**Name:** Interview-06-20220719

**Created On:** 8/9/2022 4:49:34 PM

**Created By:** WJ

**Modified On:** 8/10/2022 5:27:27 PM

**Modified By:** WJ

**Size:** 12 KB

Speaker 1:

Okay. So first I'd like to ask you some questions about your background. So the background questions can be answered briefly so that we can save time for the other questions. So first, could you briefly describe your role in your team?

Speaker 2:

All right. It's just only for current role or... Okay. So now I'm a second year PhD student in Cardiff University and I am working on an NLP research and my research focus is on, I would say a kind of application side, not a theoretical aspect. I would say most of my projects are related to language model fine tuning. And I am the one who maintain the code, training a model, monitoring everything together. And that's, I would say, my role.1

Speaker 1:

Okay. Can you tell me a recent time when you used a pre-trained model from an external model hub?

Speaker 2:

Oh, like this morning.

Speaker 1:

Oh, okay. So what did you do with the model?

Speaker 2:

So the most frequent usage in my situation is that, let's say in every single project I have a task where we use language model fine tuning, but we want to compare a different language model, like a BERT model, GPT model. And the fine tuning itself is fine, but after the fine tuning, the model checkpoint is like five or six gigabyte. And we do not want to keep every model on the server or on the workstation because it consume a lot of storage. Instead I push the model to the model hub, in this case I use the HuggingFace. Once the model training has been done, I push every model to the model hub under my account. And once it's done, I can just delete all the checkpoint in the local, in the server.

And once I uploaded the model to the model hub, I will run evaluation or anything on, let's say my local computer, because the evaluation, we don't use GPUs for higher computation, for the evaluation or just the qualitative analysis of prediction. So for me, it's not like to share the model to the community, it's more like, just a way to save the storage and a better way to organize the models. Because HuggingFace, I don't think they offer a maximum number of models you can push. You can push everything, whatever you want. So yeah, that's the main use case. This morning I finished a few model training, so I pushed the model to the model hub and tomorrow now on we start evaluation on the model.2

Speaker 1:

Okay. So for your model, do you fine tune other models from the model hub or you like fine tune-

Speaker 2:

Yeah.

Speaker 1:

Okay.

Speaker 2:

Yes, exactly.

Speaker 1:

So do you retrain any model from model hub?

Speaker 2:

Yeah. So in my project, I haven't never trained the language model from scratch. I usually take the pre-trained language model on the model hub and further fine tune that on the task we were interested in. But in other projects we have fine tuned... No, no, we have trained language model from scratch and pushed the model to the model hub. So yeah, we have both use cases.3

Speaker 1:

Okay. So have you ever used any model as like a backbone?

Speaker 2:

Yeah, pretty much. Like the language model and... Well, actually a language model and the multimodal model, like CLIP or those kind of vision language multimodal model.4

Speaker 1:

Okay. So the next set of questions is related to how you select a pre-trained model. So we are trying to understand the process that software engineers follow as they decide which pre-trained model to re-use in their project. So can you think about one time when you choose a pre-trained model from a model hub? So how did you choose which model you should use? Can you summarize your decision making process?

Speaker 2:

Yes. I think this is pretty difficult and it depends on the project, but in most cases, like 60 or 70%, in most case we have a task and there's a leaderboard on the task. Let's say we have a document classification and there's a leaderboard which shows what model is current state of the art, the best model in that task. And usually we start from that model. And if leaderboard said that the best model in document classification at this moment is GPT-2, then we take GPT-2 as a first trial. And then from that best model, we will also compare that against other architecture. Like the GPT-2, is the recurrent language model. And we were interested in other architecture, like sequence to sequence, which is like T5 BERT model, while we can try a must language model like RoBERTa and [inaudible 00:06:42]

So we always start from the best model. And then we switch to other architectures. That's the most common way to select model. Or, if you have a specific domain, like we have a product which is about the NLP on Twitter. So it's not normal text, its tweet from Twitter. So in such a case, we expect that the normal language model wouldn't work that well on the tweet. So on the model hub we have RoBERTa trained on tweets specifically. So in such a case, we use the Twitter version of RoBERTa instead of the normal one, because we know that would work well with our data set because ours are tweet as well. So yeah, that's the, I think majority concern when we select the model.5

Speaker 1:

Okay. So where do you usually select the pre-trained models? Do you select that the models from a model hub or you select the models from GitHub more?

