**Name:** Interview-08-20220727

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

**Created By:** WJ

**Modified On:** 8/12/2022 2:41:34 PM

**Modified By:** WJ

**Size:** 15 KB

Speaker 1:

Yeah. So, first I'd like to ask some questions about your background. The background questions can be answered briefly so that we can save time for the other questions.

Speaker 2:

Yes.

Speaker 1:

Could you briefly describe your role in your team?

Speaker 2:

Okay. All my research work has been mainly independent work because I'm actually a [inaudible 00:00:23] who's actually just joining a company. I just graduated this year, in fact. When it comes to research level work, my main role is to actually work on the models themselves. I primarily use [inaudible 00:00:37], and I have been using [inaudible 00:00:39] for a while now to write models of my own. As for my industry level work, I mainly work in analytics at a big tech company, and yeah, that's pretty much it.1

Speaker 1:

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

Speaker 2:

Yeah. I use pre-trained models a lot of times, especially since my work is pretty much 100% in NLP, I tend to use a lot of pre-trained models like BERT, GPT-2, and so on for either fine tuning on my own applications or to just build on top of them. In fact, one of my recent papers, which was accepted at [inaudible 00:01:28], it builds on top of the BERT architecture, and it utilizes new novel pre-training architecture, which uses pre-trained models and enhances them using a better learning step.

Speaker 1:

I'm not sure if I understand correctly. Does that mean you use the pre-trained model and you try to improve the performance of them?

Speaker 2:

Yes. I use them for fine tuning as well as my other, this paper which I was talking about, in which we show that, we are using different pre-training methods or by using a separate, additional pre-training method, you can actually make the model work better. Now, what my work does is it is primarily for code mixed languages, which is essentially a language where you have, let's say, multiple languages mixed in. For example, this can be French and English. But primarily our work is for the Hindi and English language, because I'm from India, and that's the first thing we experimented on. For the English languages, we see that normal pre-trained models like BERT and so on do not really get a good performance on benchmark datasets. So we came up with our own pre-training objective, which actually beats the benchmarks which are already present. In fact, we have a 9% improvement over the current benchmark.2

Speaker 1:

Okay. have you retrained any pre-trained models?

Speaker 2:

Retrained? Thing is we don't really have the resources to completely train a pre-trained model, so the only thing that I have done so far is fine tuning. But when it comes to pre-training from scratch, we have done them on smaller scale datasets. For example, not like the ones that Google uses. We don't have petabytes of data data. But we do do them on small, maybe gigabytes of data here and there for our applications.3

Speaker 1:

Okay. Have you used any model as a backbone and add some extended layers to them?

Speaker 2:

Yes. I have used the main BERT model as a layer in our network, and we have extended them by adding more layers to it. That is actually how the paper introduces our model, where we take the BERT model and we actually pre-train it using a CME's network architecture where we add more layers to each stack and we build upon that architecture for learning embeddings.4

Speaker 1:

Okay. So, we'll move on to the next set of questions, which is related to how you select a pre-trained model. Here we are trying to understand the process that software engineers follow as they decide which pre-trained model to use in their project. Can you think about the last time, why 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. The first thing is always the application which you're trying to use it for, because pre-trained models do not work for every use case. For example, if you were to use a GPT-2 model, it is primarily used for text generation. Whereas if you try to use, let's say, a BERT model, it is mainly used for mass language model. So based on the actual application that you have in mind, you can cut down the options that you have by a huge margin, and once you actually filter through the options that are there, it is mainly based on the language which you're trying to do, and the dataset it could be trained on.

For example, if you're trying to work on, let's say, a language generation type of problem statement, and you are trying to do it for, let's say, a non... If it's a language that models are not generally built on, for example, if you're trying to do for Hindi, then the options that you have get cut down by a huge factor, and you do not really have too many options remaining. In fact, there are only a few options that are there for the Hindi language, one of them being [inaudible 00:05:26].

So, first step would be the application. The next thing is to see if within that application there exists any work or where your domains overlap or where your use cases overlap, and that is how you filter through them, and you pick the model that you think would work best for you.5

Speaker 1:

Okay. So where do you select the pre-trained models from?

Speaker 2:

Mainly it's HuggingFace. I've been using that for about two or more years now.

Speaker 1:

So, is there a reason why you choose HuggingFace?

