WEBVTT

1

00:00:06.830 --> 00:00:14.309

Jisun An: Thanks for joining today's attendance code is llama. So please mark your attendance.

2

00:00:26.490 --> 00:00:35.240

Jisun An: The like the just a reminder. The 1st theoretical assignment has been published.

3

00:00:35.340 --> 00:00:56.629

Jisun An: I've seen a few students have already finished it. Thanks for doing it in a very in advance. But but I think they are not difficult in particular, but they will help you to recap some of the contents that we have been doing so. I hope they are helpful. And also we also have

4

00:00:57.840 --> 00:01:03.429

Jisun An: Another thing to do, which is information. So

5

00:01:04.310 --> 00:01:16.069

Jisun An: yeah, I mean, you still have some time. But yeah, just just a reminder that you need to do that. Yeah, just for those who are just got arrived today's passcode is lama.

6

00:01:18.570 --> 00:01:19.425

Jisun An: So

7

00:01:20.570 --> 00:01:34.759

Jisun An: so I want to start from here attention again, because there was a 1 thing that I found that it wasn't very clear to you. And also and and I was also I wasn't very clear. Explaining it. So

8

00:01:35.190 --> 00:01:45.680

Jisun An: the and also I was. I think I was. I got I just double check whether with how it actually works. So so here.

9

00:01:47.110 --> 00:02:11.570

Jisun An: how? The I mean. So once again, the the idea of the attention was, basically we want to compute the relationship among all the tokens. Right? Given an input, or if there's like input and output, you want to find the relationship between those tokens of the input as output. So here, basically, if you have a 1 sentence, it will like separate them into the tokens. And these key

10

00:02:11.870 --> 00:02:26.750

Jisun An: key value calculation will be done for each of the or token. But now the Gpu can do parallel processing, so everything can be processed at once simultaneously. But we are giving just one example here. And

11

00:02:27.060 --> 00:02:56.399

Jisun An: so the the input to the attention. module will be these embeddings. But so, even though speak the the things that were confused because I was explaining the position encoding in before this attention slide. But then here, because this was illustrative purposes, we, we say, as a token embedding, but so that embedding itself is actually like the finer input, embedding. So that's the that should be in theory. The the

12

00:02:56.400 --> 00:03:17.269

Jisun An: the summation of the token token embedding and the position embedding. So that's that value. And then the query and key and the value vectors. So when you have this new token to be computed for the attention. Each of our tokens in one sentence will create their own query, key and vector

13

00:03:18.430 --> 00:03:20.170

Jisun An: vectors by

14

00:03:20.620 --> 00:03:28.350

Jisun An: by multiplying between the input embedding and the the weight metrics for each of key and query and value.

15

00:03:29.080 --> 00:03:47.719

Jisun An: So you can imagine. For each of token they will have each of you. I mean, they will be computed based on the multiplication of the token input embedding, and they are weight metrics for corresponding like query or key or value. So that's the what this diagram say. So once again

16

00:03:47.760 --> 00:04:02.910

Jisun An: giving a sentence for each of the token, and this embedding itself is already the the token embedding plus position encoding. And then this weight matrix is is something that will be multiplied to this embedding, and that will be the resulting key embeddings that you will have.

17

00:04:03.400 --> 00:04:07.341

Jisun An: and and within, and also the value also has the same thing.

18

00:04:07.850 --> 00:04:20.249

Jisun An: so to compute the value embeddings, they, for each of the token there will be, do the linear transformation with these wave factories for the value. So each token will have their own valuing embeddings.

19

00:04:20.836 --> 00:04:28.239

Jisun An: And then so these are the way that they are computing some kind of score within the attention, and then so.

20

00:04:28.460 --> 00:04:49.499

Jisun An: by multiplying the the key embeddings and the query embeddings. They will get this relevant score, and then the weighted sum of the value embeddings using this relevant score will result in a new embedding. That is basically the reinterpretation of the bank. So now our new bank

21

00:04:50.012 --> 00:05:01.607

Jisun An: embeddings, we'll have some part of the bank and some parts of the river, and some part of set and combination with them will be our reinterpretation of the bank. So because the diagram was not

22

00:05:02.150 --> 00:05:08.329

Jisun An: visualized as like embedding. I think there was some confusion, but I just wanted to clarify that a little bit more.

23

00:05:14.510 --> 00:05:40.010

Jisun An: so just just to be very clear the when the E is the input token. Embedding P is the position embedding the summation of these 2 will be now our input to the transformer model. And this X and the the multiplication with their corresponding weight metrics, it will be resulting into the query vector key vector and the value vector for each of the key, each of the token in in this sentence.

24

00:05:41.020 --> 00:05:54.849

Jisun An: And another key thing is so, what are the the vectors that are learned so that will be our token embeddings that is coming from the previous, before the attention mechanism. And all these 3 weight matrixes

25

00:05:54.850 --> 00:06:13.749

Jisun An: will be learned from the architecture themselves, and I mean the learning itself is very similar to any other models that we have been talking with. So, after the attention and many other modules, we will just go through all the networks, and then the finer score, and then we will have some cross entropy

26

00:06:14.100 --> 00:06:22.280

Jisun An: loss, because next predict, next token, prediction is essentially classification problem. So once you get the score

27

00:06:22.280 --> 00:06:45.840

Jisun An: and then compute the loss, and then that will be back propagated, and then they will come down to the attention for each of these weights, and to each of the embedding token embedding themselves. The positional encoding usually is either absolute, or the relative is usually not learnable because they found that the loanable embedding also has some limitation. But then the input embedding, input token, embedding

28

00:06:45.870 --> 00:07:01.710

Jisun An: and the these within the attention, these 3 weight. Metrics is for query, key, and value. They will be learned from the training. So that was the the 2 thing that is very important. So how and so so the key and the query and the

29

00:07:01.910 --> 00:07:15.770

Jisun An: value embeddings are just the product of multiplication of the input embedding and the wave metrics. But that embedding themselves is not, is not learning anything. So I hope that's a bit more clear about the attention.

30

00:07:17.104 --> 00:07:39.950

Jisun An: So I mean, we were kind of in a bit of rush. But we were looking at all of these components. So input the positional encoding and the mass multi health attention, the additions and the layer normalization fit for network and another addition and the normalization. And finally, that would be the entire structure. So

31

00:07:39.970 --> 00:07:55.860

Jisun An: so you know, in a way that the feed board network, I mean, I mean the neural network. The the reason that we are using the neural network is to learn the combination of the features. Right? So you can consider this must multi head attentions. And because the multi head means that we have multiple hats that

32

00:07:55.860 --> 00:08:10.640

Jisun An: probably look at like the different aspects of the language and the neural network. The fit for network will now try to combine which of the multihead attentions need to be combined together and etc. So that's the how this entire transformer structure would work.

33

00:08:10.680 --> 00:08:30.179

Jisun An: So once you have this architecture, the way that this network is trained is similar to any other model that we've been talking about. So for each of the input it will go through the entire architecture, and at the end they will have the. After this optimus, they will have the probability for all the tokens for to become next.

34

00:08:30.180 --> 00:08:44.240

Jisun An: and then it will be compared with the the rear value, and then the the loss will be back prop through the the architecture, and then all the parameters, which will be basically in the fit for network and the attention and the input embedding so they will be updated.

35

00:08:46.200 --> 00:08:50.449

Jisun An: Is it? Is it clear any any question on the transformer architecture?

36

00:08:56.770 --> 00:09:25.863

Jisun An: Okay, so now that we, I mean, so these are like the fundamental architecture. Of the transformer and I mean, different model was used different parts of this architecture. So if we come, I mean, this was the the 1st figure that we've seen. When we start the transformer encoder only encoder, decoder model and the decoder only model. And now, if you look at, look at them in detail, then you will see what are the differences? So?

37

00:09:26.570 --> 00:09:39.259

Jisun An: yeah. Firstly, the encoder only model, meaning that they use the encoder. Can you? Can you tell? I mean, if you can see them, can you? Can you tell what's the main difference? In compare with

38

00:09:39.814 --> 00:09:42.240

Jisun An: encoder and the decoder only model.