Speaker 2:

Ah, right. That's a good question. So we first explore on the model hub, but sometime the best model is not on the model hub. And in such a case, we try to spend a bit time whether we can use the model released on their own GitHub, but usually it's very hard because it's not in the same format, it's not a universal format. They have their own script to use the language model to fine tune anything. So honestly, a hundred percent of my work based on the model hub. If the best model is not on the model hub, we just skip to use the model. Because we don't think it's worth to spend much time on trying to understand the script from the GitHub.6

Speaker 1:

Okay. So while selecting the model, do you care more about the model's performance than the architecture?

Speaker 2:

Yeah, I think... I would say architecture is a bit more important than performance because, in some case we have a very specific language model that works in your task the best, but because language model has a very specific architecture, to fine tune this language model requires some custom engineering. And in such a case, we usually avoid that because that takes a bit more time. So we just rely on, within the basic architecture, the best model. And we do not choose even a weird architecture, even that achieved quite high performance.7

Speaker 1:

Okay. So do you think the pre-trained model available in the model hubs can accurately describe their behavior?

Speaker 2:

Yeah. Well, in the model hub... So the README on the model, the model card, I don't think that's very comprehensive. You have to read the paper and only the model card, you wouldn't be able to understand the full picture of the model.8

Speaker 1:

Okay. So to what extent do you think the discrepancies of performance metrics can affect your decision making? So here for the discrepancy, sometimes the actual performance can be different from the claimed performance in the model cards.

Speaker 2:

Yeah, I think... So again, I think it's pretty related to the previous question, whether we prioritize the performance or the architecture. So even if the model says that performs pretty well across the task, if the architecture is quite different from the standard language model, we would [inaudible 00:11:47] It's hard to, interpret the result because, if this architecture is very similar to what we have been seeing in the literature, we would know that it works that well, it's kind of expected, but [inaudible 00:12:35] So yeah, I think the metric lets [inaudible 00:13:06]

Speaker 1:

I think the connection is unstable.

Speaker 2:

Yeah sorry, I was dropped. Can you hear me now?

Speaker 1:

Yeah, yeah, yeah. I think maybe you can stop your video so that it can be more frequent.

Speaker 2:

Okay.

Speaker 1:

Yeah.

Speaker 2:

All right.

Speaker 1:

So I didn't hear your answer so clear, so can you like briefly repeat that again? I'm sorry about that.

Speaker 2:

Yeah sure. So if I understand correctly that the question was whether the performance matters at the model selection phase.

Speaker 1:

Yes.

Speaker 2:

Yeah. So in brief, I would say that matters at some point, but it's not the biggest concern usually. And I would say that the biggest concern would be the popularity of the model.

Speaker 1:

Okay. So I think here for the performance metrics, we mean not only the accuracy, it's also like some other metrics which can be used to measure the performances, like latency or something.

Speaker 2:

Ah, yeah.

Speaker 1:

So will the discrepancy of this also affect your decision making?

Speaker 2:

Yeah. So latency or the model size is also a biggest factor. Because these days language models, parameter size grows every day and on a usual computer, those a trillion, a billion model wouldn't fit. Actually even in the paper we compare up to a few hundred millions of model and we do not touch anything with a model over a billions of parameters. So that's definitely a huge concern. And probably that's more important than the performance, because even if the performance is higher, if the model is like a hundred trillion, then no one will be able to use that in the end.9

Speaker 1:

Yeah. Okay. So to what extent do you think the robustness of the models can affect your decision?