Speaker 2:

From whatever I've noticed that over the last two years, primarily since transformers have been getting more and more popularity, a lot of developers, including me, tend to upload their pre-trained models on Hugging Face, and it generally has a huge collection of models. The other thing about HuggingFace is that a lot of it is plug and play. So, if you want to quickly test out a model or you want to substitute, let's say, a model with another one in your own, let's say, bigger model or in a layer, then it's easier to do so.6

Speaker 1:

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

Speaker 2:

Hello?

Speaker 1:

I'm sorry, I cut off. Nope. Okay. So, come back to the last question. When selecting the model, do you care more about the model's performance than the architecture?

Speaker 2:

I think both of them need to be considered hand-in-hand, because if you're, let's say, trying to extend upon a model's existing architecture, you need to make sure there's at least some certain degree of compatibility between the models which you are trying to build and the model that already exists. But at the same time, performance is also a key factor, because if you do not have, let's say, sufficient performance on the pre-trained model, then you would probably have to end up pre-training from scratch on your entire work. So I think it's a fair 50/50 split on whether you want to focus on performance, and whether you also want to see if it can actually be applied to my network.7

Speaker 1:

Okay. Do you think the pre-trained models available in the model hubs accurately describes their behavior?

Speaker 2:

I don't think so, no. Of course, some of the major models that are published by developers, they are pretty accurate. They come up with [inaudible 00:08:23] detailed documentations, and they also explain the performances. But there are also a lot of models which are not really documented well, and they do not talk about maybe the consequences or even the performances. I think one of the most recent examples was I think there was this GPT-3 model which was trained on 4chan, and I remember remember it. So someone had essentially created a bot, and it was just published onto one of the forums, and it created a huge mayhem.9

So, these kind of things were freely available on HuggingFace hub, and these were not documented well with the consequences of using these kind of models. So, at the end of the day, developers still have to ensure that whatever work they're putting out there for the world to see needs to be documented well and should have sufficient information about what could happen when you use these kind of models.8

Speaker 1:

Okay. So, to what extent do you think the discrepancies of the performance metrics can affect your decision making?

Speaker 2:

Sorry, I didn't catch that. Could you repeat that?

Speaker 1:

To what extent do you think the discrepancies of performance metrics can affect your decision making? So here, for the discrepancies, we mean the actual performances can be different from the claimed performances in the model cards.

Speaker 2:

Okay, so that is obviously a big red flag, right?

Speaker 1:

Yep.

Speaker 2:

Because let's say you have a pre-trained model, and let's say you decide to download it, and you evaluate it on the exact same dataset that is mentioned in the paper, and you see that, okay, instead of getting, let's say, a 72% accuracy, you have a 62%. Now, that is a massive difference in the claimed value and the actual obtained value on your system, and you might spend, let's say, days trying to configure your environment or trying to figure out why this is not working, and then probably it hits you that, okay, maybe it was a one-off thing, or maybe the model does not work well under some certain environment conditions and so on.

So, at that point in time, these kind of things do play a huge role, because not only are you wasting time trying to get an existing model to work, but at the same time, puts that doubt into your mind whether, okay, should I actually go ahead with this pre-trained model and use it in my own model architecture or not? Because if I cannot get the same accuracy that is claimed in the paper, how do I know that is going to maybe benefit my model or not benefit my model and so on.

So that risk always remains, whether using a pre-trained model, whether it's being developed by smaller developers or being published by a huge organization, that kind of risk always remains. But generally you try to evaluate it on, let's say, a few model or a few datasets, and if it's within an acceptable range, usually go ahead with it. That is what I usually do.10

Speaker 1:

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

Speaker 2:

What exactly do you mean by robustness here?

Speaker 1:

Do you know some adversarial attacks, like adversarial data, and if you feed this kind of data into the model and it can behave differently?

Speaker 2:

Yeah. Okay. Models do require themselves to be robust. In fact, when the first BERT pre-trained model came out, there was the issue of the male and female bias, where occupations were categorized as more manly for certain occupations, like I think doctors and so on. So remember that kind of bias did exist initially, and I think it is still there in one of the few pre-trained models.

