WASP: Software Engineering: Assignment 2

Tobias Karlsson
Chalmers University of Technology
tobiaka@chalmers.se

1 My research

My research is about using machine learning to build generative models for protein sequence generation. Designing proteins is important in many areas of application, for example in the design of vaccines. The search space is huge and therefore it is difficult to find which sequence that will be the best with respect to the design constraint. Machine learning can be used to approximate some landscape of fitness in sequence space.

2 Three topics

1. Automated Software Testing: Manual testing of software can be time consuming, automation of tests saves resources and is critical for continuous delivery. The automation can be done in many different ways. One way that we have discussed in the course and that I plan to incorporate into my personal projects is unit testing with git integration. This way, a number of predefined tests are run on the software each time new updates are committed to the repository. For a personal project this may or may not be necessary, but for larger projects with many authors I think a good way to test the code to assure that it fulfills a number of criterias and catch trivial bugs are of uttermost importance. When it comes to testing of software for ML/AI more specifically, I think one of the obvious things that could use automated tests is the data pipeline. Data which are intented for training of the model should go through checks that it contains what we think it does. This could be done by checking that the distribution fulfills some criteria. Also, we don't want data which has missing values or similar defects. Another thing that I consider could be useful is to run the training pipeline with some dummy model and data which is many orders of magnitude smaller to check that there is not an error in any of the post training code, since it can be very wasteful of resources when you have run your training for hours to only then realize that you had a bug in the part where you save the model.
2. Project management: Project management for software is a very broad topic. One of the things I want to discuss that may fall under this topic, where I currently see limitations, are how to write and debug code when working with very large machine learning models. My current pipeline is that I write the code locally which I push to a git repository which I pull from on a high performance node which I can access via ssh. This way, there is a lot of pull/push back and fourth between my local computer and the node. Further, on my local computer I can not load the models since they are to large to fit in my RAM and GPU. Sometimes there are errors that occurs on the node with the full sized model which do not happen locally with the smaller sized dummy model I use for working with the code. This can be problematic even if IDEs like vscode provide some remote debugging tools. This is something I want to investigate more since I think it can improve my typical software pipeline.
3. Regulations and Compliance: In order to successfully create machine learning systems, good data is key. Therefore it is not surprising that different applications collect data about its users some of which may be sold in between companies. There are regulations with respect to how user data can be stored, for example GDPR. Still, there are many parts of the

world where GDPR is not being used and even if it is used there is a lot of concern when it comes to data collection with respect to privacy. This issue will increase in relevance when more and more of the devices around is are connected to the cloud. There are two conflicting goals. Firstly, we want smart systems around us which are potentially also personalized, but we don't want to share private data. There is a need for research and awareness as well as regulations to assure that there is a compromise between these two goald and how they can be achieved. Further, there needs to be a discussion for how we as society wants to tackle this problem, what we consider as private data etc.

3 Future trends and directions of Software Engineering

Regarding the idea that ML will "eat Software", I agree to some extent. Much of the code we use are snippets of copy paste smaller units into our specific combination of glueing together these units which together form the program. There is already promising research in the topic of machine learning programs generating code given a prompt input for the desired functionality of the code. For example [1] or github copilot. I think this is a trend that will only get stronger and mature in the coming years. I think this is great since it may speed up software development if more time can be spent on thinking about the overall architecture and design of the system and less with writing the details of code which has already been written thousands of times. However, there are dangers coming with relying on automatically generated code. Firstly, the programmers need to be at least as skilled as before when they wrote more code themselves, they need to understand the suggested code generated by the ML system and evaluate whether the suggested solution actually do what is wished for or whether a better solution exists. One also needs to understand where the generated lines come from, that they probably reflect some distribution of the training data, which currently is public repositories from github, and that they are probably imperfect and biased.

I don't understand the idea that machine learning should not be a subset of software engineering. To me it feels obvious that it is a strict subset of software engineering. One could argue that data collection etc is outside the set of software engineering. But to me that statement feels equivalent to the statement that users interacting with an application or webpage is outside the domain of software engineering, or usage of a database. Regardless, I think it is not an interesting question since it is basically just a discussion of definitions.

I think one strong trend when it comes to software and machine learning, is an increaing amount of machine learning operation tools being developed which provide utility in different ways. Monitoring, grouping and labeling of experiments. Automated hyperparameter search. APIs for easily loading state of the art pretrained models etc. This is related to what I discussed earlier and I think it is a trend which is helpful for the machine learning community since many of us have probably experienced the challenges associated with running a large number of experiments and keeping track of all the interesting results and being able to replicate them.

References

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.