Speaker 2:

Yeah. So in general, I think the community wouldn't concern about the robustness. And so all the language models... In the ordinary machine learning model, you would run model training with a different random seed, and you quantify the robustness of the model, but in language model because the one single run takes a few days or a few weeks, we have no way to run with a different random seed or different configuration. We usually just do one shot and regard that as the model. So yeah, we do care about the robustness, but in reality, we have no way to... There's no measure in terms of the robustness when selecting the model.10

Speaker 1:

Okay. So to what extent do you think the explainability of the model can affect your decision?

Speaker 2:

I think it depends on whether you are in industry or a research community. In industry, usually you do not care about the interpretability, all that you want to do is the high performance model. So probably you would choose some model, you don't care about the interpretability of model. But in research community on the other hand, it still depends on the research topic, but usually it's preferable to have model which can give us a [inaudible 00:18:13] interpretation of the model prediction. So yeah, it depends on whether the application is in the research or industry.11

Speaker 1:

Okay. The next question is, how frequently do you retrain the model?

Speaker 2:

Retrain the model means the fine tuning on a task?

Speaker 1:

Here I mean, retrain the model from scratch or...

Speaker 2:

So from scratch we usually do not do that. But again, as I mentioned in the, in the RA, in the talk, we have a project where we train language model on a text from a specific domain. In that project, we train language model from scratch on that domain specific [inaudible 00:19:20] but that's the only predict we have where we train from scratch. And usually we just fine tune existing language model.

Speaker 1:

Okay. So how frequently do you fine tune the models?

Speaker 2:

Fine tune the model, I would say 80 or 90% of my project are fine tuning the language model. So it's pretty frequent.

Speaker 1:

So for the other 10 to 20%, do you just reuse the model directly?

Speaker 2:

Yeah. In that project, we need the representation from the vanilla language model without any fine tuning. That's because of the project's topic and we use the [inaudible 00:20:09] language model.12

Speaker 1:

Okay. So do you think the lack of trainability or fine-tunability is a problem when we're using a pre-trained model?

Speaker 2:

Yeah. I think the lack of re-trainability and the fine-tunability comes from the size of the language model. Because as I mentioned, fine tuning requires a lot of GBs, especially when we try to function a model with billions of parameter. Naturally it's not feasible, but that means we can only do experiments with smaller model. If there's any way to fine tune a billions of language model, we would try because we want to see how it goes when we scale up the language model.13

Speaker 1:

Okay.

Speaker 2:

I do think that we should have a way to fine tune, the fine-tunability is very important.

Speaker 1:

Okay. So what other challenges do you face when selecting a pre-trained model from a model hub?

Speaker 2:

Can you repeat it again?

Speaker 1:

So is there any other challenges you met before when you select a model from a model hub?

Speaker 2:

Other things when I choose to the model from model hub. So the size and the performance, interpretability. Yeah, I would say that's all.14

Speaker 1:

Okay. Let me share my screen again. So the next set of questions is about the Deep Learning software attributes. So here we are trying to find what information is useful to engineers who reuse pre-trained models. And let me share my screen. Can you see my screen?

Speaker 2:

Yeah.

Speaker 1:

So here are three traditional attributes, which are defined by npm's, which is used for the JavaScript packages. So you can take a look at some definitions here, like the first paragraph for each attributes here and let me know when you are ready.

Speaker 2:

Okay. Yeah, should be fine.

Speaker 1:

So what do you think would best help your team select a pre-trained model from the model hub?

Speaker 2:

I would say all matter, but if I have to choose one from these, then the popularity would be the most important factor.

Speaker 1:

Okay.

Speaker 2:

Since, as I mentioned in the previous question, even if the model report that in some task it achieved a pretty high accuracy, still, it is the case in the community that the BERT model is released in 2018 and now it's already four years passed from the BERT release, and we have bunch of language model that achieves better result than BERT. But the original BERT model is still being used in many application. And I think that's because it has a popularity, and community improved the usability of the model. And you feel very safe as long as you are using the BERT model. And that reflects the importance of the popularity, I would say.

