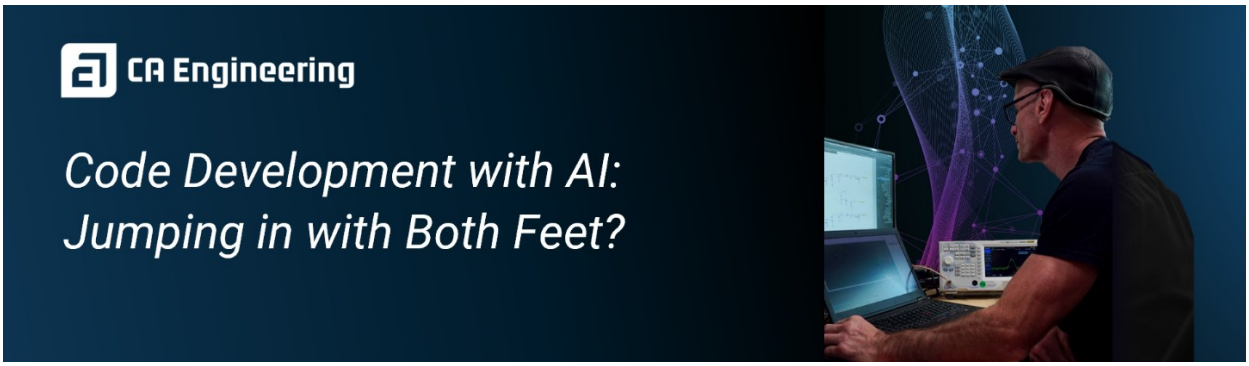

Jon Rhees, Senior Engineer
October 10, 2024

As a youth, I would often travel to Lake Powell (a reservoir along the Colorado river) to cliff dive. Although the sheer drop-off of the surrounding cliffs and the immense depth of the water typically meant I could expect a safe and predictable (and fun) experience, I could not simply assume that—it was critical to check the dive area before making any jump. In other words, 'testing the water' (so to speak) before 'jumping in with both feet' was a potentially life-saving exercise.

Steep Cliffs: The Sudden Rise of Generative AI

The last few years have seen dramatic advancements in the field of Generative AI. Using Large Language Models (LLMs) with *billions* of parameters have created AI that can digest increasingly complex natural language prompts and generate convincingly real-sounding (as opposed to artificial) responses.

Code development, which is based on strict language models, enjoys perhaps the most benefit from these new LLMs. New AI-based code development tools are appearing on an almost-*daily* basis, promising to not only *assist* in the development of code, but in some cases provide the *entire* development stack – from project management, to architecture, to implementation, to testing. These advancements, while often disconcerting to software developers, are simultaneously tempting to businesses. After all, the potential cost and time savings could be enormous!

So, what is holding you back? Should you 'jump in with both feet' into AI code development?

At CA Engineering, we are keeping abreast of developments in AI as they evolve, and believe that, at this point, a measured approach to the use of AI code development is best.

Hidden Rocks: Understanding the Advantages and Risks of Using Generative AI for Code Development

We believe generative AI should be considered and used as a *tool*, and not as a solution in and of itself. We use tools, like IDEs, compilers, debuggers, linters and source-code managers, in the software development industry every day. For every tool, understanding its capabilities, limitations, risks, and suitable application is critical. Generative AI, while rapidly evolving in capability, is no different.

Let's take a look at some of the advantages, limitations, and risks associated with these new tools:

Advantages

Generative AI can revolutionize the way developers approach coding by automating tedious tasks and providing powerful coding assistance. Some key benefits include:

1. **Increased Productivity**: AI can generate boilerplate code, offer suggestions, or auto-complete complex structures, allowing developers to focus on higher-level logic and design. This can drastically speed up the development process, especially for repetitive tasks.
2. **Error Detection**: Generative AI can assist in identifying bugs, optimizing code, and suggesting improvements based on vast datasets of best practices, leading to better code quality.
3. **Learning Aid**: For new developers or teams learning new languages or frameworks, AI tools can serve as interactive, on-demand guides, making coding more accessible and providing real-time feedback.
4. **Rapid Prototyping**: AI can assist in quickly building prototypes by generating large portions of code based on minimal inputs. This is particularly useful for startups and agile teams that need to iterate quickly.

