**Name:** Interview-05-20220715

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

**Created By:** WJ

**Modified On:** 8/9/2022 7:07:08 PM

**Modified By:** WJ

**Size:** 9 KB

Speaker 1:

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

Speaker 2:

Okay. So now I'm basically trying to be a founder of a startup project.

Speaker 1:

Yeah. Okay.

Speaker 2:

But before, like a few months back, I was a director for a professional services team at Salesforce.1

Speaker 1:

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

Speaker 2:

The reasons why I'm using external models from a model hub, right?

Speaker 1:

A recent time.

Speaker 2:

Oh, a recent time.

Speaker 1:

Yeah.

Speaker 2:

I'm using it every day because I'm trying to build a product. So I use HuggingFace to train various, more of a natural processing models. Right? Like bird. And then I also do GPT-2 or other models.2

Speaker 1:

Okay.

Speaker 2:

Yeah.

Speaker 1:

So, do you retrain or fine tune any models?

Speaker 2:

Yeah, I do fine tune.3

Speaker 1:

Okay. Have you used any models as a backbone?

Speaker 2:

As a backbone? What, in which sense, you mean as a backbone?

Speaker 1:

Use, for pre-trained ways as a backbone, and then as some extended layers on some different tasks?

Speaker 2:

Oh, no, no, no. I'm using them as they are.4

Speaker 1:

Okay.

Speaker 2:

Yeah.

Speaker 1:

Okay. Now we're move on to the second part, which is about the model selection. We are trying to understand the process that software engineers follow as they decide, which pre-trained model to reuse in their projects. Can you think about the last time when you choose a pre-trained model from a model hub? How did you choose it? Can you summarize your decision making process?

Speaker 2:

Yeah, so basically what I look for is some level of domain adaptation if I can find it. So for example, my project is related to cooking. So I try to find some model, let's imagine I have to classify tokens in a sentence. Right? I would like to find a model that has some domain adaptation, so I can spare that step and proceed to the fine tuning to get the results I need. Right? Otherwise, I would have to do that domain adaptation first to boost the performance, and that's kind of the main reason. Aside, obviously, from whatever task I looking for.5

Speaker 1:

Okay. So where do you usually select the pre-trained models, whether it's from a model hub or is from the open source projects in GitHub?

Speaker 2:

Mostly I do select them from the model hub in HuggingFace. But many times I also, if I'm coming from reading a research paper, I try to find the sources or the resources from that paper and then I move on to see if I have find something similar on HuggingFace.6

Speaker 1:

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

Speaker 2:

I care. The main thing I care about is the inference performance. Yes.7

Speaker 1:

Okay.

Speaker 2:

Yeah.

Speaker 1:

So do you think the pre-trained models available in the model registries or model hub can accurately describe their behavior?

Speaker 2:

Yes. Many of them, especially because of the naming conventions that they have and those things, you know what to expect when you're using them.

Speaker 1:

Yeah.

Speaker 2:

Some are not really welled documented8, but...

Speaker 1:

Okay. To what extent do you think the discrepancies of the performance metrics can affect your decision making? Here, for the discrepancies, I mean, the actual performance of the model can be different from the claim performance in the model cards?

Speaker 2:

Yeah. Hmm. That's an interesting question. I honestly give them a try more than... I don't read the metrics, but I try to give them a try and use my gut feeling to understand if I will, after fine tuning, I would get to where I want.9

Speaker 1:

Okay.

Speaker 2:

Yeah.

Speaker 1:

So to what extent do you think the robustness of the model can affect your decision?

Speaker 2:

I think since, I may be a little biased because I kind of always been using transformer based models. Right. And then. I know many of the architectures that are using transformers, and then I know what to expect on that. So it's just for me trying to put it on a box there and say, okay, this is a similar to BERT, or this is similar to, I don't know, T5 or whatever. Right? It's like, this is a decoder, an encoder, things of this sort. Right?10

Speaker 1:

Okay.

Speaker 2:

So, yeah.

Speaker 1:

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

Speaker 2:

I think if the model is better documented, and then I would say it has some practical examples of what I'm looking for, I'm probably going to go for it. Look, at least I found, let's say, when I was experimenting with chatbots, I found a few chatbots that I got to use because in the page of the hub, there was a really good working example of it.11

Speaker 1:

Okay. So how frequently do you retrain the models?

Speaker 2:

Very frequently. Yeah. Whenever I'm doing like the research part or the development part, I would maybe run fine tunings every day or maybe several times a day.

