Responsible LLMOps: Integrating Responsible AI practices into LLMOps

Debmalya Biswas, Dipta Chakraborty, Bhargav Mitra[1,*]

[1]Wipro AI, Switzerland

Abstract

While we see growing adoption of both LLMOps and Responsible AI practices in Gen AI implementations, the discussions are often occurring in different communities with a big disconnect between the checklists and their downstream implementation / tooling. In this paper, we aim to bring the two frameworks together with a unified approach to deploying use-cases in a scalable and responsible fashion. We highlight LLM specific issues, deep dive into the relevant LLMOps architectural patterns, and propose solution approaches to establish an integrated LLMOps platform governed by Responsible AI practices.

Keywords

Responsible AI, LLMOps, Large Language Models (LLMs), Generative AI (Gen AI), Data Quality

1. Introduction

We see growing excitement about Large Language Models (LLMs) and their potential to disrupt enterprise use-cases. However, this exploration is mostly limited to Proof-of-Concepts (POCs) today. To truly scale Gen AI use-cases in the enterprise, there is a need to establish a scalable LLM platform with LLMOps capabilities and the right level of LLM Governance.

In an enterprise context, scaling AI/ML use-cases requires enabling MLOps with the right tooling and frameworks. MLOPs [1] is that framework by which one can manage end-to-end ML lifecycle at scale by enabling technology for the development, deployment, monitoring, and ongoing management of ML models. However, LLMOps (MLOps for LLMs) is very different from MLOPs and it poses many challenges which are difficult to address by MLOPs such as:

- Unstructured data: Supervised (Predictive) ML primarily deals with structured data, in the form of labeled relational data, time series data, etc. on which models are trained. In the case of LLMs, we are mostly dealing with unstructured data, documents, multi-modal data consisting of text, images, audio, video, files.
- Pre-trained foundational LLMs: Instead of training ML models from scratch, the most prevalent scenario is to fine-tune pre-trained foundational LLMs trained on a large corpus of generic data.
- The generative nature of LLMs implies that new content is generated in real-time based on user prompts/responses. This leads to the potential of user inputs being used as training data and guardrails to prevent the generated responses from hallucinating.
- (Human) Feedback loops are an essential part of the training (and continuous improvement) of LLMs to improve the quality of the responses. Reinforcement Learning helps to perform this improvement in a targeted fashion by leveraging a Rewards strategy, e.g., Proximal Policy Optimization, to fine-tune the responses.

In parallel, we also see heightened emphasis on deploying LLMs responsibly [2] as there have been many cases of LLMs providing toxic, biased, sexual, and violent content. LLMs have also been found to hallucinate - providing biased and misguided responses. To mitigate these issues, it is of utmost importance that we train and deploy LLMs responsibly.

The rest of the paper is organized as follows: In Section 2, we introduce LLMOps and deep dive into the relevant LLMOps architectural patterns. We consider the different Responsible AI dimensions in Section 3 in terms of Data Quality, Reproducibility, Explainability and Data Privacy to enable a well governed LLM Platform. We highlight LLM specific issues and propose solution approaches to establish an integrated LLMOps platform governed by Responsible AI practices - leading to the establishment of a Gen AI CoE delivering strategic enterprise use-cases at scale. Section 4 concludes the paper and provides some directions for future work.

2. LLMOps Architecture Patterns

Gen AI solutions are varied in scope, and we expect them to enter the enterprise landscape via different applications / platforms. For instance, this can be via direct usage of a LLM based Application, e.g., ChatGPT; LLMs embedded within a SaaS product or enterprise platform, e.g., Salesforce, ServiceNow; or a foundational model fine-tuned with enterprise data for strategic use-cases. So as a first step, we identify and outline the five most prevalent Gen AI architectural patterns today.

From a Responsible AI perspective, we then identify the key challenges in integrating Responsible AI dimensions (e.g., Reliability, Explainability, Privacy and Security) into the outlined Gen AI architectural patterns. For instance, explainability becomes more challenging given the large corpus of (generic) training data involved in the case of LLMs. Novel privacy challenges arise given that user inputs can potentially be used as training data to fine-tune the LLMs.

Hallucinations remain the key issue to be addressed given the generative nature of LLMs. To address the above challenges, we propose a consolidated Responsible AI framework for LLM platforms in this article, taking into consideration the best practices and design patterns to integrate necessary governance and guardrails at various stages of the LLMOps pipeline.

