Introducing Supermaven, the first code completion tool with a 300,000-token context window

Jacob Jackson
Thu Feb 22 2024

Tool	Frames	Latency (ms)
Supermaven	15	250
Copilot	47	783
Codeium	53	883
Tabnine	50	833
Cursor	113	1,883

See below for details.

Code completion has come a long way. When I created TabNine in 2018, the first tool to use deep learning for code completion, there were few developer tools that used AI. The models were small, the capabilities were limited, and people were skeptical that the tools were useful enough to be worth paying for.

Since then, AI coding tools have been widely adopted: GitHub recently announced that its tool Copilot had reached $100 million in annual recurring revenue. Millions of developers now use AI tools every day. This adoption is driven by the deployment of large models with tens of billions of parameters that are much more useful than the models of 2018.

Given that code completion has "gone mainstream", given the growing size of the largest language models, and given the enormous costs involved in training the largest models, it raises the question: is there still room for small startups? Has code completion become the exclusive province of large players like Microsoft or Google?

At Supermaven we believe the question is not yet settled. The reason is the high serving costs of large models. While a model like GPT-4 offers unmatched suggestion quality, it's impossible to run on every keystroke (unless you charge users $1,000/month). Today, the deployment of an AI coding tool is limited just as much by the cost of the GPUs to run the tool as by the cost to develop the tool itself.

That's why Copilot has made efforts to reduce costs by limiting the contexts in which they offer suggestions.

The high cost of serving a code completion tool levels the playing field: it means that everyone, whether a small startup or Microsoft, must use a small model to remain profitable. The best product is the one which uses a small model most effectively.

So how is Supermaven different?

300,000-token context window

The context window of a code completion model determines how much of your code it can take into account when making its suggestions. A long context window is crucial to delivering good completions - that's why Copilot recently expanded its context window to 8,192 tokens.

At Supermaven we've developed and trained from scratch a new neural network architecture which is more efficient than a Transformer (the current standard architecture) at integrating information across a long context window. This allows us to offer a 300,000-token context window while keeping the cost and latency the same as a Transformer with a 4,000-token context window.

As a user, that means that after Supermaven spends 10-20 seconds processing your repository, it will become familiar with your APIs and the unique conventions of your codebase. That fixes one of the most common complaints about Copilot: that while it's great at working with well-known APIs and libraries that have good coverage in the training data, it struggles with the large, idiosyncratic codebases that you get when using it for real work.

Here's a simple example of the difference that a 300,000-token context window makes. In this example, I ran Supermaven and Copilot on the same code and asked them to generate a function called `doubleIndentation` taking a value of type `Indentation`. Because the `Indentation` struct isn't defined or used anywhere in the file, Copilot doesn't have any information about it in context, so it generates code that doesn't compile. Supermaven generates correct code because its context window is large enough to fit the entire 50,000-token repository.

Speed

Supermaven is fast: we've built custom infrastructure to keep it responsive while working with the long prompts that come with large codebases.

I designed a simple test to compare Supermaven to the competition. To ensure each tool has the prompt in cache, I started with a context where every tool agrees the most likely completion is `for (let i`. Then I typed `for (let j`, forcing each tool to generate a new completion, and measured the number of frames between the appearance of 'j' on the screen and the appearance of the completion. Here are the results:

Tool	Frames	Latency (ms)
Supermaven	15	250
Copilot	47	783
Codeium	53	883
Tabnine	50	833
Cursor	113	1,883

0:00

Trained on edit sequences, not just files

Unlike most code assistants, Supermaven doesn't see your code as a sequence of files. Instead, Supermaven sees the sequence of edits you've made to your codebase - similar to what you'd see if you ran `git diff`. This helps Supermaven quickly understand what you're trying to accomplish, and it's great for refactoring. Here are some examples:

0:00

Try it now

Download Supermaven to try it for yourself. We're excited to see what you think of it.

If you don't use VS Code, follow us on Twitter to find out when Supermaven is ready for your preferred editor.