Speaker 1:

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

Speaker 2:

I'm a machine learning engineer at a cyber security startup. My role mostly involves around working partly with ML ops, where we just deploy the models where we are figuring out the stability, doing the monitoring and everything. And the other half is to develop solutions which are going to be specific to cyber security because we don't have any DJ models, so any idea to go about from. So that is like mixture of both. Before this, I was working in NLP. So I think that is where you saw HuggingFace model hub.1

Speaker 1:

Yeah.

Speaker 2:

Yeah.

Speaker 1:

So can you tell me a recent time when you used a pre-trained model from an external model hub, like HuggingFace?

Speaker 2:

Well, around a year back. Yeah, so it's been a while.2

Speaker 1:

Okay. So do you remember if you have met any challenges when you were using these models?

Speaker 2:

Made any what?

Speaker 1:

Do you remember if you have met any challenges when you are reusing these pre-trained models?

Speaker 2:

Most of the times, yes, because the purpose that we use pre-trained models was to get ... so we didn't use classifiers and everything. We used just the language model in our previous workplace and then modified it on top of that. So most of the times, it was just used to just take a pre-trained model. They have ... at least HuggingFace model hub at that time had an option to download the entire model. And then just use that as a base model and train a second downstream task.3

Speaker 1:

Okay. So then the next set of questions is related to how you select pre-trained neural networks. So we are trying to understand the process that software engineers follow as they decide which pre-trained neural network to reuse in their projects. So first, I have some definitions here from some registries. This means we specify the traditional package registries, like npm Pypi, Maven, and for the machinery model registries, we define the HuggingFace, [inaudible 00:02:29] hub and [inaudible 00:02:29] hub as model registries here. So for the attributes here, we mean some attributes like quality maintenance popularity, and for the model attributes, we mean the provenance reproducibility and deployment constraints. For these parts, I will define this later. And so can you think about the last time when you choose pre-trained neural network? So how did you choose it? Can you summarize your decision making process with model?

Speaker 2:

So mostly one thing that is a bit constrained is what language it is based on, given that we worked in a context where multilingual languages were essential. So we generally preferred models which were trained in cross-trained methods, because that way, we'll be able to use it on multiple languages. So I don't remember the model name, but generally, these models were trained for across hundred languages. So then you can just use it for a rest of the downstream task. And now, once I have this bunch of models which are cross trained, like LMs and everything, to select the best one is again, just either go through the papers and see which paper is the latest one, which has the latest research.4

Speaker 2:

I personally use papers with code to get the general idea of what they did behind the scenes to train it. And if I don't have absolutely any idea and I have to select a model, which I ideally wouldn't go by this method, but I would just go by which is the most downloaded one. But this would be an option where it is just, you have to get the model trained quickly and you don't have a solution, so you have to do trial and error, and then you just rank it by the number of downloads.5

Speaker 1:

Okay. So do you have any preference between the pre-trained models from a model hub or is models from GitHub?

Speaker 2:

I prefer HuggingFace model hub because you can directly integrate with inference API. And that was really helpful because it reduced the inference time completely. So you just had to write a rest call to the API and that way, it was easier than actually creating a model, deploying it on your own, then maintaining it and checking the latency every time. But yeah, so that is the preference.6

Speaker 1:

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

Speaker 2:

Mostly performance. At least when it comes to the purpose that we used it for, architecture is secondary, as long as the results are there. Because as I mentioned, we again, take the model and do further processing to fit it for the downstream task. So the model is not itself the most crucial part.7

Speaker 1:

Okay. So do you think the pre-trained neural networks which are available in the model registries accurately describe their behavior like in the documentations?

Speaker 2:

I don't know, actually. So I've never tried to go and retest it or reproduce the results. So ideally, I would trust the model because ... I don't see the incentive of models not behaving the way the incentive behave. So yeah, I probably would try to test it on my data set, whether this fits for my work, but I'm not sure I have considered whether it is accurate to what they have described. It might be a good fit or bad fit for the research or for the purpose, but not might be ... I have not tried to validate their own results.8

Speaker 1:

Okay. So to what extent do you think the discrepancies of the performance metrics, for example, the actual performance, are different from the claim performance in the documentation can affect your decision making?