These kind of biases do creep in with the data that you use. Of course this is primarily down to the pre-training that is done for these models, and what I think is that robustness is a very key factor to consider when you are not only developing, but also deploying your machine learning models in the real world. Because when you are using a machine learning model is different because you might tend to use it on a few test cases, or you might use it with a smaller team and you don't really know what are the actual impacts that this could have, but when you deploy it in the real world where thousands of them could potentially use your model, then there are much larger consequences which are being taken into view, and at that point in time, you need to ensure that bias such as this is minimized as much as possible.11

Speaker 1:

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

Speaker 2:

That's one of the funny terms in deep learning, right? Because a lot of deep learning involves lot of non-explainability as well. A lot of models out there you can't really explain how they work, but you do know that maybe if I do something like this, or if I pre-train it in a certain way, this tends to work for a particular problem statement.

So, explainability in models is important to explain that, okay, it's because of these factors or it's because of these inputs I get a certain kind of expected value. But sometimes it's not always possible to explain why something is working. So if you have, let's say, a model which has very good explainability but does not perform too well, and if you have a model that has, let's say, a low amount of explainability, but performs really well with your, let's say, test environment, then I think I would go with the one with lesser explainability, because it is possible that in the downtime, I might be able to come up with an explanation to why this model behaves in a certain manner, while at the same time also ensuring that a good degree of performance is shared with the public.12

Speaker 1:

Okay. Next question is how frequently do you fine tune the models?

Speaker 2:

A lot of my work that I do is primarily aimed towards the Indian continent and the languages, so it comes to Indian social media feeds, news articles, and so on. So, a lot of times fine tuning is pretty much a very necessary step, because all models are pre-trained on English, and they don't really work well for, let's say, even a code mixed language like English. So, fine tuning is something that I do a lot, I would say at least 90% of the time.13

Speaker 1:

Okay. Do you think the lack of fine-tune-ability or trainability is a problem when we are using a pre-trained model?

Speaker 2:

No. Most models nowadays can be fine tuned easily on your own datasets, and in fact, due to how models are built nowadays, you don't even need a vast amount of data to fine tune your models. Of course, the more data you have, it is the better, but it is observed that even with, let's say, a few, maybe one GB of data or more, you can quickly fine tune your models to have that very good accuracy that you're aiming for to at least start off your work with.14

Speaker 1:

Okay.

Speaker 2:

Fine tuning has become really... Sorry. Sorry, I was continuing. Fine tuning has become really easy over the last few years, not only because these machine learning model registries, but also because the architectures themselves that support fine tuning from other researchers.

Speaker 1:

Yeah. What other challenges do you face when selecting a pre-trained model implementation from a model registry?

Speaker 2:

Sorry, I didn't catch that. Could you repeat it?

Speaker 1:

Are there any other challenges you faced before when selecting a model?

Speaker 2:

The one challenge that I sometimes do face is the language issue, which I mentioned. Apart from that, what happens is some of these pre-trained models that exist, they do not really work too well directly, even on their own datasets, and they require a lot of tuning of parameters here and there to figure out what exact parameters were used to actually run the models on their datasets. Sometimes you might spend a few days trying to figure it out to get something relatively close to what you are hoping for. So that's one of the things.

The other things is that nowadays pre-trained models are getting larger and larger and larger in size. It's not like a few years ago when pre-trained models were much smaller and you could easily fine tune them. For example, the new GPT-based models like the GPT-J and GPT-3 and so on, these have become so massive in size, it's not really possible for most hobby researchers to actually run these kind of models on their systems and fine tune on their systems. At that point in time, you need to rent servers where you can have good GPU access and you can easily fine tune and not worry. But most hobby researchers like students myself, it's really difficult to come across such resources in an easy manner, and even if you do gain access, it's not a permanent solution. It's probably for a few weeks at a time.

So, that's probably one of the things that is being noticed, although I don't think there's a solution to it. Models are getting larger in size, they're getting better, so people aren't really complaining about that.15

Speaker 1:

Okay. So then we'll move on to the next set of questions, which is about the software attributes. We want to learn what sort of information is useful to engineers who use the pre-trained models. So, here MPM defines these three attributes for the JavaScript packages, and they define the quality, popularity, and maintenance here. Can you take a look at the first one or two sentences of each attribute here and let me know what you are reading?

Speaker 2:

Yeah, okay. Has a read me, a [inaudible 00:19:00] table, has tests. Okay. Stars, folks, subscribers, contributors. Okay. Open issues, total issues, time taken to close issues, recent comments, [inaudible 00:19:12] frequencies. Okay. All right.

Speaker 1:

Yeah.

Speaker 2:

I have a general idea of what these mean now.