Speaker 1:12

Okay. So do you think the lack of accountability or fine tune ability is a problem when you're using a pre-trained model?

Speaker 2:

Yes. I can give you an example. I'm also a user of GPT-3, open AI, GPT-3.

Speaker 1:

Yeah.

Speaker 2:

And then fine tuning. That model itself is pretty impressive as a toy, let's say, but, they do provide some fine tuning capabilities. But then, given that once you fine tune, then you have to... Firstly, fine tuning is more expensive, but then you need to wait until the model loads. Then that makes it not really usable at least for production applications or for real time production applications. So definitely not being able to fine tune, it's an issue.13

Speaker 1:

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

Speaker 2:

That's basically it. Right? The only other thing is, sometimes I'm not sure if I can rely on the search functionality, especially on HuggingFace. Search doesn't always take me to models. I find, let's say with a Google search... Google, it's actually more effective than the inside search, which probably shouldn't be the case.14

Speaker 1:

Yeah. Okay. So the next set of questions is about the depending software attributes. So we would like to learn about what sort of information is useful to engineer who use a pre-trained models. So here I'd like to show you the... Here are the three traditional attributes, which are defined by the MPM, which is used for the JavaScript packages.

Speaker 2:

Yeah.

Speaker 1:

So can you briefly read the first paragraph of each attribute here and let me know when you are ready?

Speaker 2:

Okay. So quality. Okay. Popularity. Mm-hmm (affirmative). Maintenance. Okay.

Speaker 1:

Okay. So the question is, what do you think would best help your team select a pre-trained model from model registries?

Speaker 2:

Okay. If I had to put this quality, this attributes in order, I would say popularity, it's the first thing I look for. Right? I look for popularity. And then if the model is quite popular, usually that goes side by side with maintenance. So that means that the owners do good job at keeping up to date and people excited about it. And then also popularity might be an indication of the quality of the model. That means it fits for many person's use case.15

Speaker 1:

Okay. All right. Here we propose some new, different specific attributes. And I have several questions about each of these attributes and we'll start from the provenance, here. So for the provenance, we defined as a measure of the model lineage or the traceability. Some examples are like, whether the model card contains a link to the paper, whether it contains a GitHub page or an external website. So can you think about a time when you met provenance problems when using the pre-trained model before?

Speaker 2:

I typically get to the models from the papers and not the other way around. Right? If I'm looking for a specific research, and then I just get to the papers, read them, and then if the information is useful, I go to the resources. Yeah.

Speaker 1:

So if you are looking at a specific model in, let's say, the HuggingFace, have you met any provenance problems here? Or you always start from the paper?

Speaker 2:

If I'm looking for information, that type of traceability, I would start from the paper. The only thing, sometimes I would use this attribute is to make sure that the model I find is exactly the one I'm looking for. So if it has any link to the paper, then it gives me a confirmation that that's actually the model that I'm looking for.16

Speaker 1:

Okay.

Speaker 2:

Yeah.

Speaker 1:

So is there any other factors which would be useful to know beforehand in order to solve these problems?

Speaker 2:

Not sure. I think with just the proper links in the documentation, I'm not sure if there's much more to that.

Speaker 1:

Okay.

Speaker 2:

Yeah. Because, if it's linked to the... I would imagine that as a user, I would feel maybe I would have the question if the materials were reproducing the same way as the paper. Right? But then that's maybe a documentation problem. Because at the end of the day, you don't have a way to know it unless they document all the weights.

Speaker 1:

Yeah.

Speaker 2:

But you don't have a way to know it for sure.17

Speaker 1:

Okay. Okay. Then we'll discuss about the reproducibility. So here we define it as basically of preparing practitioners produce the same accuracy and the training or evaluation time from a pre-trained model as defined in the paper source code or the group.

Speaker 2:

Yeah.

Speaker 1:

Some examples are the hyper parameter, the training configuration, the hardware, and also whether the model registry contain any demos. Can you think about a time where you made any reproducibility problems while using the pre-trained model before?

Speaker 2:

Well, I think that, at least in HuggingFace, this is a really good thing to have an inference API with the model. So then you can really quickly test what you're looking for and reproduce it there on the spot most of the time. And then for the cases where the usage of the model is a little bit more complex, I have always found that the creators provided, say, a script or something that you can run and get the results. So I'm not sure if I had a particular issue with that. I don't remember an instance where I've tried to reproduce something and I couldn't.

