WEBVTT

1

00:00:08.860 --> 00:00:18.169

Jisun An: Alright let's get started. Thanks for joining today. The passport is today's prompting. Please mark your attendance

2

00:00:20.260 --> 00:00:24.499

Jisun An: right? So as you know that. 2 of

3

00:00:25.280 --> 00:00:33.260

Jisun An: running assignments, the theoretical assignments due by the end of this Sunday. Please make sure that you complete it.

4

00:00:33.920 --> 00:00:43.430

Jisun An: We will have one extra day if you are not able to submit by the end of this Sunday, but you will receive only the 50% of the credit.

5

00:00:44.140 --> 00:00:53.650

Jisun An: And another thing is the team formation, which is also due by the end of the Sunday. So if you find the team members, then please mark your team.

6

00:00:55.650 --> 00:01:03.330

Jisun An: If you don't find a team member by the end of this Sunday I will just randomly group them. So if you prefer to work

7

00:01:03.490 --> 00:01:08.019

Jisun An: individually or particular preference, please leave the notes.

8

00:01:09.650 --> 00:01:11.569

Jisun An: So please do so.

9

00:01:14.790 --> 00:01:35.539

Jisun An: And I just had a question about like the scope of the projects. And some someone may be interested in more research oriented projects, but some may interested in application level projects, and both are fine. So the scope of the projects is quite broad for this class. So feel free to

10

00:01:35.680 --> 00:01:41.690

Jisun An: work on what you are interested in. And I think, yeah, generally

11

00:01:42.000 --> 00:02:08.379

Jisun An: like some problem that you are interested in, and then try to apply the Edm. For that problem would be a good way to think about the projects, and also because since we are not offering any Gpu resources in this class, you will be need to depending on the collapse or some other limited resources that are lobby providing. So we will also take that into consideration. So

12

00:02:08.380 --> 00:02:17.870

Jisun An: rather than bidding for or trying to achieve, the sota results like some results that you can achieve with this more model. I think that should be good enough.

13

00:02:18.060 --> 00:02:42.519

Jisun An: So as long as the problem is clear and the way that you conduct the experiments and how you kind of develop. Your own methodology is valid and clear. That should be the key aspects of it. I will. I will unload the criteria for the rubrics for the projects soon. But so hopefully that will be more clear. But for now I mean considering this a very open projects.

14

00:02:44.679 --> 00:02:47.599

Jisun An: Today's passcode is prompting.

15

00:02:51.150 --> 00:03:00.150

Jisun An: Okay, you can. You can do it. Just yeah, if you need to. Yeah, yes, yes. Please. Yeah.

16

00:03:05.520 --> 00:03:06.690

Jisun An: Case study.

17

00:03:06.900 --> 00:03:17.010

Jisun An: Like on I see the product needs. Okay, I. I was under assumption that you.

18

00:03:17.840 --> 00:03:30.679

Jisun An: you'd have follow some kind of typical project. So I so you need to have some kind of implementation aspects in the project. So I don't know what you mean by the case study.

19

00:03:33.447 --> 00:03:41.300

Jisun An: Have some expectations.

20

00:03:42.660 --> 00:03:47.090

Jisun An: Right? Cycle.

21

00:03:47.770 --> 00:03:55.750

Jisun An: Okay? Okay, did. I had a case like case study in.

22

00:03:56.400 --> 00:04:06.329

Jisun An: So maybe the the way that we how we interpret as a case study might be slightly different. So can we chat like later, like, what is your expectation? That?

23

00:04:07.570 --> 00:04:10.340

Jisun An: Yeah, I see.

24

00:04:11.120 --> 00:04:13.060

Jisun An: I sorry I I

25

00:04:13.570 --> 00:04:22.440

Jisun An: don't remember exactly the what part of the Project school was the case study. So I will. I will check back later, and then let's talk. Yes.

26

00:04:25.000 --> 00:04:25.690

Jisun An: huh.

27

00:04:25.900 --> 00:04:33.540

Jisun An: okay. So we will continue the rest of the fine tuning parts and then moving on to the prompting so

28

00:04:34.246 --> 00:04:40.753

Jisun An: so I I we have this one question about we didn't defined tuning. So the

29

00:04:41.240 --> 00:05:08.819

Jisun An: the full fine tuning. Basically. So here, basically, you have a given sentence. And then you have, like predicting some sentiment. If the task was the sentiment analysis. Then one of your output embedding can be set to predict the label, either positive or the negative, and then, after the loss will be backdrop through this network. And then every parameters in this pre-trained decoder. And also

30

00:05:08.820 --> 00:05:11.869

Jisun An: the input embeddings will be updated.

31

00:05:11.870 --> 00:05:33.790

Jisun An: And so here, basically, this pre-trained decoder, this box that I mentioned this. Basically in the transformer architecture, this is the part where this black this box is covering, for so it includes the the multi head attention, part and the feed for that. So these are essential of the transformer architecture. And basically there are n of them.

32

00:05:34.450 --> 00:05:54.720

Jisun An: you can kind of assume that there are enough to make a transformer. So in this case, the way that this model will be tuned is basically every parameters in the transformer model will be fine tuned, updated, based on the results of this comparison between the predicted label versus the gold label.

33

00:05:54.720 --> 00:06:18.889

Jisun An: Once again, the the transformer architecture usually have the parameters, for in the attention mechanism, so wet weight, metrics for the query key, and the value, and also all the parameters in the fit for the networks and also input embedding these will be updated. But, on the other hand, the prompt tuning was some interesting technique that basically the idea was, they will

34

00:06:18.890 --> 00:06:36.309

Jisun An: freeze all the parameters that are originally trained for the pre-trained model. But they will have this additional or extra embeddings input embeddings for dedicated for prompt itself. And then those input embeddings will be only fine-tuned.

35

00:06:37.030 --> 00:06:53.210

Jisun An: So the the sequence or the way that it will be fine tuned will be very similar. So instead of now, instead of having these 4 tokens for this input, you will have 2 additional embedding. So in this case we are assuming there are 2 different tasks.

36

00:06:53.210 --> 00:07:17.160

Jisun An: So e, 1 and E 2 represent different tasks. So maybe e 1 here doing sentiment analysis, they need to maybe doing something else but just, I gave it as an example. So in this case, if we are predicting the positive, then, rather than updating all the other parameters, they will only update the e 1 and E 2, which is the new embedding for this prompt

37

00:07:17.674 --> 00:07:22.289

Jisun An: and then this will be updated, based on the using the gradients.

38

00:07:22.600 --> 00:07:40.579

Jisun An: so the the rules will be back propagated. But then the the architecture itself may, will not update. But these impacts will only go into these embeddings. And here it may be also confusing, because I also made it as a e 1 e. 2, both like solid line, but technically or in in.

39

00:07:40.580 --> 00:07:55.069

Jisun An: I mean, because these are all tuned, they will try to impact. But then, if one embedding was dedicated for one task, then basically e, 1 will be updated. But E, 2 may not be really updated if it is not relevant to the task itself.

40

00:07:55.393 --> 00:08:01.209

Jisun An: So that is the something that the prompt training is doing. So the major difference was like

41

00:08:01.550 --> 00:08:29.480

Jisun An: which parameters are basically updated doing the doing kind of fine tune this pre-trained model to do like sentiment analysis. So the question was, will this prompt tuning save time to train this in? Compare with a full model? Full model? Fine tuning. So what do you think that? Would it? Would this prompt tuning save the time to train them, train the model for that particular task?

42

00:08:32.590 --> 00:08:34.780

Jisun An: Maybe maybe not.

43

00:08:40.610 --> 00:08:50.660

Jisun An: So well, I mean technically if we consider that it will also take some time to update the parameters.

44

00:08:50.660 --> 00:09:15.460

Jisun An: If we are doing full model fine tuning that it will take slightly more because you need to update all the parameters. But if the majority of the time spent for fine tuning is dedicated for running through all the examples to fine tune, then that will be consuming like the major time. So in that sense the 2 model will need to go through the exact same steps, right? They need to go through all

45

00:09:15.460 --> 00:09:34.639

Jisun An: 1,000 training example to fine tune in. So they also need to do all the forward pass to predict the label for the input sentence, and then they need to backdrop. So these, all the steps of that required for the fine tuning they both model need to go through exactly the same.