39

00:09:44.250 --> 00:09:50.580

Jisun An: Anyone found the main difference between encoder only and the decoder only model.

40

00:09:53.160 --> 00:09:54.090

Jisun An: Yes.

41

00:09:54.290 --> 00:10:11.240

Jisun An: that's correct. So these 2 are exactly the same, because both are like predicting. Next token, I mean they, they could be trained with different different prediction tasks, but the architecture is literally the same. The only differences is whether they are using the multi-head attention or masked multiheaded attention.

42

00:10:11.240 --> 00:10:36.190

Jisun An: So the encoder only model doesn't need to use the masks masked multi head, because I mean they can. So the reason that we are using the masked attention was because if you are generating text, then you are coming from left to the right. So you basically, you cannot generate something. Given the future tokens right? And so that assumption just doesn't make sense in the 4 d.

43

00:10:36.190 --> 00:10:54.740

Jisun An: so they just basically changing the architecture so that when they train the attention compute the attention rates, they just make it, they cannot see the future tokens. So whatever prediction this old probability was only depends on the historical tokens that they have been already generated.

44

00:10:55.550 --> 00:11:06.250

Jisun An: So that's the that's the major difference. So given that what would be the benefits of the encoder? Only model any anyone have any idea what?

45

00:11:07.290 --> 00:11:13.859

Jisun An: What would? What would be good. Okay, what about them then? Decoder. What would be the benefit of the decoder?

46

00:11:15.560 --> 00:11:18.719

Jisun An: Only for the and then the context.

47

00:11:19.080 --> 00:11:32.079

Jisun An: right? Great. Yeah. So that's so basically because you know that how this word is there from like like both direction. So basically, you have a mature information and context about this particular word.

48

00:11:32.090 --> 00:11:53.670

Jisun An: and lee, like leading to that that will meaning that. So basically, this encoder models only models are good at, I mean representation or the encoding. So if you just want to represent a sentence or the word, then encoder, only model actually could work better because they have more information about all the surrounding words.

49

00:11:53.750 --> 00:12:18.079

Jisun An: So the decoder only model on the wire. Basically, they have no idea about the future tokens. So they have a in a way that they have a limited context. So in terms of the representation, it may have slightly, I mean, so different task requires different information. Right? So if you have any test that is really requires about understanding of the language, then decoder only model may not be very good at it.

50

00:12:18.940 --> 00:12:27.529

Jisun An: but then, but then at the same. Oh, did I didn't bring my coffee today. Then what would be the decoder? Only model? Good at

51

00:12:27.640 --> 00:12:29.089

Jisun An: or benefits of it?

52

00:12:31.580 --> 00:12:39.560

Jisun An: Generation. Right? So that's like, I mean, the exactly. The purpose of the decoder model is the generation. So they'll be better at generating. Yeah.

53

00:12:39.880 --> 00:12:49.049

Jisun An: So I mean, if you I mean you can still generate using the encoder only model. Just the the readers may not be as desirable as you would imagine.

54

00:12:50.690 --> 00:13:10.809

Jisun An: and then the then the encoder and decoder model have, like both the good aspects of the both, so they will have a good understanding of the the language themselves, and also they will be able to generate something. So the encoder decoder model are still powerful. But it may not be very efficient, because basically you have.

55

00:13:11.110 --> 00:13:39.329

Jisun An: If you think about this multi head, it will be like a giant weight matrix. It will have a really multi-head right? And then for the you will probably have a separate weight for encoder and the decoder as well. So basically, the the size of the parameter or the model itself will be doubled compared to the encoder only, or the decoder only. So basically, it's just a bit more bigger, meaning that it will take more time to train, and it will just compute like compute cost will be much higher, and also take a long time to train it.

56

00:13:39.480 --> 00:13:53.115

Jisun An: even though the architecture itself is powerful. But just efficiency might not be as good as compared to the other. And I think that's the reason that the modern large language model stick to the decoder only model, and that's the

57

00:13:53.610 --> 00:14:01.380

Jisun An: what you will see most of the current, I mean, the latest models are mostly using this decoder only model.

58

00:14:03.690 --> 00:14:25.634

Jisun An: So that's the where. But we have it here. So encoder only model is better at like natural language, understanding, task, and the example is the birth and the decoder only model. They are good at like natural language generation and the gpt and llama are the that family and the encoder decoder model, I mean the

59

00:14:26.170 --> 00:14:49.410

Jisun An: The transformer itself was presented as an encoder, decoder model, which were test to do the machine translation and some other. The sequence to sequence kind of tests. But then, after that, they were like T. 5, or the part are the 2 other model popular model that adopted this particular architecture, and it has really like the both advantages

60

00:14:50.979 --> 00:15:04.960

Jisun An: so just quick overview of each of this model. So the Bart is short for bidirectional encoder representations from Transformer. So now I hope that this name really gives you a clear idea.

61

00:15:05.830 --> 00:15:27.089

Jisun An: I mean, so basically, the way that the transformer architecture works is help to learn the context from the both directions. So they basically be a encoder that takes the like both the directions and enable to represent the text properly using this transformer architect

62

00:15:27.220 --> 00:15:50.899

Jisun An: and the bird paper itself was introduced. This diagram, where basically there are 2 phases, so you can pre-train a model with without any labels. So these are like a self supervision where you don't need a label for a text to train a model. But if you have just a text

63

00:15:51.170 --> 00:16:09.326

Jisun An: and you can create, use this auxiliary tests to create the language model which we called as a pre-trained. And then, once you have that pre-trained model, you can fine tune this model to for a specific task which they called it as a fine tuning and

64

00:16:09.990 --> 00:16:23.180

Jisun An: So this was the something that that was presented here. So in the pre-training once again the the architecture is more or less similar to what you've seen, the encoder only model. But then so what task

65

00:16:23.180 --> 00:16:42.230

Jisun An: they can use? I think that can also differ, and in Bert in particular, because they go with instead of predicting next to tokens, because I guess they really didn't care like the sequence themselves. But they just wanted to understand the context themselves. So instead of just

66

00:16:42.797 --> 00:16:45.662

Jisun An: using the task that is more

67

00:16:46.540 --> 00:16:49.579

Jisun An: better for the generation they use this masked.

68

00:16:49.830 --> 00:17:12.589

Jisun An: predicting the masked token. So within a sentence, they just masked some of the word of tokens in the sentence, and then they just ask to predict, what is it? So once again, they are like classification task, and also the word to use another task which is next sentence prediction. So that was the something that they also had.

69

00:17:13.704 --> 00:17:27.145

Jisun An: So they basically, they use these 2 auxiliary tests. In other words, so it doesn't require any like labels. But then, just using just the text information, they were able to train the birds

70

00:17:28.020 --> 00:17:37.370

Jisun An: and the pre-trained birth could have been like fine-tuned in different tech con. I mean tests. So that was the I guess.

71

00:17:38.055 --> 00:17:46.570

Jisun An: The birth is still very powerful, and there has been, though there are, like different variants of the births. And

72

00:17:47.616 --> 00:18:08.240

Jisun An: so this was the master language modeling. There are. Still, it is popularly used. It's a small enough easy to handle, and for any task that, like requires a very quick task worth is still, I think, performs as good as some of the large language models performance.

73

00:18:08.850 --> 00:18:28.011

Jisun An: and I think, like a few weeks ago or last last month, I think there was a yeah, there was a new model presented which is the modem. But so so I mean. So these are all variation of the birth, and they show, like the different kind of

74

00:18:28.760 --> 00:18:54.750

Jisun An: performance. And so far, I think the Roberta once again, architecture will be more or less similar, but they just tweak a little bit in the architect in their architecture and the way that they train, or using maybe the different data set the modem birth was something appeared nowadays, and then, basically, they are far more faster than the existing port. So if you ever need to use the bird encoder based model. Then the modem birth will be a good choice.

75

00:18:56.010 --> 00:19:05.590

Jisun An: And the as a family of the bird. Just, I'd like to mention quickly about this sentence part. So the sentence part was also another variation of the bird.

76

00:19:06.040 --> 00:19:22.001

Jisun An: but then their goal is to understanding better of the the sentence similarity. So they use the similar sentence pairs as a training set, and they use it to train this particular model.