Woodstock'22: Symposium on the irreproducible science, June 07–11, 2022, Woodstock, NY

*All authors contributed equally.

firstname.lastname@wipro.com (D. B. D. C. B. Mitra)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

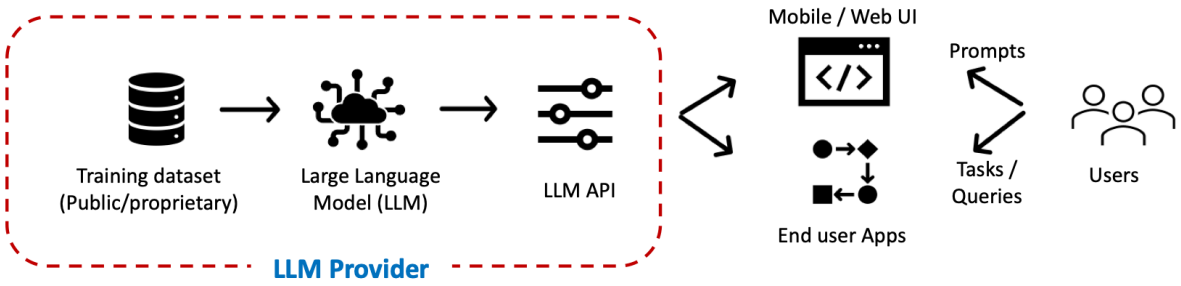

Figure 1: LLM APIs

Black-box LLM APIs

This is the classic ChatGPT scenario, where we have black-box access to a LLM API/UI. Similar LLM APIs can be considered for other Natural Language Processing (NLP) core tasks, e.g., Knowledge Retrieval, Summarization, Auto-Correct, Translation, Natural Language Generation (NLG).

Prompts are the primary interaction mechanism here and can encompass user queries and tasks. Prompts refer to adapting the user input, providing the right context and guidance to the LLM API - to maximize the chances of getting the 'right' response. It has led to the rise of Prompt Engineering as a professional discipline, where prompt engineers systematically perform trials, recording their findings, to arrive at the 'right' prompt to elicit the 'best' response.

Embedded LLM Apps

In this section, we talk about LLMs embedded within enterprise platforms, such as, Salesforce, SAP, ServiceNow; or available as ready-to-use enterprise apps available on the App Stores of LLM providers, such as, Open AI.

Enterprise LLM Apps have the potential to accelerate LLM adoption by providing an enterprise ready solution. However, the same caution needs to be exercised as you would do before using a 3rd party ML model - validate LLM/training data ownership, IP, liability clauses.

Data ownership: Data is critical for Supervised AI/ML systems, esp. so for LLMs which are often trained on public datasets, whose data usage rights for AI/ML training are not well defined and can evolve in future. For example, Reddit recently announced that it will start charging for Enterprise AI/ML models learning from its extremely human archives.

Given this, negotiation of ownership issues around not only training data, but input data, output data, and other generated data is critical. On the other hand, it is also important to understand / assess how the Enterprise App Provider will be using the data received / generated as a result of its interactions with the users.

LLM Fine-tuning / Domain specific SLMs

LLMs are generic in nature. To realize the full potential of LLMs for Enterprises, they need to be contextualized with enterprise knowledge captured in terms of documents, wikis, business processes, etc. This contextualization is achieved in most cases by fine-tuning a Large Language Model (LLM), with enterprise data, creating a domain specific Small Language Model (SLM).

Fine-tuning entails taking a pre-trained Large Language Model (LLM) and retraining it with (smaller) enterprise data. Technically, this implies updating the weights of the last layer(s) of the trained neural network to reflect the enterprise data and task. Given this, access to the base model weights is needed to perform fine-tuning, which is not possible for closed models, e.g., ChatGPT. This is where open-source pre-trained LLMs come to the rescue, e.g., Meta AI, who recently open sourced their LLaMA LLM. The Stanford Alpaca project showed that it is possible to fine-tune LLaMA for $600 to a model performance comparable with ChatGPT. So, fine-tuning a LLM does not necessarily need to be complex or expensive.