46

00:09:34.640 --> 00:09:48.629

Jisun An: The only difference is just. One is updating the parameter, the other is not up in the parameter, but that would be because they also do the back problem without like actually updating the value. That would be the only difference. So

47

00:09:49.054 --> 00:10:13.379

Jisun An: the prompt tuning and the full model. Fine tuning. All the steps that they need to go through will be very similar, and those the time it would require will be also similar, even though the full model will take a little bit more time, because they also need to update the parameters while doing the backdrop. But both need to go through the old 1,000 samples. They also need to do all the poor passes they need to

48

00:10:13.380 --> 00:10:36.029

Jisun An: predict the label. They also need to compute the loss. Also, they need to back. Prop those value till the the all the 1st layers of the network, which is the input embedding so prompt tuning. They update the embeddings that exist in the input so every values will be need to come down to the 1st layer. So the both model will require the entire path of

49

00:10:36.030 --> 00:10:37.270

Jisun An: fine-tuning.

50

00:10:37.530 --> 00:10:44.269

Jisun An: so in terms of the saving time, both model may be more or less similar, but full mode is slightly longer.

51

00:10:44.570 --> 00:10:52.779

Jisun An: but then the the rear win is mainly coming from the save of the memory. So basically, the

52

00:10:52.890 --> 00:11:06.459

Jisun An: the number of the parameter that need to be updated in the full model it, it basically depending on the size of the model, which can be very, very large. So that will be the major difference. So that's the one thing that I wanted to highlight in the prompt training.

53

00:11:07.366 --> 00:11:08.260

Jisun An: Any question.

54

00:11:13.000 --> 00:11:13.875

Jisun An: Okay,

55

00:11:15.040 --> 00:11:39.450

Jisun An: so. And so if you think about why the prompt tuning is important now, probably you've seen that, like Chat, Gpt also support for the fine tuning, even though that's like insanely expensive. But they are still support the service for fine tuning their model. And but then they never actually say what particular technique they are using. So it's likely that they are either using this prompt tuning or some other

56

00:11:39.761 --> 00:11:49.409

Jisun An: Laura, that I'm going to show you very, very shortly, and the reason that we people assume that they probably use one of these techniques because I mean, they cannot

57

00:11:49.410 --> 00:12:15.250

Jisun An: maintain, like different models for single different downstream tests that, like user retest for right, it will be just giant amount of the memories and the storage that would be required. So they are likely to use this kind of prompt training methods. So in this case, because all the parameters are fixed and froze, so the pre-trained parameters will not change at all, and you only need to update the prompt embeddings.

58

00:12:15.250 --> 00:12:37.420

Jisun An: And then, if you ever want to share these tasks to the others, then you only need to just share the parameters that updated parameters of these tests. So that's also the efficiency and also good for the community as well. So from tuning is the one of the popular techniques that we assume that that would be used within different services.

59

00:12:39.219 --> 00:12:53.049

Jisun An: Yeah. So that was the prompt tuning. And we talked about that. Basically it especially, for like sizable models from tuning, was good enough, comparable with the more model tuning performance as well.

60

00:12:53.080 --> 00:13:19.350

Jisun An: even though what this graph actually shows that. So the prompt tuning is. Actually, you are fine, tune the model with the examples and the prompt design here is just we are using the prompt prompting, the kind of prompting which we will talk today more about it. Then there are some gaps between the performance but so prompt tuning was good enough compared with the model tuning

61

00:13:20.480 --> 00:13:47.580

Jisun An: and and the. And apart from this prompt tuning, a most commonly used technique is actually the Laura low rank adaptation methodology that presented in 2012 one. And this one is also initially, people didn't buy. The methodology was not very popular, but nowadays I think this is the mostly commonly used method, because they found that this is actually quite effective and also working very well.

62

00:13:47.620 --> 00:13:50.748

Jisun An: And here the idea is now

63

00:13:51.540 --> 00:14:16.129

Jisun An: They found that to do fine tuning. They can. They realize that it actually works by updating only the low rank matrices. So basically they freeze the pre-trained rate. And they define a new, like low rank matrices that need to be updated during the fine tuning, and they just combine some these 2 matrices, and they found that this is working

64

00:14:16.190 --> 00:14:26.039

Jisun An: quite well. So this is the Laura. So for those who are not very familiar with the rolling, low, low link, low rank metrics.

65

00:14:26.680 --> 00:14:45.999

Jisun An: So the low rank metrics is is the metrics whose rank so rank is like number of the linearly independent rows or the columns, is much smaller than its full dimension, so a low rank or a matrix is rolling. If there exists a decomposition of U metrics, U. And W,

66

00:14:46.000 --> 00:15:06.425

Jisun An: where the multiplication of the UN transformation of V turned back to the matrix a so and so, if this relationship can be found, then these, a a matrix is easy, low rank, and they found that just updating these role rank metrics is good enough to

67

00:15:07.110 --> 00:15:26.150

Jisun An: learned the information from the fine tuned training examples. So so this idea was now applied to the Laura, and they found it something quite interesting. So as an example. So assuming that we have this weight matrix, that is like 10 by 10. So this is just an example.

68

00:15:26.150 --> 00:15:45.679

Jisun An: So the low rank weight matrix is rank with the rank. One would be looking something like this. So now instead of so if you have 10 by 10 metrics, then your total number of parameter is 100. But then your low rank metrics will have basically 20 parameters. So it's about like 5th one over 5, th

69

00:15:45.780 --> 00:16:11.070

Jisun An: 5 times smaller than your entire metrics. And if the rank is 2, then basically, you use like 10 by 2, and 2 by 10 metrics. So here, if you multiplying the 10 by one and one by 10 metrics, then you will have back 10 by 10 metrics. Right? So. But instead of updating all those 100 parameters.

70

00:16:11.070 --> 00:16:37.150

Jisun An: they found that just updating this 20 parameters are good enough, and they can contain all the information that require for the fine tuning. So the Laura is like actively adapting this idea. So coming back to here now, hopefully this idea, this figure bit more clear to you. So for so and so, this metrics, you can think any metrics that exist in the transformer metrics. You can apply the Laura

71

00:16:37.522 --> 00:16:44.979

Jisun An: so, for example, assuming this weight matrix, here is the WK. So these are weight metrics for the keys.

72

00:16:45.760 --> 00:17:14.510

Jisun An: Then you don't touch the the pre-trained weight weight metric itself, but instead, you create a new set of low rank metrics that are the multiplication of the 2 became the same size of the Wk. And then you. So this will be like far less number of the parameters that need to be fine tuned. So so you train. So you you just freeze these pre-trained weight, but you only fine tune those the low, rank metrics.

73

00:17:14.550 --> 00:17:34.242

Jisun An: And then, after you and then, when you actually use these metrics, you just combine these 2 because they are the same same size of the metrics. So you can just sum them. And then you can use it for the back end. I mean, every everything else can be done similarly, like the forward path and the backwards

74

00:17:35.506 --> 00:17:42.110

Jisun An: can be done in this low rank kind of metrics. So that was the entire idea. And

75

00:17:42.290 --> 00:18:04.019

Jisun An: even though there are many metrics that you can update in the transformer metric, though this paper found that just updating the key and query and value matrixes are enough. So you don't even need to update the feed forwards. Weight matrixes. They don't need to be updated for showing the good results. So just

76

00:18:04.020 --> 00:18:22.350

Jisun An: applying the Rora to the key and Korean value, matrices were showing a good performance on different Nfp tasks, and that was the main research that this paper found, and the Lora has been now the most common practice to be used for fine-tuning the large large language models.

77

00:18:22.350 --> 00:18:40.576

Jisun An: And here, before we were, I was kind of talking more on the aspects of like the parts, which is quite small model. But Laura is now used for much more bigger models, at least a few 1 billion parameters. So, using the low rank metrics can save a lot a lot of

78

00:18:40.990 --> 00:18:43.180

Jisun An: I mean, increase a lot of efficiency.

