WEBVTT

1

00:00:08.100 --> 00:00:17.659

Jisun An: So why, thanks for joining today. Today's pass code, for the attendance is instructions.

2

00:00:17.870 --> 00:00:19.980

Jisun An: Please mark your attendance.

3

00:00:20.848 --> 00:00:45.550

Jisun An: So we to have, like 2 running assignments. One is the theoretical assignments. So the due is by the end of this week, so please make sure that you complete it before the due date. Yeah, these are 25 multiple choice questions. It'll be easy enough, but help you to review just some of the concepts and the ideas.

4

00:00:48.640 --> 00:00:54.630

Jisun An: And then, more importantly, the team formation deadline is also by the end of this week.

5

00:00:55.226 --> 00:01:06.900

Jisun An: So I mean, I saw a few few students left a note on what they interested in. So you can also do the same on the spreadsheet to find the people.

6

00:01:06.990 --> 00:01:26.250

Jisun An: Okay. So by the end of the this Sunday. If you didn't form the group, then I'm just randomly match and create a group. I will. I will do that. So if you are intention, I mean. Still, you can leave a note to like Random. Then I will. I will know that you. Your intention to be grouped as a random group. So it'd be nice if you can just leave some comments or

7

00:01:26.250 --> 00:01:38.949

Jisun An: so. Once again, the team formation. If you go to canvas, if you go to under the homepage Projects team roster. If you click the team roster, you will see the Google Spreadsheet where you have these 3 columns.

8

00:01:38.950 --> 00:01:45.449

Jisun An: So just write the team name. If you're ready from the group or leave a note, if you want to be randomly married.

9

00:01:46.690 --> 00:01:52.690

Jisun An: to do is also again by the end of this this week. So please, please.

10

00:01:53.210 --> 00:01:55.030

Jisun An: And so

11

00:01:59.140 --> 00:02:02.449

Jisun An: any questions about the information at the moment.

12

00:02:03.600 --> 00:02:04.669

Jisun An: All right.

13

00:02:07.310 --> 00:02:36.339

Jisun An: all right. So so back to the lecture. So today we will talk about fine tuning and instruction tuning, which is quite interesting topic. So some of you may already feel that, and especially those who have taken like machine learning courses. The fine tuning steps may be quite familiar with what you already know, but I mean, basically, you will see how the language models are being fine-tuned, and at least the the idea how how they are trained.

14

00:02:37.710 --> 00:02:46.719

Jisun An: so just a bit of recap. So last week we talked about basically the transformer architecture, and then and then start to talk about

15

00:02:46.970 --> 00:02:58.929

Jisun An: the the architecture itself is just the architecture. And you need to train the model. And then there are actually many steps until that you see the edit, and that you are actually be familiar with.

16

00:02:58.930 --> 00:03:23.909

Jisun An: So start from the like, like, just something random, basically through the language modeling, which we call as a pre-training. You have a base rrm, and based on the base. Rrm, you are doing a fine tuning with the instruction data. And then this is the instruction tuned, meaning that this Rrm. Is now able to follow your instruction, and on top of that which we will, which on top of that now, people

17

00:03:23.910 --> 00:03:33.359

Jisun An: are using preference data set to fine tune this model. And this is the like, the final version, or the version that you are like getting more

18

00:03:33.950 --> 00:03:48.469

Jisun An: of used to like Chatgpt or cloud, or Gemini. These are, and mama, these are the models that are usually going through these 3 steps. And so whatever you are interacting with, they go through these 3 steps.

19

00:03:49.260 --> 00:03:51.549

Jisun An: So a recap of pre-training.

20

00:03:51.890 --> 00:04:10.809

Jisun An: so pre-training. Basically, they had a self-supervised objective for the language modeling. So it was essentially using unlabeled data sets. So you can literally use any text of data available in the world and and using the self-supervised objective, you can learn the

21

00:04:11.450 --> 00:04:13.990

Jisun An: build the to the language modeling.

22

00:04:14.160 --> 00:04:41.238

Jisun An: You can use as much as data you can find. And also you can choose, like the biggest model you can afford. But then the goal of the pre-trading is that you want to create a model that understands the many linguistic properties. And here the linguistic properties can be different things. It can be grammar. It can be like general word knowledge, or it can be emergent properties that that itself has, like a

23

00:04:42.033 --> 00:04:51.140

Jisun An: reasoning capabilities. So even without any instruction, it may be able to answer given on given on particular prompts.

24

00:04:52.370 --> 00:05:02.909

Jisun An: So and and the the pre-training, basically, they are not focused on a any specific task or the application. These are like a generic model that just understands about this language.

25

00:05:03.570 --> 00:05:25.540

Jisun An: And now the fine tuning is using this trained model. You assume, have you? We assume that you have a smaller like label data set corresponding to a particular task, or something that domain that you are interested in. And you want you want to now tune this pre-trained model to tackle that particular task

26

00:05:25.540 --> 00:05:34.380

Jisun An: so it could be like sentiment analysis. It could be question and answer. It could be Muslim translations or any other task that you can think of.

27

00:05:35.180 --> 00:05:42.089

Jisun An: So the goal of the fine tuning is maximize. The performance on that particular task, or that particular domain?

28

00:05:42.455 --> 00:06:12.090

Jisun An: So here the domain would be mean that so you have the pre-trained model. And if you are interested in, like the medical domain. Then you can also collect all the medical related text, and then you can also fine tune the model so that this model can be more. Learn more about this medical domain. Right? So it may not be a particular task, but you can also think it. You can fine tune a train model for a particular domain. So so now there are like.

29

00:06:12.100 --> 00:06:26.729

Jisun An: for example, like the birth, there are mental birth. There are math birds. So there are like different versions of the birth, like right, the fine tuned for different domains, so both are for for a single task, or for a particular domain. You can fine tune these models.

30

00:06:27.540 --> 00:06:40.480

Jisun An: and in a higher level the fine tuning is basically you are adapt the change. The parameters of the pre-trained model given this particular specific test for the domain, and

31

00:06:40.800 --> 00:06:55.019

Jisun An: and we will talk a little bit more about like parameter efficient. So the patient. So while you are updating these parameters, we've been talking about, these large languages are really big, right? So they may have, like the millions, billions of

32

00:06:55.350 --> 00:07:20.159

Jisun An: more than like parameters, and especially if you train them. It's not just the size of the model, but you probably need the same set of the parameter for the gradients, and you may also have like optimizers. So you may have, like a different, far more parameters than the model size themselves. So we want to use smaller number of parameters to update to fine tune. So that's the parameter efficiency adaptation. And we will talk a little bit more about that later.

33

00:07:21.160 --> 00:07:24.589

Jisun An: So so let's talk about fine tuning.

34

00:07:25.218 --> 00:07:37.729

Jisun An: So let me start with the fine tuning, Berts, which probably some of you may be familiar with. So Bert, once again, as a recap is an example of the encoder. Only transformer

35

00:07:38.780 --> 00:08:01.019

Jisun An: and it's training it's also self-supervised objective but but they basically having this master edit, which I will. Which I will talk very briefly. And then, once you have this bird pre pre trained model, then you can find tune this model by adapting some of the parameter for a particular downstream task.

36

00:08:04.180 --> 00:08:22.310

Jisun An: So as a recap how the bird was pre-trained. Was that. So assuming that as our one of the input example, we have this sentence, let's say we have students opened their books. So these are one example that are used for training the bird. And we are still talking about pre-training, not fine tuning.

37

00:08:22.380 --> 00:08:45.470

Jisun An: And the word was used. The masks must techniques. So instead of using all these 4 tokens as input, basically, they must randomly some portion of these words. So, for example, we must be opened. And then, now that embeddings of each of these tokens fed into

38

00:08:45.770 --> 00:08:51.589

Jisun An: the the this almost multi-head self-attention module.

39

00:08:51.750 --> 00:09:05.349

Jisun An: and then, which now here basically include all the attention mechanisms, and also, like the pit, 4 networks. So if you go through that entire transformer architecture, then the output will be some other hidden embeddings of hidden state, the last layer.

40

00:09:05.760 --> 00:09:09.290

Jisun An: And what they hear they do is

41

00:09:10.346 --> 00:09:20.339

Jisun An: basically, they didn't try to predict all different words, but they predict only for those masked tokens.