Retrieval Augmented Generation (RAG)

Fine-tuning is a computationally intensive process. RAG provides a viable alternative to fine-tuning by providing additional context with the prompts - grounding the retrieval / responses to the given context. This can be in the form of a set of documents that are first retrieved using an indexed document or vector search, and then provided as context with the prompts to limit the responses. Most LLM Platforms allow today allow prompts to be relatively, so it is possible to embed this enterprise context as part of the prompt.

AI Agents - LLM Orchestration

This is the future where enterprises will be able to develop new Enterprise AI Agents by orchestrating / composing [3] multiple existing AI Apps. The discussion around ChatGPT has evolved into AutoGPT. While ChatGPT is primarily a Chatbot that can generate text responses, AutoGPT is a more powerful AI Agent that can execute complex tasks, e.g., make a sale, plan a trip, make a flight booking, book a contractor to do a house job, order a pizza. LangChain is a good example of a mature framework today to compose LLMs.

AI Agents [4] follow a long history of research around Autonomous Agents, especially, Goal oriented Agents [5]. A high-level approach to solving such complex tasks involves: (a) decomposition of the given complex task into (a hierarchy or workflow of) simple tasks, followed by (b) composition of agents able to execute the simple(r) tasks. This can be achieved in a dynamic or static manner. In the dynamic approach, given a complex user task, the system comes up with a plan to fulfill the request depending on the capabilities of available agents at run-time. In the static approach, given a set of agents, composite agents are defined manually at design-time combining their capabilities.