79

00:18:43.766 --> 00:18:48.530

Jisun An: So that was the something, Laura. Any any questions.

80

00:19:01.740 --> 00:19:05.970

Jisun An: She's welcoming us.

81

00:19:07.120 --> 00:19:15.989

Jisun An: So so the the basics of the machine learning is they don't choose which parameter to update.

82

00:19:16.110 --> 00:19:18.650

Jisun An: They just update all the parameters.

83

00:19:20.170 --> 00:19:26.330

Jisun An: So here the the the yellow one that will, every parameter will be updated.

84

00:19:26.800 --> 00:19:30.969

Jisun An: It's just a far smaller than the the original metrics.

85

00:19:35.390 --> 00:19:38.520

Jisun An: So but then and then to continue to that.

86

00:19:38.630 --> 00:19:54.140

Jisun An: so there could be different parameters to updates. But then this paper found that the Wq. Which is the rate metrics for the keys and weight. Metrics for the value and weight metrics for the query are good enough to updates.

87

00:19:55.080 --> 00:19:56.340

Jisun An: Does it make sense?

88

00:19:57.320 --> 00:19:58.010

Jisun An: Excuse me?

89

00:19:58.750 --> 00:20:04.779

Jisun An: Oh, cool, yes. So once again, so, assuming this.

90

00:20:05.090 --> 00:20:08.849

Jisun An: 10 by 10 would be the weight metrics for the keys.

91

00:20:10.000 --> 00:20:16.510

Jisun An: But then, instead of updating that one, the the right, the right one, which is the low rank metrics. You can just

92

00:20:16.830 --> 00:20:27.039

Jisun An: use smaller number of parameters to update for the fine tuning, and that was still performing as good as updating like entire metrics.

93

00:20:30.980 --> 00:20:44.319

Jisun An: So these are not coming from. You are not selecting parameter from them. You just create new new 2 metrics which is initialized randomly, and then those value will be just updated.

94

00:20:44.740 --> 00:20:57.779

Jisun An: Yeah, so it's not choosing from this. Yeah, yeah, yeah. But I, yeah. So they just create the the the random and create with the proper rank. And then they just randomly initiate. And then they update.

95

00:20:58.160 --> 00:20:59.649

Jisun An: yeah, okay, thank you.

96

00:20:59.990 --> 00:21:02.460

Jisun An: Okay, yeah, that's that's a good question. Thank you.

97

00:21:04.320 --> 00:21:21.550

Jisun An: Yeah, I think if they just show that it's as good as fully fine-tuning, fully fine-tuning will be always better so. Laura sacrificed a little bit of performance. But it's far more efficient. So I think they are just accepting that degrade.

98

00:21:21.550 --> 00:21:44.799

Jisun An: So if some task is really important, like 0% 1%, like performance is important, then they and they can afford. Then they probably need to do fine full, more than fine tuning. But the efficiency is just like a lot different.

99

00:21:46.370 --> 00:21:51.340

Jisun An: you but so, but the each each mark, each

100

00:21:54.690 --> 00:22:00.039

Jisun An: So so what they found is that each layer they don't need to update it

101

00:22:00.250 --> 00:22:11.209

Jisun An: so the attention mechanism. They don't have layer, but they only have key query, value matrices. So like updating, I mean applying the Laura to those metrics were already effective enough.

102

00:22:11.500 --> 00:22:15.059

Jisun An: That's the they are finding. But it may depending on also different tasks.

103

00:22:15.190 --> 00:22:32.160

Jisun An: But my understanding is that even though in a different layers, you can also choose which layers to freeze, and which layers to apply to Laura as well. So there are a lot of diversity and freedom. But the the interesting idea here was that these can be applied any parts of the

104

00:22:32.623 --> 00:22:54.329

Jisun An: in the transformer architecture, and you can choose where you want to add, now that we became more engineering kind of efforts. Technically, it can be anything or everywhere. And maybe that will result in good results. But I think you are just trading up between the accuracy and the efficiency here. But that's that's a great point as well. Yeah.

105

00:22:59.686 --> 00:23:03.640

Jisun An: No fine tuning time, inference.

106

00:23:03.760 --> 00:23:08.390

Jisun An: inference will be the same, because eventually they will use both the full metric.

107

00:23:08.590 --> 00:23:19.339

Jisun An: the the pre-trained weight, and then the Laura fine-tuned metric, and they will use the sum of them as our next input of the next layer, so the inference will be the same.

108

00:23:20.920 --> 00:23:22.310

Jisun An: It increases paying.

109

00:23:23.810 --> 00:23:39.460

Jisun An: So pre-trained is already done. It's only the fine tuning. So once again, these are also going through all the examples for the fine tuning. So time wise, I think there's no no saving it's only the Gp memory saving.

110

00:23:40.390 --> 00:23:47.020

Jisun An: but it has even compared with prom tuning which you are. Didn't touch the architecture at all.

111

00:23:47.330 --> 00:23:57.479

Jisun An: At least the Laura should update some part of the attention mechanism as well, so it will learn more than simple, prompt tuning.

112

00:23:57.780 --> 00:23:58.540

Jisun An: So

113

00:24:02.550 --> 00:24:18.420

Jisun An: so you can think it. This Laura metrics are assisting the the pre-trained metrics and then and then and we'll have the similar impacts of the what the attention mechanism or fit forward layers do but just instead of

114

00:24:18.470 --> 00:24:44.370

Jisun An: tuning the entire parameter, they could just use this low rank metrics to assist each component within the architecture, and they will learn that's needed for for them. So once again, like attention, mechanism learns about relationship among these words, and the fit for network will learn about combination of all those features as well. So there could be something slightly different. But but

115

00:24:44.460 --> 00:25:06.790

Jisun An: so, yeah, maybe for some tests, maybe the attention may not be important, but then maybe the fit forward network could be more important. There could be some cases, but in general, among general Nlp tests. They found that updating the attention mechanism. I mean using the Laura for the attention mechanism was more important than using the fit forward part and etc.

116

00:25:07.960 --> 00:25:11.150

Jisun An: Oh, yep, any other question.

117

00:25:15.260 --> 00:25:41.909

Jisun An: Okay. So even though this Laura sounds very simple, if we are doing actually running, it takes forever. And we will actually do do do try to how to use the Laura in the lab session. Even for a very small data. It takes really, Collab, it takes like an hour or so but still, it's kind of interesting that you can actually fine tune the large model with this technique with like 3 each

118

00:25:42.255 --> 00:25:50.550

Jisun An: platform. So that'll be yours. So that will be part of the lab in the next next week the fine tuning session.

119

00:25:50.750 --> 00:25:52.889

Jisun An: So we will try to use the Laura.

120

00:25:52.930 --> 00:26:11.080

Jisun An: And last thing that I want to talk with in defined tuning is the Q. Laura, and this is the quantized Laura and I. This is like also in. There are so much work here. So now we talk about efficiency because the model is really big

121

00:26:11.080 --> 00:26:27.290

Jisun An: is not accessible to like everyone. So people are really trying to use various technique to reduce the the use of the memories of use of the like training times, and etc. So there are many different techniques, and

122

00:26:27.390 --> 00:26:38.800

Jisun An: and this the quantized Laura, is that they they also applied the they want to apply the Laura. But in the quantization model.

123

00:26:39.240 --> 00:26:42.334

Jisun An: So what is the quantization? Is?

124

00:26:43.436 --> 00:26:50.229

Jisun An: so so we talked about so to represent a single floating points.

125

00:26:50.230 --> 00:27:14.239

Jisun An: So the Fp floating point 32 or Fp. 16, or the the Bp. 16. So we talked about how to use a smaller number of bits to present the floating numbers right? And now. So what it means is that to present one floating point you need 32 Byte, and then here the quantization is now, instead of using those 32

126

00:27:14.480 --> 00:27:36.639

Jisun An: by bits. You are using 4 bits to to represent all those values. So it may sound really crazy, because I mean floating points are basically just the infinite numbers. And how then, can we represent them in like 4 bits? But these are now we are talking. So we have the model that is already trained.