77

00:19:24.270 --> 00:19:35.340

Jisun An: by by using this semi cmes network. So the the idea was super simple. So here the bird was. The pre-trained bird was used to

78

00:19:36.778 --> 00:19:49.661

Jisun An: to to be fine-tuned for each of the sentence they just get the the embeddings. And then basically, they were measured. How similar to sentences are, and then the similarity themselves was

79

00:19:50.040 --> 00:20:10.929

Jisun An: compared with their actual answer. And then the model was trained. I mean, they actually had a like different objectives. But just I'm I'm including here. They also formulated as a classification test, whether 2 sentences are similar or not. Like binary classification. They also had, like use the Triplet laws, where they create a triplet, where.

80

00:20:10.930 --> 00:20:30.259

Jisun An: within anchor sentence, they had a 2 different positive and negative example, where positive examples are more similar to the original example, and the negative examples are like dissimilar to that sentence. So they kind of use like 3 different objectives and tested it. And then sentence burst is really good at

81

00:20:30.590 --> 00:20:58.409

Jisun An: finding the which 2 sentences have similar semantics, and it has been widely used in semantic search and clustering, and the question and answering, etc. So once again we already went through this. I mean we practiced it with our 1st lab. If you need to encode a sentence sentence. Birth is also a good way, or you can also fine tune the sentence birth himself. If you have a particular purpose as well.

82

00:21:01.130 --> 00:21:22.459

Jisun An: and the t 5 is the encoder, decoder, model and I mean, once again, I'm just like like giving you very high level of like introduction of each of this model. But then they had this idea that I mean, these are presented in 2019. So it's actually only stage

83

00:21:22.943 --> 00:21:33.570

Jisun An: that every things are kind of developed their idea is that basically, every task can be represented as a text.

84

00:21:34.285 --> 00:21:38.100

Jisun An: Task, was their idea. So instead of

85

00:21:38.120 --> 00:22:05.340

Jisun An: so before the T. 5, every different task was trained differently. So if you think about the birth. So there's a pre-trained birth, and for each of different tasks they actually created different birth model. So if you have 5 different tasks, basically, you have 5 fine-tuned births. But then, now, T. 5 have an idea. If the the model may be able to. They can generate a. They can generate a generalized model where they can solve like different tasks

86

00:22:05.340 --> 00:22:07.690

Jisun An: and all tasks can be

87

00:22:08.869 --> 00:22:36.579

Jisun An: represented as our text. So they treated all nap task as text to text translation, and they trained the model, and apparently it was working quite well. So, even though. And also T, 5 is a still very strong model. If you need this sequence to sequence prediction like I mean, anything can be turning into probably text to text translation. But

88

00:22:37.270 --> 00:22:58.409

Jisun An: the key idea that T. 5 brought was that, oh, you can basically build a generalized model that can do everything or anything, so that this was the one of 1st paper that can bring that idea into the language model. So now we know that I mean Gpt can do many different tests. But that entire idea came from this paper.

89

00:23:00.460 --> 00:23:22.656

Jisun An: And then finally, the Gpt. They were trained, based on the decoder on the model. And Gpt, even though unfortunately, I mean, Gpt did not release any technical reports about how the model actually trained. So no one knows exactly what's actually happening. And there are only a few things that we know. And

90

00:23:25.132 --> 00:23:50.300

Jisun An: basically the differences between Gpt 1, 2, 3 for sure. Basically, they are getting larger and larger. So they using like larger tokens as on when they train this mode that they are using, like the larger number of the tokens, meaning that they probably need also more gpus and etc. But the and also they change it slightly.

91

00:23:50.640 --> 00:23:53.100

Jisun An: I'm just looking at whether there's any.

92

00:23:53.340 --> 00:24:14.429

Jisun An: So the if you compare, like Gpt-one and Gpt-two, the layer norm, the position of it slightly change it so, as we mentioned before, like whether the layer norm should be done before and after each of the attention or the fit forward was something debatable, and the Gpt-two actually take that pre normalization.

93

00:24:14.967 --> 00:24:26.079

Jisun An: But then, other than that, basically, they are getting bigger and getting bigger, was one of the main reason that they perform, or I'm showing the highest good performance on various tasks.

94

00:24:29.489 --> 00:24:33.581

Jisun An: And the one thing that I pointed out is the

95

00:24:36.142 --> 00:24:40.270

Jisun An: this comparison between the the original transformer versus the llama.

96

00:24:40.480 --> 00:24:48.629

Jisun An: so all the other architectures are more or less similar. But then Llama took the some variations of this.

97

00:24:48.670 --> 00:25:16.410

Jisun An: So, for example, instead of the post nom. They took the pre-num and the normalization, also the initial. Originally they used the layer norm, but they now use the Rms nom, and instead of the value in the fit world network. They use the Silu and also position encoding. They use now the relative position encoding, and these were the major changes that llama had, in compared with the transformer architecture, once again the decoder on the model.

98

00:25:17.390 --> 00:25:43.757

Jisun An: But then but then, even so, these are all kind of the efficiency for the developed for the efficiency. So here, if you look at this graph, basically here, the transformer plus plus is like llama, and basically the they build a stronger architecture, and that just became like 10 times more efficient in in training these models. So

99

00:25:44.310 --> 00:25:57.650

Jisun An: so the thing that I wanted to highlight is that transformer is very strong architecture, and there are a lot of work going on that each module and each part to be more efficient or more strong, but the core structure is the same.

100

00:25:58.760 --> 00:26:07.549

Jisun An: and these are some summarization of the advantages and disadvantages of each of this model. And I mean, you can have. Take a look when you have time.

101

00:26:08.280 --> 00:26:18.015

Jisun An: and if you want to look more deeper in the transformer, how it actually built, then they they have this really nice

102

00:26:19.820 --> 00:26:31.269

Jisun An: article that goes through like line by line transformer code and explain everything about it. So it'll be some interesting read if you're interested in this kind of like architecture of these models.

103

00:26:32.471 --> 00:26:37.930

Jisun An: Yeah. So this is the last page of the transformer slide. Do you have any questions.

104

00:26:39.940 --> 00:26:45.629

Jisun An: You need to be created by open or things like.

105

00:26:46.750 --> 00:26:51.880

Jisun An: no, the Gpt is by the open. AI, yeah. Yeah. Yes.

106

00:26:52.070 --> 00:27:14.300

Jisun An: So, yeah, like, I guess the architecture is architecture, and the model is now depending on various things like what kind of data they have which we will actually talk about today. So what kind of data they use, and how the decision of each of these components of the architecture, and like like some other things like running late, how many batch they use. So all these

107

00:27:14.470 --> 00:27:29.509

Jisun An: parameters, I mean, there's a like hyper parameter. There's a loanable parameter, and there are like hyper parameter that requires for train. This model? So the yeah, there are a bit difference between the architecture and the model itself. So. But I may use it instead.

108

00:27:29.970 --> 00:27:31.949

Jisun An: both light. But hopefully, yep.

109

00:27:32.090 --> 00:27:37.720

Jisun An: So the Gpt is the the model that is created by the open data.

110

00:27:40.000 --> 00:27:42.250

Jisun An: Thanks for the question. Any other question.

111

00:27:45.240 --> 00:27:52.740

Jisun An: Okay, so then let me move to E

112

00:27:53.467 --> 00:27:56.870

Jisun An: pre-training and pre-trained model. So

113

00:27:59.550 --> 00:28:28.093

Jisun An: so now we know that the the architecture of I should have brought the water. The architecture of the model itself. But then, now if you train the model just with this architecture, the research that the generating output will be quite different from what you are having at the moment. So there are a few more steps that to achieve the performance that you are getting from like Gpt or cloud, or the other models.

114

00:28:28.680 --> 00:28:43.839

Jisun An: so so 3 steps of creating a high quality. And M would be so assuming that you start from some untrained, basically nothing there. But then, now you have this using the language, modeling what we just discussed like transmor based.