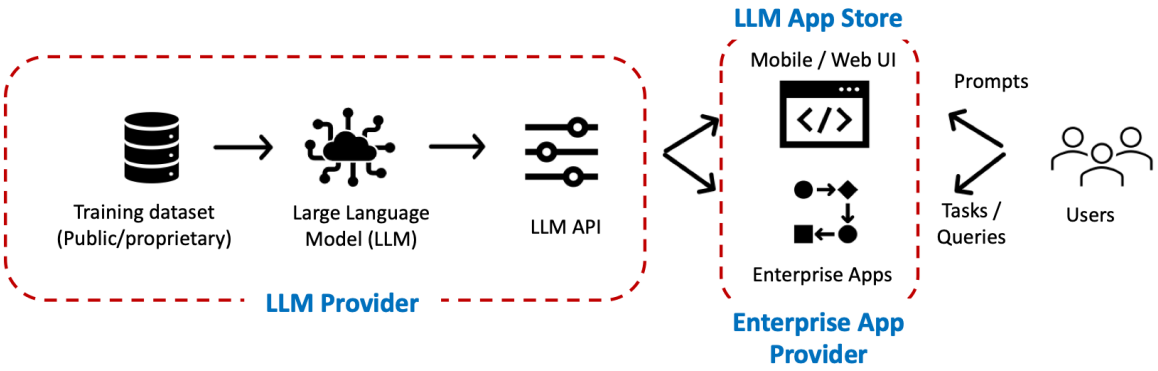

Figure 2: LLM apps embedded within Enterprise Apps / Platforms

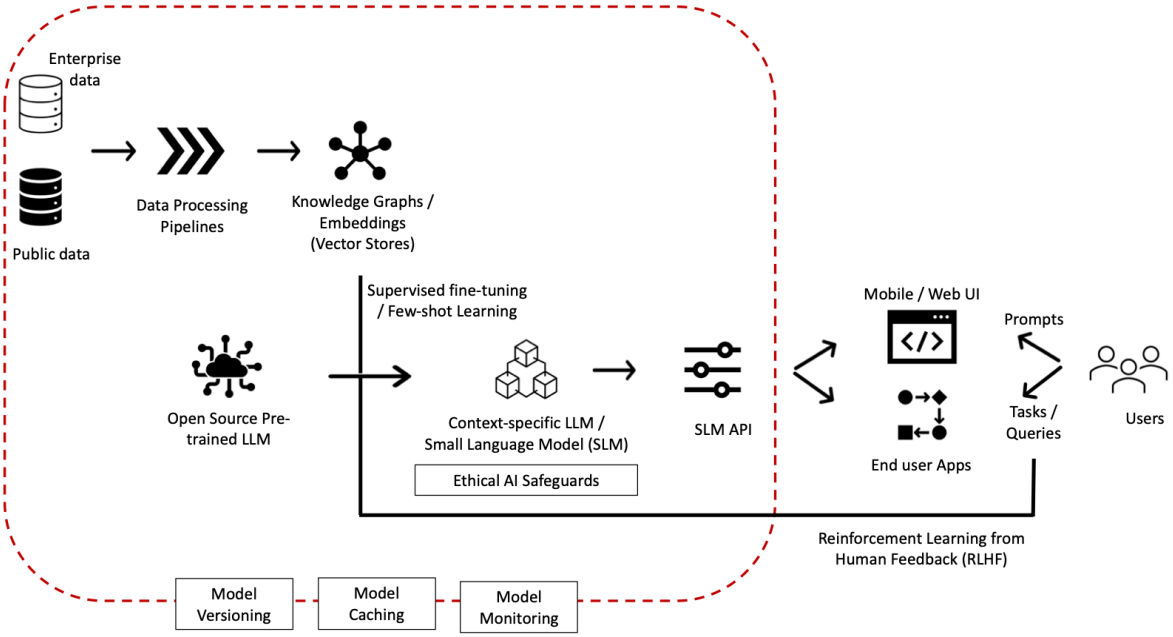

Figure 3: LLM fine-tuning with Enterprise data

3. Responsible AI Framework for LLMs

The growing adoption of Generative AI, esp. LLMs, has reignited the discussion around AI Regulations to ensure that AI/ML systems are responsibly trained and deployed. Unfortunately, this effort is complicated by different governmental organizations and regulatory bodies releasing their own guidelines and policies with little to no agreement on the definition of terms.

The specification language is also (sometimes intentionally) kept at such an important level that it is difficult to map them to an implementation / enforcement mechanism. For example, the EU AI Act mandates a different set of dos and don'ts depending on the 'risk level' of an AI application. However, quantifying the risk level of an AI application is

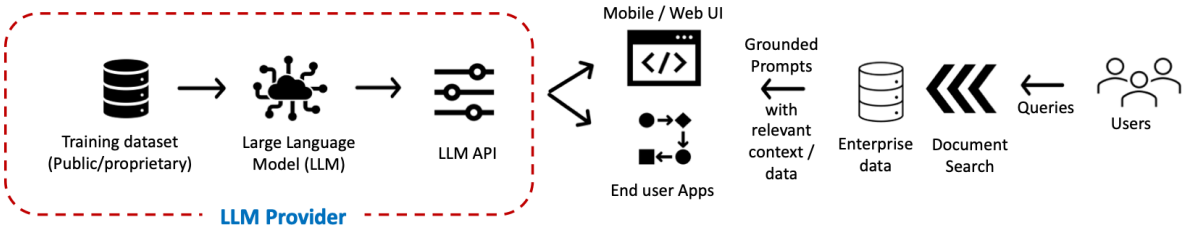

Figure 4: Retrieval Augmented Generation (RAG)