42

00:09:20.500 --> 00:09:46.240

Jisun An: So you know. So once again, I mean, we are not talking really details about all this training procedure. But whenever we talk about how to update this parameter. I encourage you to think back of the work to back models, which was far more simpler, right? But then the the basic logics are all the same. So that you have, input you have output. I mean, predicted output. And you have rear output. You just compare

43

00:09:46.420 --> 00:09:56.709

Jisun An: between the 2 values, and you compute the loss, and then everything will be back propagated, and they'll take the prominent right? So this logic will be like exactly the same across all different models.

44

00:09:56.890 --> 00:10:22.880

Jisun An: So I will also soon talk about the pre-training of the decoder only model, but for decoder only model for the each of the output token they predict the next token. So that's the literally apply the next token prediction. But in Bert case they used this must strategy. So they randomly select a few tokens from the input, they must it. And they. Their task was to predict those master tokens.

45

00:10:23.060 --> 00:10:34.240

Jisun An: And here one thing to note is that the mask here in the input token is different from the masked multihead self attention. So here unmasked means that when we

46

00:10:34.390 --> 00:10:41.899

Jisun An: compute the attention themselves the the Attention mechanism didn't have any masks right? So the decoder will only know that had a mask.

47

00:10:42.330 --> 00:10:45.339

Jisun An: But these 2 are different masks, so don't get confused by it.

48

00:10:45.520 --> 00:10:46.630

Jisun An: So

49

00:10:47.327 --> 00:10:55.054

Jisun An: so that was the how you predict the I mean what the exact task that this worked was

50

00:10:56.060 --> 00:11:22.400

Jisun An: have to do. And and that's the reason that the birthday is also called as a master language modeling. So instead of predicting the next token, so now you are predicting wi a particular token given, and all these surrounding words right? So I small, I mean the all the word, all the other tokens smaller than I, and all the other token greater than the I. So you're using the surrounding wordings information to predict the I token.

51

00:11:22.520 --> 00:11:27.339

Jisun An: And let's see what this port how that day is that's been portrayed.

52

00:11:28.380 --> 00:11:36.390

Jisun An: And so this was like basic architecture that we described. But then any questions up to here

53

00:11:37.590 --> 00:11:38.550

Jisun An: or clear.

54

00:11:39.210 --> 00:11:52.300

Jisun An: Okay? So there was one thing that I didn't explicitly mention is the oh and sorry that if there are. So, basically this multi head, we have multiple of them. And this entire architecture is called as a birth.

55

00:11:53.420 --> 00:12:19.119

Jisun An: So there was actually one additional token that I didn't mention, which is Cls token. So if you ever read the hedge reports paper, then you will see this. Cls, and maybe just. You are wondering what that is, and this is like a separate like separate embedding. That is not the part of any of the existing tokens. It's just a separate Cs token, and it occurs at the beginning of every input sequence.

56

00:12:19.340 --> 00:12:33.100

Jisun An: And the reason that they have this Cs token is a bit bit special, because and also it's a choice of the design, because back then, when they were building this bird, they were interested in classification problems. So even

57

00:12:33.100 --> 00:13:00.060

Jisun An: because this was the moment that they even haven't think about like the proper text generation. So this was the time when, okay, let's understand better of the languages. And let's do the classification better. And now we feel that classification is kind of dumb problem. But I think back then it was still there. So they really wanted to tackle the classification problem. So they used this Cns token to tackle the classification problem in particular.

58

00:13:00.630 --> 00:13:16.190

Jisun An: So you can imagine that for every input sequence there's always they always attach it. This one extra embedding at at the start of this sequence. And then, after going through the bird architecture at the last layer, they will also have these last tokens

59

00:13:16.980 --> 00:13:25.230

Jisun An: and this this another output embedding were used to predict the next sequence prediction.

60

00:13:25.500 --> 00:13:38.149

Jisun An: So so, even though I didn't, so I think I briefly mentioned that Bert was trained, based on 2 different tasks, the masked token prediction, and the next sentence, prediction as well. So how? What?

61

00:13:38.300 --> 00:13:51.140

Jisun An: What input the bird was bad was for every basically they had a 2 sentences that concatenated together and proceeded with a Cls token. And this Cls token was

62

00:13:51.140 --> 00:14:13.970

Jisun An: basically had an idea whether these 2 sentences are actually the next sequence in the corpus or not, so they could just sample from the the actual 2 sentences that were followed by each other, or they randomly sampled the 2 sentences. And, as you can see, their output would be basically yes or no. So whether the 2 sentences were consecutive or not. Right? So

63

00:14:14.190 --> 00:14:17.970

Jisun An: now you can see that this next, because this 1st

64

00:14:18.472 --> 00:14:42.010

Jisun An: the output token output embedding were test to predict the next, whether the 2 sentences are next, I mean sequential or not. The answer was yes or no. So these are again binary classification problems. So they use the cross entropy loss to compute the loss, and then they were able to back, propagate their gradients, and similarly, for each of those masked tokens.

65

00:14:42.010 --> 00:14:49.489

Jisun An: They were also computing the cross entropy, and then they back prop all the gradients to

66

00:14:49.700 --> 00:14:55.589

Jisun An: update the parameters. So this was the how Bert was actually trained.

67

00:14:55.880 --> 00:15:04.069

Jisun An: and and the purpose of the Cs. Is just a special token to deal with the classification from the model a bit more easily

68

00:15:07.030 --> 00:15:08.210

Jisun An: does it make sense.

69

00:15:12.940 --> 00:15:16.054

Jisun An: Alright. So now, assuming that we are

70

00:15:17.510 --> 00:15:46.490

Jisun An: we are doing fine tuning, for example, like sentiment analysis. So given an input, we want to predict like positive or negative. So as a input, now we have okay, the movie was good. And once again, for every input, we will have this Cls token. So these factors will be fed to the bird, and they will have the embeddings at the last layer. And basically this first, st the output embeddings will predict whether they are positive, or will be test

71

00:15:46.490 --> 00:15:56.240

Jisun An: to predict whether they are positive or negative. So you can. So these are very similar to the next sequence prediction. And this is the first.st

72

00:15:56.360 --> 00:16:22.389

Jisun An: The output embedding will be test to deal with that classification problem. So this will be exactly the same how it was pre-trained. But now you are having your own small labeled data, which is labeled as a positive or negative. So, given an input, you will predict whether something is positive or not, you will check with the actual answer. And basically the loss between the 2 will be back propagated to

73

00:16:22.580 --> 00:16:28.400

Jisun An: through the network. And then coming to to the update all the things.

74

00:16:28.800 --> 00:16:56.359

Jisun An: So in a way that and to predict whether something is positive or negative. This last layer basically will have the multiple create like a new set of weight. So the Hcls, which is the hidden embeddings of the hidden state of this Cls token will be multiplied by this new set of parameters wo which is just there for predicting these 2 classes.

75

00:16:56.730 --> 00:17:02.269

Jisun An: and then they will take this up to Max to get the probability between these 2 labels. So

76

00:17:05.190 --> 00:17:23.379

Jisun An: and then things will be back propagated through their networks and also through to the embeddings, the input embeddings as well. So I mean, so usually the birds fine tuning. They update all the parameters. So what kind of the parameters do we have in the birds architecture. Do you remember

77

00:17:23.720 --> 00:17:27.070

Jisun An: what parameters do we have like in the birth architecture?

78

00:17:36.560 --> 00:17:58.800

Jisun An: So if I mean basically birthdays, just I don't know what happens. Microsoft updates going on? So if you remember the transformer architecture, there were a few metrics weight matrixes that were updated. So some of the examples are key key. Metrics is, wait, key up, wait for the keys, wait for the values, wait for the queries.

79

00:17:58.800 --> 00:18:06.980

Jisun An: and also all the all the parameters in the fit for the networks. So these are all the parameters that are updated of

80

00:18:06.980 --> 00:18:34.790

Jisun An: doing while doing the fine tuning. So you can imagine that these parameters were set after the pre-training right? They are just fixed values. And then during the fine training, they are just like updated based on. So these, these, all these values will be other just slightly to better reflect which of these words are more associated with the positive or the negative, so they they will just change their values based on that.