Speaker 1:

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

Speaker 2:

The first thing would be the popularity. If you know that more people are using it, it generally tends to point towards the fact that, okay, the model is useful, the model can be used easily. People are using the models, a large number of people are using the model, so it could provide some kind of trust in the model.

The second thing would be, let's say, I think maintenance, because pre-trained models do tend to get updated from time to time, and these updates are important. They sometimes remove bias, they sometimes are pre-trained on more data. So of course that helps as well.

Speaker 1:

Okay.

Speaker 2:

As for quality, I think quality will change when you fine tune the model after that, because at the end of the day, the pre-trained model has to be fine tuned for your application, so quality will greatly be affected by what your use case is rather than the pre-trained model that is there.16

Speaker 1:

Okay. So, here we propose three different specific attributes, which are the provenance, reproducibility, and portability. So, I will ask several questions about each of these attributes here. First I'd like to define the provenance here, which is the measure of model lineage or traceability, and some examples are whether the model contains a link to the paper, whether it has a GitHub page or external website, whether it has the datasets or architecture. So can you think about the time where you met the provenance problems when you were using the pre-trained models before?

Speaker 2:

Hello?

Speaker 1:

Sorry, can you hear me?

Speaker 2:

Yeah, I can hear you now.

Speaker 1:

So, can you think about a time where you've met any provenance problems before?

Speaker 2:

Yeah. Sometimes what happens is you come across, let's say, a paper which has, let's say, a very good architecture, or you think that it could really benefit your model, and you think that, okay, you want to give it a shot, but then you realize that it does not really point to any kind of GitHub page related to it, and might not have any kind of other implementation information which could help you out. That's one of the main things that I do look at. Because let's say if I come across a model, and one of the first things that I try to do is I try to understand how they have implemented it to look at what are the quirks, what are the things to keep note of, and how exactly they have defined the architecture. Because whatever architecture used by them is primarily going to also determine to a certain extent how it's going to tie in with my model. So yes, provenance is an important factor that I take into consideration.

Now, when it comes to external websites, not a lot of models come with external websites, so I don't think so that is too big of a deal. But dataset yes is important because I need to know how the pre-trained model first works on their own dataset, whether the claims in the paper actually match the actual code that is present with it. As far as research or commercial groups, again, sometimes not all papers come out out of a research oriented group. Sometimes they're from students and so on. So again, that is not a big factor for me. I think the main things are the link to the GitHub page, the dataset, and whether the architecture can be used.17

Speaker 1:

Okay. Is there any other things would be useful to know beforehand in order to solve these problems?

Speaker 2:

I'm sorry, I didn't catch the first part?

Speaker 1:

Is there any other things which can be useful to know beforehand in order to solve these problems?

Speaker 2:

Okay. The good thing about a few papers is that they already attach the GitHub repositories within the paper where they submitted for reviews, so when they get published, they do come out with the GitHub repositories. Maybe the other thing that we could keep in mind or what can be done is for these researchers to actually publish their pre-trained model itself so this can be directly evaluated on the dataset that exists, because most of them just release the source code for it, and the dataset might be massive, so you end up spending time pre-training it again with maybe your own hyperparameters, because sometimes they might not mention that as well. Maybe that's the thing that could probably solve this provenance issue where researchers could try to given their actual pre-trained model through some kind of platform the way HuggingFace does it, and then this can be used to actually evaluate on the datasets that's provided.18

Speaker 1:

Yeah. Okay.

Speaker 2:

So, it removes that additional step that you have of evaluating their model before going to your model.

Speaker 1:

Okay. So then we will talk about the reproducibility. Here we define it as the ability of [inaudible 00:25:07] practitioner to produce the same accuracy and training evaluation time from a pre-trained model as defined in its paper, source code, or the original group. So, can you think about the time when you met any reproducibility problems while using the pre-trained models before?

Speaker 2:

Yes. I was working on this problem for generating math word problems, and one of these recent papers, in fact, they had spoken of this architecture and there was no GitHub page attached. But the thing is, we were pretty confident of the architecture that they had mentioned, and we thought that it could go in well with the extensions that we were planning to that architecture, and so we tried it out, and it didn't really work at all.

In fact, it was a huge setback for us, because I think we spent about a week implementing it and testing it, and we did everything we could. We actually used the same hyperparameters, we used the same pre-trained models, and we were in fact even running it on similar architecture, similar GPU resources as what was mentioned in the paper, and it still did not work. It was a pretty big setback for us because we had quite a lot of high hopes from it. I wouldn't say it completely did not work. A portion of it did work, but the entire thing as a whole that was promised did not really work. As for the accuracy metrics as well, when we evaluated it on their own dataset, there was a huge difference in the scores that we received.