115

00:28:43.840 --> 00:29:10.159

Jisun An: And now you have base added them. And usually this training is done with this unlabeled data, and then you can use auxiliary task to train the model like next token, prediction. Next, sentence, prediction or vast token prediction. So these are all auxiliary tasks that once again, you don't require any label data. You just need a text, and you can train the model to learn about the language itself, which is the language modeling that we do.

116

00:29:10.190 --> 00:29:14.619

Jisun An: Then, once you have this baseline, another thing that they do is

117

00:29:15.266 --> 00:29:30.730

Jisun An: they basically do fine tune. This model and fine tune means that, given the base model. Now, you will once again update some parts of these networks, parameters to be to do some more specific, to do something more specific.

118

00:29:31.120 --> 00:29:57.809

Jisun An: And and in the high quality Edm case they fine tune this base model to follow a particular instruction or instructions. And these interest instructions are basically what what we would prefer to see. So you probably seen that something was instruction tuned. So it other in other mean. In other words, that would mean that this Rrm is fine tuned with the instruction data. And this instruction data would be so

119

00:29:57.810 --> 00:30:06.657

Jisun An: like chat, Gpt voice example. That's like the instruction tuned model and instruction. Tuned means that if we so, for example,

120

00:30:07.536 --> 00:30:36.840

Jisun An: I mean, they basically they will follow if you let the language model just to create the text that they will just keep, keep, generate the text that they think that most likely to come up. But then I mean but then that may not be the best way to interact with the human. So the instruction data set, at least basically gives. Okay, these are the types of conversation that we'd like to have. And and basically the model will be fine-tuned to that.

121

00:30:37.940 --> 00:31:07.849

Jisun An: and then there are like the free preference tuned. So instead of like having a set of the instructions, now, the preference tuned is also kind of fine tuning. But here, now, using more like preference data. So even for the same output, you may have like better preference to this particular answer over the other one, and given that information now you can even fine tune the model more and that would be the preference tuned.

122

00:31:08.300 --> 00:31:18.679

Jisun An: And this preference tuning is done in various way. But one of the way was using the reinforcement learning. And that's something that we will talk in about 2, 3 weeks.

123

00:31:20.330 --> 00:31:23.700

Jisun An: So what you've interacted with

124

00:31:24.064 --> 00:31:52.680

Jisun An: the larger language models are the outcome of these 3 steps. So they created, based on the language modeling. And then they instruction tuned, meaning that they will basically answer to as our we prefer. And then I mean the prefer means that they will follow some template that we would interact with easily. And then the they also tuned based on the preference. So I mean, basically, the one of the main reason that this preference tuned came out is because

125

00:31:52.690 --> 00:32:13.700

Jisun An: without it they can easily generate something very harmful or something very toxic, something that's not desirable. So this was kind of the steps to ensure the safety of the Lms. And etc. So so that's the how the this preference shouldn't kind of come out. But we will talk more about it in the following lecture.

126

00:32:14.380 --> 00:32:17.449

Jisun An: So there's an example. So assuming that if

127

00:32:18.054 --> 00:32:24.925

Jisun An: we give a question like, what is one plus one, then this base model. Basically may

128

00:32:25.950 --> 00:32:44.659

Jisun An: generate like one plus one plus one plus one plus and double ending because it. It's I mean, that's probably these are the tokens that are most probable to come next, based on this model. After it's instruction tuned, then it may be able to just generate 2. Because now, with

129

00:32:44.660 --> 00:32:58.839

Jisun An: the different data set of like question and answers, the model is fine-tuned. So now they know that. Okay, okay, this. Now they can understand. This is the question that requires an answer, a particular answer, so the instruction tuned model will be able to generate 2,

130

00:32:58.840 --> 00:33:15.629

Jisun An: and then the preference model. I mean, this may not be the best example to example the I mean demonstrate the preference. But basically, instead of the 2, maybe you prefer to have, the answer is 2. And then this kind of thing can be also learned and fine-tuned to the model. Further.

131

00:33:16.410 --> 00:33:18.669

Jisun An: so these are the 3 steps

132

00:33:18.830 --> 00:33:35.710

Jisun An: of the high quality edit them. And so just for a definition. So the pre-training is training a model on a large data set to learn general patterns, and the representation supervised fine-tuning is now train the model to learn test specific capabilities.

133

00:33:36.070 --> 00:33:42.139

Jisun An: Instruction. Fine-tuning is basically is the kinds of disprovised learning. But

134

00:33:42.440 --> 00:33:50.270

Jisun An: they train the model to follow user instructions. So basically, they're using different data set and and they'll code it as an instruction. Fine tuning.

135

00:33:50.270 --> 00:34:15.170

Jisun An: but nothing different from the normal supervised learning. And then the preference, fine tuning is using labeled preference data to fine tune an M, and you probably heard the alignments here and there, and the alignment is, basically, it's a general notion of training model to meter the user desire. So whether the model, whatever model generates, whether they are aligned well with our desire is.

136

00:34:15.170 --> 00:34:23.960

Jisun An: see the alignment all right. But any any question.

137

00:34:26.020 --> 00:34:27.889

Jisun An: Right? So

138

00:34:30.010 --> 00:34:44.160

Jisun An: so so we will talk a little bit about the pre-training today. How the edit Ms. Are trained, and as a like, we will use llama as an example. And and one of the reason is because the like the

139

00:34:44.580 --> 00:35:06.790

Jisun An: the commercialized models. Basically, they haven't disclosed any anything about their model. So there's not not much information about it. Llama, at least have some information how they they train their model, so we will base at it. But we assume that I mean the goal is to understand what are the things that is required to pre-train these, and

140

00:35:06.820 --> 00:35:19.960

Jisun An: there are. We will introduce, like bunch of other existing items, and they will more or less the same with a little tweaks but so hopefully, this will be a good guideline for that free training.

141

00:35:20.860 --> 00:35:43.850

Jisun An: So I mean, I just borrowed this image again. That why is it for the pre-training, because it's weird, right? Because it's training the model of a live pre-training. And the notion is coming from this birth paper where they basically split this. I mean, this was the 1st paper probably that distinguished between these 2 steps. So and then they decided to code it as a pre-training, and the fine tuning.

142

00:35:43.850 --> 00:35:51.080

Jisun An: and that's the reason that it became pre-training. And we are keep using this word as a pre-training. But it's just a training the model.

143

00:35:51.080 --> 00:35:57.480

Jisun An: So we have the architecture and the llama architecture that I will show you later. But

144

00:35:57.830 --> 00:36:16.759

Jisun An: to train this large language model, the most important thing is the data, right? So you need a lot. And and this model, you, you need a lot of data to and that would only able to make the model to learn all different relationship of this language.

145

00:36:17.390 --> 00:36:44.600

Jisun An: So in llama case, they basically use the web data that can be collected on the web. So the common core is something that I mean, basically see everything that you see the Internet. So they started from some seed website. And then they collected all the, all the other pages that were linked back to that page, and then they do like snowballing kind of collection, and these are the common core. I mean, basically, they are like HTML files. And then they extracted all the text, and then they keep that in the way.

146

00:36:44.600 --> 00:36:54.449

Jisun An: And they also using, like like the data from the Kita or Wikipedia and the books and archives and State Exchange. So these were some of the sources that the llama was was using

147

00:36:54.770 --> 00:36:56.489

Jisun An: for training their data.

148

00:36:57.532 --> 00:37:03.860

Jisun An: So in oh, and

149

00:37:04.800 --> 00:37:24.699

Jisun An: and so llama one, basically so surprisingly. Lama one was released in 2023. So it's not that long ago. It's very recent, but I guess just last 2 years was crazy with the edit them. So so Llama one appeared in 2,003, and it was trained with the 1.4 trillion tokens.

150

00:37:25.020 --> 00:37:50.650

Jisun An: And just I wanted to mention how big is this 1.4 trillion tokens. So if we so basically, if we convert these tokens into a books, then it became basically 10 billion books. And if we assume that if we compare it with like our reading, then so on average person read about 200 words per minute. So this would mean that we will need to

151

00:37:50.890 --> 00:38:12.835

Jisun An: read 16,000 years of nonstar reading to process that much text. So the llama one is still very small and very old model, but still that was trained with the 1.4 trillion tokens. So basically, this is something that human cannot do right. And I think there's the beauty of the language models.