127

00:27:36.640 --> 00:27:46.260

Jisun An: So we have the parameter that are already fixed. So you have fixed number of parameters, and you just quantize. You just

128

00:27:46.290 --> 00:27:53.860

Jisun An: put those values into, quantize it into the 4 bits. So in that sense it would be possible, because you have

129

00:27:53.860 --> 00:28:22.768

Jisun An: the like, not infinite number. But you have a finite numbers of the parameter, and you just somehow put all of them into the 4 piece. So so there are different techniques. How you can transform the this the floating values in represented in ft. 32 into the int. 4. And the easiest way that you can think is you can find the maximum and the minimum value from the your numbers, and then you also map exactly

130

00:28:23.450 --> 00:28:47.970

Jisun An: to the into 4 and but then that may not be really accurate, because there will be like smudging some numbers. So there are like different things that you now create, like the different blocks. And then you are kind of map those values. So once again, there are different methods. How to you how you can actually do better on this quantization. But then that's the basic idea. So you are now squeezing the values that are represented in

131

00:28:47.970 --> 00:28:56.859

Jisun An: fp, 32, or like Bp, 16. And you kind of map them into the int 4. So you are representing 4 bits.

132

00:28:58.135 --> 00:28:59.410

Jisun An: quantization.

133

00:28:59.650 --> 00:29:25.350

Jisun An: And and then also another technique that they used is instead of using Gpu. They also use like CPU memory. So they like basically move some information to the CPU, and then just read them when they are calculating this thing. So so what they found that in using this kind of technique they found they were able to kind of train 65 billion models.

134

00:29:25.350 --> 00:29:35.019

Jisun An: 48 GB. Gpu. This is another like efficiency kind of tips and techniques that have been done in in it.

135

00:29:35.100 --> 00:29:56.670

Jisun An: And the reason that I'm introducing this is some very interesting ongoing research, and also in our lab. We will also use the quantized model version of the model, because that is far more smaller. And so the goal here is that they are using less memory, but keep the performance once again. Here they also have a bit of degradation in the performance.

136

00:29:56.670 --> 00:30:06.079

Jisun An: but the the size of the model will be far smaller than what's was originally trained. So that will see something that we will also see.

137

00:30:07.970 --> 00:30:12.730

Jisun An: Okay, but that's the basic idea of this quantization. And yes.

138

00:30:16.500 --> 00:30:31.840

Jisun An: yes. So even for training. Now, people are sometimes using the the, this, this technique in particular. But here is the particular one that applied for the laura. So yeah.

139

00:30:33.090 --> 00:30:36.670

Jisun An: but the quantization itself can be used for other purposes as well.

140

00:30:40.100 --> 00:30:52.764

Jisun An: All right. So that was the last slide for the fine tuning. So I hope that that's

141

00:30:54.200 --> 00:30:57.800

Jisun An: that things were interesting.

142

00:30:58.554 --> 00:31:06.900

Jisun An: So I guess the most important part was like instruction. Tuning is basically

143

00:31:07.600 --> 00:31:18.840

Jisun An: fine-tuning, but with a very special data set. And to do it, you needed kind of like different masking, like the partial masking to do. But then

144

00:31:19.030 --> 00:31:40.630

Jisun An: so the masking is the only thing that you need to apply for, and then the rest of the training. The fine tuning is quite similar to like very general kind of training the model. So each of the output embedding will be test to predicting the next to tokens. So

145

00:31:41.520 --> 00:31:43.559

Jisun An: that's the something. And then

146

00:31:45.450 --> 00:32:08.170

Jisun An: but that this was like fine tuning for text generation. But if you are interested in fine-tuning your model for a particular tasks like sentiment analysis, then you can do like prompt tuning, or this Laura or Q. Laura. And also there are some other methodology, obviously. But but these are some of the things that are most popular and commonly used. These days.

147

00:32:10.100 --> 00:32:13.139

Jisun An: So I will move on to the prompting. Yes.

148

00:32:13.620 --> 00:32:42.910

Jisun An: So you have researched about a different. So is there any research that says, like every every fine tuning technique like, there will be some research like. They only talk about particular fine tuning. So is there any research that say that which one is better or less, not in terms of which one is better like, of course. But is there any combination of fine tuning techniques that someone else

149

00:32:43.860 --> 00:32:50.420

Jisun An: I see? So so I I guess these are

150

00:32:50.770 --> 00:32:54.859

Jisun An: especially for Prom tuning and the Laura they are.

151

00:32:56.730 --> 00:33:07.229

Jisun An: They can be combined as well. Right? There could be a paper that explore that as well. Actually, that's a good idea. I haven't seen it personally, but I assume there should be some

152

00:33:07.690 --> 00:33:16.820

Jisun An: paper that have tried that as well. But that's actually good. Good suggestion. Yeah, you you can, you can.

153

00:33:19.810 --> 00:33:20.760

Jisun An: But

154

00:33:23.360 --> 00:33:26.900

Jisun An: But to me.

155

00:33:27.160 --> 00:33:34.770

Jisun An: eventually fine tuning will depends on how more number of parameters you are using to fine tune.

156

00:33:35.010 --> 00:33:52.860

Jisun An: So I think that prompt tuning will be the the minimum number of parameters that you will update, and then the prompt Laura will at least have few more parameters to fine tune, and also it will involve the attention, mechanism, etc. So I think this.

157

00:33:52.990 --> 00:34:00.980

Jisun An: the the Laura will kind of cover the performance. The expected performance increase from the home tuning. So.

158

00:34:01.420 --> 00:34:15.689

Jisun An: even though you're combining them, I think the performance will be Laura. Only model would be as good as combining them. But but who knows what happens? So there are many mysteries left here. Yeah. But thanks for yeah, it is an interesting idea.

159

00:34:17.760 --> 00:34:26.005

Jisun An: Okay, move on to prompting. I hope today, trade class will be more fun, and it will be very

160

00:34:26.580 --> 00:34:54.780

Jisun An: just different ideas and different attempts people have been doing with the prompting. So I will try to introduce those aspects. So I've been already using this term of the prompting. But, like kind of former definition would be like encouraging a pre-training model to make particular pro predictions given the prompt. So by providing our the texture prompt and this prompt will have, like some tasks that we want to be done.

161

00:34:55.739 --> 00:35:20.130

Jisun An: So we want that these models are given our prompts, that they that they will predict the we hope that the next token predicted is answering to our prompt, and that's the generally code as a prompting, and we will talk about like basics of the prompting, and like the different method that has been developed and and tackled. And I guess nowadays from engineering, are

162

00:35:21.940 --> 00:35:39.509

Jisun An: So for some models, or for for some cases it would be very important, but in generally, as the model, getting larger and larger from things are slightly getting less important. But still I think it is useful to know these techniques and and try it out, so I will introduce them

163

00:35:40.250 --> 00:35:57.070

Jisun An: before talking about prompting. The one important concept to talk is the decoding or the inference. So how these models, the pre-trained models, are actually generating or choosing. The next token is the code as a decoding or the inference.

164

00:35:57.390 --> 00:36:14.759

Jisun An: So assuming. So now your input is, see, I am driving a and then you are now asking the giving me like the next token, then how the model choose, like, what token to generate. And if you think about how the model is working now, is that given this input

165

00:36:15.470 --> 00:36:35.039

Jisun An: for all the tokens, they will have the probability or the likelihood that which word is coming next. So they will have list of the tokens with the corresponding probability or the likelihood, and then they will select the one, and then they will just generate it right? So what would be what would be the easiest choice here

166

00:36:37.260 --> 00:36:55.539

Jisun An: the highest likelihood. Right? So that's the code, the greedy decoding. So if you have an input, and if you select the word with the highest likelihood, then that's called as a greedy decoding because they are greedingly choose the next most probable words.

167

00:36:55.890 --> 00:37:10.989

Jisun An: So there are different algorithms, algorithms decoding algorithms. And the 1st one is this greedy decoding that, as you mentioned, it's a simple enough. We are just selecting the most probable words, or, in other words, we are saying that we are using the arguments because

168

00:37:11.020 --> 00:37:27.820

Jisun An: you have the metrics. The list of the tokens with the corresponding likelihood, and you are selecting the index of that word with the largest value. So if you are taking the arguments, and basically, that will get the most probable words.