Limitations

Generative AI models for coding are essentially a sum of their parts—they are an expression of the data set upon which they were trained. This leads to the following caveats:

1. **Limited Training Data**: While the data sets used to train prevailing LLMs are enormous, they are *not* infinite. Typically, there is a 'cutoff' point—data newer than a specific date is not included. Also, the training data consists of publicly available code scraped from the internet. This means that the output capability (the generated code) will never (in theory) exceed the input quality (the training dataset).
2. **Customization**: The models and datasets used for training may not fully align with the specific requirements or preferences of individual projects or development teams. Customizing (at present) generally involves training a new model, which is not practical.

Quality Risks

Generative AI in code development presents several risks, particularly because it is still an **emergent technology**. These risks include:

1. **Code Quality and Accuracy**: While generative AI can write code, it does not always guarantee that the output will be efficient, secure, or bug-free. It often lacks the nuanced understanding of context that human developers possess, potentially leading to suboptimal or error-prone code. Blindly relying on AI-generated code can introduce hidden issues that might only surface later in production.
2. **Lack of Accountability**: AI-generated code may work initially, but if errors occur later, it can be hard to trace the root cause or understand the reasoning behind certain code blocks. This lack of transparency can lead to maintenance challenges.
3. **Over-reliance**: Developers may become overly dependent on AI tools, diminishing their problem-solving abilities and understanding of the codebase. This could eventually lead to a decrease in core programming skills.

Privacy and Security Risks

Generative AI poses particular risks in terms of **privacy** and **security**:

1. **Data Exposure**: AI tools often rely on massive datasets for training, which can include proprietary or sensitive code. There is a risk that confidential information or trade secrets may inadvertently be used in training models, raising concerns over **intellectual property theft** and **data privacy**. Additionally, AI tools sometimes generate code based on patterns they've encountered, and this could unintentionally leak sensitive snippets or expose vulnerabilities.

2. **Insecure Code Generation**: AI-generated code may not always follow best practices for security, leading to potential vulnerabilities, such as SQL injection, buffer overflows, or inadequate input validation. If developers do not carefully review AI-generated code, insecure practices might slip into production, creating exploitable weaknesses in the system.

3. **Data Collection**: Many AI-based development tools collect user data to improve their models. This presents privacy concerns, especially if sensitive or proprietary code is processed and stored without proper oversight or safeguards. Even anonymized data can sometimes be de-anonymized, posing a risk to organizations relying on AI tools.

Deep Waters: The Evolving Landscape of Generative AI

One thing is for certain: the face of AI is in a rapid state of flux. What is state-of-the-art today will be obsolete a month or two from now. Many of the limitations and risks listed above will be mitigated or possibly eliminated in the future.

For example, newer code-assist tools are becoming available that allow LLMs to be custom-trained on local customer data and code, generating code that is better suited to specific projects, free from accountability concerns of public data sets. Hardware is also evolving, making it more practical to run LLMs on local servers versus utilizing cloud services, helping to ensure privacy.

Jumping In: A Measured Approach

Generative AI offers immense potential for accelerating code development, improving efficiency, and aiding both beginners and experts. However, its emergent nature also brings significant risks, especially around privacy, security, and code quality. A measured approach today can be a great way to ‘dive in’ and start taking advantage of this new technology in a responsible way, stay balanced while plunging into this new AI revolution, while avoiding any ‘hidden’ rocks that lie beneath the surface.

"Adopting a measured approach allows for a responsible entry into the AI revolution, enabling access to the benefits that drive innovation and accelerate time to market while maintaining balance and steering clear of potential ‘hidden rocks’ beneath the surface."

We Can Help

CA Engineering