152

00:38:13.440 --> 00:38:17.860

Jisun An: they just learn, read so so much more than we do.

153

00:38:19.080 --> 00:38:39.857

Jisun An: And, as I told you, they used this multiple sources to train the model. But they actually did a sample from the reliable sources. So if you think about like common core, these are just random web pages on the Internet, and not all languages or the contents would be useful or or

154

00:38:42.141 --> 00:38:51.359

Jisun An: so instead of just I mean as all like, once more step they up sampled the or train more with the Wikipedia and the book data.

155

00:38:52.118 --> 00:38:55.950

Jisun An: So that that was one thing like particular from the llama.

156

00:38:56.090 --> 00:39:15.030

Jisun An: And then, in terms of the tokenizer, they use the byte pair encodings. And then, yeah. So I mean, and then they use this implementation from the sentence piece. So the sentence piece had the 2 implementation, the by code by pair encoding, or the unigram model and llama just happened to choose this one.

157

00:39:16.170 --> 00:39:22.446

Jisun An: and here the right side is the architecture of the llama and

158

00:39:24.310 --> 00:39:46.900

Jisun An: and so sorry the the Pixar font is really small, but so the left one is just our transformer, original transformer, and the right one is the llama architecture and the the and I also had that table that show the differences between the llama and the original transformer, and they are like more or less the same as we see the decoder only model.

159

00:39:47.400 --> 00:39:51.740

Jisun An: But then there was one significant difference is this group

160

00:39:51.860 --> 00:40:03.820

Jisun An: query, attention. So here the the orange box that you are seeing? They are using. This is the attention. But then, instead of the the attention, the multi multi head attention they used.

161

00:40:03.980 --> 00:40:11.730

Jisun An: I mean, they still using multi-head attention. But they use this multi grouped multi-courry attention, mechanism, which is

162

00:40:12.173 --> 00:40:29.970

Jisun An: so. And this was for the efficiency. So if if you know, like normal multi-head cases, you would have create the weight matrices for each of the query and key and value, and if you have n head, then you will have

163

00:40:29.970 --> 00:40:42.723

Jisun An: and like multiply by 3, and if the weight matrix is like, I don't know how big that would be if, like 1,000 dimensions, then basically, you will have that many

164

00:40:43.865 --> 00:40:57.064

Jisun An: parameters to train. So basically, these are very expensive model to train because it has so many parameters. So as an alternative. So someone wanted to reduce the number of parameters

165

00:40:58.864 --> 00:41:07.940

Jisun An: to handle this multi multi hat. And then so the 1st idea was emergency query. So instead of having

166

00:41:08.210 --> 00:41:18.219

Jisun An: separate keys or separate value, they decided, okay, we can have just multiple queries, but one key and the one value embedding. So this was the multi key attention.

167

00:41:18.580 --> 00:41:41.709

Jisun An: And so literally, this will reduce the number of parameters that is required, but at the same time it also loads the performance because each of the head, the key and value weight will learn something different. But then, basically, you are removing them. So, even though in terms of the computation, this will be efficient. But then probably the performance may not be as good as the mercy had.

168

00:41:42.260 --> 00:42:00.770

Jisun An: But so the group query is basically kind of meet in the middle kind of idea. So it doesn't have the the weight metrics for every key and value. But they decided to just group some of the queries, so they just share the weight metrics for the key and value for multiple queries.

169

00:42:00.880 --> 00:42:19.379

Jisun An: So in a way that it's just once again, these are the choice for the efficiency. When you infer or even train as well. Because if you have, this means that basically, you have more parameters and more parameter means that you will take more time to train, and also, more importantly, when you do the inference

170

00:42:19.380 --> 00:42:36.269

Jisun An: for each of the next token, you need to go through that the transformer architecture. So you need to compute the attention for all the words that you, all the tokens that you have to predict the next token. So if you have, like smaller number of key and value

171

00:42:36.470 --> 00:42:54.429

Jisun An: vectors. The weight matrix is then basically the the times. To compute all these attention will be less, and then there will be efficiency in the inference. So so this was the main reason that why they came up with this different mechanism of the the attention and the llama used this query, attention.

172

00:42:55.730 --> 00:43:18.370

Jisun An: So here, basically, these are the Mha is the multi-head. Gq, is the group query, Mq is the multi-courry and the large model and extra extra. I don't know how it how to pronounce it is. Xxl is like the largest model, so obviously the multi-head, and with the largest model would

173

00:43:18.370 --> 00:43:43.130

Jisun An: perform the best. So the X axis is performance. Y axis is the time per sample for the inference time. So obviously, if you are using full multi head model attention, model architecture, then they need to perform the best, but at the same time it will also take long time to inference. But if you are using the multi query or the grouped query, the inference time is now getting far less. But then

174

00:43:43.310 --> 00:44:04.899

Jisun An: but then, once again, the multi query will basically I mean, they give up some performance right? So but then the group query, they found that they are compatible with the multi hat performance. But at the same time achieved a very fast efficient inference. And here basically shows that the the large model and the Xxl models the differences. So so.

175

00:44:04.900 --> 00:44:17.770

Jisun An: And if you are using smaller model, the inference time will be faster, because smaller model means that basically, you have less number of head. And so, even though you have multi head, you will have, like less number of the head and etc.

176

00:44:19.172 --> 00:44:24.487

Jisun An: So that was the main idea of the grouped Cory. Attention and Lama kind of take this

177

00:44:24.960 --> 00:44:27.929

Jisun An: architecture into their architecture.

178

00:44:29.300 --> 00:44:31.439

Jisun An: Oh, any questions here?

179

00:44:38.640 --> 00:44:39.330

Jisun An: Right?

180

00:44:39.640 --> 00:44:51.560

Jisun An: So right great but different queries are going to

181

00:44:52.450 --> 00:44:55.040

Jisun An: have the same result when you have any.

182

00:44:55.820 --> 00:44:58.235

Jisun An: So so so basically,

183

00:45:00.260 --> 00:45:10.789

Jisun An: I I guess. I guess it's it's hard to explain in like plain English, but just in terms of the computation. So

184

00:45:11.280 --> 00:45:14.585

Jisun An: for so

185

00:45:22.380 --> 00:45:27.207

Jisun An: right? I I understand what what you got confused here.

186

00:45:28.470 --> 00:45:34.250

Jisun An: so there are. So multi head means that when you have a new sentence.

187

00:45:34.967 --> 00:45:56.379

Jisun An: and then this new sentence will go to end. Times, I mean will be an input to the end. Different head, and then and then do, and then going through the key value and attention. Computation and get each multi head will learn something different.

188

00:45:57.121 --> 00:46:05.359

Jisun An: But then, in this case, basically, the key and value will not have those weight metrics to learn different things.

189

00:46:06.300 --> 00:46:15.699

Jisun An: So there will be just one giant wave matrix, where serving as a key and value embeddings.

190

00:46:17.120 --> 00:46:28.899

Jisun An: Does that make so each each column is the each head?

191

00:46:34.130 --> 00:46:40.640

Jisun An: Oh, no, no! These are each head or use Google tab very clear about you.

192

00:46:40.970 --> 00:46:50.389

Jisun An: So for for each of token they will have multiple queries, but they will only have 1 1 key and one value

193

00:46:52.690 --> 00:46:59.959

Jisun An: like, are all the queries representing the same content, but learning different aspects of the same content? Or is it different?

194

00:47:02.320 --> 00:47:14.719

Jisun An: Oh, so yes, you. You're right. So so they are all the same. I mean, they are multi head. It means that the same token will go through all the queries. But then, yeah, same keys and same.

195

00:47:15.990 --> 00:47:23.390

Jisun An: it is not saying in office, okay.

196

00:47:24.630 --> 00:47:29.240

Jisun An: But I I wonder what would be the better way to describe this.

197

00:47:29.540 --> 00:47:34.945

Jisun An: So just remove s, 20, right? Right? Right?

198

00:47:36.160 --> 00:47:53.129

Jisun An: like, like the blue rectangles, the multi clear art, the blue rectangles all like different inputs, or like, if the multi head is is that you're you put the same input 2 end times. Yeah.