169

00:37:28.610 --> 00:37:54.620

Jisun An: And then you are now repeating this. So after I'm driving a you select the car. Then, maybe then given, I'm driving a car as an input and then you are doing this, repeat and then selecting the next problem token. And then this will lead to the text generation. That's the what the reserves that you are seeing from like the Chattpc. Then there could be some any. There could be other algorithms as well. So what would be alternative? Any ideas

170

00:37:55.420 --> 00:38:02.529

Jisun An: instead of most probable world. What would be the good way? What would be any any way to choose the token

171

00:38:06.970 --> 00:38:31.900

Jisun An: sampling? You can just randomly sample right out of the token, and that's the second method, sampling method. And and these are now compared to greedy, decoding, sampling method can get more diversity or the randomness, because I mean choosing the most probable may not be always the best way to do it. So sampling method, it's another one, and the within sampling method also, there are 2 different methods.

172

00:38:32.080 --> 00:38:48.219

Jisun An: One is the pure sampling. So at each of the steps, when you need to generate the next token, you just randomly sample one from the probability distribution, and just obtain the next word. But if you think about this, the pure sampling may not be desirable. I mean you

173

00:38:48.390 --> 00:39:02.429

Jisun An: train this model with so much money and so much time, and you are picking the random one, then it's literally the the performance should be more or less similar to without the model, without any training. Right? Because you're gonna randomly sample the tokens.

174

00:39:02.430 --> 00:39:24.690

Jisun An: So I mean, pure sampling is just the conceptual idea. But then, what normally people do is the top end sampling. So now you just rank all these tokens based on the likelihood, and then you choose. So if N is equal 10. Then you basically select the top 10, most likely word, and then sample so randomly, sample out of this 10.

175

00:39:24.690 --> 00:39:39.650

Jisun An: So that's the top end sampling. And this is the mostly commonly used method, because it's not as greedy as greedy, decoding. It gives us some diversity, but also it keeps the merit of the our trained likelihood.

176

00:39:40.470 --> 00:40:04.610

Jisun An: So these are the 2. The. But if you think about these 2, these are just there's no going back after you choose the next token. Right? So these 2 methods are just you're choosing next one and the next one and the next one. So you have only single one option, and you just don't go back to I mean. But if you think about there could be better way to generate

177

00:40:04.700 --> 00:40:21.269

Jisun An: a better solution or better way to select these ones. So so the 3rd general idea was, the beam searches. So those here. The idea is rather than keep one output, one token, you have at least like 2

178

00:40:21.480 --> 00:40:30.870

Jisun An: or more possible sentences, keep them alive, and then, you see which one lead to better results.

179

00:40:31.040 --> 00:40:32.606

Jisun An: So it

180

00:40:33.810 --> 00:40:41.119

Jisun An: So that's the idea of the beam search. So they explore several different hypotheses instead of just one single one.

181

00:40:41.120 --> 00:41:06.070

Jisun An: meaning that they keep track of K most probable partial translation at each decoder step and beam size usually like 5 to 10. So here, if the beam size is too small, then it'll be more similar to like greedy decoding or the top end sampling. But then, if the beam size is really really big, then basically, the inference computational cost will be quite large. So you need to select some reasonable

182

00:41:06.070 --> 00:41:14.759

Jisun An: K, and these are example of the beam. Search so, and the number here are like the measures of the

183

00:41:15.275 --> 00:41:23.789

Jisun An: I think this was so. The the likelihood of the sentence or the next word next probable words

184

00:41:24.230 --> 00:41:36.490

Jisun An: given the historical context. So what you can see. Is that so? They will always keep the. They will expand the search of the next token for those 2 example that has the highest likelihood.

185

00:41:36.490 --> 00:41:56.279

Jisun An: So they start, and then there are 2 options, he and I. So you just keep these 2 as tracking these 2. And then so for each of these 2 tokens. They also had a 2 other options, and out of these 4 you now choose the 2, the heat and the was because these are the 2 values, 2 tokens that are having low

186

00:41:56.280 --> 00:42:10.839

Jisun An: probabilities. And then so you, if expending like 2 possible tokens from that one as well, and then you choose 2, and then you keep trying it, and then at the end of it, you choose the sentence that leading to the highest probability. So that's the like

187

00:42:10.840 --> 00:42:39.930

Jisun An: very high level idea of the beam search. So we're not going to go more details on the inference. There are also a whole lot of literatures here, but in this course I think we will just keep it here. So there are 3D. Decoding sampling and the beam search most commonly used are top end sampling, and but the beam search could be expensive. But still it's also the option that give you some opportunity diversity, and maybe better results on the generated research.

188

00:42:40.560 --> 00:42:41.780

Jisun An: Any questions?

189

00:42:46.210 --> 00:42:48.991

Jisun An: Okay? So I will move on to the prompt

190

00:42:49.740 --> 00:43:03.189

Jisun An: so the prompting is so so the the basic idea is now given a whatever input that we are giving, we want to the models to predict the following text.

191

00:43:03.390 --> 00:43:28.120

Jisun An: So and so this, like the input, that we are giving it as a code, as a prompt. And so if we are giving like, when a dog sees our score, it usually, if we are giving this as an input to like Gpt, 2 models which are now like grandpa, the then there will be, I mean, still kind of makes sense. You can see that these tokens are just generated, based on their likelihood.

192

00:43:28.431 --> 00:43:33.420

Jisun An: It will be usually be afraid of anything unusual which doesn't make much sense, but like

193

00:43:33.997 --> 00:43:48.920

Jisun An: leak the score, I don't know. And I I tried this sentence with the Claude yesterday, and they said, like, yeah, it would usually bark, so I think the bar could be something more probable. So but these are like super what prompting is.

194

00:43:49.700 --> 00:44:13.209

Jisun An: and so prompting also mean many different things. But here I will talk about. We are using prompting to do some kind of task. So once again, let's assuming that we are doing like sentiment analysis. So from that moment, from that aspect prompting workflow usually follow these 3 steps. So we feel a prompt template, and we predict the answers, and you probably need to do a little bit of post

195

00:44:13.210 --> 00:44:29.370

Jisun An: process. The answer to see whether the answer is correct or not, or answer with, solve your task or not. So the fill so the prompt template is usually go with some kind of instructions and the the actual input

196

00:44:29.900 --> 00:44:54.030

Jisun An: so in this case. This is once again example of the sentiment analysis, and our template would be so given an input overall, it was like output. So we are giving a template or the instructions, what would be our expected input and what would be our expected output. And then. So input now, there can be many different inputs, then then the actual. So you will.

197

00:44:54.380 --> 00:45:23.389

Jisun An: I guess this is an example for the actual programming. So if we are doing the prompting in the with the code. Then you will kind of see this kind of thing. So the template basically will have the space variable for where that will be replaced based on the inputs and the output. So if we have this kind of template, then if we have this, input then our actual prompt will be, I love this movie overall. It was jets, and that will be the our expected output.

198

00:45:24.590 --> 00:45:53.689

Jisun An: And so, and if you are actually implemented with the code, then this would be the particular template that are commonly used nowadays, and this is the chat template that are introduced by the open AI, and many of the models are now adapting the same format. So this messages has the different roles, and here roles are the system and the users in assistance.

199

00:45:53.690 --> 00:46:17.270

Jisun An: and the system message is now. You can imagine it as testing can be used for instructions. But the system messages are the something that what you you want the model to? Not you want the model, not forgetting these particular messages. So the system messages can define the expected behavior of the

200

00:46:17.628 --> 00:46:31.959

Jisun An: That for your own task. So, including instructions or some other information that you want, edit them not to forget can be go into the system messages and the user messages is the like. The general input by the users.

201

00:46:31.960 --> 00:46:37.189

Jisun An: and the assistant message will be now the output of the system, meaning that the answers from the

202

00:46:37.750 --> 00:46:39.079

Jisun An: the model itself.

203

00:46:39.529 --> 00:46:50.719

Jisun An: So I mean. So the next Tuesday we will do the lab with for the prompting. So you will see more in in the Python code. But this will be, some general format that you will. You will see

204

00:46:51.490 --> 00:46:55.890

Jisun An: the and the the reason that they have this system.