Sometimes I might have... So I've tried this model that generates cooking recipes and it was from a paper, and then I had a little bit of trouble trying to find some hyper parameters to get the results that the paper said. But then it was a matter of reading and getting a little bit more familiar with the hyper parameters. But it was just one rare occasion.18

Speaker 1:

Okay. So is there any other thing you think could be useful to know beforehand in order to solve the reproducibility problems?

Speaker 2:

No. I think it's just also a documentation thing. I think if the creator documents well, and then maybe, I don't know, add a notebook or something of the sort, that shouldn't be that big of a problem.19

Speaker 1:

Okay. So the last attribute here is portability. Here we define it as the ease waste, which an engineer can take a pre-trained model and reuse it in another environment or a software project. So can you think about time when you met any portability problems?

Speaker 2:

Oh, yeah. This is the main thing for me. Since running all of these models is kind of expensive, sometimes I need to run the models on the client side, and sometimes I need to run them on the server side and make it strike a balance with that. Right?

Speaker 1:

Yeah.

Speaker 2:

So, many of these models are, especially on the HuggingFace site, they're on PyTorch or on TensorFlow. And what I want is maybe ONNX. Because with ONNX, there's a variety of run times, and then it's an open format that can potentially be, I don't know, run on an iPhone or whatever. Right?

Speaker 1:

Yeah.

Speaker 2:

So I always have to build and port, and then probably build even tools around it because I feel that ONNX maybe, it's a great concept, but it needs some maturity. Definitely I feel that I've seen myself building tools to port models to different platforms.20

Speaker 1:

Okay. I think I have one follow up question here.

Speaker 2:

Yeah.

Speaker 1:

Do you think if the model registries, if they provide the compressed models, the quantized models, which could be helpful here?

Speaker 2:

I think it would be helpful. Yeah. Similar to, I'm not sure if you're familiar with, probably you are, with CDN and the JavaScript libraries, right?

Speaker 1:

Yeah.

Speaker 2:

You have the versions that are not minimized and the version that are minimized. Right?

Speaker 1:

Yeah.

Speaker 2:

And then it's good to have both because if you're developing and then you need to troubleshoot, it's good to be able to go back to the source code. But then if you're on a production deployment, you want them to be fast. You don't want to trace it. Right? So for sure, I think that having a quantized version of the model, that I can grab with my continuous deployment and say, "Okay, this is production. I'm going to the quantized model. And I don't have to worry about building it." Currently, that's my problem. I need to build in and test and then having the quantized model. But then on the flip side, I also see the problem where if you fine tune, then you cannot have a quantized model on the hub. Right? There should be maybe a kind of a trigger that once you push a model, then the quantizes it for you. Which is, I don't know. I would have to see it.21

Speaker 1:

Okay. Okay. So is there any other thing you think could be useful to know beforehand in order to solve the portability problems?

Speaker 2:

No, not really. So I think that maybe it would be interesting to see if that's fully supported by ONNX or other format that's easily portable to other platforms.22

Speaker 1:

Okay. So the last question for this part is, except for these three attributes, do you think there are any other attributes could be helpful for the pre-trained model you're using?

Speaker 2:

No, I think what we have done, at least it covers everything that I would find useful when I'm working with these things. Let me see. I'm trying to remember cases where I'm working with models.

Speaker 1:

Yeah,

Speaker 2:

No, aside from that, if you want to smooth or make it easier for people who are new to this, maybe it would be interesting to see the recommended scenarios or the scenarios that those models are designed for. So it softens the curve for new people, let's say. Or they don't try to do things that the model is not meant for. And then something like nowadays, if you have not read papers or books about it, you don't really know what the model is intended for.

Speaker 1:

Okay.

Speaker 2:

Or maybe there's tasks that could be achieved on different architectures, but then the way that you achieve those task tasks is completely different. So I think I feel that might be something that could be23 useful.

Speaker 1:

Okay. Yeah. So the last set of question here is about the pre-trained model trustworthiness. So we are trying to understand how pre-trained model shortcomings can affect the engineer's ability to rely on or use them. So the first question here is which aspect of the pre-trained model do you assume are trustworthy?

Speaker 2:

When it comes from a user, as opposed to, let's say, a company or something of the sort. So maybe if I go to the hub and see just an individual name, I know that I would have to take care. Mostly I would have to just make a copy of the model and take care of it. Right? But if it comes from a Microsoft or if it comes... Or any company, right? Or HuggingFace, then I know it would probably be maintained. And then I would consider not having to have a copy of that or using it in my processes without having to worry that model will not be there anymore or change.24

Speaker 1:

Okay. Okay. So have you found any discrepancies between the clamp pre-trained models and the downloading version in terms of the accuracy, latency, and architecture?