81

00:18:39.310 --> 00:18:41.190

Jisun An: Any any question

82

00:18:49.460 --> 00:18:51.650

Jisun An: so fine-tuning all tips

83

00:19:03.840 --> 00:19:08.630

Jisun An: that's so the the black lines are forward packed

84

00:19:09.070 --> 00:19:11.100

Jisun An: and the blue lines are backpack.

85

00:19:11.110 --> 00:19:38.790

Jisun An: Yeah? So so you can see that for each of the example. The yeah. Personally, they will given this input sentence, they will. So the parameters already have some values. Right? So given this input going through this architecture, they will compute, they will do all the computation within the attentions and within the fit for networks, and at the end of the last layer you will have also some other values for representing I mean embeddings of each of these tokens.

86

00:19:38.790 --> 00:19:43.550

Jisun An: And then they don't care much about these other 4 input

87

00:19:44.007 --> 00:20:00.670

Jisun An: input part, because they don't do anything about it. But they only focus on this 1st token, where they are dedicated to predict the positive or negative. So these embedding will represents either positive or negative. Basically have 2 value. I mean

88

00:20:00.730 --> 00:20:23.709

Jisun An: the way the multiplication really weight matrix will reserve in 2 values positive, the probability to be positive, or probability to be negative. I mean, after this of 2 Max. And then and then that value will give you label right? So you will. The model itself will evaluate, evaluate, eventually assign a label either positive or negative after the 4 path.

89

00:20:23.970 --> 00:20:29.329

Jisun An: and then that label now will be compared with the actual answer, and then compute the loss.

90

00:20:29.490 --> 00:20:33.539

Jisun An: cross entropy, loss, and then base. That value will now

91

00:20:33.740 --> 00:20:36.030

Jisun An: coming back through the backdrop. Yes.

92

00:20:37.050 --> 00:20:39.860

Jisun An: yes, that's the that's the correct sequence.

93

00:20:40.620 --> 00:20:42.080

Jisun An: Any other question.

94

00:20:42.810 --> 00:20:44.860

Jisun An: So if this is not clear.

95

00:20:45.860 --> 00:20:53.273

Jisun An: it'll be a it'll be an issue, because I will repeat this again and and also this is like the key

96

00:20:54.380 --> 00:20:58.269

Jisun An: key concept. I mean, basically, it's the same for all the models. So

97

00:21:00.500 --> 00:21:05.609

Jisun An: any any questions that you don't, you don't clear you, you find it not clear here.

98

00:21:10.580 --> 00:21:11.520

Jisun An: Okay,

99

00:21:16.120 --> 00:21:29.558

Jisun An: So yeah. So the fine tuning is not self supervised. Meaning that to fine tune, you need a label, the data, right? So you have an input you need an input and the corresponding labels. And also it requires

100

00:21:30.857 --> 00:21:41.959

Jisun An: I mean, it required training data, but far less than the pre-training. So usually pre-training. You need billions of text, but then, for fine tuning, you probably need a far more smaller data set.

101

00:21:42.610 --> 00:21:55.895

Jisun An: So so now I will move on to the decoder only model, which is like decoder. Only model is like a llama. Gpt, they are all the decoder only model. So let's just look back how these were kind of trained.

102

00:21:56.490 --> 00:22:10.579

Jisun An: so assuming that we have 3 on input with the 3 tokens, students opened there and then they, these initial embeddings is going into the masked multi header self attention. So once again.

103

00:22:10.800 --> 00:22:18.930

Jisun An: the reason that we are here using the mouse to multi-head self attention is because we are

104

00:22:20.460 --> 00:22:38.333

Jisun An: because you cannot use the information of your future tokens. Right? So if you remember the masks here in the attention mechanism is basically, it was the mask that was like triangular shape where you basically multiplying all the

105

00:22:39.340 --> 00:22:41.610

Jisun An: the minus infinite value.

106

00:22:42.180 --> 00:22:58.509

Jisun An: so that you, you masks all the future tokens when you are adding the input, so what? What that? What that means is that when you compute okay, after this architecture, you will eventually have, like 3 output embeddings.

107

00:22:59.360 --> 00:23:17.689

Jisun An: So this must means that when you compute this last vector for the student, you will only use the student information. And when you you. When you compute the second vector for the opened, you will use the information of students and opened, but not there.

108

00:23:17.690 --> 00:23:39.529

Jisun An: And when you compute the information of the there. Then you will compute across the students opened there. So basically, I guess the only example that makes sense is the opened. So when you compute something for the opened token. The second token. You only use the the students and the opened, but not the there, and the mask was kind of helping to.

109

00:23:39.570 --> 00:23:46.529

Jisun An: I mean able to do that by masking, like the half of the future tokens in in your attention mechanism.

110

00:23:50.470 --> 00:24:20.180

Jisun An: And then, after you get this last embedding each of so now decoder, only model are each of the output. Tokens are each of output embedding. I'm sorry that I'm still like confusing the word. So each of the output embeddings are trying to predict the next tokens. So for this 1st one, given students, they try to predict the opened and the second one given students opened. They try to predict it there, and this last one. Given these 1st 3 words, they try to predict the books.

111

00:24:20.750 --> 00:24:35.919

Jisun An: and these. So for the decoder only, unlike the encoder only model that we just seen, where they only predict for the master token, decoder, only model, they just predict the next token for for every single input, token.

112

00:24:36.610 --> 00:24:58.780

Jisun An: And once again, these computation can be done parallelly. And that's the reason that if you have, like a more powerful computation. And more Gpus, then you can basically train this model far, like fast. And you can train also larger data at once. So that's the beauty of the attention. Unlike the Rnn, so that's also one thing that we talked before.

113

00:24:59.740 --> 00:25:23.189

Jisun An: and and the way that the predict opened means that. So what? What we're going to happen here is basically you will have this probability across all vocabulary or tokens that you have in the model, and you will have a probability distribution, and this one trying to predict open the means that the which that distribution that it had will be

114

00:25:23.280 --> 00:25:31.880

Jisun An: try to maximize the the probability of opened to be higher compared to the other tokens. And that

115

00:25:32.460 --> 00:25:49.399

Jisun An: comparison between these 2 distribution is basically the cross entropy loss. Once you compute that they will just back propagate. So each of these output embedding will compute their own loss, and then these, each of the loss will be back prop through the this, this

116

00:25:49.710 --> 00:25:52.480

Jisun An: models and and to the input embeddings.

117

00:25:55.110 --> 00:25:56.450

Jisun An: So

118

00:25:57.290 --> 00:26:16.130

Jisun An: so once again, these are now like, hopefully, you can kind of have an idea that the models are more or less similar, but some parts are slightly different, so some configurations are slightly different to each other, and that determines whether it's the encoder, whether you see decoder, only model and also I've introduced something else very soon.

119

00:26:16.840 --> 00:26:22.970

Jisun An: and because of this characteristic, these decoder only models are useful for the text generation.

120

00:26:24.533 --> 00:26:47.189

Jisun An: and but then, if you think about how you interact with the like, Gpt, these days, they usually don't work in this way right? So they usually work in a way that you given you give a particular question, and then they are answering to it right? So you are giving some kind of prompt, and then they are start to generate the text after that

121

00:26:47.290 --> 00:27:06.319

Jisun An: prompt. So you are not saying students, and then they will generate. I mean, that's not, I mean, that's what this model can do. But that's not the way that you are interact with these models. Right? So that's the reason that people also came up with this prefix language model. So the idea is once again it looks more similar. Just the configurations are a little different.

122

00:27:07.150 --> 00:27:31.419

Jisun An: So the prefix Lm is now something to help to interact better with the human. And the idea was very simple. So we have now going more so, assuming that you have tokens that you want to generate, but we just have a few other tokens that are serving as a prompt, so, given a particular prompt, then they will start to generate the token. So that's the whole idea of this prefix. Lm.

123

00:27:31.880 --> 00:27:41.780

Jisun An: so let's say, these 1st tokens are prompt token, and these next 3 tokens are the tokens that we want to generate.

124

00:27:45.230 --> 00:27:53.840

Jisun An: And now these or embeddings are feeding into once the partially masked, multi-headed self attention, architecture.