Speaker 1:

Okay. So for quality and maintenance, which one do you care more about?

Speaker 2:

I think maintenance should be a bit more prioritized. Because sometimes it happens that even if the quality is good, because that has not been tested by any other people, or if you are a first person to test the library or the pre-trained language model, you would definitely have some issues. While if there's maintenance, when I say maintenance it means that, that language model has been used by various people. And even if there's no README, people are reporting the issue and the person who released the language model has trying to spend some time on fix that issue and those communication, the active communications would be more important than the quality.15

Speaker 1:

Okay. Oh I think we are running out of time. And if you have to leave, let me know.

Speaker 2:

Yeah it's okay.

Speaker 1:

Okay. Okay. Thank you. So then here are three different specific attributes we define here, including the provenance, reproducibility and portability. So I will start from provenance, for each attribute here I will ask several questions. So for provenance, here we defined as a measure of model lineage or traceability. So here are some examples, like whether the documentation contains the link to the paper, a GitHub page or dataset architecture, something like that. So can you think about time when you met provenance problems when using the pre-trained model before,

Speaker 2:

Right. I think this happens a lot before the HuggingFace have existence, as everyone released the model on their own GitHub have repository. And usually they lack of... Well I think they at least had link to the paper, but the usage, how to use the language model, how to reproduce the result on the paper, those are usually lacking. But once we upload the model on the HuggingFace model hub, then I see less issues in terms of the problems. Because usually they have a link to the paper, and I think the biggest contribution is that you put the model on the HuggingFace model hub, and you don't have to write how to use the model because it's on the HuggingFace model hub. You know how to use the model. I think that's the reason why we don't see much provenance issues. At least the model is on the HuggingFace repos.16

Speaker 1:

Okay. So what do you think would be useful to know beforehand in order to solve the provenance problems except for the device here?

Speaker 2:

So it's very useful if you have the... First of all, you should have the performance table. Usually if you wrote a paper about proposing new language model, you have the accuracy on the downstream tasks, like document classification 90%, and other task 80%. Those tables should be in the README in both of the GitHub and the model hub. And what else? You put the data set on the list, but there's two version of the data set. One is the data set for evaluation, the task. Even if you say document classification, what data set used to perform document classification. And that's the one task data set, while people report this task data set frequently on the README but the dataset they used to train the language model, the pre-trained dataset that's usually has omitted on the README. So I would appreciate if they have a section where they explain what kind of dataset the language model has been trained on, like what percentage of the topics like news 80% and website 10% or something like that.17

Speaker 1:

Okay. Okay. Then we're discuss about the reproducibility. So here we define it as a ability of a different practitioner to produce the same accuracy and trending or evaluation time from a pre-trained model as defined in the paper, source code or the groups. So can you think about the time where you met any reproducibility problems when using the pre-trained model before?

Speaker 2:

Yeah, I do have this issues quite a lot. And there is little situation where I don't have this reproducibility issue. The thing is because they list language model itself on the model hub, but they do not release the fine tune version of the language model. Like one paper argue that language model is pretty good and they shows that it achieves quite high performance in downstream task, but they only release the language model itself and do not release the fine tune language model. Even though they say that we release the script to fine tune language model, usually that do not provide you the performance reported on the paper because they are very sensitive, again, related to robustness issue. The language model is so sensitive to the hyper parameter. And even if you change random seed the result would be changed dramatically. And so yeah, we do have this reproducibility issue quite often.18

Speaker 1:

Okay. So what do you think would be useful to know beforehand in order to solve these problems?

Speaker 2:

So I think people should push the fine tuned version of the model on the model hub separately. That's solve quite a lot of [inaudible 00:32:49] and also we should have a unified way to fine tune the language model to achieve the same performance reported on the paper. Somehow people are very... I don't know, it's really hard to reproduce the result on the paper, especially the fine tuning process. So we should come up with some much more wiser way to organize those fine tuning process.19

Speaker 1:

Okay. So the last attribute here is portability. So here what you find as is with which an engineer can take a pre-trained model and reuse it in another environment or software project. So can you think about the time where you met any portability problems before?

