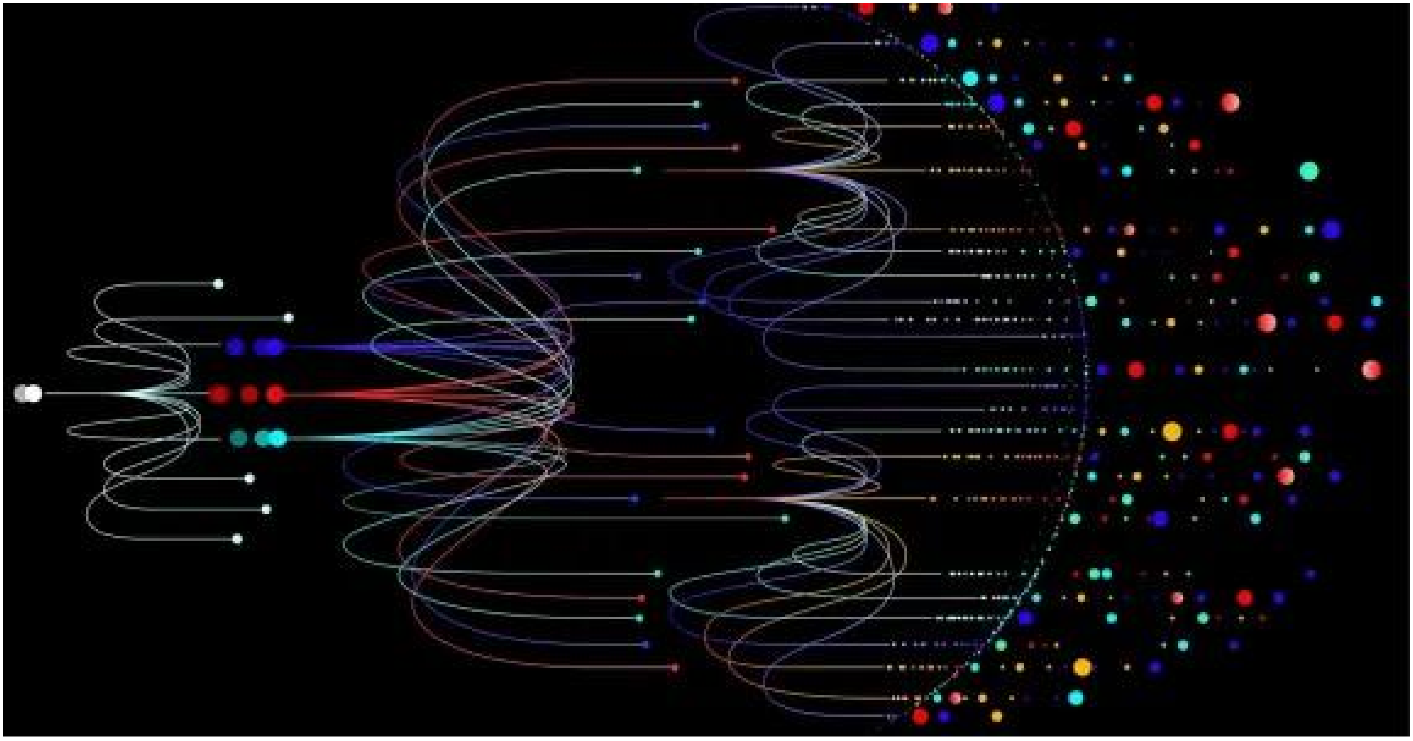

Four months of experimenting with GenAI

25 June 2024, 13:34

Nick Boucart
Marouene Oueslati

Our experience with building LLM-based apps

Is it hard to build a GenAI app? That question was worth an experiment, so we started working with LLM-based apps. Four months later, it's time to share some of our experiences.

Together with Sirris experts Niels Holvoet and Marouene Oueslati we experimented with building LLM (large language model)-based apps. We built some proofs of concept and discovered that the path from PoC to production is not a smooth one. We created a startup researcher who, based on a high-level pitch from a founder, searches for colleagues online and summarises their findings in a report. We built a chatbot to chat with a number of PDFs - possibly the quintessential use case for RAG (retrieval-augmented generation). With crewAI we even built multi-agent applications, which are apps where LLMs 'talk to each other' to solve more complex problems together.

Some observations

- Libraries such as LangChain (a Python library that makes it easy to build an interface with LLMs) and Gradio (a Python library for building web interfaces quickly) make it easy for someone with some Python skills to interact with LLMs.

- By using tools such as Ollama or lmstudio, you can easily work with local LLMs so you don't have to keep buying tokens from OpenAI - and LangChain makes it trivially easy to switch LLMs.
- The effect of seeing a computer system perform tasks that until recently seemed the exclusive province of humans still hasn't worn off.
- A GenAI PoC, however, is far from production-ready.

Some less rosy findings

Although LangChain makes it easy on a technical level to work with different LLMs, they are not interchangeable: prompts that produce good results with one LLM will not necessarily do so with another. A lot of time is spent on prompt engineering, which, in practice, involves a lot of trial & error.

Speaking of trial & error, you naturally regularly test whether the system provides the output you want. Since that output is the result of less trivial tasks (e.g., summarising a text, examining results from Google searches, ...), it turns out to be quite a job to assess those results: on the one hand, it’s great that you can build a computer system able to summarise texts at all, especially with only vague notions of NLP (natural language processing). On the other hand, and with a bit more scepticism, the result is often good only if the questions have been carefully formulated. If you're not so careful, the quality of the answers can fall off very quickly. This makes the system fragile and doesn’t inspire confidence in rolling out these PoCs on a larger scale. As a software engineer, you might then wonder how you can 'unit test' an LLM-based system.

Like many things, experimenting with LLMs costs money. Larger models (GPT-3.5 or GPT-4, open-source models with more than 7B parameters, ...) cannot be run locally unless you have a powerful graphics card. You therefore do this with LLM providers such as OpenAI, or you deploy those models yourself on Runpods, AWS Bedrock or other providers offering sturdy GPUs. This is naturally not free. The cost of experimenting with crewAI and a larger LLM model for a few hours quickly runs into dozens of euros, and that's just to tune the prompts. And we haven’t even dealt with rolling out your solution to multiple users. So it will certainly be important to monitor your costs and ensure that the ROI of your app, despite all this, remains positive.

While smaller models are more sensitive, the larger LLMs, such as GPT-4, handle suboptimal prompts slightly better. Conversely, if you spend more time building a good prompt for some tasks (e.g., few-shot prompting, ...), a smaller LLM can sometimes achieve the correct accuracy at a fraction of the cost.

The term “LLMOps” may already be familiar to you. In any case, there’s at least something we can imagine: all monitoring for quality assurance of answers and inputs, the deployment of the right LLMs (with associated prompts) in real-time and based on the task requested, and all this in a way that balances costs and benefits, ... We expect to hear something about this in the coming months.

Do you have experiences of your own that you would like to share with us? Or are there challenges you want to take on?

Let us know!

Since we see this problem arising for many people, we have started the collective research project LISA (LLM Implementation, Security & Adaptation), together with DistriNet, to work on this matter. Does this sound interesting?

Then be sure to contact us!

Authors

Nick Boucart

Marouene Oueslati