Speaker 2:

It would actually make a huge difference because if ... For example, if it was a model that is used for question answering, for example, if it does not have a high [inaudible 00:06:48] score, then it is no use for me because I don't want to give inaccurate answers. So in that way, other metrics do matter, but then again, it is based on ... The use case itself might be different and the model might not be the best fit for my use case.9

Speaker 1:

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

Speaker 2:

It would be highly important. I would probably prefer a model that is much more robust.10

Speaker 1:

Okay. So could you explain that in detail, like, why that can affect your decision?

Speaker 2:

One is because if you're deploying a model and if you are doing the whole decision making process, we have already spent three to four months, or at least a huge amount of time making that decision. So lets just say that it will just be a lot of research work to be done after it is deployed. And once it's deployed, you don't want to keep changing the base architecture or the base model. You just want to be able to update your performance by the newer entities. So the assumption has always been that all the effort to select the model has to be put in before. A model needs to be ... one is that it needs to have good performance as well as it needs to be robust. And also, it can't be slow. So I think that is the third thing that is important.11

Speaker 2:

Other things would still be slightly ... we would compromise on, but the latency is not something that we can compromise on. So to give a little context, what I worked at previously was a chat automation company. So it's very annoying if you have a model and if it is doing ... so one of the tasks was to detect names and if a person is saying, "My name is this," and it takes proper one second for the bot to reply, then we just have too many unhappy customers.12

Speaker 1:

Okay. So next question is, to what extent do you think the explainability of the models can affect your decision?

Speaker 2:

I would say it doesn't affect at all. I probably have never bothered with it. And I don't think in purposes of NLP, at least, most of the people ... not NLP, but chat automation, explainability is not a huge factor. It is used for debugging once you are sure of a system that works. But the assumption is that these systems are buggy, these systems ... we have put in other guard rails later on. So yeah. Explainability is not something at least I have even tried to attempt to look at afterwards.13

Speaker 1:

Okay. So next question is, how frequently do you retrain the models?

Speaker 2:

So with models from HuggingFace, I don't think we need to retrain them, right?14

Speaker 1:

Yeah. Oh, do you think the lack of the fine-tuneability is a problem when using a pre-trained neural network?

Speaker 2:

Oh, very highly. Yes. I think that was the best part when all these zero shot models came in, because you don't have to worry about fine tuning. But before that, it just felt like impossible task. If you're not able to fine-tune the model, then you can't use it. Then you have to, again, just train the whole model on your own. So yeah, lack of fine-tuning is a huge issue. And to not be able to do it on fly, like just not take the model that is there and the repository and then just make a command for inference APIs and the data and fine-tune it. Like, you don't have an instance of a model that is specific to yours, a folk model, probably.15

Speaker 1:

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

Speaker 2:

I would say one issue is obviously security and privacy concerns, which I ... because these are not hosted on our own VPC, the problem is that the data is being sent to another network that is unreliable. We can't send a lot of data. We can't send any data that is, as I mentioned, name data, because it's ... So inference API is out of the question. When you use just the model directly from our HuggingFace hub, you still have to do that part of be able to interact with it on your own. Like, have a piece of code, build a docker file which downloads around 3 GB of model, which is a huge docker file. So yeah, these are couple of issues, the security aspect of it, about sending the data either to a hosted environment, which is not in your control. And if it is in your control, then it's the same problem that you're just, every time you're trying to install, you have to install it and you are making calls to HuggingFace, then it's installing on your VM and then you have to again, use it. So setup time is huge.16

Speaker 1:

Okay. So the next set of questions is related to the [inaudible 00:12:12] software attributes. So first, I'd like to show some ... this as a traditional attributes and the definitions here are from the npm. So you can take a look at these definitions and specifically the first one or two sentences of each attributes here and let me know when you finish reading them.

Speaker 2:

Okay.

Speaker 1:

So what do you think would best help your teams [inaudible 00:12:52] a pre-trained model from model registries for these three attributes?