Speaker 2:

Yeah. So first of all, that's the model issues. So usually a language model has a different model size, like from a small one to the extra large model. So it's better that we have a choice to change the model size, depends on the system we want to deploy the model on, that's a good thing. I think as long as the model is on the model... The HuggingFace model have this, portability is not that issue, just the model size matters.20

Speaker 1:

Okay. So is there any other things you think would be useful to know beforehand for the portability problems?

Speaker 2:

Yeah, I would say it's better to know the size of the memory footprint or GPU footprint that the model would consume when on the server, because usually they report the model size only like 100, 10,000 millions of the model, but they usually do not report how much GPU you would need for inference or how much GPU you use or how much CPU memory you need to fine tune this language model. Those hardware specification are not usually on the README and we have to know that by trying by ourself to try to put it on the server and see how much it costs. That's something we should know beforehand so that we can choose appropriate server beforehand.21

Speaker 1:

Okay. One follow up question here. So except for the GPU memory, do you think the GPU version also matters here?

Speaker 2:

I don't think so. Because I remember that when I used TensorFlow, it was an issue. Sometimes the GPU version and driver version are not appropriate and they raised a inconsistency, incomparability issues, errors, but I switched PyTorch a few years ago and from that I haven't been using TensorFlow at all, but the PyTorch I don't see much issues about the incompatibility. So yeah, that's the reason why I think the version wouldn't affect a lot.

Speaker 1:

Okay. So the last question for this part is, except for these three attributes, do you think there are any other attributes that would be helpful for the pre trained model reusing?

Speaker 2:

Well, let's see. So I think one aspect that we care is the cost, the money we spend on the server, and that would be something related to the portability, but we have multiple options to build the pipeline and each of them has a benefit and the drawbacks and it's actually pretty hard to estimate the cost. For example, I have a project where I should have a demo where we host the language model to perform the prediction. And we have two candidates of the design of the pipeline. One is that we use the GCP server where we clone the model from the HuggingFace model hub, and on that server, we run an API server which use that model. And we have a front end that called the API, which served by the GCP. And in this situation, we have to pay pretty much on the GCP solver. And that was quite quite expensive because to host language model, we need pretty much a lot of memory in the CPU for the server.

But then we switched that from that GCP based pipeline to the HuggingFace solution where, I don't know if you have used that, but I think last year HuggingFace launched the HuggingFace inference API that hosts language model, instead of using GCP. Basically we can query to the HuggingFace inference API and get the prediction from the model. So we have no need to host the language model by ourselves. That's not free subscription, but in the end that's 10 times or 20 times cheaper than using GCP to host the language model. So this kind of calculation is really hard. And I wish we could have some sort of website or service that gives you a ball park cost estimation, depends on what service you would use to create this pipeline design.22

Speaker 1:

Okay. We have the last four questions about the trustworthiness of the pre-trained models. So we are trying to understand how the pre-trained model shortcomings affect engineer's ability to rely on and reuse them. So first, which aspects of the pre-trained model do you assume are trustworthy?

Speaker 2:

Trustworthy. Can you go back to the slide where you explain the trustworthy?

Speaker 1:

Yeah. Well, I don't think I have the slides for the trustworthy. So for the trustworthy, here are some examples, the model architecture is different from what the documentation or what the model card said, or the performance which is incorrect. Something like this.

Speaker 2:

Okay. This shouldn't be happening, from the perspective of academic community, this shouldn't be happening, and I don't think we had that performance discrepancy in terms of the reported one and actual one. So as a software engineer, machine learning engineer, I think they can just trust what reported on the paper. Because I don't think there's so much cases where the false performance are reported.23

Speaker 1:

Okay. So have you found any discrepancy between the [inaudible 00:42:30] pre-trained model and the downloaded version in terms of the accuracy, latency and also architecture?