205

00:46:56.570 --> 00:47:00.887

Jisun An: as like separation of the system. And the user is because,

206

00:47:01.814 --> 00:47:22.050

Jisun An: because how they actually trained is is also closely related to how they've been trained. So we we talked about this instruction tuned instruction tuning, and when they were doing instruction tuning, they considered this instruction as a part of the system. And then they had this user input as a

207

00:47:22.593 --> 00:47:31.003

Jisun An: like different inputs. So so they wanted to distinguish between what is the instruction and the what is the input part? And I think that

208

00:47:32.300 --> 00:47:52.579

Jisun An: that's just kind of leading to. Why, we have this chat chat template at the moment. So the llama and the arpa has also, I mean, behind the scene they all have a different kind of exact tokens that used for distinguishing these things, but more or less, they have a similar ideas of the system, messages and user and the assistance.

209

00:47:55.600 --> 00:48:14.440

Jisun An: So now so once we have a particular template for a particular task then given this prompt, then you will predict the answers, and then. Now, to to get this answer you can use like different inference algorithm and like, the the models that they don't.

210

00:48:14.530 --> 00:48:35.290

Jisun An: the libraries that we will use they will define. You can use like a different inference algorithm that we just discussed. And you, you can also adjust the levels of how much you want to encourage in the creativity, or how much you want to make it more strict. You can also adjust those as a parameters.

211

00:48:35.650 --> 00:49:03.180

Jisun An: and then, after you get the prediction. Sometimes you will see that I mean those models that are large. They are they? The answer, like they. They are giving the expected answer more aligning with your prompts, or, as you expected, but as the model is smaller and smaller, you will see that the answers are not as expected, and they will be very varying. So you may need to do some post processing for these answers.

212

00:49:04.481 --> 00:49:30.528

Jisun An: So I mean you can, taking it as a taking the output as it is. But then you may also need to do some formatting, or you may need to select only the parts that you want to use, or you may need to map those output to a particular actions that you would like to. So in terms of the output formatting I mean, you can kind of do a bit of using asking to create a markdown

213

00:49:30.960 --> 00:49:44.009

Jisun An: style, or you can give us a code depending on like your usage. So if you get the output and they use it directly for your other code, then you can ask it to get it like Jason format. So that'll be one option

214

00:49:44.960 --> 00:50:02.180

Jisun An: or so even though you are expected to do like sentiment analysis. But then they could like generate overall. It was like movie that was simply fantastic. Then maybe you need to extract some information that needed for your own task.

215

00:50:02.495 --> 00:50:24.890

Jisun An: So if if this was like a classification test, then you may need to identify some of the keywords. If it is the regressions, and you may identify numbers, or if it's the code. Then you may look for those like triple tactics to identify the code blocks. So this kind of like post processing may be needed, depending on what kind of task that you are doing with the prompting

216

00:50:30.430 --> 00:50:45.240

Jisun An: so, and and lastly, like. You may also need to do some kind of output mapping as well. So so these were all the things that was done very actively like few years ago, and now the bigger models are tend to give like

217

00:50:45.410 --> 00:51:04.860

Jisun An: very aligned features, but sometimes, you may need it for a smaller model. So depending on the task that that you are kind of heading. So so keep in mind that even though the output that you received is not what you expected, there should be some way that you can transform this to the way that you want.

218

00:51:06.680 --> 00:51:16.960

Jisun An: yeah. So that was the some page of framework. I I think this is straightforward to understand. But any any question in just in case.

219

00:51:17.760 --> 00:51:25.530

Jisun An: okay, so the generation future and prompting, which is also called as an in context learning.

220

00:51:26.600 --> 00:51:43.789

Jisun An: So general sharp prompting is. Now we just give a prompt that defines the task that you want to solve, and you just give that prompt without any other information. And then you just get the prediction. So that's the general sharp prompting

221

00:51:44.350 --> 00:51:55.195

Jisun An: so essentially in the chat. Gpt, if you just like chat with it. Then that's like the general prompting, because you are just chatting, and there's no much other

222

00:51:56.301 --> 00:51:58.809

Jisun An: information that you are providing.

223

00:52:00.080 --> 00:52:03.669

Jisun An: But then so that's called as a 0 shot, and

224

00:52:03.700 --> 00:52:26.359

Jisun An: now the few. So once again, these are these are all the terms that are defined. Given the assumption that you are trying to solve a particular task. So these are terms that are task oriented. So just make keep that in your mind, and the future prompting is now instead of the on top of the instruction. You are giving a few examples together. So now.

225

00:52:26.360 --> 00:52:39.409

Jisun An: rather than just saying that, please classify movie review as positive or negative, you are also giving some examples or the demonstration. So given this, input the output is this, given this, input the output is this.

226

00:52:39.610 --> 00:52:44.930

Jisun An: So this code is a few top prompting because you are giving few examples.

227

00:52:45.770 --> 00:53:05.490

Jisun An: and and interestingly, these were even not an option for the language modeling up until Gpt. I think, Gpt. 3. So even before that even this kind of few shot or 0 shot was even not possible, they were never giving the correct answer. But then up to like

228

00:53:05.590 --> 00:53:23.430

Jisun An: starting from Gpt-three, they realized that now these large models can do this kind of future prompting. So why this future prompting works is there are many research about trying to understand why future actually works.

229

00:53:24.703 --> 00:53:31.390

Jisun An: The most recent theory is that so so in a simpler way, you can think as

230

00:53:31.550 --> 00:53:54.820

Jisun An: as they are adding these inputs, they will activate different parts of the attention mechanism. And that kind of helps to basically find the better. Next token. That's the most intuitive idea. But recently they found that these models are implicitly set up a small model, and then they may do actually do

231

00:53:54.950 --> 00:54:01.346

Jisun An: run some kind of like fine-tuning process. Within this, in context, learning

232

00:54:02.000 --> 00:54:28.150

Jisun An: So there was some evidence that that the trained, pre-trained models are able to implicitly create a model and train the model with this small example. I don't exactly know how that actually works, but there are some theory or hypothesis that this may be the case, and they showed a proof with the mathematical examples.

233

00:54:28.150 --> 00:54:44.351

Jisun An: But we still don't know whether the same approach was applicable for the text generation, but that was something that people something that's like actively going on. They really try to understand why the in-context learning is working, but no definite answers. Not yet.

234

00:54:44.930 --> 00:54:52.089

Jisun An: But at a high level. The yeah, this somehow started to work.

235

00:54:52.810 --> 00:54:59.410

Jisun An: Yet once again, still few years back they found that

236

00:55:00.430 --> 00:55:16.499

Jisun An: The language models are very sensitive to these in context examples. So the left one was a very interesting work about. So now they have different demonstration examples, and then they change the order of the examples.

237

00:55:16.500 --> 00:55:41.270

Jisun An: and then the Y-axis, the X-axis. Here, on the left figure, the X-axis is the model parameter, so as is getting right is the bigger model. The Y-axis is the variance of the accuracies by changing the order of the examples. So what it means is that so? Simply by changing the number, the orders of these examples.

238

00:55:41.270 --> 00:55:46.219

Jisun An: the performance were just significantly change it. So that's the what they observed.

239

00:55:46.260 --> 00:56:07.349

Jisun An: and on the right side. They also checked the impacts of the label balance or the label coverage. So the label balance means that. So if you are giving 5 of the 4 demonstration example with the binary classification. So they were kind of checking when they giving one like one

240

00:56:07.570 --> 00:56:26.775

Jisun An: positive example, and 2 and 3 and 4 positive example. So 4 positive example means that the demonstration didn't have any negative examples. So they looked at the impacts of the label balance. And they found that having 4 positive, 3 or 4 positive examples actually lead to higher accuracy.

241

00:56:27.610 --> 00:56:43.669

Jisun An: and and this was not the case. So these are the 2 different data sets. So the the results are not consistent. But what it kind of tells is that basically the results can be sensitive to the label balance. So whether you are. If you have 2 labels, if you are giving balance.