Speaker 2:

Maintenance. Highly maintenance, because I post ... even if it is not ML code or even if it is not a model, knowing that it's constantly maintained and doesn't have a lot of open issues as reliable. Second is that if you have an issue and if you know that, for example, if the code is being updated or if you raise a bug, then someone will help you out and it won't just be unanswered, that's highly important because again, you're relying on someone else to get the model. So you want to be able to build that trust factor there.17

Speaker 1:

Okay. So these are the traditional attributes and here, we also propose some different and specific attributes. And we're going to start from the provenance. So we define provenance as a measure of the model lineage or traceability. So for example, the link to the paper, the GitHub page, external websites for the model. So could you tell me about the time when you met provenance problems when using the pre-trained neural networks?

Speaker 2:

I think a lot of them in HuggingFace have readme, but none of them are associated with paper. I've seen very few of them associated with paper. In fact, it is the other way around how I find the models. It's like, you find a paper which is linking it to the model and then you find those 10, 15 models which are linked with paper, but that's it. So that is something that I definitely haven't seen a lot, but I think it also depends. As I mentioned, right now, I work in security so I can see issues of the way I used to use this before, which can be further exploited. But if you have a model which you don't know what is the association or who's creating it or where it is coming from, you're just creating a sense of just a risk.18

Speaker 2:

The model can have a bias that you can't predict. A model can have the kind of security flaws that you won't be able to predict. And because you don't know who created it, you don't know where the trained data was, because none of them have to declare the trained data. They just have to upload them all.19

Speaker 1:

Yeah. So what do you think could be useful to know beforehand in order to solve the provenance problems?

Speaker 2:

Data set on which it was trained, definitely. And is the data set publicly accessible or it's not? And how is the data being updated? Is the model ... does it have an ability to update the data? Of there's a bias, have they updated the data? Is the new data version controlled? Can I trace back to which particular model version is mapped to data set version? [inaudible 00:15:52].20

Speaker 1:

Okay. So next, we are going to talk about the reproducibility. So here, we define the reproducibility as the ability of the [inaudible 00:16:01] practitioner to produce the same accuracy and training or evaluation time from a pre-trained neural network as defined in the paper and source code or the engineering group. So can you tell me about the time when you met any reproducibility problems while using the pre-trained neural networks?

Speaker 2:

A lot? Have you heard of Vampire?

Speaker 1:

What?

Speaker 2:

Vampire?

Speaker 1:

I think I've heard about that. Yeah.

Speaker 2:

The least reproducible ML model out there. I think it was named Vampire because it sucks the blood out of you. But the problem with that is that when you try to train Vampire or something similar model, which are hard to reproduce, again, you're not aware that it is not reproducible beforehand. So you have to spend in the amount of effort and amount of time. And you do have limited time for figuring out which model you have to choose. I had a huge issue using Vampire. Other than that, just using direct models, which ... there used to be this whole ... Everyone used to release models on GitHub releases, where you downloaded the model. And sometimes it was just inaccurately named, like an English model won't be English. It might be some ... a missed thing and then you have to find the new release. And even once the model has downloaded, the whole reproducible ... the script that they gave, sometimes it was broken. Sometimes it is just generally not there. It's like, you have to figure it out yourself. So that's a huge issue.21

Speaker 1:

Okay. So what do you think can be useful to know beforehand in order to solve the reproducibility problems?

Speaker 2:

I don't know. A badge saying is it reproducible? But also, just a way to run it online before downloading it. Have you seen binder? The way you have a Jupyter Notebook and it automatically runs in its own. So if there's a binder available with your model and I can just test it. If there is an API or if there's just some sort of interface where I can just see that model is working, whether it'll work for my data, that'll be amazing. And I think you've mentioned even Docker image, right?22

Speaker 1:

Yeah.

Speaker 2:

So if there's a Docker, prebuilt Docker image that someone can just run on it and just make inferences before they can play around with the model, that'll also be very helpful.23

Speaker 1:

So do you mean like automated validation for the validation data set or a notebook demo can be helpful? Like which one?

Speaker 2:

A notebook demo, but with the model inside. Like a notebook demo where ... A Google collab will be slightly difficult because you have to install the model. And if it is a big model, then you are again facing the same problem that you can't reproduce the environment on what it was trained. You don't have the same libraries, but if there is something that is just PIP install model, model is there and then you can run it, or a notebook which already has the environment in which the model will run smoothly. I want to be able to run five cells and see that, "Oh, fine. This is doing what I was expecting it to do. It will work on my data set. Now I can take this and then do the further development."24

Speaker 1:

Okay. So then we'll talk about the probability here. So we define probability as is with which an engineer can take a pre-trained neural network and reuse it in a different environment or software projects. So could you tell me a time when you met the deployment constraints problem or the probability problems while using the pre-trained neural networks?