125

00:27:54.500 --> 00:27:58.039

Jisun An: and each of these tokens also will have some output embeddings.

126

00:27:58.660 --> 00:28:03.890

Jisun An: And now here, what they are trying to do is

127

00:28:06.020 --> 00:28:15.349

Jisun An: so let's say that our prompt token was like complete. This phrase, and then the token we wanted to generate is like, students opened there.

128

00:28:15.430 --> 00:28:42.994

Jisun An: Then, in this case. Now we let this model to train only for predicting the next token, starting from this c 1 token. So they basically, even though the model have these prompt tokens. The prompt output embeddings of the tokens do not anything, but only these c, 1 c. 2 c. 2 c. 3. They will ask, I mean test to predict the next token

129

00:28:43.700 --> 00:29:06.249

Jisun An: for each of them. So this so once again ignore. So these P. The 1st 3 output embedding will not do much, but then the 4th embedding will be tested to predict the next token, which is the opened, and for that is the there, and for the that is the books. So this, the second part is exactly the same as decoder, only model. But now.

130

00:29:06.553 --> 00:29:16.279

Jisun An: if you train this model using some prompt then now they will, they will know, like app given this, prompt to like, what they want, what they need to kind of generate.

131

00:29:17.160 --> 00:29:18.310

Jisun An: So

132

00:29:19.310 --> 00:29:43.355

Jisun An: these are kind of some idea that's very similar to that I will introduce later is to the instruction tuning. So we basically want to train a model to answer to your particular question. I mean, it's not necessarily question, and that's the reason that we call this a prompt right. And now you can. You can understand how these models were trained. Right? So

133

00:29:44.280 --> 00:30:03.509

Jisun An: basically, your input is now the concatenation of the prompt and the answer. And you ask the model to predict the next token for those answers only, and in that way the model will learn what to do or basically answer to your particular point.

134

00:30:05.400 --> 00:30:06.060

Jisun An: Yes.

135

00:30:09.750 --> 00:30:36.889

Jisun An: right? Right? I mean, so these are just the the illustrative results. So it may not be very clear. But in this case, because we asked to complete this phrase. So in reality, you probably need to also have the student so that they can complete the phrase.

136

00:30:37.430 --> 00:30:51.430

Jisun An: But but that's basically this is wrong wrong test wrong sample. So I I understand why? Why you are confused. But technically you should only give the prompt tokens.

137

00:30:52.170 --> 00:30:56.950

Jisun An: Yeah, just the example is wrong. Yeah, yeah, bye, yeah.

138

00:30:57.770 --> 00:31:03.810

Jisun An: I mean, when you're interacting with this model. But when you train the model, basically, these are just the training data.

139

00:31:05.600 --> 00:31:24.629

Jisun An: Does that doesn't make sense. Yeah. So the what we provide here, these are like training data. But once you train your model with this data, then when you interact with this model you can give the prompt, which is P's, the set of P's, and then the model will start from C's

140

00:31:29.890 --> 00:31:46.040

Jisun An: and so so in in terms of the architecture. If you compare with this one and the decoder only model, then the only differences here is the they are using partially maxed multi-head self attention. And I haven't mentioned about what that is. And

141

00:31:46.749 --> 00:32:06.449

Jisun An: basically. So in the decoder only model, we were using this decoder masks. So if you remember, then these are the the token to token metrics and basically these masking with the minus infinite value will prevent for you to see the future tokens.

142

00:32:07.420 --> 00:32:18.229

Jisun An: so in the prefix edit them. So, assuming that we have these metrics where you have prompt tokens and the the tokens that you want to generate, then

143

00:32:18.400 --> 00:32:21.610

Jisun An: can you? Can you imagine what this would this would be? Look like?

144

00:32:22.490 --> 00:32:25.440

Jisun An: Which part will be must or unmasked

145

00:32:31.190 --> 00:32:36.770

Jisun An: left? 3 will be masked, mask or mask

146

00:32:37.150 --> 00:32:57.359

Jisun An: must, so maybe the meaning of the mask. You may be a little confusing so the left will be unmasked, because unmasked means that they will be able to see each other everything. So basically, the prompts doesn't need to be hide everything right? So so they are the unmasked.

147

00:32:57.890 --> 00:33:10.350

Jisun An: So once again, if you think back to the birth model, Bert was using unmasked self-attention multi hats because they just want to get all the information from the surrounding worlds.

148

00:33:11.280 --> 00:33:13.525

Jisun An: and then and then for this

149

00:33:14.371 --> 00:33:24.650

Jisun An: tokens to generate it. They are also partially. It marks, because these are the part where you want to generate. So given the future tokens, you are, I mean.

150

00:33:24.860 --> 00:33:36.960

Jisun An: sorry. Given the historical tokens you are generating, so the so the the white part is the masked, and the the gray part is the unmasked part, and unmasked means that

151

00:33:37.170 --> 00:33:43.990

Jisun An: for those tokens that are unmasked they can share the information, and for the area that is, must

152

00:33:44.140 --> 00:33:55.219

Jisun An: those information will not be shared. And you can see this as each of the role. Because that would be, we are predicting like the next token So

153

00:33:55.580 --> 00:34:01.930

Jisun An: so the figure should be interpreted from the each. Each role. In this speaker

154

00:34:04.030 --> 00:34:05.840

Jisun An: is this, does this make sense?

155

00:34:05.970 --> 00:34:07.230

Jisun An: Any questions?

156

00:34:14.790 --> 00:34:15.520

Jisun An: Yes.

157

00:34:21.230 --> 00:34:29.100

Jisun An: What's the most common way that decide how long, how long? What

158

00:34:31.231 --> 00:34:38.610

Jisun An: well, that you can decide from the training data. So it will be changing dynamically. Okay.

159

00:34:39.199 --> 00:34:41.179

Jisun An: you know, it's not like a.

160

00:34:44.429 --> 00:34:47.719

Jisun An: So with the training data, that's all.

161

00:34:48.770 --> 00:34:59.830

Jisun An: You decide how much mask. Oh, you mean, you mean the mask part, but but this one is next token prediction, so the next one will be masked anyhow.

162

00:35:00.930 --> 00:35:04.330

Jisun An: unmasked one will be they. They are not masked right?

163

00:35:06.165 --> 00:35:11.030

Jisun An: Asking. So like, if you have a bunch of different, I guess

164

00:35:12.090 --> 00:35:21.110

Jisun An: sentences or paragraphs, or whatever that happens, half of their their data. Not you.

165

00:35:21.340 --> 00:35:28.530

Jisun An: No, no, no. So the decoder model doesn't use the masked language. That's only for part.

166

00:35:30.070 --> 00:35:38.549

Jisun An: right? But I guess I shouldn't say last so like, but they're getting completely. No, they are complete. They are complete sentences, they.

167

00:35:38.980 --> 00:35:39.810

Jisun An: Okay.

168

00:35:40.440 --> 00:35:45.370

Jisun An: So let me go back and ask you one question.

169

00:35:52.670 --> 00:36:02.259

Jisun An: so I'm coming back to the birth. And we we talked about the this 1st output embedding is

170

00:36:03.281 --> 00:36:08.209

Jisun An: dedicated to predict these sequence

171

00:36:08.400 --> 00:36:12.409

Jisun An: or okay, actually, let me. Let me come.

172

00:36:13.640 --> 00:36:15.180

Jisun An: Come back here. Sorry.

173

00:36:15.970 --> 00:36:28.179

Jisun An: So here in the fine tuning in in the Birth example. I I said that the 1st this this 1st output. Embedding will be used for predicting the positive or negative. But

174

00:36:28.310 --> 00:36:42.910

Jisun An: so basically, we will use only this 1st embedding to predict whether the sentence is positive or negative. But do you do you think this 1st output embedding actually encode all the information of this sentence?

175

00:36:44.950 --> 00:36:48.889

Jisun An: No. And and why- why is no, just yeah.

176

00:36:50.910 --> 00:37:05.140

Jisun An: I just get any any other opinions. Do you think this the output embedding of the Cls token encode information about the other tokens. The movie was good.

177

00:37:07.060 --> 00:37:20.069

Jisun An: I mean, in other words, if it's not, how can you predict it? Right? But but just to think about it like is is this output embedding the Cls output embedding? Does it encode informations about other tokens?

