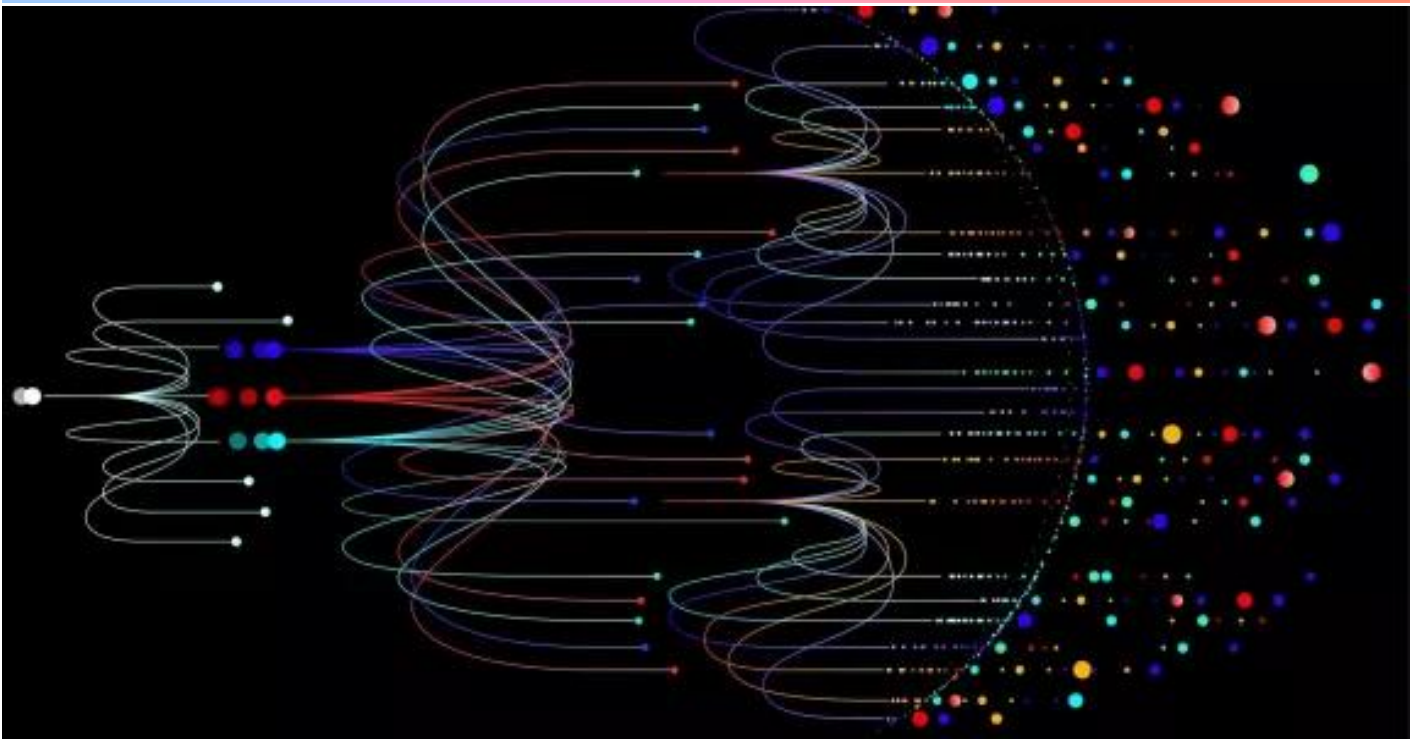




Vier maanden experimenteren met GenAI

25 juni 2024, 13:21

Nick Boucart

Marouene Oueslati

Onze ervaring met LLM-gebaseerde apps bouwen

Is het moeilijk om een GenAI-app te bouwen? Die vraag was een experiment waard: we gingen aan de slag met LLM-gebaseerde apps. Vier maanden later is het tijd om enkele van onze ervaringen te delen.

Samen met Sirris-experts Niels Holvoet en Marouene Oueslati werden experimenten uitgevoerd met het bouwen van LLM (large language model)-gebaseerde apps. We bouwde enkele proofs of concept, en ontdekten dat de weg van PoC tot productie niet over rozen gaat.. We maakten een startup researcher die, op basis van een high-level pitch van een founder, online op zoek gaat naar concullega's en die zijn bevindingen samenvat in een rapportje. We bouwden een chatbot om met een aantal pdf's te chatten, mogelijk wel de quintessential use case voor RAG (retrievel augmented generation). Met crewAI bouwden we zelfs multi-agent-toepassingen (dat zijn apps waarbij LLM's 'met elkaar praten' om samen complexere taken op te lossen).