Speaker 2:

They're huge. That's the first problem. You can't use it in a Docker image. They have latency issues, as I mentioned, that even if the model is perfect for you, if you have to deploy it in a small ... you still have to make sure that it will be able to handle a lot of calls because if I have 2000 calls per second, that's still something that the model has to keep working on and keep giving output. And I can't have latency because of that. So yeah, just the size of it surely is high. And even when you ... inferences are generally, at least till now, as much as they have experience, are not preferred to be done on GPUs. The inference machine is going to be a CPU, which is going to be interacting with others. So yeah, if the model is ... as I mentioned, I like inference API purely because of that, because it's much easier to just make a call on a machine which is much more reliable than do the deployment on my own. So yeah.25

Speaker 1:

So what do you think can be useful to know beforehand to solve the probability problems?

Speaker 2:

I don't know. Generally, the time constraint, the latencies of how much the model takes, whether just a general idea of how long it will ... how it will perform in a production environment. So yeah, I don't know how that will be possible, but yeah.26

Speaker 1:

So do you think the performance in different hardware environment can be helpful?

Speaker 2:

I guess, but ideally, I would say that most of the people are deploying are going to be deploying it on Linux. So maybe not different hardware environments, but just a general ... just an idea of how to work on a default ec2, because everyone is going to be just using ec2 in the end. So how is it doing on ec2? If it is slow on my Mac, I'm fine, as long as I know that it'll perform well on ec2.27

Speaker 1:

Okay. So the last questions for this part is, except for these three different specific attributes, do you think there are any other attributes can be helpful for the pre-trained neural networks?

Speaker 2:

I think one thing I mentioned is bias because if I'm able to figure out what kind of bias it has, because no model has zero bias, but to be able to understand what kind of bias it has, then I can counteract it beforehand. For example, if it is a model that detects names, if I'm able to understand that it is terrible with female names, if it is terrible with particular kind of names, then I can just counteract by that. I can figure other ways out, but we generally don't have any idea about the bias in the model, so that is one thing.28

Speaker 1:

So is there another thing you want to talk about?

Speaker 2:

No, that's it. I can't think of others. Yeah.

Speaker 1:

Okay. So the last set of questions is about pre-trained neural network trustworthiness. So we are trying to understand how the pre-trained neural network's shortcomings affect the engineer's ability to rely on and reuse them. Which aspects of the pre-trained neural network do you assume are trustworthy?

Speaker 2:

One thing I would believe is trustworthy is if it is a big pre-trained neural network by established labs and everything, I would expect it to behave the exact same way as it is written in the paper. Like, I don't think I will try to reproduce it or I don't think I'll try to validate it. That'll be my basic assumption that it ... whatever accuracy is written in the paper is the truth. And then I have to figure my way out from that. So that is there. Like, that is what makes a model trustworthy for me. But again, you take everyone's word for it. I don't think I've thought about trustworthiness particularly while thinking of models.29

Speaker 1:

Okay. So have you found any discrepancies between the [inaudible 00:24:04] pre-trained models and the downloaded version? So in terms of the accuracy, latency and model architecture.

Speaker 2:

Ah, model architecture, no, because it would be very obvious to see if it is a difference because it won't just work with your code. So it'll just break because the rest of the framework is written around it. So I've never faced any model architecture issues. Accuracy, again, never validated it, but there might be some difference, but most of the times, it depends on the data because even if the model is accurate and it is accurate on the data that they gave, the data that most of the people will be using is completely different. I am not going to be using any data that is similar to the huge data that pre-trained model was trained on. So it's just going to be small slice. And even specifically, it's going to be a domain specific slice. So yeah. So accuracy, again, I don't see a point in validating the given accuracy. It might not do well, but again, the assumption is that it's not doing well because our data is very specific.30

Speaker 1:

Okay. So to what extent do you think the discrepancies of the accuracy are acceptable?