242

00:56:43.670 --> 00:57:05.689

Jisun An: 2, 2 kind of examples or 3 to one. All this work there was no consistency patterns. I think so. They wanted to alarm. These models are very sensitive to these changes also, the label coverage means that. So now these are examples that having more than 2 labels like 4 and 6 labels, and also the results were kind of changing, depending on, like, what? How many labels you are

243

00:57:06.060 --> 00:57:09.520

Jisun An: presenting within your demonstrating examples.

244

00:57:11.150 --> 00:57:33.689

Jisun An: And and so this was something that I mean, this kind of makes sense. Maybe they are sensitive to it. But there were some very interesting counterintuitive research as well. And here now, what they were do is what they did was so they are still giving the examples. But in the example. Now they started to give the wrong answers.

245

00:57:33.750 --> 00:57:57.949

Jisun An: So here they added the so here the 1st figure, the 100% correct, means that all the demonstration examples were having the correct answers. And then 75% correct means that 25% of the examples were having like noisy examples. And then they were kind of increasing these values. And they found that for some cases the performance differences were basically marginal.

246

00:57:57.950 --> 00:58:04.259

Jisun An: So you can see that, like the the blue bar is the No demos, so that definitely, there's impacts of the future

247

00:58:04.260 --> 00:58:25.211

Jisun An: learning. But then, like even though you are not giving the correct label. The the performance were still quite, quite good in compare with when you gave, like all the correct answers, so this now indicate that what they learn is they learn, like the types of the task that they are

248

00:58:25.890 --> 00:58:38.970

Jisun An: expected to have and types of length of the maybe the tests and the topical distribution of the tests, so they may learn those things. So it was not eventually. So

249

00:58:39.010 --> 00:58:52.170

Jisun An: let the model know what the test is in other events is more important than giving them the right answer. So, without even giving these answers, they were still able to get the correct answers, meaning that

250

00:58:52.170 --> 00:59:09.459

Jisun An: the next token prediction will also depending on your input the finer question. So your demonstration can be anything but just giving them some information about the task was helping for edit them to know, like the guide Edm, to which token to generates

251

00:59:10.873 --> 00:59:39.986

Jisun An: and also sometimes the the the letter. The the below figure shows that the compare between the gold values and the random values. And but here they wanted to see, like how many demonstration is needed. But then they found that if you are giving too many demonstrations, and now it starts to hurt the accuracy. That was the some of the findings and some of the efforts trying to see what learns from the in context learning?

252

00:59:40.610 --> 00:59:44.459

Jisun An: and I think that's the something quite interesting.

253

00:59:45.736 --> 00:59:47.290

Jisun An: Any question.

254

00:59:49.350 --> 01:00:12.104

Jisun An: I mean, it sounds very obvious, but I mean because someone did a new search. Now it sounds obvious, but I think when the general shot and the few shot firstly came out. I think everything was kind of very black boxes, so people tried to understand from different perspective. Still, no, no clear answer for it. But yeah, that's the

255

01:00:14.090 --> 01:00:20.199

Jisun An: But at least that give us some opportunity to work with different things. So I think that sounds interesting.

256

01:00:20.840 --> 01:00:43.019

Jisun An: So now I will go on to some basics of the prompts engineering. So I mean, so prom engineering in a way, you can do this manually. And there's a this really good guideline. So you can visit this website a page and then see some general ideas of what is the good instructions how to

257

01:00:43.320 --> 01:00:56.670

Jisun An: write a good instruction for your own task. I guess most important thing is like, be really precise. So, as you be more precise, your answer will be far better

258

01:00:57.040 --> 01:01:00.220

Jisun An: generating, and something like

259

01:01:00.830 --> 01:01:11.259

Jisun An: at the moment like they don't complain when you are vague, so if your question is fake, their answer will be also vague, so trying to be as precise as pre possible.

260

01:01:11.879 --> 01:01:30.709

Jisun An: So once again, that's a good example. So, and there are many tips. But I'm just highlighting a few things that I find it important. It may also worth it to pay attention to the prompt format. And once again, this may not happen in the larger model. But this was the done experiment on the llama. 2.

261

01:01:31.380 --> 01:01:49.870

Jisun An: What they found is that so? They were changing the formatting like, little by little. So the original formatting was passage colon space and the text and then answer colon space and text. And now they change it like removing the line, break.

262

01:01:49.870 --> 01:02:12.310

Jisun An: or removing the colon, or changing the lowercase to the capital letter, or like name them, remove the space. So they did this minor modification, and they found that the results were changing. So they suggested that you need to check the format that your trained model was used, and then following them would be helpful for getting the correct or better performance.

263

01:02:12.310 --> 01:02:25.859

Jisun An: But but once again, this may not be a problem for the larger models. These are the issues that are more obvious for those smaller models. But but I found it quite interesting. So just wanted to introduce one

264

01:02:26.594 --> 01:02:39.819

Jisun An: and another thing is, there are different components. Then you can think about when you're writing your prompt, you can consider like a like a persona, meaning that you can say that. What? Who you are? I mean, what? What is the

265

01:02:40.413 --> 01:03:05.220

Jisun An: the role that the Edith should assume? You can also give a clear instruction. You can also give more context. When you talk about that particular instruction, you can also talk about the format. So what is the desired desired outputs you can also mention, like the audience. So who? What? The who should be intended? recipients of these outputs?

266

01:03:05.260 --> 01:03:19.870

Jisun An: And also you can ask for the tone. And also, you can also ask, talk more about the data. So these are the components that may or may not be applicable for your own test. But this would be all good components to think about.

267

01:03:19.870 --> 01:03:30.940

Jisun An: So this was like the one example. So you are an expert in the so these are giving the persona, and the instruction is summarize the key findings of the papers.

268

01:03:30.940 --> 01:03:51.939

Jisun An: But then the the context can give. Okay, what are the points that you want from to for this sub summary, and what should be the format like, including, like the bullet points of the summary. The audience is for the Bg researchers, and the tone should be like professional and clear. So these are some of the factors that you may consider, and

269

01:03:52.110 --> 01:04:01.739

Jisun An: you may do like the different iterations, and see each of the components may or may not be helpful for the task. So this would be basic prompting techniques that you would do

270

01:04:01.800 --> 01:04:24.630

Jisun An: some other factors like there could be hallucinations. So you can also add the prompts that if you don't know, then to say I don't know rather than generating something. You can also identify that to prevent any hallucination, and also the order may also be important once again, this may not be a problem for the bigger models, but

271

01:04:24.630 --> 01:04:42.509

Jisun An: they tend to forget some things in the middle. So the pride there are 2 primary effects and the recency effects so they could focus on the start and the end. But then, so the middle ones are can be forgotten easily. So that may be also something that to think about while you are designing your own prompt.

272

01:04:42.750 --> 01:04:44.833

Jisun An: But these are very generic

273

01:04:45.520 --> 01:04:52.349

Jisun An: ideas, and but but hopefully this will be like a good starting point to think about how to design your prompt

274

01:04:52.937 --> 01:05:02.279

Jisun An: what I like to more on is these more advanced prompting method or or reasoning in the Lm, so

275

01:05:03.140 --> 01:05:27.179

Jisun An: now, since the 0 shot and the few shots people realize that can actually do something more than they expected. And now they like very actively look at and measure their ability of reasoning. So the reasoning is using evidence and logic to arrive at conclusion and make judgment. There could be like the different systems, and they try to find out which systems are using and how to

276

01:05:27.180 --> 01:05:45.849

Jisun An: how to make them to. So if you think about next token prediction, they usually going with the system one. But they are now trying to how to make edit them, to go with the system. 2. So think really about it, and then answer before as generating their outcomes.

277

01:05:47.450 --> 01:06:08.241

Jisun An: so I'm sorry that I'm just doing this lecturing, but I have a bit of contents today. So I'm kind of watching it. But please stop me whenever you have. You need any clarification. So the the 1st and most interesting attempts in this reasoning was this chain of thoughts prompting? And here the idea is,

278

01:06:08.600 --> 01:06:26.609

Jisun An: so rather than the rather so. So here they are, asking, like some simple logic question. Logic has 5 tennis ball. He buys 2 more tens of tennis ball. Each can has 3 tennis ball. How many tennis balls does he have now.

