PL
TRANSFORMING ERP
SAP ECC & S/4HANA
AGILE INDOOR CONNECTIVITY WITH 5G, DAS & WIFI 6/7 FROM SOLiD
STRENGTHENING LOYALTY & ENGAGEMENTS WITH TransUnion
THE RETAIL ROBOTICS REVOLUTION WITH BADGER TECHNOLOGIES
5-STEP ROADMAP FOR THE AI AGE
AI & EQ FOR TECH HIRING
SUSTAINABLE NETWORK DEPLOYMENTS
REINVENTING U.S. MVNOS
STREAMLINING COLLECTIONS
OPTIMIZING LLM CONFIGURATION
OPERATIONAL AGILITY

www.pipelinepub.com
Volume 21, Issue 10

Right LLM and Configuration for Your Operations Agent

By: Mark Cummings, Ph.D.

When planning for an AI agent to achieve operational agility and efficiency, it may be tempting to base it on the standard online version of the most leading-edge Co-Pilot or GenAI system. Doing so can be a mistake. It is important to pick an LLM and its configuration that is the best fit for your agent. A plan for monitoring agent performance and AI technology evolution is important. Because LLM technology and business practices are evolving so fast, you need to have a way to determine when to update and have the necessary data documented and in place to do it efficiently.

To start the process, it is good to review the application requirements. The requirements posed as questions below are particularly important for decisions around developing your agent.

Interaction Between Requirements and GenAI Selection

The first question should be, do you need GenAI or can you use conventional programming? For example, if your application is essentially based on a fixed well structured decision tree (series of: if; then; else decisions), conventional programming may be the best development technique. If that is the direction chosen, attention can turn to whether and how GenAI might assist with code development.

If the application has some complexity, pattern recognition tasks, dynamic nature, etc., and a GenAI agent approach is being explored, the next set of questions involves privacy and security. Does your agent handle proprietary data? If so, many online GenAI systems collect user input and use it in training. This practice can create what has come to be called an IP Leakage. That is an Intellectual Property Leakage. In addition, there are emerging security vulnerabilities involving online AI. For example, OpenAI has been breached 1,000 times. Other online AIs having been breached with Microsoft Co-Pilot having a serious vulnerability.

If privacy and security are important requirements, basing your agent on a local GenAI implementation may be a better approach. There are two related questions with a similar answer. The first concerns uptime. Online GenAI systems are subject to periods of unavailability.

The second concerns latency. That is how fast your agent has to act. Online systems processing can be fairly fast, but they have two serious latency constraints. The first is propagation delay. That is the time it takes for data to travel from the point of origination to the online system and back to the point of action. The second has to do with congestion. Congestion in the network and in the online system. So, if any of the three are a concern, investigate the option of a local implementation.

Finally, you need to determine the application's sensitivity to the quality of output. An over-simplified approach to quality can be broken down into: probability of hallucinations, and precision of output.

When a GenAI hallucinates, the output appears to be authoritative, well-structured, and well-presented. Unfortunately, it has little or no relationship to reality. The hallucination question is, how sensitive to hallucinations is your application? Are you developing a system that will be an adviser to operations staff that can deal with hallucinations? Will your system include actuators that independently take action? Are the actuators able to filter out hallucinations?

The point is, if your application is sensitive to hallucinations, you need to take steps to shield it. GenAI hallucination is an area of intense study. Progress is being made at a fast pace. At the same time, progress in other areas seems to have an impact. For example, recently, several sources, including OpenAI, said that early reasoning models (frontier LLMs that are designed and trained to do something akin to human reasoning) had a higher rate of hallucination than did previous generations of non-reasoning models. By the time you read this, that may no longer be true.

Suppose reasoning models continue to have a higher probability of hallucination, and your application is sensitive to hallucination. In that case, you may need to choose an LLM that does not have reasoning capability.

Some speculate that hallucinations can occur because inference requests fall outside of what the LLM has been trained on. Also, other aspects of performance can be affected by training data. Thus, you may want to consider how the LLM was trained. Here, the question is, what skills does your LLM need? In the case of LLMs, skills
relate directly to the corpus (set of training data) used to train the LLM. If the skills required are within the normal types of training sets commonly used, this may not be a concern. But if your agent will be operating in a very highly specialized field, or you wish better performance, extra precision, etc., you may wish to do additional training. This extra training is commonly referred to as fine-tuning. Fine- tuning data sets are generally of the order of 1% the size of the full training corpus.

Configuring LLM’s for Your Requirements

Below is not an exhaustive list of configuration choices nor mechanisms for configuration. Here again, things are changing so fast that configuration parameter choices and consequences of various settings will change over time. As GenAI continues to evolve, configuration choices are not likely to disappear. So, the following examples are illustrative.

The platform you choose to support your LLM generally provides control of a number of these external variables.. If you are using an online GenAI service provider’s LLM, that service provider has already chosen the platform. Some service providers will work with you to set up a version of their GenAI system on your premises or on a public Cloud you select. Here again, the platform is generally chosen for you.