Speaker 2:

They're acceptable. Let's just say not huge discrepancies. Like, if it is a classification model and if it says it's performing at 80% accuracy, if it is still 75, 70, it's fine. But anything lower than that, I'll just be like, "Okay, no." And it's 75 on my data. Like, if it is 80% on the data that is supposed to be their data, and if I'm trying to fit it for a different purpose and use it for a different person, I'll be okay with a slightly lower accuracy, but not too low, because then it just changes the whole output. I would rather not have something that says it's ... because in classification or anything else, the problem will be that you don't want to give false positives because false positives in any sense are very annoying. And if the accuracy keeps letting down or the metrics are not as well, then I think slight bit of thing, changes are okay, but nothing beyond 10%. 10% is also huge according to me.31

Speaker 1:

So there's things that discrepancies can ... where have the significant impacts on the software.

Speaker 2:

Yes, but I think the model discrepancies will be taken care of by the tests that are written. So most of the times, during the deployment, we at least write behavioral tests. So if the model's accuracy for some reason is dropped, then the model itself will fail in the next behavioral testing round of phase. So it won't get deployed itself.32

Speaker 1:

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

**Annotations**

1 Subject is a ML Engineer at a cyber security startup.

-Works mostly on OPS or DEV

2 Has not used a PTNN in about a year

3 Useing PTNNs had challenges most of the time.

This subject only used language models as a base model and trained a second model for a downstream task.

4 Due to use case models needed to be multilingual

Cross trained models were preferred.

If multiple models met requirements, the dates on papers were used to determine which one had the latest research.

5 Papers were most helpful when they include code so that subject can understand behind the scene operation.

Worst case, download count was used to rank models in a trail and error list.

6 HuggingFace is preffered

inference APT reduced inference time

7 Performace is more important than Architecture

Architecture is secondary

Due to further processing downstream the model is not the "most crucial part"

8 Subject has never tried to validate claimed results of a PTNN

If model works with subjects data, that is good enough

9 A discrepancy in performance metrics would "make a huge difference"

Different use cases need to take metrics into concideration

10 Robustness of a model is "highly important"

11 Robustness is important because of the time spent selecting adn deploying the model (3-4 months)

3 things are important

1 Good performance

2 Robust

3 Not slow

12 Other that the 3 mentioned above, compromise is an option

Latency can not be compromised on

Example given of a chat bot learnign names too slowly

13 Expliainability does not matter in the decicion making process

Subject has "never bothered with it"

14 Subject does not retrain models from HuggingFace

Possible good quote

15 Lack of fine-tunability is a problem

That is why subject appreciated zero shot models

Finetuneing used to feel like an "impossible task"

If a model can't be finetuned you can't use it

16 Other challenges:

Security and privacy concerns if model is not run on subject owned VPC

Transfering data accross networks (files can be 3GB)

Setup time is "huge"

17 Maintance is most important

Having someone help out with issues is #2 - creates trust

18 Knowing the creators of a model (or reading a paper) might reduce risk

Very few models are currently associated with a paper (~10-15 might be linked to paper)

19 Models may have unpredictable bias or security flaws - knowing who made the model would help

20 Data set used for training would be useful to know

Is it public, controlled, biased, often updated, automatically updated, traceable?

21 Vampire is the least reproducible ML model

If a model is not reproducable subject does not know beforehand - this wastes time and effort

Some models and not named correctly

Sometimes reproducability scripts are broken

22 A badge indicating reproducability or way to run the model prior to download would be helpful to solve reproducibility problems.

23 A pre-built docker image would also be helpful for reproducability

24 A notebook demo would be nice so that a model can be validated to behave as expected with minimal set up.

25 Probabilty problems are huge

Docker image can't be used due to latency

Models need to handle many calls and keep responding

26 The time constraint and latency would be useful to know beforhand to solve probability problems

27 Performance in ec2 is all that matters

28 Understanding bias was not mentioned but important

29 PTNN from estabilished labs are assumed to be trustworthy

Everyone is taken at their word

30 Discrepancy between models and downloaded version?

Arcitecture - never

Accuracy - possbile small discepancy but never measured

31 Small discepancies in accuracy are acceptable

10% + or - threshhold was given

32 Models are tesed prior to deployment so a drop in accuracy should not get deployed