## Augmented Large Language Models for Software Engineering

**The rise of new advanced generative AI models like ChatGPT has been a game changer for how companies work, presenting a large range of opportunities. Programming and software development is one of the areas where high hopes have emerged, with prominent CEOs stating that English will be the new programming language. However, beyond the hype, a bunch of challenges have to be addressed to actually make these models useful - some technical, others regarding how we interact with this technology.**

Along with every new groundbreaking technology comes hopes for productivity gains and economic growth - as seen for example during the dot-com boom. The same is now happening with *generative AI*, meaning artificial intelligence that can generate text, images and more in response to *prompts*. At the forefront of this development are companies such as OpenAI, Microsoft, Google and Meta with their *large language models* (LLMs) GPT, Gemini and LLaMa. These are trained on billions of words and terabytes of data, to be able to write text and computer code similarly to humans.

For software programmers, there are already many tools available tailored to the specific use case of writing code. These can suggest what to write and answer questions, and some companies even claim to have developed full-fletched "AI software engineers". These fascinating capabilities have contributed to lots of hype, and alongside also fears of mass job replacement. But do they actually work well enough to be helpful?

Surveying software engineers at an innovative start-up company, we found that these tools can be a double-edged sword. As the models can't actually reason logically like a human, they play a very advanced guessing game based on statistics and probabilities – sometimes they get it right, other times it's plainly wrong and misleading. Though there are some ways of bringing their answers closer in line with expectations.

As models generate their answer based on an input (the prompt), making sure to give them the right instructions and providing them access to relevant information are useful techniques to improve results. To try this, we developed a prototype for a chatbot with the ability to search through and access the company's source code based on software engineers' questions.

Testing the chatbot, we found that these techniques can contribute to better performance of such AI tools. They can help programmers work faster as they don't need to copy and paste their code into a chatbot as they need to with ChatGPT. Though, it is still important to think critically about what tasks to give to a chatbot, how to define them, and whether to trust the output. Giving generative AI models access to more information may improve the output, but does not magically grant the ability of logical reasoning. This makes us believe that AI can augment humans, but is far away from replacing us.

**Generative AI** – Artificial intelligence with the ability to produce content.

**Large Language Model, LLM** – The AI models powering for example ChatGPT.

**Prompt** – The instructions you give to the LLM.

*Authors: Fabian Rosén and Moritz Rübner*