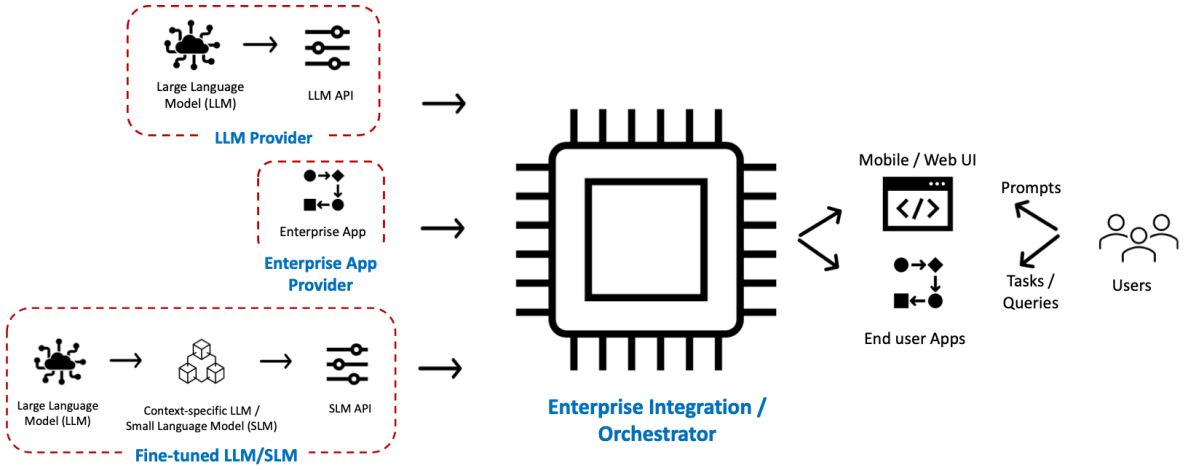

Figure 5: AI Agents - LLM Orchestration

Responsible AI Dimensions	Factors	LLM APIs	Fine-tuned LLMs	LLMs with RAG	Orchestrating LLMs
Reliability	Data Consistency	Adherence to the consistency of data during prompting	The training data should be consistent and balanced	The data should be aligned with the prompting and be consistent	Adherence to the consistency of data during prompting and while passing to other models
	Bias/Fairness	Prebuilt LLMs can perpetuate and amplify harmful biases present in the training data.	Fine-tuning the LLMs with unbiased data reduces the chance of unfair responses	RAGs with unbiased data reduces the chance of unfair responses	Chances of unfair and biased responses can get amplified by using multiple LLMs, but can be reduced by using prompts that contain unbiased data
	Hallucination	Hallucinations are likely as the model gives responses based on large training data	Reduced to some extent as the model is re-trained with curated enterprise data	Reduced to a significant extent by limiting the space of the generated responses	Hallucination likelihood is amplified as a result of using multiple LLMs
	Accountability	Human should be in the loop while training the LLMs or during build phase so that the output of the model can be verified before deployment, also human feedback is leveraged for continuous improvement of the model.			
Reproducibility	Evaluation during Training	LLMs' performance can be evaluated either by manual testing by keeping humans in loop or using statistical measures. There are different statistical measures available to evaluate the performance of LLMs: Perplexity, BLEU, ROGUE etc.			
	Inference Evaluation	Metrics available to measure the LLM performance during productionization in terms of handling incoming requests are: Completed requests per minute, Time to first token (TTFT), Inter-token latency (ITL), End-to-end Latency, etc.			
Explainability	Chain of Thought (CoT)/Provide Evidence	CoT prompting can be used to provide the logic behind the LLM response.	Training data can be adjusted in such a way that the LLM response consists of the CoT as well	RAG plus CoT prompting can solve the gap of providing logic to the LLM response	Difficult to produce the underlying logic as this involves multiple LLMs
Security	Data Guardrail	Secure cloud platforms to train and deploy LLMs are provided by hyperscalers, e.g., Azure, AWS, GCP. They provide the necessary landing zones, inc. infrastructure and tooling to securely host the data and models.			
	Access Control	Access control to the data used for LLMs can be configured within the cloud platform as per the use case. Cloud platforms also provide default standard protocol to access the data used in LLMs.			

Figure 6: Responsible AI integrated with LLMOps

easier said than done as it depends on multiple parameters, and basically requires one to classify how the capabilities of a non-deterministic system will impact users and systems who might interact with it in the future.

The table in Fig. 6 summarizes the key challenges and solutions in implementing Responsible AI for the different Gen AI architectural patterns. We expand on the above points in the rest of the article to enable an integrated LLMOps pipeline with Responsible LLM Governance.

3.1. Data Quality and Reliability

With respect to the data quality/reliability, we need to consider certain dimensions of the quality of the data to enhance the reliability of the overall LLM ecosystem. Majorly, we can speak about these dimensions either during training of a LLM or using the LLM using prompt engineering-in both the cases the reliability of the data is of utter importance for enhancing the trust of the LLM. The following data quality checks should be considered while using/training the LLMs:

- Data Consistency: The data used for training (esp., fine-tuning) the LLM should be accurate and precise, which means the relevant data pertaining to the specific use-case should be used to train the LLMs, e.g. if the use case is to generate summary of a medical prescription, the user should not use other data like Q&A of a diagnosis, user

must use only medical prescriptions and corresponding summarization of the prescription. Extra caution should be exercised when incorporating time-related data, ensuring mindfulness of frequency and temporal periods. Many a times, data pipelines need to be created to ingest the data and feed that to LLMs. In such scenarios, extra caution needs to be exercised to consume the running text fields as these fields hold mostly inconsistent and incorrect data.