199

00:47:53.560 --> 00:47:59.360

Jisun An: right, is it same info going into?

200

00:48:00.160 --> 00:48:04.460

Jisun An: But so okay, so so the

201

00:48:05.250 --> 00:48:34.233

Jisun An: 1st coming back. So in the multi head case each column is multi head, and the same token will be, go to all these 8 different path, and this 8 different path will learn something different. But then, in the multi curry the same token, we also go to all the 8 queries, but then they will, just when they compute, like what to their relevance for or their value, they will just use the same

202

00:48:35.090 --> 00:48:40.539

Jisun An: same key and value weight metrics and the embeddings.

203

00:48:40.810 --> 00:49:03.979

Jisun An: So basically. But still query will have more more. I don't know whether it's a steel code as a multi multi head. So. But still the query, there will be 8 different different weights, so they will learn something about the difference of the keys, but then eventually the they will share the key and value embeddings the weight matrixes.

204

00:49:08.560 --> 00:49:10.000

Jisun An: There are multiple.

205

00:49:10.320 --> 00:49:17.180

Jisun An: So they're getting invited if sorry, I don't understand

206

00:49:18.930 --> 00:49:21.769

Jisun An: he is joining what she can do.

207

00:49:22.060 --> 00:49:26.280

Jisun An: Right? Right? Yes, yes.

208

00:49:27.470 --> 00:49:34.969

Jisun An: In group query, we have multiple for and divided in value.

209

00:49:35.180 --> 00:49:38.350

Jisun An: We have multiple queries, that one

210

00:49:39.400 --> 00:49:44.316

Jisun An: right? Yes, yes, that's correct.

211

00:49:46.610 --> 00:50:01.589

Jisun An: So maybe it'll be easier if I get like some how the maybe the quote that, how it actually works. So maybe let me come back to this like next next week, maybe. I will try to find a better way to come

212

00:50:01.720 --> 00:50:04.959

Jisun An: displaying this one. Yeah, yeah. Yeah.

213

00:50:05.470 --> 00:50:10.639

Jisun An: But just I'm also worried like when they doing this group query, how exactly they this?

214

00:50:13.390 --> 00:50:23.760

Jisun An: Yeah, I mean, in in terms of the implementation, I think, is is doable. Basically, you are using the same key weights to compute the relevant score.

215

00:50:25.410 --> 00:50:38.049

Jisun An: So in the multi head case, you have each key embedding also have different key weights, weight metrics. So your relevance score will be different for each of the

216

00:50:38.820 --> 00:50:41.070

Jisun An: curry and key combination.

217

00:50:41.790 --> 00:50:53.159

Jisun An: and also in the multicore example the value will be different because of the key. The query has the weight metrics. They are all different, but then the keys weight metrics will be just one.

218

00:50:55.380 --> 00:51:15.130

Jisun An: So in my interpretation, just just they will learn far less than the multihead, because multi head have 8 different 8 different key weights and 8 different value weight matrixes which can enable the model to learn different aspects of the language.

219

00:51:15.310 --> 00:51:21.600

Jisun An: So the easiest way to understand this is, they just have far less

220

00:51:22.170 --> 00:51:47.139

Jisun An: parameters to to learn learnable parameters. So the multicore is still enabled to compute the relevant score and compute the weighted sum from the value vector and then getting the final output. But just there are far less parameters there. So like once again, it's the trade-off between the performance and the time for the inference.

221

00:51:48.910 --> 00:52:06.269

Jisun An: But let me check. If there's any better way to explain this, or or how we should interpret this, I think that's the I mean. So I I thought this was more for efficiency. But maybe there are something more that it leads to the interpretation of how the attention actually works.

222

00:52:07.130 --> 00:52:09.220

Jisun An: which I'm not aware at the moment.

223

00:52:10.430 --> 00:52:35.960

Jisun An: All right. And and these were some other setups and basically, when Lama came they had 4 different models with the different parameters with the different dimensions. So basically, the number of head. So what it mean by the model is getting larger is basically they have bigger dimension. They have more head. They have more layers in the feed for network. So that's the how the model gets

224

00:52:35.960 --> 00:52:42.610

Jisun An: larger and larger. And if you have larger model once again, you have more parameters to tune, and

225

00:52:42.610 --> 00:52:53.679

Jisun An: and there will be better way to capture different combination of all these parameters. And so that's how the language model learns about the language. So I mean, many people say that like

226

00:52:53.710 --> 00:52:58.590

Jisun An: these models are working like a magic. And yeah, sometimes I also feel bad in in that way.

227

00:52:59.217 --> 00:53:04.389

Jisun An: But then, yeah, so these were some of the details. And yeah.

228

00:53:05.080 --> 00:53:32.790

Jisun An: And I mean, there are like details like optimizers. But I mean, unfortunately, we, I mean, we will not talk much about these pre-training parts, I think, because I need to kind of keep the balance between this and also the application. But, all these parameters are needed for training these models, and it's a lot of efforts of just try and editor, and they try to find the best parameters for their own models.

229

00:53:34.860 --> 00:53:54.919

Jisun An: and the the only thing is so. The batch size here is like 4 million. I just wanted to point that out. So the one thing that that to note is that basically only like Gpu, even for the CPU like 10 operation of size, one is much slower than one operation of size. 10.

230

00:53:55.390 --> 00:54:20.000

Jisun An: So basically, they combine all the small operation into a big one, and apparently that is far faster. So they try to when they try to compute something, they are batching the examples together. And now, and also, once you have this, so Batch means that at one step. How many examples you are using to train the model. So we so far we have talked about

231

00:54:20.150 --> 00:54:25.741

Jisun An: how each token actually going through this transform architecture, but

232

00:54:27.820 --> 00:54:42.178

Jisun An: but but basically, when in actual training, they have 4 million batchy size for each step, meaning that they probably need a lot of Gpus to

233

00:54:42.750 --> 00:55:00.858

Jisun An: to train this model. The llama trained on the Nvidia a 100, and they have 4 different parameters, and the largest one, the 65 billion parameter. One was trained using 2,000 gpus. So the 2,000. And and that's the

234

00:55:01.730 --> 00:55:08.200

Jisun An: probably something that was needed for the processing 4 million tokens at once.

235

00:55:08.832 --> 00:55:12.370

Jisun An: So these were so that's, I think.

236

00:55:12.520 --> 00:55:16.046

Jisun An: very interesting. I don't know like

237

00:55:17.330 --> 00:55:22.320

Jisun An: a lot of efforts out as well. Because the

238

00:55:22.900 --> 00:55:35.649

Jisun An: when you kind of I mean it also, it takes a long time so there could have been stopped at after like running for a few months, and then you probably need reruns, or there was a lot of prior errors as well.

239

00:55:36.580 --> 00:55:56.139

Jisun An: And another thing is that they also use the B plot. Bf, 16, precision for optimized performance and the memory efficiency. Once again, this is another efficiency trick for training the large length model and the here. The B flot is the short for brain flot.

240

00:55:56.140 --> 00:56:06.869

Jisun An: I don't know why the name is coming from. So the floor 32 is something that have been commonly used. So basically to represent our the floating points.

241

00:56:07.300 --> 00:56:31.309

Jisun An: you can use 32 bits to represent them. And here the 1st one is usually the sign whether it's a plus or minus, and the exponents, the 8 bits are good for the representing the exponents and the exponents is now determines the range of how many, how large range of numbers you can represent using the computer and the rest of 23 bits were used for fractions.