If you are running an open-source LLM in-house on a central site or distributed environment, you have to choose such a platform. Choosing such a platform is generally not difficult. The most important consideration is that it supports the LLM you have selected. In an organization, you may wish to standardize on one particular platform for ease of maintenance, training, support, etc. If so, this may limit the alternative LLM options.

One example of such external variables is Quantization. Quantization can impact the quality of GenAI output and the amount of computing resources required. Quantization refers to how many binary digits (quantity of digits) are used to encode (represent) each parameter. In a one trillion parameter model, reducing the quantization from 16 bits to 8 or 4 bits can have a big impact on the amount of computing resources required and the latency (speed of output) delivered. Here, the question is, what level of precision does your application need? Your answer to this question may impact how you need to configure the quantization external variable to achieve the necessary precision and the highest possible efficiency.

Another external variable is Temperature. There is controversy around whether temperature settings can increase the probability of hallucinations. Without a temperature setting, LLMs choose the highest probability symbol in their vocabulary. For online chatting applications, this was found to produce correct inferences, but less interesting ones. Temperature settings were developed to allow LLMs to select from a larger palette of high-probability symbols. You can see something similar in human- written poetry, where ending a phrase with an unexpected word can have a big impact. Making for a more interesting poem. There is disagreement about whether temperature settings can lead to higher probabilities of hallucinations.

What seems clear is that it may be a mistake to take an LLM configured for one purpose (I.e., to support online chatting) and not reconfigure it for use for another purpose (supporting an agent).

Development, Verification and Test

A development process common in semiconductor design is becoming common in GenAI agent development. Semiconductor design uses very specialized languages to describe in detail all the pieces of a semiconductor device, commonly called a chip. Verification is the process of testing this detailed design before it is sent to the fab (fabrication facility), where it is converted into an actual completed chip. Fab processing is very expensive. So, the objective is to avoid having to redo it by trying to find any potential problems in verification. Once the chip has been fabbed, it is tested to make sure it does what it is supposed to do and doesn't do what it is not supposed to do.

This two-step verification and test process is entering the development cycle using agents. A GenAI system is used in an iterative process to evaluate GenAI-created implementations. There may be two separate GenAI systems. One for development and one for verification. They work iteratively. The first develops an implementation, and the second verifies it. Here, the objective is to minimize cost by reducing the time human developers have to work to get the GenAI developed system to the point where it can be implemented. The verification GenAI system has a set of criteria that must be met before declaring verification successfully completed. Until those criteria are met, the verification GenAI system keeps iterating with the GenAI development system. This can go on for many cycles, but relatively short calendar time.

The verified version is presented to staff members who seek to perfect it. Once perfected, it goes through a final test process. Depending on the outcome of the final test process, it may be further iterated.

In large enough organizations, separate groups are responsible for the perfection phase and the final test phase. Separating these functions is valuable because it removes normal human biases.

LLM Monitoring and Updating Plan

Because AI technology is evolving so quickly, it is important to create a plan to evaluate, including ongoing performance of the agent, and emerging GenAI technology improvements in general, and newly released LMMs in particular.

Agents at installation may perform well. Over time, their performance may degrade. Performance may degrade because the conditions that the model was created to deal with have changed, rising user expectations, or technical limitations.

For example, early medical agent implementations found that their AI agents worked well when first installed, but after a while, their performance degraded. Institutions using them reported that they had installed the agents in large part because they anticipated significant cost savings. What they found instead was that they had to add expensive staff to monitor and update the agents.

Industry speculation was that these performance problems occurred because the Context Window of the LLM supporting the agent became full. The context window provides information to the LLM that is important in making the next inference request. Information on each inference request and result is added to the context window. Frontier models keep increasing the size of the context window. Frontier models have also been developing ways of editing the context window. But even so, over time, the window can become full. As LLM technology continues its rapid evolution, this and other technical limitations may fade away.

Only to have new ones appear. Thus, having a plan for monitoring performance is necessary.

Not only is GenAI technology evolving rapidly, but the appearance of the technology is generating rapid changes in many of the underlying businesses seeking to take advantage of it. As the business requirements change, LLM technology evolves, end user expectations rise, or problems appear with existing agents, it will be necessary to consider updating an agent. To do so, it is important to establish a set of criteria for deciding when it makes sense to consider updating an agent. Over time, it may be necessary to update these criteria as the technology changes. But care should be taken to make sure that the criteria are not changed a priori just to justify a decision that has been made for political or other reasons.

To support this expected monitoring of performance and possible updating, documentation should be captured and retained in a way that makes it easily available. This documentation should contain the original requirements, significant experience during development, testing process, and results, and a way to capture the ongoing performance monitoring results. The AI agent monitoring process needs to be combined with the results of regularly evaluating alternative LLMs as they appear. The result needs to be kept in a secure manner that is easily accessible to update or when needed in planning and execution.

Conclusion

Choosing the optimal LLM and configuring it correctly is critical for a successful AI agent implementation. Matching the selection process to the requirements is key. Properly configuring the selected LLM is also very important. Monitoring agent performance after installation and monitoring LLM technology evolution are important steps in maintaining good ongoing operational efficiency.