet score. So the equation that you are seeing here is X minus their average, divided by the standard deviation. This is like the typical jet score equation where you do

307

01:05:53.626 --> 01:06:05.980

Jisun An: normalize certain lane range of the distribution to the range from 0 to one. And the the reason that they are adding just to prevent that, the divided by the 0

308

01:06:06.110 --> 01:06:12.104

Jisun An: and the Y basically with the sum rate, the the normalization will be done with some kind of rates.

309

01:06:12.910 --> 01:06:32.779

Jisun An: and the layer normalization in particular. And there are also like a batch normalization. You can also. So now, given, you have the data, there could be different way to normalize it batch normalization will be normalized across the data. But the layer normalization is normalized per data. So for example, within this 3 example.

310

01:06:33.180 --> 01:06:44.740

Jisun An: assuming that we have for the 1st tokens of the each sentence assume that these are the embeddings that we have, and we are normalizing for each of the token at at like

311

01:06:45.336 --> 01:06:47.689

Jisun An: based on their own value. So we

312

01:06:47.860 --> 01:06:58.870

Jisun An: compute the average and the standard deviation for each of the token. And we take the equation that we've seen in the previous slide, so that each of these embedding value will be just normalized.

313

01:06:58.970 --> 01:07:20.130

Jisun An: So once again, that's the layer normalization. And it's it's the purpose of the normalization once again is to treat all the features equally, so that, like we not see unnecessary weight for a feature that has embedding dimension that has, like a larger range.

314

01:07:20.580 --> 01:07:21.760

Jisun An: bigger range.

315

01:07:22.200 --> 01:07:39.080

Jisun An: And instead of the layer, normalization root, mean square normalization is something more commonly used. The idea itself is extremely the same. But instead of the using the mean and the standard deviation value, they simply divided this value by the root mean square.

316

01:07:39.100 --> 01:07:55.289

Jisun An: so that it's not will be. The range will not change to 0 to one, but it will be a certain value that is reasonable enough to be similar to each other. But this simply provides the computation themselves. So it basically, this is for the efficiency of the computations.

317

01:07:58.580 --> 01:08:17.139

Jisun An: And the register connection is the in particular is the the part where you see this connection from here to here. once again.

318

01:08:17.399 --> 01:08:20.497

Jisun An: Connection. And the idea is really simple.

319

01:08:21.210 --> 01:08:25.959

Jisun An: they simply take the whatever output you had here.

320

01:08:26.534 --> 01:08:35.029

Jisun An: Whichever output you had here. You just add that exact embedding at this moment. So that's just the register connection.

321

01:08:35.800 --> 01:08:40.879

Jisun An: Just add the additive connection between the input and output. And the reason that they are doing is

322

01:08:41.060 --> 01:09:02.729

Jisun An: so somehow, this input encoding should be, I mean that input encoding should be contained the most important information. But then, after the attention mechanism, sometimes they are just losing what was the most important. So they just added this simple mechanism where they just regain those embeddings and then added to it.

323

01:09:02.899 --> 01:09:26.709

Jisun An: And somehow that turns out it just improved the performance a lot. Once again these multi head also became a very complex computations that may or may not lose some information about the inputs. So they just want to keep some of the information of the input. So they simply take the embeddings of the input, and they just add it at the end of the attention.

324

01:09:27.189 --> 01:09:43.690

Jisun An: And they also do again. For after the feed forward networks again. So they just you can just I mean, think it as a they highlight more about the input embedding themselves so that they keep track of some important parts of the input embeddings.

325

01:09:44.399 --> 01:10:04.059

Jisun An: So this also, like prevents like vanishing gradients, and also allows to learn the difference from the inputs themselves. I mean, basically, this networks or model gets really big and deep. So they want to just empathize more on the input themselves, and by simply adding the embedding in the different part of the model, they just were able to do that.

326

01:10:05.870 --> 01:10:28.009

Jisun An: And interestingly, the register connection. The concept itself was presented in some other work in 2015. And this is really simple idea. You just take the input embedding and then add it to in some different part of the model, and this itself has been used in different neural network based models. And this is paper itself cited like 250,000 times

327

01:10:30.490 --> 01:10:54.560

Jisun An: and and now, so you are doing this addition and the layer normalization. And now the matter is when you actually do this normalization and the addition so initial transformer architecture, or have it at the after the attention and after the fit for network. But then this paper proposed that actually, if you do this before the attention, then

328

01:10:54.610 --> 01:11:13.509

Jisun An: layer normalization. If you are doing that before the attention, then it actually learns better, and it actually makes sense right. The attention mechanism is essentially learning all this weight. So you need to normalize the values before doing all the learnings. It would be actually helping to stable, the learn, the parameters.