242

00:56:31.850 --> 00:56:57.739

Jisun An: And so but then there's also like the version of the flood. But then this means that basically for representing one number, you need basically 32 bits. And once again, if you think about the model, then if it gets basically bigger than the model basically requires, like the 2 twice bigger memory to be trained. So there are also there were some efforts that trying to like

243

00:56:57.740 --> 00:57:03.399

Jisun An: by using this different plotting point format, they try to find some efficiency

244

00:57:03.570 --> 00:57:24.060

Jisun An: in terms of the memory use. So the float 16 would be one option, and this will literally reduce, like the memory uses in the half, to train the model or doing the inference as well. But then the float 60, 16, using 5 bits to represent these points, numbers meaning that and then, and the 10 bits for diffraction, so

245

00:57:24.250 --> 00:57:43.820

Jisun An: meaning that the range that you can represent is far smaller than the plot. 32. And the the brain plot basically using 8 bit for the exponents and 7 bit for the fraction. So it keeps the the range of the number that they can represent. But they reduce basically the precision. So instead of

246

00:57:43.820 --> 00:57:56.361

Jisun An: so, for example, like plot 32 can well represent 0 point 0 7 6 5, 6, 5, 7, 4. But then maybe the B flot may not be able to represent exactly that, but probably

247

00:57:56.710 --> 00:57:58.220

Jisun An: like I mean some

248

00:57:58.380 --> 00:58:14.736

Jisun An: giving up some precision. But then they found that for the training the large language model. The people are working better. So having the diverse range was more important than being the precision. So there was also some research that looking at what is the benefits and the

249

00:58:15.270 --> 00:58:22.050

Jisun An: And they found that the beef lot is. And so that became one of the common practice that used for training these models as well.

250

00:58:23.953 --> 00:58:43.429

Jisun An: And then, yeah. So this graph shows the training loss over the number of the tokens that were trained for different size of the models. And here the training loss was measured by the Cross entropy loss, because it was the next token prediction task.

251

00:58:43.470 --> 00:58:56.899

Jisun An: So it says once again, that's the classification task which can be measured based on those loss. And yeah, they will. Basically, you see that the loss is getting lower and lower, and I don't know. They stopped at certain points.

252

00:58:58.830 --> 00:59:04.110

Jisun An: So so this was the how the llama was trained, and

253

00:59:04.980 --> 00:59:08.160

Jisun An: many the other models may have, like different

254

00:59:09.113 --> 00:59:33.130

Jisun An: the changes in the architecture, or these parameters, but the the framework to train the model is more or less similar to this, and even though we don't know how the the commercial ones are actually trained, but but these are only the pre-training, so we haven't went for the instruction tuning or the the preference tuning. But these are just the pre-training part.

255

00:59:34.954 --> 00:59:49.035

Jisun An: And so regarding the pre-training, there are a few concepts that I'd like to discuss scaling loss and the emergent behavior. So the scaling loss is something that was also

256

00:59:49.800 --> 01:00:07.330

Jisun An: presented by the Openai and the mind that they essentially what they found is that the performance of the language model is just getting better and better with more computation and larger data set and more parameters.

257

01:00:07.922 --> 01:00:16.137

Jisun An: So that was. And and that I mean that relationship was explained, based on the power law.

258

01:00:16.900 --> 01:00:42.010

Jisun An: So the scaling law of the larger models trained on more data, with more compute to lead to better performance. And this predictability found the power law relationship. So this was presented in 2020, and since then it was one of the dominant law that was governing this pre-training world, which has been changed slightly recently. But

259

01:00:42.847 --> 01:00:47.379

Jisun An: so the key insight from the scaling law is

260

01:00:47.630 --> 01:01:17.200

Jisun An: basically, if you make the model larger and larger and using more and more data, then the performance will get better. So, in other words, that was the major reason why the Nvidia has been has been so successful, and their stock has been increasing because people thought. And now people want to create an edit them that are really generalizable for any test. And to make such large language model you basically the larger model, meaning that it needs more higher computation, power, meaning more gpus, right?

261

01:01:17.580 --> 01:01:19.889

Jisun An: And that was the main reason that

262

01:01:20.905 --> 01:01:26.828

Jisun An: I mean, they. They found that the performance was like following this. And

263

01:01:28.160 --> 01:01:32.709

Jisun An: That was the one of the reason that why the gpu became so important.

264

01:01:33.354 --> 01:01:36.826

Jisun An: Which which actually change it a little bit.

265

01:01:38.970 --> 01:01:46.290

Jisun An: Then the another thing that I'd like to mention is the emergent ability of the rrm.

266

01:01:47.150 --> 01:02:13.210

Jisun An: so, people, I'm still. I think it's a still very work so well, and I don't think this is the true explanation. But some people say that potential explanation is emergent abilities and the emergent abilities that the ability is emergent if it is presented in a larger model, but not in the smaller model.

267

01:02:14.270 --> 01:02:30.160

Jisun An: So it was very. There was very weird phenomena that it hasn't been really predictable. And there was no way that you can explore extrapolate from the smaller models. But some performance was just

268

01:02:31.140 --> 01:02:37.740

Jisun An: basically, it was near random performance until certain threshold, and then it improved heavily at some point.

269

01:02:38.252 --> 01:03:06.209

Jisun An: So these are the paper that presented this emergent ability. And the different graph here shows the different tasks, like like reasonings, or like a multi-step reasonings like math or even human understanding. So these are testing on different data set. And X-axis is like model scale. Here, in in terms of the computations, how many computation it requires to to train that model. So you you see that this, like.

270

01:03:06.210 --> 01:03:10.935

Jisun An: like steep increase in the performance for every different

271

01:03:11.870 --> 01:03:15.120

Jisun An: task. And this is the something very

272

01:03:15.865 --> 01:03:30.509

Jisun An: unexpected. So basically, this ability is not linearly increasing, but it was just random wasn't random, and at some point it just at. If the the model gets large enough, then suddenly, this ability just came out.

273

01:03:30.890 --> 01:03:40.272

Jisun An: So it's some interesting phenomena and not fully explained yet, and I think people trying to find out what's actually going on, but I think there's no

274

01:03:40.910 --> 01:03:56.589

Jisun An: great answer. So you you've heard about like 0 shot prompting like few shot 0 shot learning, future learning. This is also part of the emergent ability. So few shot learning means that the that the large language model can

275

01:03:57.452 --> 01:04:13.570

Jisun An: can infer based on a few examples. And then once again, this view shot also was not able to do within a smaller model, but for the larger model. Suddenly it was possible to be done

276

01:04:14.300 --> 01:04:19.370

Jisun An: so. These are like the 2 characteristics that that you can observe from this pre-trained model.

277

01:04:21.610 --> 01:04:37.558

Jisun An: and I mean, there are open and the closed models, and depending on whether they are open for weights or code or training code, or the data it can be or differently defined. But

278

01:04:38.677 --> 01:04:52.942

Jisun An: and another thing is, there are also for each of these model that you will use. There are having may have different licenses. So these are something that if you go to industry and start to use this model for your own company, this would be very important.

279

01:04:53.680 --> 01:05:02.589

Jisun An: some have, like mit, which is like restricted but very few restriction. And it basically, I mean.

280

01:05:02.890 --> 01:05:10.675

Jisun An: different. Model would have different license, their own license, for example. So, for example, llama

281

01:05:11.310 --> 01:05:27.630

Jisun An: they they generally allow for the general purposes of the use. But then, if you want to commercialize it, using the llama model, then you may be restricted. So yeah, so make sure that what kind of license each of models are using and using it based on your own purpose.

282

01:05:28.830 --> 01:05:34.359

Jisun An: And another thing that I'd like like to mention is the fair use. So

283

01:05:34.440 --> 01:05:49.180

Jisun An: there has been a lot of discussions about. So to train these models, they use a lot of data that collected from the Internet. And probably you've heard that there are many lawsuits and discussions about whether it's really fair to use all this data to train this model.

284

01:05:49.240 --> 01:06:11.489

Jisun An: And the Us. Fair use doctrine is they can use like copyrighted material in some cases, so they allow to certain extent. So I mean, these are very simplified version. But like quoting of them may be okay. Maybe it doesn't diminish any commercial value. Possibly okay. And use for non-commercial purposes, possibly okay. And then.

285

01:06:11.680 --> 01:06:28.020

Jisun An: Now, whether the model training if and I mean, people are using using them for model trainings are kind of fair use, but but still I think it's still very debatable, and there are a few lawsuits going on from the New York Times and the Github.

