WASP Software Engineering and Cloud Computing Course

Assignment 2 Module 2 - Essay

Simona Gugliermo

Automated planning and scheduling (denoted as simply AI planning) methods have the potential to introduce disruptive changes in industrial transport applications, as they automate the key process of assigning resources to tasks, deciding when tasks need to be carried out, and ensuring that plans remain feasible over time in the face of contingencies. However, AI planning methods rely on manually specified knowledge encoded in the so-called planning domain. Obtaining accurate and complete domain descriptions is a notoriously difficult and time-consuming process, even for domain experts, as it requires a significant knowledge engineering effort. The multitude of factors and quality metrics that should be accounted for in order to obtain a feasible plan and optimize its quality are primarily responsible for this difficulty. Many of these factors are known to and learned by human planning experts from experience. Machine Learning (ML) can enhance AI planning in order to learn from human planning experts and from traces of plan executions. Furthermore, learning algorithms could support automated planners in order to save human effort by producing domains that lead to good quality plans in less time and overcome the challenge of depending upon the fleet transport managers experience. Finally, the idea is to use the results of this research in the context of fleets of autonomous vehicles.

Automated testing

Automated testing is a process that validates whether the software is working properly and meeting requirements before being released into production. Indeed, it can help identify bugs and flaws early in the development lifecycle, which can save time and money on fixing them later on. Automated tests are also known to be more reliable than manual tests because they are less prone to human errors. Automated testing and Machine Learning (ML) can be “combined” in two ways. There have been studies on how to use ML for automated testing and, conversely, on how to use automated testing in ML. The former is used to address the problem of continually updating test cases when changing software. The ML model can help automatically generate and update test cases and improve the scope of current code, thus producing more quality and quantity work in less time. Therefore, ML has an enormous potential to significantly improve the existing test automation environment by accelerating speed and accuracy of tests and, therefore, of the overall software. On the other hand, using automated testing in ML is challenging and my understanding is that there is no one standard way of doing it. Indeed, ML projects are heavily dependent on data and models that cannot be specified a priori. This entails that testing ML projects is more complicated than conventional software testing. Moreover, in contrast to most conventional hand-coded software tests, ML projects have much more uncertainty than traditional software. Finally, when using automated testing in ML, it is crucial to make sure that the system is functioning correctly not only in development but especially in production. Indeed, over time in ML projects, some external conditions might cause data shifts or we might change the data source and provider and, therefore, testing only the development environment is not enough. However, automated testing in ML would help prevent to (i) accumulate technical debt, (ii) reduce developers effort and overall costs, and (iii) improve quality and consistency. For this reason, I believe this area of research will be a trend of the future as the results will be greatly exploited by companies.

Project Management

Software Project Management (SPM) is an established way of planning and leading software projects. Its purpose is to facilitate the planning and tracking of project components, stakeholders and resources. As for automated testing, I think that the role of the ML in project management can be dual. Indeed, there are studies on how to use ML to improve software project management techniques for traditional software but also on how to properly manage a software of an ML application[^1]. In this essay, I will focus on the latter problem. It is challenging to manage ML projects as these typically differ from other types of software development projects. A key difference is the nondeterminism. This entails that predicting how long tasks will take to complete is difficult. Moreover, it is challenging to set clearly defined goals, as well as setting clearly defined expectations because it is often not clear the value/quality of a potential predictive model prior to a project. Indeed, ML models are programmed implicitly, not explicitly, and the performance are not known a priori. Therefore, it is hard to know in advance what tasks are prone to fail or what are prone to succeed thus leading to hard times in project management. Nevertheless, a proper SPM is extremely important in order to achieve the desired goals. Therefore, I believe that this area presents many research opportunities in order to overcome these challenges

Sustainability

Sustainable software development is an approach to software design that emphasizes energy efficiency and environmental sustainability. The goal is to minimize the impact of applications and the infrastructure that hosts them on the planet. A software can become more sustainable by making it more efficient and reducing its energy consumptions. Training ML models demands a significant amount of energy consumption and computational power. Garcìa-Martìn et al.[^2] provide useful guidelines to give energy estimation values for machine learning algorithms. Moreover, many ML applications are deployed on data centers. The latter require enormous amounts of energy to operate. I think that given the impending climate crisis,it is necessary to focus on how making the code more efficient in order to reduce the required energy consumption but also to make the data center sustainable by using for example renewable energy sources.

Conclusion and Reflections

I was fascinated by the sentence in the assignment specification "*ML will eat Software*" as I had never heard about this future trend and, therefore, driven by curiosity I did some research. According to Billings et al. (2017)[^3], there's a high chance that AI will replace software developers as early as 2040. They state *"Programming trends suggest that software development will undergo a radical change in the future: the combination of machine learning, artificial intelligence, natural language processing, and code generation technologies will improve in such a way that machines, instead of humans, will write most of their own code by 2040"*. I believe that AI and ML technologies will be used more and more in the future. This will certainly affect the figure of the software developer. Indeed, ML is a tool or technique that a software engineer can use to get his/her job done thus reducing the maintenance burden in software industries. However, I think the verb "eat" is too strong. ML will not "eat" the software development, but it will definitely change it. Indeed, although the software development will be aided by ML algorithms, the latter need to be trained. Problem and goal definition, data collection, data preparation, model learning, model deployment and integration, and model management are key components of this process. Therefore, the software developers of the future will find and compose large data sets to train applications to be intelligent, instead of hard coding the desired capabilities. More in general, I think that as automation levels increase, people feel that, in the long run, human skills and abilities may be lost as they are replaced by new technologies. I believe that some jobs will disappear over time, others will change, and new ones will be born. From my point of view, it is not a loss, but a normal mutation caused by the technological development. Indeed, in the past there were jobs that no

longer exist, such as the switchboard operator who connected long-distance calls putting the caller in contact with the recipient and skills that are no longer needed today. I think ML will change the skills of software developers who, with a reshaped skill set, will still be needed in the future.

[^1]: Mahdi, M.N.; Mohamed Zabil, M.H.; Ahmad, A.R.; Ismail, R.; Yusoff, Y.; Cheng, L.K.; Azmi, M.S.B.M.; Natiq, H.; Happala Naidu, H. Software Project Management Using Machine Learning Technique—A Review. Appl. Sci. 2021, 11, 5183.

[^2]: Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, Håkan Grahn, Estimation of energy consumption in machine learning, Journal of Parallel and Distributed Computing, Volume 134, 2019, Pages 75-88,

[^3]: Billings, Jay Jay, Alexander J. McCaskey, Geoffroy R. Vallée and Gregory R. Watson. “Will humans even write code in 2040 and what would that mean for extreme heterogeneity in computing?” ArXiv abs/1712.00676 (2017)