So, reproducibility is a huge factor when it comes to this, because researchers will first tend to evaluate how the pre-trained model works on what has been promised before going on to evaluate it on their own fine tuned model, and if that is not runnable, if that itself creates issues, you lose faith in the model that has been given, because you don't really know whether you should proceed, whether you should search for a better model, because on one hand, there is proof on paper that the model works. But on the second hand, you are not able to get it to work. You are stuck in a dilemma which you don't know whether it's because of the model, you don't know whether it's because of the implementation, you think it's something wrong with your implementation. So, it's a long spiral of a lot of things that could go wrong and you don't know which is the right thing.

So, reproducibility is I think a huge thing to make note of. Researchers should try to explain how to reproduce their model in as much detail as possible, right down to the exact parameters used, right down to the type of data being used, how the model has been implemented, and in these cases, having not only the pre-trained model, but also the GitHub repository is very important.19

Speaker 1:

Yeah. So, what do you think the model registries can offer to help you better solve these reproducibility problems?

Speaker 2:

One of the things that actually the points which you have mentioned here where you have [inaudible 00:28:18] images of the pre-trained models that actually help you quickly set up and run your particular pre-trained evaluation. This easily helps because if you have, let's say, [inaudible 00:28:30] image is the same for everyone. It quickly helps you evaluate the pre-trained model and helps you quickly come to the conclusion that, okay, this pre-trained model actually works.

The other thing is some repositories are kind enough to provide scripts that actually show how the model can be initialized to evaluate it on a few standard datasets which the paper has used. Those also help actually test the paper quickly will come up with a conclusion to see if it works and then proceed.

Speaker 1:

Okay.

Speaker 2:

Apart from that, the general standard practice of mentioning all the hyperparameters, mentioning all possible configurations that are present, that were at ever present the time of testing the model, and so on, those have to be present regardless of what happens.20

Speaker 1:

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

Speaker 2:

The only portability problems I've encountered so far is when a few models have been written in slightly different versions of [inaudible 00:29:52], and that creates a compatibility issue, the one which I have. Or sometimes the code isn't completely optimized for [inaudible 00:29:59], and what happens is you need to go through the core and see which exact layer is causing the issue. That's one of the things that the architecture itself has some issues with, with the compatibility. So that's one of them. But those are relatively easier to fix, I'd say, because if you're writing the entire model or extending the model, you tend to go over all the layers that are present, so sooner or later, you will be able to fix that issue. But yes, it's one of the factors that is present.

The other thing is present in your slide about fine tune-ability. Yes, the ability to fine tune it also matters to see how easy it is that it can be quickly ported to your particular task. Some tasks are easier to port to, but some are not, based on the pre-training that is done. But again, that depends on your use case, the model that is chosen, and so on.21

Speaker 1:

So, what do you think the model registries can offer to help you solve these problems?

Speaker 2:

Regarding portability, I am actually not sure, because you tend to run into these issues only during implementation and testing of the models. So, I'm not really sure how a model registry might be able to counter that. Yeah.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 that would be helpful for the pre-trained model we're using?

Speaker 2:

Provenance... There's nothing coming to the top of my head right now, but because your points actually cover everything.23

Speaker 1:

Okay.

Speaker 2:

So I'd say everything that has ever come to me gets covered under these three categories here. So no, at the top of my mind, not really.

Speaker 1:

Okay. Here we have a last set of questions about the model trustworthiness, and I think we are running out of time here, so you can let me know if you have to leave.

Speaker 2:

No, no, it's okay.

Speaker 1:

Okay, thank you. So, here we are trying to understand how pre-trained model shortcomings can affect the engineer's ability to rely on and reuse them. First, which aspects of the pre-trained model do you assume are trustworthy?

Speaker 2:

Okay. One of the things is, as I mentioned before, the bias which is present in the pre-trained models. People will test a model more if they know that there is no bias towards any kind of factor. The moment the model starts showing bias, you know that the results are not always... Or you cannot depend on the results always. So, the bias in the pre-trained model can affect the trustworthiness.