Speaker 2:

No.25

Speaker 1:

Okay.

Speaker 2:

I haven't.

Speaker 1:

To what extent do you think, if there exists discrepancies, to what extent do you think these discrepancies are acceptable?

Speaker 2:

Well, I think that maybe, or I would imagine for the way that this is built, that the only way that you would have discrepancies is when you have different defaults, right?

Speaker 1:

Yeah.

Speaker 2:

Different type of parameters or anything of the sort. Because otherwise you're just downloading waste, so it should be the same file.

Speaker 1:

Yeah.

Speaker 2:

Right? So I think that, as long as those are, I would say, not too big or well-documented... Let's say that maybe in the inference API, it says, "Okay, I'm using this hyper parameters. Right?" But I guess that they, both places, should use the same defaults to avoid this confusion.26

Speaker 1:

Okay. So the last question here is, do you think the discrepancies will have significant impacts on your reuse?

Speaker 2:

No. I think it is the same as whenever you're doing a regular framework, right? Sometimes you download it and it doesn't work as you expect, then you just go to the documentation, try to adapt it to your needs. And it might be a problem if the documentation is not sufficient or something of the sort.27

Speaker 1:

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

Speaker 2:

Yeah, sure.

**Annotations**

1 Role: founder, previously a derector of a professional secices team at Salesforce.

2 Training various models. More NLP models.

3 Fine-tune models

4 He does not use the models as backbones. Just reuse them.

5 Decision making process: domain adaptation? (can boost the performance) -> fine-tuning

6 Where: mostly from HuggingFace. Github only from research papers.

7 Inference performance > Model architecture

8 accurately describe behavior?

- Many of them: yes

- Some of them: not really well documented

9 Discrepancies:

- I don't read the metrics

- try to give them a try first, I will use my gut felling to understand if I will get to where I want.

10 Robustness: I am a little biased. Mainly use transformer based models. Compare the models to popular models like BERT

11 Explainability: better documented -> should have some practical examples -> look for them and use them (example: chatbots)

12 Retrain: maybe not

Fine-tune: everyday or maybe several times a day.

13 Retrainability/Fine-tunability:

- Example: GPT-3

- Problem: expensive, not usable at least for production applications/real time production applications

- Conclusion: definitely not being able to fine-tune

14 Other challenges:

- untrustworthy search functionality, especially on HF

- doesn't always take me to models

15 Traditional attributes:

- popularity > maintenance and quality

- popularity usually goes side by side with maintenance

- popularity indicates the quality

16 Provenance issues:

- mainly start from the papers.

- use this attribute to make sure the model is exactly I am looking for

- link to the paper -> confirmation

17 Provenance help:

- if the materials were reproducing the same way as the paper

- hard to know unless they document all the weights.

- Don't have a way to know it

18 Reproducibility issues:

- In HF, inference API is really good -> really quickly test

- a little bit more complex -> a script or something you can run and get the results

- Don't remember an isntance where I've tried to reproduce sth but I couldn't

- Hyper-parameters are hard to find (rare occasion)

19 Reproducibility help:

- documentation

- add a notebook or something of the sort

20 Portability issue: main problem

- Run the model in both client side and server side. Hard to strike a balance.

- Want ONNX. But many models are on PyTorch or TF. ONNX can be run on an iphone or whatever

- Always build tools to port models to different platforms.

- ONNX is a great concept, but it needs some maturity!

21 Portability help:

- Good to have both minimizde and not minimized versions, like CDN and JavaScript libs

- If you are developing, you need to troubleshoot, so you need to go back to sounrce code. But on a production deployment, it's fast and don't want to trace it

- Grab the model with continuous deployment

- Quantized models: good for CD and fin-tuning. I would like to see a trigger that automatedly quanitze a model for you once you push it.

22 Portabiltiy help:

- whether the model fully support ONNX or other format that's easily portable to other platforms

23 Other attributes:

- it covers everything I would find useful

- recommended scenarios/what are designed for -> soften the curve for new users

- Tasks that could be achieved on different architectures

24 Trustworthy:

- company/individual

25 Discrepancies?- No.

26 Acceptable?

- only reason: different defaults

- both documentation and inference API should use the same defaults to avoid the confusion.

27 Significant impacts?

- No

- same as whenever you are doing a regular framework

- try to adapt it to your needs

- Might be a problem if the documentation is not sufficient or something of the sort.