- Bias/Fairness: With respect to model performance and reliability, it is difficult to control undesired biases in black box LLMs, though it can be controlled to some extent by using fair and unbiased data to fine-tune the LLMs and/or contextualize the LLMs in a RAG architecture.
- Accountability: To make LLMs more reliable, it is recommended to validate the LLMs' output by humans. Involving humans ensures if LLMs hallucinate or provide wrong response, a human can evaluate and make the necessary corrections.
- Hallucination: In case of using LLM APIs or orchestrating multiple LLMs, hallucination likelihood increases as there are no guardrails. The right prompts can help but only to a limited extent. To further limit the hallucination, LLMs need to be fine-tuned with curated data and/or limit the search space of responses to relevant and recent enterprise data.

Use-case

Let us consider a chatbot for an airport to provide live status of flight and ticket availability. In this solution, user can check the flight status as well as the ticket status based on PNR. In this solution, it is extremely important to provide correct feedback to the user.

Without RAG:

User Input: What is the status of PNR 290049223?

LLM Output: Upon checking, we can confirm that your booking is confirmed for the upcoming flight ABC0012R.

In the output, the flight mentioned in wrong as well the status. This will mislead the user and create confusion. To tackle these kind of challenges, we can leverage RAG architecture as it will then contain the information regarding PNR and flight status.

With RAG:

Data stored in RAG: PNR 290049223: Alan Smith:Pending:XYH9902K.

User Input: What is the status of PNR 290049223?

LLM Output: Upon checking, we can confirm that your booking is in pending status for the upcoming flight XYH9902K.

In this output, LLM provides accurate answer with correct status and flight details.

3.2. Model Performance (Reproducibility)

With respect to model performance / reproducibility, it is important to measure the performance of the model during training and inferencing phase to evaluate if the model is performing as per expectation. During the training phase, there are many approaches available to measure the performance of the LLMs which not only ensure the performance of the model during the training phase but also during inferencing phase. We can divide these metrics into two phases: metrics during training phase and for the inferencing phase.

Model Evaluation during Training

To ensure the performance of the model, it is always right to measure the performance of the model during training phase before deploying it to the production. There are different metrics score available which statistically quantifying the model performance:

- Perplexity: This quantifies how a model predicts a text - lower the score better the model.
- BLEU (Bilingual Evaluation Understudy): BLEU is a metric commonly used in machine translation tasks. It compares the generated output with one or more reference translations and measures the similarity between them. The higher the score better the model.
- ROUGE (Recall-Oriented Understudy for Gissing Evaluation): ROUGE is a set of metrics used for evaluating the quality of summaries. It compares the generated summary with one or more reference summaries and calculates precision, recall, and F1-score. The higher the F1-Score, the better the model.
- Human Evaluation: Last but not least, human evaluation or manual testing of the model is essential to check the performance of the model for different scenarios.

Use-case

Let us consider an AI Product that provides summary of the user response of a campaign in social media based on the input provided by the Marketing manager.

User Input: Can you please provide the insights where we can do better for the upcoming campaign, especially, for chocolates?

LLM Output: For your chocolate product campaign, highlight its unique flavor profiles through visually appealing and mouth-watering imagery on social media platforms. Consider partnering with popular food bloggers or influencers to create enticing content and reach a wider audience. Offer limited-time promotions or exclusive discounts to create a sense of urgency and drive sales.

For the above example, *Perplexity* will be low, *BLEU* and *ROGUE* will be high.

Model Performance during Inferencing

Monitoring model performance during live run is an essential aspect of ensuring reliability of the model. Model performance can be monitored in two ways: 1) measure the perfectness of the model outcome and 2) checking model performance in handling request.

Measure the perfectness of the model outcome: Regular monitoring of the model outcome is essential to make the model more robust and reproducible. Certain metrics like Perplexity, BLEU, ROGUE etc. can be used to measure the model performance. If there is any significant drift in model performance, there is a need to retrain the model. For LLMs, it is essential to retrain a model from the last checkpoint of the trained model, else there will be a chance the model would forget what it learned previously and got trained only on the newly fed data.

Checking model performance in handling request: Model performance during productionization or handling

incoming requests is an important feature to evaluate the model robustness and reproducibility. The different metrics that can be used to evaluate the model performance are:

- Completed requests per minute: This measures, how many requests the model dealt with in a certain period.
- Time to first token (TTFT): TTFT is the time before LLM return the first token, in streaming applications this is a critical metric.
- Inter-token latency (ITL): It is the average time between consecutive tokens.
- End-to-end Latency: The end-to-end latency should approximately be the same as the average output length of tokens multiplied by the inter-token latency.

