Assignment – WASP Software Engineering 2024

Author: Stefan Stojanovic

Intro To My Research

My area of research is reinforcement learning (RL) and related topics. Reinforcement learning is a branch of machine learning that focuses on sequential learning problems – such as learning to move a robot arm or drive a car. Compared to other areas of machine learning (such as supervised and unsupervised learning), RL is less robust and not as well understood. A key issue specific to RL is that the learning agent has the freedom to choose data it samples, creating a vicious cycle of model update and data generation within the training process itself. In contrast, (un)supervised learning models train on fixed data; once a model is trained, it can choose to sample a different dataset and repeat the process, but this occurs only after the model has already been trained.

There is a natural transition between these areas - for example, the subfield of offline RL considers settings where the data is fixed in advance, and (almost) no additional sampling is allowed. Because of these complexities, training RL models is often not robust – it is almost standard practice to report the random seeds for which the algorithm works (and for which it does not). This lack of robustness makes RL algorithms less practical for wide deployment in the real world. So far, they have mainly been successful in the field of gaming and advanced robotics.

Nonetheless, it is worth noting that large language models (LLMs) use RL algorithms (with human preferences) in the latter stages of their training. This demonstrates that RL algorithms can play a role even in settings where (un)supervised learning has traditionally been the standard approach.

Two Valuable Ideas From Robert's Teachings

The first topic I want to discuss is software testing. According to Robert's slides: "Software testing is the process of exercising a software system using a variety of inputs with the intention of validating its behavior and discovering faults". Moreover, testing can be manual or automated, it becomes increasingly difficult with larger input-output spaces and, most importantly, requires clearly defined requirements. Thus, it is no surprise that testing modules and pipelines containing ML submodules is extremely challenging. As I mentioned in the intro, it is standard practice in the RL community to report the random seeds for which an algorithm works. Therefore, it is very difficult to test a model when you do not expect it to work consistently but only in certain random instances. Additionally, it is challenging to predict how well your algorithm should perform based on the amount and quality of data collected. On a high-level note, this course has opened my eyes to how much engineering is actually required in an ML framework.

This leads me to the second idea worth noting here, which could be concisely called "ML deployment challenges". Unfortunately, as ML researchers, we often do not get a chance to work with ML engineers and experience the difficulties of deploying and developing ML frameworks. For example, the seemingly simple task of version control becomes an engineering challenge when you start versioning the data as well. Since data collection is part of an RL algorithm, it is interesting to see how people in industry actually solve this issue: do they save and annotate all the data they collect during different training iterations? More importantly, how do they make sense of this data chaos during validation?

Two Valuable Ideas From Per's (SAAB) Lecture

First idea: The importance of behavioral software engineering. Behavioral software engineering is a subfield of software engineering that focuses on understanding and integrating human and behavioral aspects into software development processes. It applies psychological and social principles to address issues affecting software development. Per emphasized that the connections and relations between people are crucial for successful product development. I believe this idea is important for anyone working with technology and software, not just for my specific area of research. We can easily fall into the illusion that what we develop is purely objective and impersonal. However, since software is a product of collective human effort, interpersonal dynamics play a significant role in the final outcome. In ML software, where systems often contain even more interconnected submodules than traditional software, we can expect that the interconnectedness of those working on ML problems becomes even more crucial for efficient product development.

Second idea: Challenges in promoting AI adoption. Per also discussed the difficulties of promoting AI adoption, even among those working in IT/tech. For example, in his company, opinions about the relevance of software engineering in the face of increased AI/ML integration vary widely. Some believe AI will make their jobs more mundane and boring, while others think AI is overrated. Additionally, Per mentioned people's reluctance to use AI tools in their jobs (ex. in air traffic control). This is a general issue, not unique to my research area, but it underscores the importance of making new technology accessible, understandable, and approachable so that people can accept and integrate it into their everyday work environments.

CAIN Paper 1: What Bothers Practitioners? [1]

Next, we discuss a meta-summary study [1] that examines the challenges reported by industry practitioners who develop products with ML components. By filtering the results of 50 studies involving over 4700 practitioners, this paper aims to identify the main challenges encountered by the practitioners. The study considers papers that move beyond the model-centric view of classic data science workflows to focus on building automated pipelines and entire software systems with both ML and non-ML components. Some of the challenges reported in the study are related to:

- Requirements engineering: issues such as unrealistic expectations (from customers, managers, team members) often arise due to a lack of education in AI (ex. among stakeholders) or the inherent difficulty in mapping model performance to business goals. *My research:* This challenge somewhat overlaps with what I discussed in the section on Robert's lectures regarding the difficulty of defining clear requirements. In reinforcement learning, where models often interact dynamically with their environments, defining requirements is particularly tricky because the desired behavior may evolve as the agents learns. Although there is more flexibility in academic settings compared to industry, aligning RL objectives with well-defined goals remains a challenge that could benefit from better requirement engineering practices.