329

01:11:13.670 --> 01:11:29.069

Jisun An: So this was the post versus pre layer normalization. This paper found that the compared to the post layer normalization pre layer normalization works far better. And also you have to propagate the gradients.

330

01:11:29.656 --> 01:11:53.199

Jisun An: So interestingly after the transformer. Obviously, there has been extremely amount of the research that we're trying to improve like the different bits of the transformer. But this is the probably the only thing that will stick in terms of the architecture. So all, the other thing is just changing the function to something else. Maybe like for the position encoding. They were just changing, absolute to the relative or

331

01:11:53.430 --> 01:12:10.258

Jisun An: layer normalization changing to the Rms normalization. So the architecture itself hasn't been changed, even though each of the component has been improved like little by little. And this is the only architectural change that people found it useful of all the other.

332

01:12:10.840 --> 01:12:12.639

Jisun An: use all the other work.

333

01:12:14.490 --> 01:12:21.719

Jisun An: Yes, the like famous diagrams above those should be that, and one should be separated.

334

01:12:21.980 --> 01:12:27.219

Jisun An: Oh, yeah, yeah, I mean, these are, I, I guess I I yeah, they should be separated.

335

01:12:28.190 --> 01:12:36.200

Jisun An: So I think it's relatively simple and and the diagram itself is coming from 2,017. So yeah.

336

01:12:36.360 --> 01:12:41.690

Jisun An: yeah, they need to be separated, and the layer known should become before the attention, and before they fit for that.

337

01:12:43.190 --> 01:12:45.770

Jisun An: Okay, I will really stop here.

338

01:12:47.060 --> 01:12:52.870

Jisun An: even though it's even though not much left. But I will continue from here.

339

01:13:00.430 --> 01:13:03.140

Jisun An: Well, if you give me 1 min.

340

01:13:03.710 --> 01:13:12.148

Jisun An: this is just one slide. So just to so the last part is to feed forth network. And these are just fully connected layers.

341

01:13:13.060 --> 01:13:32.729

Jisun An: that will do the the combination feature thing that we're going to do. So, the whatever output that you have after input position encoding attention and the addition and the normalization that output will feed into the fit for the network. And they will that fit for the network will learn themselves some different parameters which

342

01:13:33.330 --> 01:13:48.390

Jisun An: which I don't know what I learned something about this legacy model links. So you can kind of. And then the way that this feed forward was constructed is basically it it, it added like a larger layers

343

01:13:48.720 --> 01:14:13.129

Jisun An: which have more dimensions. And then they added some nonlinearities, and then they are now coming back to the same dimensions as they started. So these are some fit forward network that they usually have. And it could be multiple layers in between these 2. But just I want to highlight what is input and the output, they will be the same dimensions, and in between there will be just all like a deeper network, deeper, larger network

344

01:14:13.130 --> 01:14:25.509

Jisun An: which this will now learn. Maybe everything about the sentences. So how the sentences are related to each other. Maybe some dependencies, some other information than the word themselves.

345

01:14:25.740 --> 01:14:37.729

Jisun An: and also this will be also depending on some task that they will be also trained on. So that will be the last component of the feedforward of this actually, architecture. And then.

346

01:14:37.730 --> 01:15:01.440

Jisun An: in terms of these nonlinear activation functions. We talked about the value. But there is also another one that is famous like the Silu. It's very similar to the value, but it just doesn't give the 0 for those value that is smaller than 0. So, and meaning that if you are taking the gradients that will be some non-zero value, and then the value will be updated, even though it is very, very small.

347

01:15:02.110 --> 01:15:10.850

Jisun An: I didn't really stop here. And and thanks a lot for, and I'm sorry that I was stopping twice in between. The yeah.

348

01:15:11.490 --> 01:15:12.290

Jisun An: if

349

01:15:12.570 --> 01:15:20.240

Jisun An: thanks a lot, and I will finish up the slide on Thursday. And if you have any questions, let me know. And

350

01:15:20.350 --> 01:15:23.369

Jisun An: yeah, thanks. Have a great day.