3.3. Model Explainability

Explainability is an umbrella term for a range of tools, algorithms and methods, which accompany AI model inferences with explanations. Explainability and transparency of AI models clearly ranks high among the list of 'non-functional' AI features to be considered first by enterprises. For example, this implies having to explain why an ML model profiled a user to be in a specific segment - which led him/her to receiving an advertisement.

Chain of Thought (CoT) is a framework that addresses how a LLM is solving a problem. CoT can be implemented using two approaches: user prompting and automated approach.

- User Prompting: Here, during prompting, user provides the logic about how to approach a certain problem and LLM will solve similar problem using same logic and return the output along with the logic.
- Automating Chain of Thought Prompting: Manual handcrafting CoT can be time consuming and provide sub optimal solution, Automatic Chain of Thought (Auto-CoT) can be leveraged to generate the reasoning chains automatically thus eliminating human intervention. Auto-CoT basically works on two processes: (1) Inter-token latency (ITL): It is the average time between consecutive tokens. (2) End-to-end Latency: The end-to-end latency should approximately be the same as the average output length of tokens multiplied by the inter-token latency. Auto-CoT works well for LLMs having approximately 100B parameters but not so accurate for the small scale LLMs.

Use-case

Let us consider a chatbot for a medical insurance company where user can check the status of a claim by providing the Claim ID. The chatbot returns not only the status of the claim but also the clarifications.

User Input: What is the status of my claim id: 123456789?

LLM Output: Upon checking, we can confirm that your claim is approved and 80% of the claim amount is finalized in accordance to the condition mentioned in the policy terms section 1.5.

3.4. Data Privacy

With respect to Data Privacy [6], we need to consider the privacy aspects of enterprise data provided as context (RAGs) and/or enterprise data used to fine-tune the LLMs. In addition, the novel privacy aspect here is consider the privacy risks of data (prompts) provided voluntarily by the end-users, which can potentially be used as training data to re-train/fine-tune the LLMs.

Cloud providers and hyperscalers providing LLMs and enabling their fine-tuning on enterprise data also provide the necessary setup / landing zone for data privacy and controlling access to the data for specific use cases. A detailed discussion of the LLM data privacy controls is beyond the scope of this article.

4. Conclusion

Gen AI is a disruptive technology, and we are seeing it evolve faster than anything we have experienced before. So, it is important to understand the key architectural patterns - the corresponding challenges and solutions in scaling Gen AI POCs to Production. This includes enabling LLMOps to efficiently manage the mix of proprietary, open-source and fine-tuned LLMs in an enterprise. Responsible usage of LLMs also requires assessing how the identified LLMOps architectural patterns impact relevant Responsible AI dimensions.

To overcome these challenges, we have proposed a blueprint towards implementing a Responsible and Governed LLM platform enabled by LLMOps frameworks and tooling in this article. We believe that this will accelerate LLM adoption and enable enterprises to scale Gen AI use-cases in a responsible fashion. This also effectively future-proofs Gen AI investments and ensures that the LLM platform will be able to cope as the Gen AI landscape evolves, new LLMs, and new training / deployment architectures, etc. emerge in the future.

References

[1] MLOps: What It Is, Why It Matters, and How to Implement It, nepture.ai, 2023. https://neptune.ai/blog/mlops.

[2] A. De, B. Mitra, D. Ghosh, S. Sen, S. Talukder, Responsible Design and Use of Large Language Models, 2023. URL: https://www.wipro.com/analytics/responsible-design-and-use-of-large-language-models/.

[3] D. Biswas., MLOps for Compositional AI, in: Proceedings of the NeurIPS Workshop on Challenges in Deploying and Monitoring Machine Learning Systems, 2022.

[4] D. Biswas., Constraints Enabled Autonomous Agent Marketplace: Discovery and Matchmaking, in: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, 2024, pp. 396–403.

[5] A. Bordes, Y.-L. Boureau, J. Weston, Learning End-to-End Goal-Oriented Dialog, 2017. arXiv:1605.07683.

[6] D. Biswas, Privacy Preserving Chatbot Conversations, in: IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2020, pp. 179–182.