- Architechture, design and implementation: Practitioners note that transitioning from model-centric to pipeline-driven product development is challenging. Moreover, the lack of encapsulation and modularity makes standard SE practices difficult to apply to ML workflows. *My research:* In reinforcement learning, building scalable and modular architectures is a significant challenge, especially when dealing with complex environments or multi-agent systems. RL systems require a high degree of flexibility and adaptability, which complicates the application of traditional SE practices. While these issues are indeed important, they are hard to tackle directly in academia due to their engineering focus. However, understanding these challenges is crucial for my future work that may involve deploying RL systems in real-world settings.

- Quality assurance: There is no strict notion of "correctness" in ML, making testing and debugging challenging. Issues like the silent failure of models (where models provide incorrect answers rather than crashing), the long tail of corner cases, and "invisible errors" are common. Unit tests are often insufficient due to the entanglement of components, and online testing is complex, expensive, and time-consuming. *My research*: In RL, quality assurance is particularly challenging because the agent's performance can be highly sensitive to the training environment, hyperparameters, and random seeds. Developing robust testing frameworks that can account for these variabilities is an ongoing area of research. Better methods for validating RL models' robustness, stability, and safety across diverse scenarios could directly impact the deployment of RL in practical applications.

- Process: Development processes are often ad-hoc and lack well-defined structures. *My research:* This is especially true in RL, where the iterative nature of exploration and exploitation makes it difficult to predict training times and outcomes. While academic research often embraces the exploratory aspect, transitioning to more structure processes would be beneficial for scaling RL research to real-world applications.

CAIN Paper 2: How Do You Choose Your Hyperparameters? [2]

As mentioned several times, improving the robustness of ML software is an open research question. One of the key challenges in achieving this is the choice of hyperparameters. It is not uncommon for ML algorithms to have dozens or even more hyperparameters that need to be set. The study discussed next examines how hyperparameters are chosen in practice by analyzing 2000 code repositories and related papers. The authors posed two research questions: 1) To what extent are ML methods configured w.r.t. their hyperparameter settings? and 2) How are hyperparameters reported in the accompanying papers?

Although there are numerous hyperparameter tuning strategies - such as manual tuning, grid and random search, Bayesian optimization, bandit-based methods etc, most ML practitioners appear to rely on very primitive techniques. This finding is surprising, given that some studies suggest that tuning the hyperparameters of an ML method is often more critical than choosing the specific ML method itself. Here are some of the main conclusions of the study:

- Most papers change only a few hyperparameters from their default values.
- The majority of hyperparameters are set to constant values rather than being adapted dynamically by a function. Consequently, frameworks like MLflow cannot be used to adapt and documents these values effectively.
- The most frequently changed hyperparameters are common ones, such as the learning rate.
- Most papers use simple search methods like grid, random or manual tuning, although some utilize highly specialized libraries such as Optuna.

What about hyperparameter optimization for RL?

RL models typically have a large number of hyperparameters that influence learning rates, exploration-exploitation balance, discount factors, and other aspects of agent behavior. Properly tuning these hyperparameters is essential for achieving robust and effective RL models, as poor choices can lead to suboptimal policies or even failure to learn. Given that RL often involves iterative training processes in dynamic environments, the ability to fine-tune hyperparameters dynamically rather than relying on static, default values, is crucial. Therefore, insights from this paper could inform better hyperparameter tuning strategies in RL, potentially improving both the stability and performance of RL algorithms.

For example, a larger AI project could benefit from implementing more sophisticated hyperparameter optimization methods beyond simple grid or manual searches. Techniques like Bayesian optimization or bandit-based methods could be employed to dynamically adapt hyperparameters based on real-time feedback from the environment. Furthermore, we could also focus on developing RL agents that are inherently more robust to hyperparameter variations by incorporating meta-learning approaches or adaptive learning rate schedules.

References

[1] Nadia Nahar, Haoran Zhang, Grace Lewis, Shurui Zhou, and Christian Kästner. A meta-summary of challenges in building products with ML components–collecting experiences from 4758+ practitioners. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 171–183. IEEE, 2023.

[2] Sebastian Simon, Nikolay Kolyada, Christopher Akiki, Martin Potthast, Benno Stein, and Norbert Siegmund. Exploring hyperparameter usage and tuning in machine learning research. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 68–79. IEEE, 2023.