Sorcery in computation

Teis Johannesen

Wizardry has through time often been used as a way to explain indescribable phenomena, even if these were later found to be explainable. In much the same way are programmers, while not masters of black magic but that of language, seen as wizards who in mysterious ways makes actions or products appear from thin air. But even though modern programming languages shares a lot of traits with that of humans, compared to their ancestors, it still requires a great amount of both practice and studying to take full advantage of each languages' complexity and depth. This added with the punctilious nature of programming greatly increases the cost of developing, maintaining and prototyping programs.

The question is then: What can we do to make software development accessible to all people without sacrificing complexity, neutrality, efficiency and ease of learning?

In the exam paper, formulated as a contribution to a future edition of the Software Studies Lexicon, I want to explore and reflect on what might be the future of software development. The primary concern of the paper will be the idea behind accessibility of programming or other tools in the creation of digital artifacts. This will include an analysis of the development of programming languages, and which direction it is going, but also how the increased focus on machine learning and AI might contribute new tools in the field of software development.

Language

To get a better idea of the intricacy in programming languages I would like to examine its evolution. A lot is done towards the translation of ideas of digital artifacts unto its creation. Tools have been made that provide users with libraries, graphical interfaces as well as en- and

decoders. While these make it easier for people with no background to develop these kind of artifacts, they don't eliminate the need for learning each tool/program. By comparing the complexity and efficiency of different tools from different ages we might get an idea of the general direction that these have been and are going. My general hypothesis behind its direction is that of Natural Language Processing (NLP). The definition of NLP, that professor Elizabeth D. Liddy offer, is:

> Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.[1]

One of these applications that Liddy mention I postulate could be that of software development. By using a decoder with full Natural Language Understanding (NLU) it would acquire the ability to retrieve data or values from a given text written in natural 'human' language. With this data it would be able to determine on its own which actions should be executed to achieve the desired outcome. Is this achievable or even desirable to have this kind of automation in a process like this? To explore more of this idea I want to bring Liddy's work and reflections into the topic of an artificial programmer.

Neutrality

My next topic in the final paper would be that of biasing in programming tools. This would concern the idea of neutrality in software development. If one were to strive for innovation or uniqueness, would using tools then be a contradiction to this goal? For this topic I wanted to investigate how each programming language and tool might have developed efficiency in creating certain programs or products in certain scenarios. While this makes each tool good by its own rules, it also limits itself in others that might be beyond the awareness of the tools creators. In others words we are provided with

powerful tools in every field imagined but none in every field that has yet to be discovered.

While the decision to concretize each tool certainly has an understandable justification, I think that awareness of said biases contribute towards a healthy amount of critique and innovation in it's applied field. A solution to this might be the aforementioned NLU to some degree eliminate all biasing but the users own, or it might be one of awareness. The latter seems easy to carry out but might require acknowledgement from both the user as well as creators or contributors behind the tools. This is also something that I want to explore in the final paper.

The Artificial Programmer

My final topic for the paper is that of artificial intelligence. By artifical intelligence I mean computers who are able to do certain actions 'requiring intelligence' by themselves. The result of this might, or might not, be predictable or comparable to the way a human would solve the given problem. My particular interest within this field is the use of artificial neural networks(ANNs) and evolutionary computation. Both of these contribute to the field of machine learning to automate or innovate in its applied study by using some level of intelligence, whether or not this intelligence might be real or simulated. I see this as a inevitable future stage of software development to let AI contribute or even completely take over the profession of a programmer. But is this a good thing and why would we even want this? Stuart Russell and Peter Norvig might provide us two great explanations to why this could be the case:

> [...] the designers cannot anticipate all changes over time; a program designed to predict tomorrow's stock market prices must learn to adapt when conditions change from boom to bust.[2]

and:

[...] sometimes human programmers have no idea how to program a solution themselves. For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task…[2]

These reasons would provide a program a great amount of complexity while still keeping the actual programming to be dealt with by the computer itself. By providing rules and learning data we might, at some point, develop a universal tool. One that not only understands how to create near-perfect programs, but will also learn and adapt the more it is used. Even in the event that one might wanna create something original, the AI then know what is not often done and could become an adviser to the user.

On the topic of AI I also wanted to make some reflections upon the neutrality that might come with this. Would a computers neutrality, compared to ours, might seem too extreme or even be a desired trait? And even if a computer was to have a complete understanding of humans and designing things for us, would we then miss the incompleteness and imperfections that is connected with products made by humans?

Notes

1. Liddy, E.D. (2001). Natural Language Processing. In Encyclopedia of Library and Information Science. 2nd ed. NY. Marcel Decker, Inc.

2. Russell, S. and Norvig, P. (2009). Artificial Intelligence, A Modern Approach. 3rd ed. Pearson Australia Pty Ltd, p.693.