286

01:06:28.920 --> 01:06:52.359

Jisun An: I'm I don't know. The stack overflow or stack overflow is one of also the thing that's going on with the lawsuit, and I don't know whether you realize it. But probably before the Gpt, you probably use that overflow a lot to ask questions. But now, who who goes to the stack? Overflow? Right? So I think there are big debates and lawsuits going on so something that to keep in mind when using these models.

287

01:06:53.632 --> 01:07:17.339

Jisun An: and the closed model, basically, they are restricting access to their model. And why do they do? I mean, there are obviously commercial concern. They want to earn money right if they have better model than they want to use it for. Make more money. But also there are like safety issues. The editing can be so powerful to be used in very harmful ways. So they also want to prevent that aspects. And also because of this copyrighted data, there could be some

288

01:07:17.430 --> 01:07:25.559

Jisun An: if they release everything. What data they have used. Then there could they probably be not like legal kind of issues. So these are one of the regions.

289

01:07:26.580 --> 01:07:31.969

Jisun An: So from here onward, these are a list of pre-trained model

290

01:07:32.660 --> 01:07:43.149

Jisun An: and these will be just optional slide, and I will just leave it to you to go through to this slide, because I don't think it is very important, but just going through

291

01:07:43.150 --> 01:08:05.820

Jisun An: to see what existing there. So we have open source model. These 3 are fully open. Everything is open, you can download these models and use it, change it based on your purpose. The open weights. They only open the weights themselves so you can still use it to infer, but you may not be able to find tune or change the model, the closed one. Nothing is opened, so you just need to use their Apis

292

01:08:06.170 --> 01:08:13.650

Jisun An: there are, and also once again go through at your pace. But I'd like to just find out a few things

293

01:08:15.276 --> 01:08:16.630

Jisun An: here. So

294

01:08:17.080 --> 01:08:26.419

Jisun An: and also so basically llama 2, the difference between lama one and 2, they tuned for safety needs

295

01:08:26.529 --> 01:08:28.269

Jisun An: strong. They did a like

296

01:08:28.510 --> 01:08:49.369

Jisun An: strong safeguards, and basically, they use different data set on safety, and they use the reinforcement learning to train. And these were evaluated, based on the helpfulness and the harmfulness. And they found that as they run over the run reinforcement learning basically helped a lot to

297

01:08:49.710 --> 01:09:04.269

Jisun An: to improve their safety needs. Once again, we will talk about the reinforcement learning from the human feedback in week step 7. So we will come back to here. But just wanted to just mention this quickly.

298

01:09:05.479 --> 01:09:22.130

Jisun An: and Lama 3.1 is more or less similar to Lama 2. But they just use larger data like more more training, larger model. So nothing very fancy. And also they. They also support multilinguality, and they also know multimodal as well. But I will not talk much about it.

299

01:09:22.720 --> 01:09:48.599

Jisun An: The Mistra is something from the Mistre AI and something once again, I mean, if now I hope if you read in, read them, then you should be able to understand most of it, because, like transformer rob, seek glu context. 4 K rms, no, right? I mean, I hope, that you are understanding. But but these Mistra actually had a 2 interesting architecture which I will just introduce. They have one sliding window attention.

300

01:09:49.310 --> 01:10:05.520

Jisun An: So the idea here is very simple. So so within the entire sentence, instead of focusing on giving attention to, or tokens in the sentence. They will just have a particular window, and they will just give the attention to a small window.

301

01:10:05.650 --> 01:10:14.790

Jisun An: makes sense right? And especially if you have a long input then you don't need to calculate attention for the entire long sentence, and

302

01:10:14.790 --> 01:10:38.189

Jisun An: probably the most important context will be nearby. So they just decided to focus on the narrow window. So that's the idea of this sliding window attention. In other words, this will reduce the computation a lot. Right? So when you compute the attention, the relevant score it was against from one query to all the keys, meaning that the all the tokens of the sentence. But if you are reducing that since that window.

303

01:10:38.190 --> 01:10:40.999

Jisun An: then you will need to compute far less than that.

304

01:10:41.000 --> 01:10:44.889

Jisun An: But that was one idea. Another thing is the mixture of experts.

305

01:10:45.230 --> 01:11:05.850

Jisun An: And here the idea is something also very interesting. So instead of the feed forward network, now they have n number of feedforward network that is supposed to be smaller than the full one, and then they will have this special module router.

306

01:11:05.850 --> 01:11:33.290

Jisun An: which is working like as a gatekeeper. So whenever, when the token comes in, this router will decide. So okay, coming back so that now they have different fit for network, and each of them are now they could consider them as an expert. So each the fit for network will learn something different, and they will be an experts for different things. The routers will now decide for each of the token which which experts this token need to be. Go

307

01:11:34.387 --> 01:11:38.442

Jisun An: so that's the mixture of experts idea and

308

01:11:39.260 --> 01:11:49.120

Jisun An: in terms of the training the model parameter size, I think, will be more or less the same, or even even more, for the mixture of experts.

309

01:11:49.170 --> 01:12:11.330

Jisun An: This bit forward network can also have another mixture of experts within it, so it can be also hierarchical Moe model. So eventually there may be more parameter to be used to be trained. But interesting part is the when the inference you, only the parameter, will be only activated for those fit for those neural network that are relevant to the tokens.

310

01:12:11.330 --> 01:12:21.860

Jisun An: So the in the inference time, the number of parameter that need to be used for calculation will be smaller than the full fit for the network, meaning that the inference time will be faster.

311

01:12:22.240 --> 01:12:33.662

Jisun An: That's the basic idea. But so this was something that the Mr. Was firstly used. So basically, they have done speed optimization. So that was something.

312

01:12:35.190 --> 01:13:02.960

Jisun An: the Qn is strong for the multilingual and then Gpt Gemini and Cloud. These are the commercial models. Just read it through slowly. The only thing that I want to mention for the last 2 min. Is this Dipsic? And this is something that I will also come back in the when we talked about the Rn. And here one of the interesting, I mean 2. I I think Dipsic was really interesting in 2 different points. So one is

313

01:13:03.120 --> 01:13:30.240

Jisun An: this far smaller model compared to the Gpt or any other models. And also the if you see. So they basically, the model has 671 billion parameter. But for the inference. They only need 37 billion parameter to be activated for the inference, meaning that their inference is far faster than any other model existing. So that's and the reason that it's it's faster is because they are using the mixture of their 1st model that I just mentioned.

314

01:13:30.860 --> 01:13:48.702

Jisun An: That's 1 interesting thing. And another thing is they use this group relative policy optimization, which is one of the Rl reinforcement learning technique. It also simplified version of the reinforcement learning algorithm, and that also helped to

315

01:13:50.110 --> 01:14:05.830

Jisun An: less computation. So that also lead it to like helping to reducing the size of the model. But more important interesting part is that we we talked about the large language model had these 3 steps, the instruction tuned and then preference tuned.

316

01:14:05.990 --> 01:14:18.889

Jisun An: Basically, they combine these 2 steps only and and replace them to this reinforcement learning. And they found that they basically don't need this instruction tuned step.

317

01:14:19.030 --> 01:14:42.696

Jisun An: They can only use the Rl using this Grpo methods. And still it shows some good reasoning performance. So these were 2 things that dipsy kind of brought. And so basically, they tell us that the the model doesn't need to be large to be good. The architecture can be more smarter, and that can be also lead to a good output. So that was the key.

318

01:14:43.190 --> 01:15:01.609

Jisun An: insight that we can learn from the dipstick, and that's the main reason why also, once again, entire stock market on the AI. Has been checking last week, so we will come back and talk more about it when we talked about the reinforcement learning. But that will be something that I prepared

319

01:15:01.980 --> 01:15:04.379

Jisun An: any questions, any last quick questions.

320

01:15:06.680 --> 01:15:23.219

Jisun An: So the the slide of these older models? They will be optional, meaning that I will probably not ask for the details. But the the yeah. I will probably not going to ask for the exam. But I will leave it to you for your own interest in checking.

321

01:15:24.580 --> 01:15:30.130

Jisun An: Alright. Thanks a lot. And I will see you next week.

322

01:15:32.120 --> 01:15:32.870

Jisun An: Yeah.

323

01:15:33.520 --> 01:15:36.019

Jisun An: Have a good weekend. I'll see you next week.