Enkele observaties

- Libraries als LangChain (een Python-library die het makkelijk maakt om met LLM's een interface te maken) en Gradio (een Python-library om snel web-interfaces te bouwen) maken het eenvoudig voor iemand met een beetje Python-skills om te interageren met LLM's.
- Door tools zoals Ollama of lmstudio te gebruiken, kan je heel eenvoudig met lokale LLM's werken, zodat je niet voortdurend tokens moet kopen bij OpenAI. Langchain maakt het trouwens triviaal om te wisselen van LLM.
- Het blijft verwonderlijk om een computersysteem taken te zien doen die tot voor kort het exclusieve terrein leken van mensen.
- Een GenAI PoC is nog lang niet productieklaar.

Enkele minder rooskleurige bevindingen

Hoewel LangChain het op technisch niveau makkelijk maakt om met verschillende LLM's te werken, zijn die niet onderling inwisselbaar: prompts die goede resultaten opleveren met de ene LLM, doen dat niet noodzakelijk bij een andere LLM. Er kruipt veel tijd in prompt engineering. Dat betekent in de praktijk veel trial & error.

Over trial & error gesproken, je test uiteraard regelmatig of het systeem de gewenste output geeft. Aangezien die output het resultaat is van minder triviale taken (bijv. samenvatten van een tekst, resultaten van Google-searches bekijken, ...), blijkt het een hele opdracht om die resultaten te beoordelen: enerzijds is het überhaupt fantastisch dat je een computersysteem kan bouwen dat teksten kan samenvatten, en dat zelfs met slechts vage noties van NLP (natural language processing). Anderzijds, en met wat meer scepticisme: vaak is het resultaat goed, zolang de vragen zorgvuldig geformuleerd zijn. Wijk je iets verder af, dan duikt de kwaliteit van een antwoord soms razendsnel naar beneden. Dat maakt het systeem fragiel en geeft weinig vertrouwen om deze PoC's uit te rollen op grotere schaal. Als software engineer kan je je dan afvragen hoe je een LLM-gebaseerd systeem kan 'unit-testen'.

Net als zoveel zaken, kost experimenteren met LLM's geld. Grotere modellen (GPT-3.5 of GPT-4, open-source modellen met meer dan 7B parameters, ...) kun je niet lokaal draaien, tenzij je over een zware grafisch kaart beschikt. Dat doe je dus bij LLM-providers zoals OpenAI, of je deployt die modellen zelf op Runpods, AWS Bedrock of andere providers die je stevige GPU's aanbieden. Dit is uiteraard niet gratis. Enkele uren experimenteren met crewAI en een groter LLM-model kost al snel tientallen euro's, en dat is enkel om prompts te tunen. Dan is er nog niet eens gesproken over het uitrollen van je oplossing naar meerdere gebruikers. Het gaat dus zeker zaak zijn om je kosten te monitoren en ervoor te zorgen dat de ROI van je app, ondanks dat, positief blijft.

Terwijl kleinere modellen gevoeliger zijn, kunnen de grotere LLM's (bijv. GPT-4) iets beter om met suboptimale prompts. Omgekeerd geldt ook: als je voor sommige taken meer tijd in het bouwen van een juiste prompt (bijv. few-shot prompting, ...) stopt, dan kan een kleinere LLM soms ook de juiste accuraatheid halen, en dit tegen een fractie van de kost.

Misschien klinkt de term LLMOps u al bekend in de oren? We kunnen er ons in elk geval zeker iets bij voorstellen: alle monitoring voor kwaliteitsbewaking van antwoorden en inputs, het in real-time en op basis van de gevraagde taak inzetten van de juiste LLM's (met bijbehorende prompts), en dit op zo'n manier dat kosten en baten in evenwicht zijn, ... We verwachten daar nog wel wat van te horen in de komende maanden.

Heeft u eigen ervaringen die u met ons wilt delen? Of zijn er uitdagingen die u wilt aangaan?

[Laat het ons weten!](#)

Aangezien we deze problematiek bij velen zien opduiken, [zijn we het collectieve onderzoeksproject LISA \(LLM Implementation, Security & Adaptation\) opgestart](#), samen met DistriNet, om rond dit thema te werken. Interesse in dit project?

[Neem dan zeker contact op!](#)

Auteurs



Nick Boucart



Marouene Oueslati