178

00:37:22.000 --> 00:37:28.479

Jisun An: I mean. Now I'm forcing you to give an answer. Yes, right? But but think about why, that's the case.

179

00:37:29.030 --> 00:37:32.370

Jisun An: And here the bird was using unmasked.

180

00:37:32.480 --> 00:37:35.680

Jisun An: Murty had self-attention unmasked

181

00:37:35.850 --> 00:37:43.559

Jisun An: in terms of the transformer. Architecture means that when they compute the attention nothing was hide.

182

00:37:43.780 --> 00:38:05.889

Jisun An: Everything was just unmasked, meaning that basically nothing was masked. I don't know whether it's better to use the unmasked or not, but basically. And then computing, the attention means that across all the different pairs of the tokens they were computing how much they are related to each other right? So they wanted to compute the relevance score for each of the token across all the other tokens

183

00:38:06.130 --> 00:38:16.463

Jisun An: and in birth, because the attention is unmasked. In other words, nothing was masked. They basically to encode the new

184

00:38:17.910 --> 00:38:24.270

Jisun An: new to encode each of the token. They use all the other information from all the other tokens.

185

00:38:26.240 --> 00:38:35.138

Jisun An: Does that make sense getting better? Slightly better? So, okay, well, okay,

186

00:38:37.290 --> 00:38:38.320

Jisun An: So

187

00:38:38.510 --> 00:39:00.019

Jisun An: in other words, even though in fine tuning, the bird basically will not do anything with this from second to the last output embeddings. But each of these are basically you can consider it as a reinterpreted embeddings of the reinterpreted embeddings of the movie, and wasn't good.

188

00:39:00.220 --> 00:39:18.410

Jisun An: And these are now. These embeddings are incorporated with all the other information of the other tokens. So what is? How should we include? How should we represent goods in relation with the doll movie? Was is the output embedding that each token has.

189

00:39:18.580 --> 00:39:35.290

Jisun An: So, even though this Cls is in coming first, st and you may be very confused. But these are not the Rnn. But these are the transformer where they just shares and compute the cross. I mean, not cross. Compute the attentions across

190

00:39:35.560 --> 00:39:39.280

Jisun An: all the pairs of the tokens in the input

191

00:39:40.340 --> 00:39:46.049

Jisun An: they supposed to know the information I supposed to encode the information about all the other tokens.

192

00:39:46.640 --> 00:39:50.017

Jisun An: So that's the key difference between the

193

00:39:50.710 --> 00:39:57.399

Jisun An: the birth, the encoder based or masked language, modeling versus the decoder, where

194

00:39:58.710 --> 00:40:08.469

Jisun An: now the decoder model are using this decoder masks. So what it means is that so this decoder mask.

195

00:40:08.730 --> 00:40:35.889

Jisun An: before applying the mask, they actually compute all the attention. So you have token 4 and 4. So you actually have 16 values, 4 by 4 matrices. But they are just removing this upper triangular part. So when they actually try to predict the next token, you just intentionally remove those information about the future tokens. And let's see what this mask is doing for decoder only model.

196

00:40:36.460 --> 00:40:49.760

Jisun An: And now the prefix at M. The the aim is, we want to predict the next token given, the prompt and the tokens, information itself, and those that will have these partially masks

197

00:40:50.340 --> 00:40:55.080

Jisun An: partially masked, meaning that all the other will be unmasked, and the only the upper party will be the masked. Yes.

198

00:40:55.270 --> 00:41:02.049

Jisun An: it's my question is still the same. How? How do you decide how much of c 1 removes?

199

00:41:02.620 --> 00:41:18.349

Jisun An: So so these are like all the p. 1. And okay, I should probably put it here. So the 1st to see one will be actually here. And then from C 2 and

200

00:41:18.700 --> 00:41:19.830

Jisun An: everything.

201

00:41:20.220 --> 00:41:32.050

Jisun An: Okay, because you are predicting the next token without the future token information. So you just mask all the future tokens for each of the inputs. Yeah.

202

00:41:33.270 --> 00:41:34.380

Jisun An: guess.

203

00:41:38.460 --> 00:41:41.779

Jisun An: Okay, I'll ask you a question. So the the

204

00:41:47.750 --> 00:41:56.310

Jisun An: so it's always it's always the same in like, regardless of the size of the phrase, it's always no, always, always like.

205

00:41:56.670 --> 00:42:02.639

Jisun An: yeah, cause you are masking all the future tokens except you and historical tokens

206

00:42:02.690 --> 00:42:30.729

Jisun An: all over the fees. So the prompt. We don't care because we actually want to the prompt tokens to be shared sharing all the information, because that's the key information right? And also we are only predicting. The reason that we are having the mask is because to generate the next token, if you know the future token just is not fair right, and it may not make much sense. So that's the reason that you are. You're hiding to get information from the future tokens

207

00:42:30.770 --> 00:42:34.670

Jisun An: for the I mean for the soft training

208

00:42:34.790 --> 00:42:37.690

Jisun An: for the or the self supervised training part.

209

00:42:38.160 --> 00:42:52.500

Jisun An: What's treated as the prompt? Is it like the 1st sentence and the next? You decide as a training as a trainer like you decide what's the prompt and what's the the token that you want to generate? Yeah, okay, cool.

210

00:42:53.020 --> 00:42:55.300

Jisun An: So so

211

00:42:55.510 --> 00:43:09.260

Jisun An: so the masking here I so once again the the masking, a few tokens from the bird when training that's different from masking the attention. So that's the 2 different masking, and within the attention. The masking is

212

00:43:09.883 --> 00:43:37.746

Jisun An: that purely for hiding the information from the future tokens so that can be appeared in the decoder, only model, or this prefix ATM. But then for the encoder the attention will be unmasked, meaning that I mean, that's just the multi heads of attention. So there will be no hiding, because they want to encode more like they want to use all possible information from their surrounding words

213

00:43:38.200 --> 00:43:50.169

Jisun An: to predict that master token. But this decoder only model was focusing on generating tokens. So they they just basically hide all the future tokens to be.

214

00:43:50.550 --> 00:43:53.630

Jisun An: And their attention basically were not considered.

215

00:44:02.310 --> 00:44:04.660

Jisun An: Okay, any question.

216

00:44:14.450 --> 00:44:30.069

Jisun An: So now, so now we know the decoder only model. And now, how can we find to this decoder model? I mean, in a way, it's a similar to what we've seen based on the encoder model. But so now there's a the movie was good

217

00:44:30.350 --> 00:44:49.030

Jisun An: and once again. These are architectural choices, so there could be better architecture or different ways. But this is just the way that people have been doing in common. So the we have 4 tokens as an input. It went through the partially masked self-attention. We have the output embeddings.

218

00:44:49.660 --> 00:44:59.307

Jisun An: And now, once again, in the decor only model, each of the output embedding. We're predicting the next tokens, right? So for the last one instead of the

219

00:45:00.887 --> 00:45:07.860

Jisun An: in basically for the last late last embedding, you can ask to predict either being positive or negative.

220

00:45:09.094 --> 00:45:28.585

Jisun An: So so that would be the way that how you can fine tune this model even though the decoder model can do far better than like generating just a single token. But still it is possible to do so. You can let the one of the outputs embedding to predict the

221

00:45:29.240 --> 00:45:52.700

Jisun An: either positive or negative. But but in this case, unlike the encoder model, where where we had only like 2 labels to predict, because these are already trained to predict the distribution of the different words. Basically, this last embedding will have, all the tokens and their probabilities.

222

00:45:53.010 --> 00:46:08.840

Jisun An: And out of these a 2 that different tokens basically, you will have 2 particular tokens that is corresponding to negative or positive, and you will try to change these 2 distribution while fine tuning. So

223

00:46:09.504 --> 00:46:22.990

Jisun An: but then but then eventually the the way that you will compute it will be the same as you've seen in the encoder. So you will, for the last output of the hidden layers.

