Nicholas Hamilton

Cognitive Generative AI Externship

June 2 2025

AI Text Completion

For this assignment, I imported my API key from a file that git is not tracking, and followed OpenAI documentation for using their API. I created a simple menu of items, and used basic functional programming to structure my file. Within response, I am able to artificially alter the temperature and max tokens. I pass these as variables so that the user can alter them from my CLI menu.

For creativity, I prompted the model to continue a sentence: “The train stopped in the middle of the desert. No one knew why.” The model output a short vignette involving a mysterious figure in red dust and silent passengers leaning out windows. It was coherent and had a decent sense of tone. It didn’t try to explain too much, which made the suspense work. When I increased the temperature to 1.0, the output got more surreal—at one point the train was described as dreaming. Creative, but also incoherent. The higher temperature led to novelty, but not always in a useful direction.

For summarization, I gave the model a few paragraphs about the Industrial Revolution and asked it to summarize. With temperature lowered to 0.3, it returned a textbook-style sentence: “The Industrial Revolution, starting in the 18th century, led to major advancements in manufacturing, agriculture, and transport, transforming economies and societies.” This was accurate, terse, and mostly fine. But increasing the temperature caused it to interpolate—adding commentary about worker suffering that wasn’t in the source. The model doesn’t seem to differentiate well between summary and interpretation unless tightly constrained.

For explanation tasks, I tried “Explain recursion like I’m five.” The model gave a decent analogy involving mirrors and mentioned that recursion means a function calling itself until the problem is small enough to solve. It was close to understandable for a beginner, though probably still too abstract for an actual five-year-old. Still, no technical errors, and it captured the essence of the concept. On a more basic prompt—“How do you calculate the area of a circle?”—it returned “The area of a circle is calculated using the formula $A = \pi r^2$, where r is the radius.” This was correct and clear. The model didn’t deviate or overcomplicate the response. These simple factual queries work reliably, especially with low temperature (0.2 or 0.3).

One of the more open-ended prompts I tried was: “Write a short dialogue between a robot and a tree.” At temperature 0.8, the model leaned poetic. The robot asked, “Do you ever move?” and the tree replied, “Only when the wind remembers me.” They exchanged a few lines about agency and control. It was stylized and coherent, and didn’t feel forced. This kind of output is where higher temperature actually helped—it injected a distinct voice into both characters.

Across all tests, the model was most reliable on shallow tasks: summarization, paraphrasing, and simple definitions. When prompted clearly and with constrained scope, it performed predictably. For creative or abstract tasks, it could produce compelling output but tended to get weirder as the temperature increased. The trade-off was always novelty versus coherence. Some responses at high temperature lost logical structure entirely, especially when asked to improvise narratives.

Its weakest area was multi-step reasoning or anything requiring internal consistency across several sentences. Even when the response began logically, later sentences could contradict earlier ones. It also tended to generalize when prompts were vague, often defaulting to bland or generic language. That makes it functional for quick drafts, but I would not use it for anything requiring precision.

If I wanted to improve performance, I'd try prompt engineering first—clarifying task boundaries, giving examples, or requesting specific formats. Adding instructions like "Be concise" or "List three bullet points" helped sometimes. Longer-term, filtering post-output for hallucinations or letting users specify tone (e.g., speculative vs. factual) could reduce errors. But fundamentally, the model works best when the user already knows what a good answer should look like. I have recently gotten into self hosting LLMs, and I feel that training those for small things has given me the best results.