Speaker 2:

No, I don't think I have seen anything like that on the HuggingFace model hub.24

Speaker 1:

Okay. So if the discrepancy exists, to what extent do you think this exist discrepancies are acceptable?

Speaker 2:

Like a few percent drops or let's say the performance accuracy reported was 80% and the actual one was 50%, that's not acceptable. From 80% to 78%, a few point drop in terms of the percentage, that would be acceptable, for me personally. Because after all the language model are very sensitive to the hyperparameters. Maybe it's having a different hyperparameter for inference.2625

Speaker 1:

So can you clarify, a few percent, is it like less than 3% or 5% or something else?

Speaker 2:

I would say 3% might be accepted, but 5% is too much. 5% is no.

Speaker 1:

Okay. The last question here is, do you think the discrepancies of the models will have significant impacts?

Speaker 2:

You mean the discrepancy-

Speaker 1:

Of like accuracy, latency or architecture will have significant impact on the engineers or the products?

Speaker 2:

Ah, right. Yeah I think so. Especially the latency would affect a lot because the pipeline or the system diagram, the system pipeline would be built on top of the model capacity or model latency. And if the model latency is too low than the reported one, then they might have to change the pipeline or service to host a language model. And so yeah, that's quite problematic, I would say.27

Speaker 1:

Okay. Thank you very much. So that's all for my questions, I will stop recording now.

Speaker 2:

Okay.

Speaker 1:

Okay.

**Annotations**

1 Role:

- PhD student

- NLP application

- language model fine-tuning

- maintain the code, train the models, monitor everything together

2 Reuse case:

- language model fine tuning

- After fine-tuning, each ckp is like 5-6 GB

- Instead of keeping every model on server or the workstation, I push the models to the model hub. Once it's done, I can delete all the checkpoints.

- After uploading the models, I will run evaluation or anything on local computer.

- Does not use GPUs for the evaluation or just the qualitative analysis of prediction.

- Not like share the model to the community, but more like a way to storage and a better way to organize the models

- Reason: HF does not have a limitation on the max number of models you can push.

3 Retrain: both use cases

- language models: never

- other tasks: yes

Fine-tune: Yes

4 Backbone?

- yes

- a language model and the multimodal model, like CLIP or those kind of vision language multimodal model

5 Decision making:

- Summary: pretty difficult, depend on the project.

- most cases (60-70%), there is a leaderboard on the task.

- usually start from the state-of-the-art (best) model from the task leaderboard as a first trial.

- Then switch to other architectures

- For specific tasks, like "NLP on Twitter", which is not normal text, we use the Twitter version instead of the normal one.

6 Where:

- spend a bit time whether we can use the models on their own GitHub

- usually it's very hard because it's not in the \*same format\*

- They have their own script to use the language model to fine tune anything

- All my work based on the model hub

- if the best model is not on the model hub, we just skip them

- We don't think it's worth to spend much time on trying to understand the script from the GitHub models

7 Model performance or architecture?

- Architecture!

- in some case, we have a very specific language model that works in your task the best.

- To fine-tune this language model requres some custom engineering. In this case, we usually avoid that becuase it taks a bit more time.

- We just rely on the best model within the basic architecture.

- We don't choose a weird architecture, even that achieved quite high performance.

8 Behavior?

- The model card is not very comprehensive.

- You have to read the paper

9 Discrepancy:

- relevant to the previous question -> whether we prioritize the performance or the architecture

- It matters at some point, but it's not the biggest concern usually.

- The biigest concern: popularity

- Latency or the model size is also a biggest factor.

- parameter size grows every day -> a huge concern

- probably more important than the perormance

10 Robustness:

- I think the community wouldn't concern about the robustness

- No way to run model training with a different random seed in language models.

- We usutally just do one shot.

- We do care about the robustness, but in reality we have no way to do it. And there is no measure in terms of it.

11 Explainability:

- Depends on whether you are in industry or a research community

- Industry: usually you do not care. All you want is the high performance model.