The second thing is about whether the model can actually be used for a malicious purpose. A lot of the models that exist in NLP nowadays can be used for malicious purposes, and as models are getting better and better, yes, they're able to solve a lot of more problems. Things that are possible today could not have been possible years ago. But at the same time, you start questioning if, let's say, a model were to fall into, let's say, the wrong kind of hands or were to be trained with malicious intents, and they could have a lot of consequences in the real world. Within a smaller group, maybe not, but when, let's say, deployed in the real world, you tend to see these issues more common.

So, two of the things that I'd say is the ability of a model to be used for malicious use, and the second thing is the bias which is present.24

Speaker 1:

So, as a follow up question, have you ever met any malicious models before from the model registries?

Speaker 2:

No. No, I have not.

Speaker 1:

Okay.

Speaker 2:

I have heard about how these have been used, but it's something that I stand against, and I haven't [inaudible 00:34:05].

Speaker 1:

Okay. Next question is, have you found any discrepancies between the [inaudible 00:34:11] pre-trained models and the downloaded version? For example, in terms of the accuracy, latency, and architecture of the model?

Speaker 2:

So, architectures and latencies, not really, but as I mentioned before, I have experienced scenarios where there's a difference in the accuracy metrics that have been mentioned.

Speaker 1:

Okay. So, how did you find the discrepancies of the accuracy?

Speaker 2:

I usually tend to evaluate it on the dataset that has been mentioned on at least one of them. I usually do it on two, but it depends on how well I get the scores for the first dataset. So, I remember I did it on the first dataset, and the scores didn't really come out okay, and I thought at that point it was probably some kind of configuration issue, and I spent a few days trying to figure out what could be the issue. I actually tried hyperparameters not even mentioned in the paper and it still didn't work out, and at that point I dropped the model.25

Speaker 1:

Okay. So, to what extent do you think these discrepancies are acceptable? For example, less than a certain percentage discrepancy of accuracy?

Speaker 2:

I usually go for about 5%. I really can withstand a decrease in 5% because I know that that's not the final accuracy I'll be getting, and if I fine tune it on my work, it is possible I can end up with a completely different result. So, I usually keep a threshold of 5% decrease in accuracy as the final limit I am willing to work with. But beyond that is just a pointer towards something wrong in the model itself.26

Speaker 1:

Okay. One follow up question here is, we mentioned you have found some accuracy discrepancy before.

Speaker 2:

Yes.

Speaker 1:

Are those discrepancies less than 5% or larger than 5%?

Speaker 2:

Oh, there were larger than 5%.

Speaker 1:

Okay.

Speaker 2:

Yeah. That was one of the things. The other thing is it was a generative kind of problem statement where we had to generate word problems, and on the same architecture on the same hyperparameters, we didn't really get anything.

Speaker 1:

Oh, okay.

Speaker 2:

It was illegible text that was being sent out, so we knew it wasn't really worth the time.

Speaker 1:

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

Speaker 2:

Yes. If you know that your pre-trained model doesn't work too well on the dataset that has been used to evaluate it in the paper, then it points towards the fact that maybe there is some discrepancy in the model architecture or the way it was trained, or maybe the researchers have not provided the truth values in the first place. So it does affect your decision making. It [inaudible 00:37:06] doubt whether you should continue with the model or not.

So yes, discrepancies do play a huge role, but at the same time, I think that you cannot let small discrepancies actually hinder your progress, because as I mentioned, you are going to fine tune your model, and you could end up with a completely different result on what happens after fine tuning, because at the end of the day, you're fine tuning on something else. So, maybe to an acceptable level, it's good to go. For example, I use 5%, some people might use lesser. But beyond a certain threshold, that's more of a problem than what can be dealt with.27

Speaker 1:

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

**Annotations**

1 Role:

- Mainly independent work

- Just graduated this year

- Research work: write models of my own

- Industry work: Mainly work in analytics as a big tech company

2 Reuse scenarios:

- Use pre-trained models a lot of times, especially since my work is pretty much 100% in NLP.

- I tend to use a lot of PTMs like BERT, GPT-2

- For either fine-tuning on my own applications or to just build on top of them.

In my recent paper, it uses PTMs and enhances them using a better learning step

- It's primarily for code mixed languages, which is essentially a language where you have multiple languages mixed in

- Normal PTMs like BERT and so on do not really get a good performence on benchmark datasets.

3 Retrain

- We don't have the resources to completely train a PTM.

- The only thing I have done so far is fine tuning.

- We have done pretraining from scratch on smaller scale datasets. (GBs of data)

4 Backbone?

- Used the mian BERT model as a layer in our network.

