An empirical study of challenges in machine learning asset management

This paper seeks to address the difficulties many users within the machine learning environment encounter. Specifically, the management of assets like datasets, models, code, configuration files etc… There hasn't been research conducted in this way specifically within the context of machine learning applications. The researchers try and shed light on the lack of real-world user challenges and solutions across different ML application tools.

The new idea proposed in this article was to gather Q&A posts that would help to identify challenges and solutions across varying forums. The researchers then deployed BERTopic, which allowed them to use topic modeling in order to get a better understanding of what the posts were related to.

The researchers split the posts into knowledge inquiries and problem inquiries. The latter dominating 60% of the 15,000 posts gathered. The importance of this distinction was to help identify what types of posts require cross-domain solutions versus those that can be self-resolved with documentation or code examples.

Lastly, the researchers broke their analysis down into three main research questions. What topics are most frequent, what topics of solutions exist, and what are the commonalities and differences between forums. The first two research questions were both dominated by the environment and dependency management macro-topic. Around 1 in 5 developers deal with these challenges as well as provide solutions regarding this topic. It's a universal pain point across all forums.

Positives

The dataset was diverse and well rounded. I think having over 15,000 posts is a legitimate amount to start generalizing forum behaviors. I also thought their method was solid. Yes there is the potential of bias being introduced into the study with the Macro-Level topic labeling step; however, I see this as positive because it allows for the researchers to be more hands on with the data they are gathering. When you are forced to sort topics manually, you understand your dataset better and the patterns that come out of it. I have had my own experience with topic modeling and found the manual aspect involved in a pipeline similar to this allows researchers to uncover nuanced, human-interpretable themes that pure automation may miss. Ultimately this method allows for raw AI clusters to be turned into meaningful insights. Lastly, this article was very practical in its intent. Can we better anticipate in our community forums the issues most developers are experiencing, or can tool makers innovate more quickly? Or is this simply an issue with poor documentation because of the dynamic inherency within the ML environment?

Negatives

There does not seem to be a lot of negatives I see from this paper. The more obvious negatives are addressed in the threats to validity section well, so I don't think I need to restate those (subjective bias, not including all forums etc...). However, I do think it's important to note that the study was only done on open-sourced tools and that the study could potentially be overlooking issues or non-issues found in proprietary tools. Secondly, were there any patterns found within the ~1,000 posts that were not labeled? Just something that would be interesting to know why they had a hard time being labeled.

Future Work

There are two direction I see future work going with this type of analysis. If we know that environment and dependency management is the number one pain point, how can we encourage the community to provide users better documentation or learning resources? We could conduct a study where we ask a specific forum like stack overflow to provide some incentive for users to provide more detailed documentation (this may be a stretch). But the idea being how could incentivizing documentation increase developer experience in environment and dependency management. And secondly, a longitudinal study could be interesting. Over a one-year period: do we see challenges and solution topics changing in their distribution? Do new topics start appearing? Or is this number one pain point inherent to the field of software development and so the ML environment will also see similar statistics.

Rating

5/5. Clear practical implications, sufficient data set, and helpful read for my own research,

In-class Discussion Summary

The first question a student asked had not to do with the article itself but software engineering research in general. That being, most research papers in this field do not mention a baseline. There was nothing addressed in the paper regarding what the baseline is outside of the ML field. The point being that if it's the same, then maybe this study doesn't really tell us anything. If environment and dependency management are typical challenges across the industry, then the findings in this paper tell us something we already know and nothing specifically pertaining to the ML field itself. The professor pointed out that it can be difficult to find a "baseline" and actually validate the baseline. This makes sense to me and I can see both sides of this argument of baselines are important, but let's not also get bogged down in the baselines which could prevent the research from being conducted.

The second discussion question I posed to the class was, "There are many dependency problems related to Docker or other environment setups especially in ML scenario." One student mentioned how the current tool set that is out there for environment management he doesn't prefer. This makes me think that maybe our tools are really lacking in this area, but then I think of the dynamics in environment management and the give-and-take a developer has to make when choosing between different tools, performance needs, and reproducibility. Suggesting the issue isn't just poor tooling, but the built in complexity of balancing flexibility, speed, and consistency in real-world ML workflows. A student chimed in here and talked about how he feels this is a really important topic because it could reshape how people think about the issues facing this topic.

I then posed my third question, regarding what role education could play to try and mitigate this consistent issue found in software development and have the opinion that maybe the reason school doesn't teach more about this topic is because the ML Ops are evolving too fast. One student chimed in here and said a community effort can improve the dilemma seen in this area and that really we can count on no one else but the developer community to help mitigate issues in environment management.

Lastly, one student brought up that Git Hooks could potentially be a solution where it automatically adds the env and host platform's information before developer code is pushed to the repository. I would agree with this approach. Having an environment.yml file can really help to get environment dependencies organized. However, this is something that I figured out as I learned. It would have made my life a bit easier to have had some prior knowledge on best practices, but I do think it is just the nature of software. You don't know something until you try and build it, then you learn it. Because of this, it makes sense why developers who have more cross-domain knowledge can more easily handle the troubles encountered when dealing with environment management and dependency hell. And maybe that is the lesson we take from this article. Either get your feet wet in lots of different areas of software development, or be a contributor to the Q&A forums.