224

00:46:23.210 --> 00:46:35.879

Jisun An: the 80 goods. You basically compute some kind of weights. And taking these off to Max, which will, turning into the probabilities across all these different tokens, and then you will compute the loss, and then there will be back. Prop

225

00:46:36.330 --> 00:46:59.809

Jisun An: so these are all similar tests. But I mean, these are just a minor thing, and not very important. But the difference between the encoder model and this one for the encoder model. When you are fine tuning you actually added the new parameter to so this W was the new parameter. But in this case, here the W. They are just using the one that

226

00:47:00.270 --> 00:47:16.070

Jisun An: because this last embedding already has the wo metrics because they were already trained to like predict the next token. So they are just updating that wo, so so basically, there's no new parameter in the decoder only model

227

00:47:20.780 --> 00:47:22.030

Jisun An: any questions.

228

00:47:27.460 --> 00:47:54.616

Jisun An: So so I guess doing fine tuning on the decoder model is quite expensive, but still it may be possible, but the reason that we are talking is to lead to this to the instruction tuning. So this is the way. But basically, the decoder only model can do far more than just predicting positive. But you can also train them to not only

229

00:47:55.010 --> 00:48:09.639

Jisun An: giving out the label. But you can also train to generate the explanations of that label as well. So, given, the movie was good, you can, as an input, you can also train them as a positive because of good.

230

00:48:09.990 --> 00:48:29.959

Jisun An: So you can have this as an output. So you can have this input and output pairs and using them to fine, tune your model and then basically to do it, you will use this partially masked model to fine tune this model to do this, does that make sense?

231

00:48:30.260 --> 00:48:43.099

Jisun An: So now, the input basically, yeah, the model will be quite similar to the prefix at M that we've seen. And using that model, you can fine tune to to generate a particular text that you'd like to have

232

00:48:51.220 --> 00:48:52.490

Jisun An: any questions

233

00:48:56.330 --> 00:49:12.959

Jisun An: right? So so this idea is now leading to the instruction tuning and the instruction tuning was the the second step that we've seen to build a good quality of rm, so and basically instruction tuning is.

234

00:49:13.120 --> 00:49:30.139

Jisun An: It's a fine tuning once again. Now, people just call them separately. So when they call the fine tuning they call them as a supervised fine tuning or sft. So if you ever seen this as sft in any papers, then it means that they are just doing fine tuning on a particular downstream task.

235

00:49:30.180 --> 00:49:52.290

Jisun An: and the instruction tuning is basically doing the fine tuning. But to be capable of following a particular instructions and method is just a standard fine tuning, but on a special data set. So that's the only difference of the instruction tuning. So this is essentially fine tuning, but using a particular data set that dedicated for the instruction following.

236

00:49:53.750 --> 00:50:18.690

Jisun An: So the instruction tuning. The 1st step to go with the instruction tuning is to collect data, the data of the instructions, and in particular what it means that so these instruction data basically have different tasks. And the output of each of these tasks. And here the goal is slightly different from like normal, supervised fine tuning, because supervised fine-tuning tend to focus

237

00:50:18.690 --> 00:50:36.319

Jisun An: on one particular downstream task. But here they actually this collection is the covers, the multiple instruction. And that's actually the aim. So they create a instruction data set that has multiple different instructions and their corresponding outputs.

238

00:50:36.320 --> 00:50:40.582

Jisun An: So here one could be sentiment analysis. So please

239

00:50:41.760 --> 00:51:09.210

Jisun An: classify the sentence of this following sentiment, following sentence in like positive or negative, so this could be your one instruction and the answer, but then you can also have, please, rate the sentiment of this sentence from one to 5, and then output and the output. So you can also have that as a X like separate instruction, you can also have a please summarize these paragraphs into

240

00:51:09.320 --> 00:51:38.619

Jisun An: 2 sentences, and then input and the output. You can also have like and I mean, do this math, I mean, solve these math equations and give the math equations. And the answer. So you can now have multiple different instructions and their corresponding outputs and construct this as an instruction data set. And you fine tune the pre-trained model which was the using the prefix kind of error method using the partially masked

241

00:51:38.640 --> 00:51:40.499

Jisun An: other potential models.

242

00:51:41.420 --> 00:52:07.859

Jisun An: So idea itself is quite simple. So like some kind of instruction like, please answer the following questions and provide the detailed justification. It usually have the inputs. What was the mobile number of Jason and the output could be, I mean, rather than they actually generate, like the random numbers for the mobile phone, you can create a data set that. Oh, I can't answer that that because it's a private information as an answer, and you can now find tune your model

243

00:52:07.860 --> 00:52:30.060

Jisun An: to based on these instructions. And you, your model will learn how to how to respond to such information. Right now you can see that you probably heard that the addms are like safeguarded. They were preventing from rebuilding private information, etc. So these are all the things done through these instruction tuning plus preference tuning that we were talking about 2 weeks.

244

00:52:31.060 --> 00:52:32.260

Jisun An: Does that make sense?

245

00:52:37.210 --> 00:52:55.359

Jisun An: And once again, this is like fine tuning. So you have pre-trained ATM, you have this instruction and the input? And you have a generation, and you compare that with the actual output, and then you will be back, propagate, and then all the parameters will be updated. And once again, even in the instruction tuning.

246

00:52:55.744 --> 00:53:09.979

Jisun An: The loss they are using for the training is a simple cross entropy loss, because it's just a classification classification problem for each of the output embeddings, whether they are predicting the token well or not.

247

00:53:15.760 --> 00:53:34.160

Jisun An: And so the one thing that is really interesting about the instruction tuning is that these instruction tuning focus on many different tasks at once, and not just the one. So that would be the crucial difference between the the normal supervised fine tuning and the instruction tuning. And

248

00:53:34.500 --> 00:53:43.740

Jisun An: interestingly, they found that this instruction tuning improved the generalization on tasks outside of the fine tuning data means that

249

00:53:43.820 --> 00:54:08.680

Jisun An: even though the. So you have a set of tasks in your instruction data. But then they found that the model fine-tuned with this instruction data can also solve other tasks, even though that task was not part of the instruction training data. So this is something that they found in the paper that suggested the flan. So fluent is another model. Lama based instruction tuned lama, basically, if I remember

250

00:54:08.680 --> 00:54:18.160

Jisun An: correctly. And this was the 1st time that this instruction tuning was introduced, and what they did was they fine-tuned the llama

251

00:54:18.240 --> 00:54:43.229

Jisun An: using many different instruction, like common sense, reasoning, and translations, sentiment, analysis, co-reference, resolution, etc. Etc. And then they tested their model on some other Nfp test. That was not part of these instruction data, and they found that here these are the results. So the Flan general shots. Basically, general shot means that just asking the question

252

00:54:43.230 --> 00:54:50.619

Jisun An: without any examples. These were better than the Gpt-three general shot. And also it was better than Gpt's free shot.

253

00:54:51.500 --> 00:54:56.340

Jisun An: example. So so basically kind of interesting idea that that

254

00:54:57.320 --> 00:55:11.909

Jisun An: just doing, I mean, you're kind of guiding the edit. And in a way to find a way to answer the question to a new question. And this was the kind of the initial idea of, or initial steps toward to

255

00:55:12.960 --> 00:55:14.460

Jisun An: towards that.

256

00:55:15.610 --> 00:55:17.809

Jisun An: So any questions?

257

00:55:23.780 --> 00:55:24.490

Jisun An: Yes.

258

00:55:26.090 --> 00:55:46.059

Jisun An: yeah. So that we will talk more about it tomorrow on Thursday. But so the 0 shot is basically you are just asking the question without any other information. So you are just you. You don't train the model with a particular task, but you just ask it without any information.

259

00:55:46.310 --> 00:56:09.170

Jisun An: So that's the general shot, because no information field shot means that you are giving a few examples. So if in a case of the sentiment analysis. So if you are doing general shot, sentiment, classification, even though now the models are all trained, based on those assuming that your model was not trained, based on the sentiment general shot is that you just ask like classify the sentiment. This sentence into like partial or negative.

260

00:56:09.310 --> 00:56:22.409

Jisun An: an input. And you are just asking for the output. The future is. Now you are given these example. So, for example, the movie is good is positive. The movie is bad, it's a negative. So you are just giving a few examples in the prompt.

261

00:56:22.710 --> 00:56:28.950

Jisun An: And then asking, then, what is the sentiment of this sentence and then asking for the output. So that's the few shot

262

00:56:29.200 --> 00:56:30.349

Jisun An: choose up from 2.