- We have extended them by adding more layers to it.

5 Decision making:

- Application which you are trying to use it for, because PTMs do not work for every use case.

- Cut down the options that you have by a huge margin

- Once you filter through the options that are there, it's mainly based on the language which you are trying to do and the dataset it could be trained on.

- In fact, there are only a few options that are there for the Hindi language

Application -> see if within that application there exists any work or where your domains overlap or where your use cases overlap, and that is how you filter through them -> pick the model that you think would work best for you

6 Where?

- HuggingFace

- Why:

- Transformers have been getting more and more popularity

- Generally, HF has a huge collection of models

- A lot of it is plug and play. -> Easy reuse and test

7 Performance or architecature

- Both of them need to be considered hand-in-hand

- If you are trying to extend upon a model's existing architecture, you need to make sure there is at least some certain degree of compatibility between the new one and existing one.

- Performance is also a key factor. If you don't have sufficient performance on PTM, then you would probably have to end up pretraining from scratch on your entire work.

- I think it's a fair 50/50 split

8 Discrepancy?

- No.

- Some of the major models that are published by developers, they are pretty accurate.

- These models have detailed documentations and they also explain the performances

- There are also a lot of models which are not really documented well. They do not talk about maybe the consequences or even the performances.

- These kind of things were freely available on HF hub. These were not documented well.

- At the end of the day, developers still have to ensure that whatever work they're putting out there for the world to see needs to be documented well and should have sufficient info about what could happen when reusing these models.

9 Security problem:

- Example: GPT3 model which was trained on 4chan, someone had essentially craeted a bot and it was just published onto one of the forums, and it created a huge mayhem.

10 Discrepancy:

- Obviously a big red flag

- discrepancy -> you might spend days trying to configure your environment or trying to figure out why this is not working

- At that point in time, these kind of things do play a huge role.

- wasting time trying to get an existing model to work

- put that doubt into your mind whether should i continue

- That kind of risk always remains, whether using a PTM developed by smaller developers or being published by a huge organization.

- Generally you try to evalaute it on a few models or a few datasets. If it's within an acceptable range, usually go ahead with it.

11 Robustness:

- Example: male and female bias

- That kind of bias did exist initially and I think it is still there in one of the few PTMs.

- The biases do creep in with the data that you use.

- Robustness is a very key factor to consider when you are not only developing, but also deploying your ML modesl in the real world.

- In development, you might tend to use it on a few test cases, or with a smaller team and you don't really know what are the actual impacts that this could have.

- But when you deploy the model in the real world, there are much larger consequences which are being taken into view.

- At that point in time, you need to ensure that bias is minimized as much as possible.

12 Explainability:

- A lot of models there you can't really explain how they work. But you do know that maybe if I do sth like this, this tends to work for a particular problems statement.

- Explainability in models is important to explain that.

- Sometimes it's not always possible to explain why sth is working.

- I would go with the one with less explainability because it is possible that in the downtime, I might be able to come up with an explanation to why this model behaves in a certain manner.

- While at the same time also ensuring that a good degree of performance is shared with the public.

13 Fine-tune: pretty much a very necessary step. At least 90% of time.

14 Fine-tunability/trainability

- No.

- Most models nowadays can be fine-tuned easily on your own datasets.

- Due to how models are built nowadays, you don't even need a vast amount of data to tine-tune your models.

- A few, maybe 1 GB of data or more, you can quickly fine tune your models to have that very good acc.

- Fine tuning has become really easy over the last few years. Not only bacause these ML model registries, but also because the architectures themselves that support fine tuning from other researchers.

15 Other challenge:

- language issue

- some of these PTMs that exist, they don't really work too well directly, even on their own datasets.

- They require a lot of tuning of parameters here and there to figure out what exact parameters were used to actually run the models on their datasets.

- The other thing: PTMs are getting larger and larger and larger in size. It's impossible for most hobby researchers to actually run these models on thier systems and fine tune them.

- Models are getting better, so people aren't really complaining about that.

16 Traditional attributes:

- Popularity > maintenance > quality

- popularity

- it could provide some kind of trust in the model

- maintenance

- PTMs do tend to get updated from time to time. These updates are important. They sometimes remove bias, some things are pre-trained on more data. Of course that helps as well.

- quality

- qulaity will change when you fine tune the model after that.