- Research: on the other hand. Still depend on the research topic. Usually it's preferable to have model which give us the interpretation of the model prediction.

12 Retrain: usually don't do that

Fine-tune: 80-90%

Reuse without any fine-tuning: 10-20%

- Example: vanilla language model

- because of the project's topic

13 Finetunability/retrainability:

- the lack of it comes from the size of the language model

- I do think we should have a way to fine tune. The fine-tunability is very important.

14 Other challenges:

- size, performance, interpretability. That's all.

15 Traditional attributes: popularity > maintenance > quality

- Populartiy: we have many models that achieves better result than BERT. But original BERT is still being used in many applications

- You feel safe as long as you are using the BERT model.

- This reflects the importance of the popularity.

- Maintenance:

- Even if the quality is good, the model has not been tested by any other people. You would definitely have some issues.

- When I say maintenance, it means that the model has been used by various people. Even if there is no README, people are reporting the issue and the active communication would be more important than the quality.

16 Provenance issue:

- This happens a lot before HF exists

- The GitHub models usually lack the usage of the model (e.g. how to use, how to reproduce)

- Less issues in terms of these problems in HF

- You know how to use the model from HF model hub.

- We don't see many provenance issues, at least for the models from HF repos.

17 Provenance help:

- performance table

- accuracy on the downstream tasks

- should be in the README in both of the GitHub and the model hub

- Put the dataset on the list

- dataset for evaluation

- dataset for training: what percentage of the topics.

18 Reproducibility issue:

- little situation where I don't have the reproducibility issue.

- They list language model itself on the model hub, but they do not release the fine-tune version of the language model.

- Usually they release the script to fine tune language model, but they do not provide you the performanc reported on the paper becuase they are sensitive, related to the robustness issue.

- The language model is so sensitive to the hyper parameter.

- If you change the random seed, the result would be changed dramatically.

- Yeah, we do have this reproducibility issue quite often.

19 Reproducibility help:

- People should push the fine tuned version of the models on the model hub separately.

- We shold have a unified way to fine tune the language model to achieve the same performance reported on the paper

- It's really hard to reproduce the result on the paper, especially the fine tuning process.

- We should come up with some much more wiser way to organize those fine tuning process.

20 Portability issue:

- usually a language model has a different model size.

- It's better that we have a choice to change the model size, depends on thesystem we want to deploy the model on.

- As long as the mode is on HF, portability is not that issue, just the model size matters.

21 Portability help:

- it's better to know the size of the memory footprint or GPU footprint that the model would consume when on the server.

- usually they report the model size, but they usually do not report how much GPU you would need for inference or how much GPU (memory) you need to fine tune this model. Those hardware specification are not usually on the README and we have to know that by trying by ourselves and see how much it costs.

- That's something we shuold know beforehand so that we can choose appropriate server beforehand.-

- GPU version: the incompatibility is an issue when using TensorFlow, but not for Pytorch. So I think theversion wouldn't affect a lot.

22 Other attributes:

- Cost: the money we spend on the server. That would be sth related to the portability

- We have multiple options to build the pipeline and each of them has a benefit and the drawbacks and it's actually pretty hard to estimate the cost

- I wish we could have some sort of website or service that gives you a ball part cost estimation, depends on what service you would use to create this pipeline design.

23 Trustworthy aspects:

- From the perspective of academic community

- This shouldn't be happening. I don't think that performance idscrepancy in terms of the reported one and actual one.

- I don't think there are so much cases where the false performance are reported.

24 Discrepancy: Not

25 Acceptable discrepancy:

- a few point drop in terms of the percentage.

- 5% is too much. 3% might be accepted.

26 Discrepancy reason:

- Maybe it's having a different hyperparameter fo rinference

27 Significant impacts?

- Yes, especially latency would affect a lot

- the system pipeline would be built on top of the model capacity or model latency.

- If the model latency is too low than the reported one, then they might have to change the pipeline or service to host the model.

- That's quite problematic.