263

00:56:30.460 --> 00:56:32.909

Jisun An: Yeah. But we we talk more on that on Thursday.

264

00:56:36.380 --> 00:56:52.859

Jisun An: All right. So yeah, these are not very important. So I will just skim through so these are. There are a few instruction tuned model that you can also play with. These are 3 most popularly used ones. So these are now means that

265

00:56:52.950 --> 00:57:16.369

Jisun An: I mean, we will also see comparison between the instruction tuned versus like the pure base model, but the pure base model. Basically, you cannot basically have a conversation with it. But then, once things are instruction tuned, then their answer will more make sense, and they will interact with you in a way that answering back to your problems or your questions. So these are model. That would be interesting to explore.

266

00:57:17.240 --> 00:57:40.510

Jisun An: And another question would be so, how do we then construct that instruction data set? I mean, that has been also one big hot topic. And I'm just. I'm just providing a few examples. So one popular data set is the natural instruction. And so what, after they found out that instruction tuning is helping for generalizing the they was crazy with

267

00:57:40.510 --> 00:58:07.379

Jisun An: creating. The 1,000 1,600 different tests, and they create this like giant instruction data set using the experts written those instructions. So they were kind of comparing with. So basically they covered like very diverse tests compared with like the fun or instruct Gpt, and they compare that they are basically have far more data set. And that helps to certain extent

268

00:58:08.480 --> 00:58:26.070

Jisun An: another way to instruct. I mean, create this. The instruction data is you can also use to generate this data. So this self instruct was the paper that they automatically generate the instruction tuning data, using the Gpt and then.

269

00:58:26.070 --> 00:58:44.079

Jisun An: yeah. And then it, they show that you're also working well. So so there was. The trend of this research they're trying to create like, more better large instruction data set. But then, interestingly, the recent work also found that

270

00:58:44.090 --> 00:59:01.539

Jisun An: you actually don't need like gigantic instruction data set, you actually need very the quality is basically important. So here, instead of using gigantic amount of data for the instruction tuning this paper. We're choosing 1,000 prompts

271

00:59:01.800 --> 00:59:18.840

Jisun An: and the responses, and like, without any further like fancy techniques. They found that these stuff were ensuring good quality of the Internet in in various tests. So that's also some interesting trend in the was in the Nfp.

272

00:59:22.180 --> 00:59:30.809

Jisun An: Right? So before moving on to here. Any questions, all right. So

273

00:59:31.020 --> 00:59:56.964

Jisun An: so if you think about not sure how much I can do. But so we now will talk about the parameter, efficient, fine tuning. So once again, the fine tuning itself is now you have bunch of different parameters in your model. We are. And we are just updating based on each of these downstream tests. Right? And there's the obvious reason that why people cares about being efficient.

274

00:59:57.560 --> 01:00:19.840

Jisun An: and this is the figure. So we will talk about like prompt tuning, which is different from different from prompting. This is a figure from the prompt tuning paper, and this shows a good motivation for why we need a being efficiency. So now the edit and become more and more powerful, and there are many different tasks you can do. But what if you

275

01:00:19.840 --> 01:00:38.720

Jisun An: had to build a new model for every single new task? Then basically, it means that like pre-trained model, is already 11 billion parameters model, then, for each of the tasks you may need, like individual separate models. Right? If you are just using the normal, supervised fine tune, fine tuning method right

276

01:00:39.400 --> 01:00:59.420

Jisun An: then. Now, this will be firstly even though we are just simply saying here as a 11 billion parameters. But if you think it's a space or the memory usage, and especially if you, a company that are serving different services based on this, and this will be just a huge waste of your all the cost.

277

01:01:00.080 --> 01:01:04.480

Jisun An: So so people wanted to find a way to

278

01:01:05.162 --> 01:01:23.350

Jisun An: have more generalized model that were used for various tests, or the way to share these parameters, that is dedicated for a particular task, and etc. So there are a couple of different way that how people go with it.

279

01:01:24.530 --> 01:01:34.769

Jisun An: So the parameter efficiency, fine tuning is based. The high level idea is so we don't want to tune or the parameter, but just some

280

01:01:35.206 --> 01:02:02.179

Jisun An: and once again, if we have 11 million parameters as a model that has then to train them. We need also their gradients and also the optimizers. So the memory requirement will be easily doubled up. So they there are different. So they kind of try to find a way to use, like lesser number of parameters, to be tuned, and see whether the performances still keep up.

281

01:02:02.310 --> 01:02:03.420

Jisun An: So

282

01:02:03.700 --> 01:02:25.857

Jisun An: that's the what the peft is try to achieve. And there are various strategies. But I will introduce the most popular ones which is prompting and prompt tuning. And Laura I mean, basically, I mean, prompting is not a strategy. But but it's kind of interesting, because even as an input,

283

01:02:26.260 --> 01:02:34.769

Jisun An: if you are giving a prompt as an input, then that will activate some part of the networks, and then that will lead to a getting the

284

01:02:35.060 --> 01:02:38.220

Jisun An: good oh.

285

01:02:38.370 --> 01:02:48.009

Jisun An: reverse. But so the prompting itself is I is is required basically like the like, the just in general parameters. So you are not

286

01:02:48.070 --> 01:03:14.999

Jisun An: updating anything. But you are just using the M is the prompting. And what you are doing or chatting with the Chatgpt is basically you are doing prompting. and it's it's a kind of interesting because we are now thinking, prompting as a 1 technique to like do these up sub downstream tasks. So, assuming that you have a particular task. You can still do the prompting, just asking the chatgpt to get the answer right? And basically, this requires no parameter adjusting.

287

01:03:15.520 --> 01:03:31.670

Jisun An: So a simple example here would be so without any fine tuning, we can just ask, like, what is the sentiment of the below sentence? Answer with either positive or negative. You will have the input sentence. You will have the output, and this output will be just coming out from the edit app.

288

01:03:31.790 --> 01:03:36.289

Jisun An: But so now, prompting may sound like very common, but

289

01:03:36.490 --> 01:03:39.323

Jisun An: actually prompting is very expensive.

290

01:03:40.855 --> 01:03:51.809

Jisun An: method, because to do the prompting, you need to have really good edit. And and I mean, that's why we are paying for like Chatgpd, sometimes. Right?

291

01:03:52.180 --> 01:04:15.950

Jisun An: So so you can still tackle different soft tests, using only really prompting. But there could be some limitations, I mean, firstly, these are just the prompting, so the it may not. It will be limited to solve something very complex reasonings or the understanding task, and also it requires a good edit them like Gpt, or even larger.

292

01:04:16.280 --> 01:04:18.140

Jisun An: But then these

293

01:04:19.200 --> 01:04:40.150

Jisun An: this larger models basically requires a huge scale pre-training. And also you need high scale instruction tuning that requires tremendous amount of the instruction tune data. And also we haven't talked about it. But the reinforcement learning from human feedback is basically they create like a free preference data. So

294

01:04:40.150 --> 01:05:05.719

Jisun An: comparing 2 different data points, and which one is actually better than the other. So these models are already tuned with multiple steps, with the large amount of data. So unfortunately, some of them. Now, if you are looking at different models. The architectures are not that different to each other, but what makes one is better than the other is the quality of these data, the instruction data and the preference rate data. So sometimes, just the

295

01:05:05.880 --> 01:05:22.129

Jisun An: now they are competing with the how good data they can, they can create. And big companies are spending lots of money to. I mean, hire people to notate all this data and based on that, they can generate good models. So so prompting.

296

01:05:23.396 --> 01:05:47.219

Jisun An: has some limitation, and this this red part is the prompt and to certain you can do. Change this prompt to improve the the the performance of the task that you have. But so the prompt and that is called as a prompt engineering. But basically, no one have found a scientific

297

01:05:47.220 --> 01:06:11.460

Jisun An: evidence that how prompt engineering is actually working. So there are a few tactics, and there are also some good problems that is helpful for reasoning in particular, but in general, and also we will talk about this from engineering on Thursday. But in general, from engineering also has some limitation to achieve a good performance on a particular downstream task.