17 Provenance issue:

- The paper dose not really point to any kind of GitHub page related to it and might not have any kind of other implementation info which could help you out.

- Provenance is an important factor that I take into consideration.

- Not a lot of models come with external websites. That's too big of a deal.

- Dataset is important because I need to know how the PTM first works on their own dataste, whether the claims in the paper actually match the actual code that is present with it.

- As far as research or commercial groups, sometimes not all papers come out of a research oriented group. Sometimes they're from students and so on. So this is not a big factor for me.

- The main thing sare the link to the GitHub page, the dataset, and whether the architecture can be used.

18 Provenance help:

- Good thing abuot a few papers is that they attach the GitHub repo within the paper submission

- For these researchers to actually publish their PTM itself so this can be directly evaluated on the dataset that exits.

- Most of them just release the source code for it. The dataset might be massive.

- you end up spending time pre-training it agian with maybe your own hyperparameters.

- Researchers could try to give their actual PTM through some platform the way HF does it, and then this can be used to actually evaluate on the datasets that's provided. So it removes the additional step that you have of evaluating their model.

19 Reproducibility issue:

- Problem: generating math word problems

- They had spoken of this architecture and there was no GitHub page attached.

- We actually used the same hyperparameters, same PTMs, and running it on similar architecture, similar GPU resources as mentioned in the paper. It still did not work.

- It was a pretty big setback.

- Researchers will first tend to evaluate how the PTM works on what has been promised before going on to evaluate it on their own fine tuned model. If that is not runnable, you lose faith in the model. Because you don't really know whether you shuold proceed, whether it works

- It's a long spiral of a lot of things that could go wrong and you don't know which is the right thing.

20 Reproducibility help:

- images of PTMs that actually help you quickly set up and run your particular pre-trained evlauation.

- It helps you quickly come to the conclusion that, this PTM actually works.

- Scripts that actually show how the models can be initialized to evaluate it on a few standard datasets which the paper has used.

- Those help actually test the paper quickly -> if the model works and whether we should proceed

- General standard practice of mentioning all the hyperparameters, metioning all possible configurations that are present.

- including the time of testing the model and so on

- Those have to be present regardless of what happens.

21 Portability issue:

1. When a few models have been written in slightly different versions -> compatibility issue

- Some times the code isn't completely optimized for ...

- You need to go through the core and see which exact layer is causing the issue.

- Those are relatively easier to fix. Because if you are writing the entire model or extending the model, you tend to go over all the layers that are present. So sooner or later, you will be able to fix that issue.

2. fine-tunability.

- Some are easier to port to, but some are not. Based on the pre-training that is done. That depends on your use case, the model that is chosen, and so on.

22 Portabiltiy help:

- Not sure. You tend to run into these issues only during implementaiton and testing of the models.

23 Other attributes:

- Your points actually cover everything. Everything that has ever come to me gets covered under these three categories here.

24 Trustworthiness:

1. The bias which is present in the PTMs.

- People will test a model more if they know that there is no bias towards

- The moment the model starts showing bias, you know that you cannot depend on the results always.

2. Whether the model can actually be used for malicious purpose.

- A lot of NLP models nowadays can be used for malicous purpose.

- If the model were to fall into the wrong kind of hands or were to be trained with malicious intents. They could have a lot of consequences in the real world.

- When deployed in the real world, you tend to see these issues more common.

- I have not met any malicious model before, but it's something I stand against.

25 Discrepancy:

- Not architectures and latencies.

- Accuracy:

- How to find it: evlauate it on the dataset that has been mentioned on at least one of them.

- I usually do it on two. But it depends on how well I get the scores for the first dataset

- I actually tried hyperparameters not even mentioned in the paper and it still didn't work out. At that point I dropped the model.

26 Acceptable discrepancy:

- 5%

- If I fine-tune it on my work, it is possible I can end up with a completely different result.

- so I usually keep a threshold of 5% in accuracy as the final limit

- Beyond that is just a pointer towards sth wrong in the model itself.

- It wasn't really worth the time because it was illegible text that was being sent out.

27 Significant impacts:

- it points towards the fact that maybe there is some discrepancy in the model architecture or teh way it was trained.

- Discrepancies do play a huge role. But at the same time, I think that you cannot let small discrepancies actually hinder your progress.

- You could end up with a completely different result on what happens after fine tuning.

- Maybe to an acceptable level, e.g. 5% or lesser. But beyond a certain threshold, that's more of a problem than what can be dealt with.