279

01:06:26.620 --> 01:06:31.579

Jisun An: So if you are just asking this question, this would be like Jerusha prompting right?

280

01:06:32.150 --> 01:06:36.513

Jisun An: And you can also have, like a future prompting. But still

281

01:06:37.010 --> 01:07:06.040

Jisun An: the model were model, tend to not be really able to answer these questions. And then here the idea was they were. They used the few shots prompting, but in this demonstration they showed what logic or what reasoning they should follow to solve this, to answer to this question. So they actually wrote the answer as chain of thought. So what thought process should be there? So instead of the answer is 11.

282

01:07:06.040 --> 01:07:29.120

Jisun An: As a like few shot example, they say, rotor started with 5 boards, 2 cans of 3 tennis boards. Each is 6 tennis ball, 5 plus 6 equal 11. The answer is 11. So they actually give a this thought process to the language model, and they call this as a chain of thoughts cot.

283

01:07:29.480 --> 01:07:52.429

Jisun An: and surprisingly giving this as an example, helped the Lm. A lot to improve their reasoning ability, and this has been used like not only this kind of logic applications, but prior to the application. This has been to the increase in the performance in the Llms.

284

01:07:52.856 --> 01:08:16.100

Jisun An: Once again. Why, this is working we still don't know. But I think the idea is similar to the other, like the few shot promptings basically. Now the rm is so huge and there are so many different paths that you can take, and by guiding this model to which path to take that generally leads to better performance or answering correct answers.

285

01:08:17.498 --> 01:08:30.169

Jisun An: Then the this one still used the few shots. So they they gave the demonstration of what is the thinking process. And the next paper found that it's even working on

286

01:08:30.210 --> 01:08:50.669

Jisun An: even working without the demonstration, so rather than providing the actual thinking process. The the d. Here is the general shot, unsupervised chain of thought. So instead of like giving all these details of the thinking process, they simply added less things step by step.

287

01:08:51.100 --> 01:09:08.120

Jisun An: And then this simple phrase led Lm. To reason better, and this was as good as the cot. So once again of this moment, I think people are all saying this as a magic, and still no one knows clearly why this is working. But

288

01:09:08.370 --> 01:09:17.290

Jisun An: this was one of the 2 of the breakthrough that that we've seen in 2022

289

01:09:17.750 --> 01:09:43.829

Jisun An: and and now we probably already quite familiar with all these things. But if you are just chatting with the Chat Gpt, they were already doing this. Now, like, if you are asking some questions. They will already do this thinking processes, because, since they found this cot, now they are instruction tuned, based on these cots, so they extensively generated the instruction data with this thinking process, and they retrain their

290

01:09:43.830 --> 01:09:54.660

Jisun An: model. So now the models Gpt. 4 0. And the higher versions are very, very good at thinking process. And these are all coming from this basic ideas.

291

01:09:56.206 --> 01:10:20.870

Jisun An: And but then, so the actual paper did actually 2 steps. So how you actually implement this in a prompting. So they firstly asked to do like things step by step, and they will give the answer. And now they give these answers again, and then they added, the prompt There, for the answer is, and then they got the actual answer. So there were actually 2 step prompting to get done this orange provide chain of thought.

292

01:10:20.880 --> 01:10:25.920

Jisun An: So this would be how you can implement this if you want to implement it.

293

01:10:26.570 --> 01:10:46.680

Jisun An: So we we talked a little bit about the emergent ability of the Rnm. And this regioning was one of those emergence ability. So what it means is that when the model is very small. Basically, they were not able to solve any of these reasoning problems. But then, as they are getting larger and at some point, they

294

01:10:46.690 --> 01:11:04.359

Jisun An: show this jump in the performance. And they call this as an emergent ability. And then, later, there's another paper. And this is actually the artifact artifacts that, coming from the way that we measure things. And if you think about

295

01:11:04.360 --> 01:11:28.999

Jisun An: so eventually, we are measuring this as a like classification problem. And then, if you are doing the reasoning to get the reasoning actually correct, then you need to like at least like 30 40 tokens to be right to get the right reasoning. So if you change the metrics rather than not measuring the accuracy of the task. But if you use some continuous measure to

296

01:11:29.490 --> 01:11:38.010

Jisun An: measure the performance of the language model, then these emergent ability plot just disappear. So so there are also.

297

01:11:38.260 --> 01:11:51.000

Jisun An: People are saying this as an emergency ability. But there are also other argues that just the artifacts of the measures rather than the actual emergent ability. So that's also the interesting aspect you may think.

298

01:11:53.720 --> 01:11:54.930

Jisun An: So

299

01:12:02.180 --> 01:12:06.949

Jisun An: these are relatively simple concepts. So

300

01:12:07.510 --> 01:12:32.429

Jisun An: you you'd better to kind of rather than if something is very complex, then you need to. It's better to split them into a smaller chunks of the task, and then solve as like answer individually, that would be actually better to solve a complex problem, because that's the what is called as a problem decomposition. So instead of like, write a blog about the risk of the AI. If you can give more

301

01:12:32.430 --> 01:12:42.589

Jisun An: details of like, what is the exact, the sections that you requires and what you want, then these generally perform much better the in in terms of the generation output.

302

01:12:43.240 --> 01:13:10.990

Jisun An: So this problem. The competition framework is separating in normally, like the 2 steps like, you need to be able to generate subproblems. And also you need to solve each of these sub problems. So in this example, we are already giving these like 6 different sections. But then this model may also be able to come up with this 6 different sub problems. Right? So this leads to most prompting was try to automate this idea of

303

01:13:11.462 --> 01:13:19.500

Jisun An: the complex problem, like sub breakdown into smaller problem and then tackle from start from the 1st

304

01:13:19.871 --> 01:13:26.020

Jisun An: questions to the last. So that was the basic idea of this leads to most prompting.

305

01:13:26.780 --> 01:13:55.539

Jisun An: So once again they use the like demonstration. So these are all future. So these are purely done within the prompting. So so they gave. For this particular question, they try to come up with what is the like? The first, st what what one need to solve? 1st to answer this particular and then in this stage 2, they answer like each sub questions, and then ideally, they will come up with the finer answers.

306

01:13:56.000 --> 01:14:02.040

Jisun An: So that's what yet extra prompting method that a polite

307

01:14:02.854 --> 01:14:16.230

Jisun An: and I have few more, but I will continue from here on next Tuesday, and then together we will do the lab for the prompting but any any custom up to here I will probably repeat, like from here.

308

01:14:16.950 --> 01:14:41.050

Jisun An: Okay, in 1 min. Just one thing. I haven't published it, and but we have. The practical assignment will be released tonight, and you can read it through. But here the idea is so we are trying to impersonate a teammate using Rrm, so you will be paired up so you will need to find another. Also. 2

309

01:14:41.610 --> 01:14:49.439

Jisun An: students will be your team, and you, too, will create an data set by answering a questions.

310

01:14:50.120 --> 01:14:59.019

Jisun An: And then your task is to impersonate your teammate. So I mean, once again, we we are not expecting you to get this Sota

311

01:14:59.020 --> 01:15:10.760

Jisun An: performance, but I think this will be a good practice for you to create the data and applying some of the methods and also define the evaluation metrics. So so you need to think about

312

01:15:10.760 --> 01:15:33.319

Jisun An: the best way to create a data set, and the best evaluation metric that you can do and what method to pick. So I will post it tonight. You can read it, and I will give you more details. Probably next Tuesday. We have so much thing to do next Tuesday, but but I was trying to do give all the details, but that will be something. So I will publish it tonight. But yeah.

313

01:15:33.570 --> 01:15:37.310

Jisun An: oh, okay, that will be awful today.

314

01:15:38.030 --> 01:15:39.810

Jisun An: Okay.

315

01:15:40.590 --> 01:15:48.499

Jisun An: thank you. Have a good weekend we'll see you next Tuesday. If you have any questions, I will be here for a couple more minutes. So

316

01:15:48.720 --> 01:15:50.469

Jisun An: yeah, thank you.