298

01:06:12.220 --> 01:06:19.569

Jisun An: and that is the one of the results that we seen from the prompt tuning. And here the

299

01:06:19.991 --> 01:06:43.900

Jisun An: the blue line, is the the achievements performance achievements based on, only based on the prompt design. So basically, you can assume that red part. You just change it a little bit, little by little, and even though you are get, you are doing your best to get the best result. So the other 3 leaders are basically the tuned, Richard, so you can kind of see that.

300

01:06:44.285 --> 01:07:01.250

Jisun An: The the difference in the performance are almost like 20 points in this particular tests. And also there are some improvement over the larger models. But the there are some limitations that the prompting itself is actually can give back to you.

301

01:07:01.710 --> 01:07:24.899

Jisun An: So the point here. If you want to really achieve a good performance for a particular subtask, then you probably need to do some kind of fine tuning or or tuning on on the on the edit, and it depends on like what your what what you will work on it. But so fine tuning will be really helpful, and the efficient way of doing fine tuning will be very important.

302

01:07:26.335 --> 01:07:52.640

Jisun An: So so we talked about the fine tuning. But let me just quickly go over so assuming that we have these inputs, and we are now these inputs, we're going through the pre-trained, the decoders. And we have the output last hidden layers of the output embeddings. And then the the last one basically can be tested to predict something. Also. These are like fine tuning, the for the

303

01:07:52.870 --> 01:08:21.569

Jisun An: the full model. So here we already talked about it. But once you get the output, then you will compare with the we are true value, and then compute the loss, and then the loss will be back. Propagate through all this entire network. And here we are interested in computing all the gradients. Of the loss in the respect to data data is the all the parameters that in the architecture. So.

304

01:08:21.930 --> 01:08:43.244

Jisun An: for example, in the even in any other models. So we already talked about that these are the parameters that are in the transformer architecture. So it would be all the attention parameters. So the Wq. Wk. Wv, and also all the parameters in the feed forward networks. So let's say, Wjet, and also the input embedding. So

305

01:08:43.939 --> 01:08:49.160

Jisun An: the loss will be propagated back to all the parameters.

306

01:08:49.430 --> 01:09:10.585

Jisun An: But then, once again. But then, if the model is now getting bigger and bigger that there are just far more parameters to be updated. So the prompt tuning was one of the 1st idea that let's fine tune the model more efficiently. And here the idea was very interesting. And so on top of I mean,

307

01:09:11.300 --> 01:09:22.040

Jisun An: so so we we have this input embeddings. And we actually have these 2 additional embeddings, and we concatenate them in front of these inputs.

308

01:09:22.490 --> 01:09:27.180

Jisun An: And now these embeddings are serving as a

309

01:09:28.375 --> 01:09:43.420

Jisun An: embeddings that representing the prompt itself. So you are trying to. So the model will be trained to find I mean the optimize for these particular embeddings that are dedicated for the prompt for a particular task.

310

01:09:43.720 --> 01:09:44.779

Jisun An: So

311

01:09:45.359 --> 01:10:07.150

Jisun An: so what we're gonna happen. So once again, these e, 1 and E 2 is nothing is not it's not any, not. It's not the any of the tokens that you are observing from the input. It's just a new embeddings. And you are just adding it in front of these inputs, and then they will be fit into the decoders. And then in the last layer you will predict, and then

312

01:10:07.310 --> 01:10:12.789

Jisun An: what we are only interested in is the these sort of updating these 2 embeddings.

313

01:10:13.710 --> 01:10:31.529

Jisun An: So we are just interested in the loss. The the derivatives of the loss in respect to the e 1 and E 2. So the model will just compute these derivatives, and they will update the e 1 and E 2 only. So basically the in the from from tuning

314

01:10:32.100 --> 01:10:43.219

Jisun An: they will fix and frozen all the remaining parameters, and they will only update this e 1 and e. 2, using the fine, but using the same logic of the fine, tuning the model.

315

01:10:45.820 --> 01:11:00.069

Jisun An: So once for each of the example, you will also compute the loss, and then the backdrop will be now coming through this architecture. But then the actual update will be only done by the e 1 and E. 2.

316

01:11:02.950 --> 01:11:04.409

Jisun An: Does this make sense?

317

01:11:06.150 --> 01:11:12.122

Jisun An: So you can also this you can. You can imagine that

318

01:11:13.890 --> 01:11:19.570

Jisun An: in the prompting you are writing like, okay, like, like, classify the

319

01:11:19.880 --> 01:11:25.049

Jisun An: please answer the sentiment of this sentence, right? So you, we were writing in human text.

320

01:11:25.360 --> 01:11:38.169

Jisun An: But you can imagine that this e, 1 or E 2, each of them are basically for separate task. And this embedding is representing that sentence. Please, like classify this sentence of this.

321

01:11:38.290 --> 01:11:46.069

Jisun An: or classify the sentiment of the sentence, is the represented by this e, 1 or e 2. So these like

322

01:11:46.750 --> 01:11:59.749

Jisun An: prefix kind of embeddings and the the model will this model will learn to the best embeddings of these e. 1, so that this embedding will represent that particular prompt.

323

01:11:59.870 --> 01:12:03.250

Jisun An: And that's the why they called it as a prom tuning

324

01:12:11.120 --> 01:12:12.270

Jisun An: so

325

01:12:12.500 --> 01:12:22.139

Jisun An: so compared with the full model. Fine tuning and this prompt tuning? Do you think the prompt tuning will save the time of training?

326

01:12:23.750 --> 01:12:26.850

Jisun An: And why is it? Why, you would be saving the time?

327

01:12:28.910 --> 01:12:37.280

Jisun An: Come on right? Right? So what you mean is that the storage will be saved right?

328

01:12:37.660 --> 01:12:44.239

Jisun An: But I'm asking whether the time for training will be saved or not in compared with the full model functioning?

329

01:12:44.830 --> 01:12:49.579

Jisun An: Will the time? How long it takes to do fine tuning. Will it be saved?

330

01:12:52.390 --> 01:12:53.919

Jisun An: Maybe, or maybe not.

331

01:12:54.460 --> 01:13:14.727

Jisun An: I will give you the answer in the in the in the Thursday. Yeah. So that's a good thing to think about, whether in compared with the full model train. Fine tuning, whether the prompt tuning will save the time, meaning that how long it takes to train with the train given. They have exactly the same fine-tuned

332

01:13:15.240 --> 01:13:21.534

Jisun An: label data, whether they will be fine-tuned or not. Right? So yeah.

333

01:13:24.130 --> 01:13:49.819

Jisun An: So now, if you look back to this figure, the right side will be makes more, much sense so compared to the model tuning, meaning that for each of these tasks that you basically train on new model independently. But in the prompt tuning you can have a like set of tests and really a few examples of them, and then you can just have

334

01:13:50.374 --> 01:13:57.295

Jisun An: train, just one model, and then that will work for all these subtests. And

335

01:13:57.920 --> 01:14:22.469

Jisun An: If you have like, 3 different different tasks, then you probably need like more embedding, so like embedding for each task, so it can be more than 2 that I like than the example that I just showed in the previous slide. But this will be far more efficient in terms of the memory definitely. Because if you're not updates any other parameter, but only the input embeddings

336

01:14:22.961 --> 01:14:31.310

Jisun An: and but that still found that the prompt tuning was still quite effective. And so the

337

01:14:31.600 --> 01:14:57.547

Jisun An: the the orange and the red lines are the model tuning. So these are like the full, fine tuning versus the the green one is the prompt tuning, so if the model size is more, then there are still a bit of gap, but they both are not really working well. And then, as the models are getting increasing, the prompt tuning impact is getting increasing, and is as part as like similar to the model tuning. So that was their main reserves.

338

01:14:58.170 --> 01:14:59.859

Jisun An: that they've been talking

339

01:15:00.730 --> 01:15:09.530

Jisun An: alright so I will come back to here and then wrap it up from on on Thursday. So

340

01:15:10.060 --> 01:15:16.870

Jisun An: thanks a lot. Any last quick question. I'm sorry that I'm a little over time, but

341

01:15:17.340 --> 01:15:29.910

Jisun An: yeah, so I will. I will start from this prompt tuning on Thursday. So if you have any questions, we can also talk then, alright thanks a lot. And see you on Thursday.