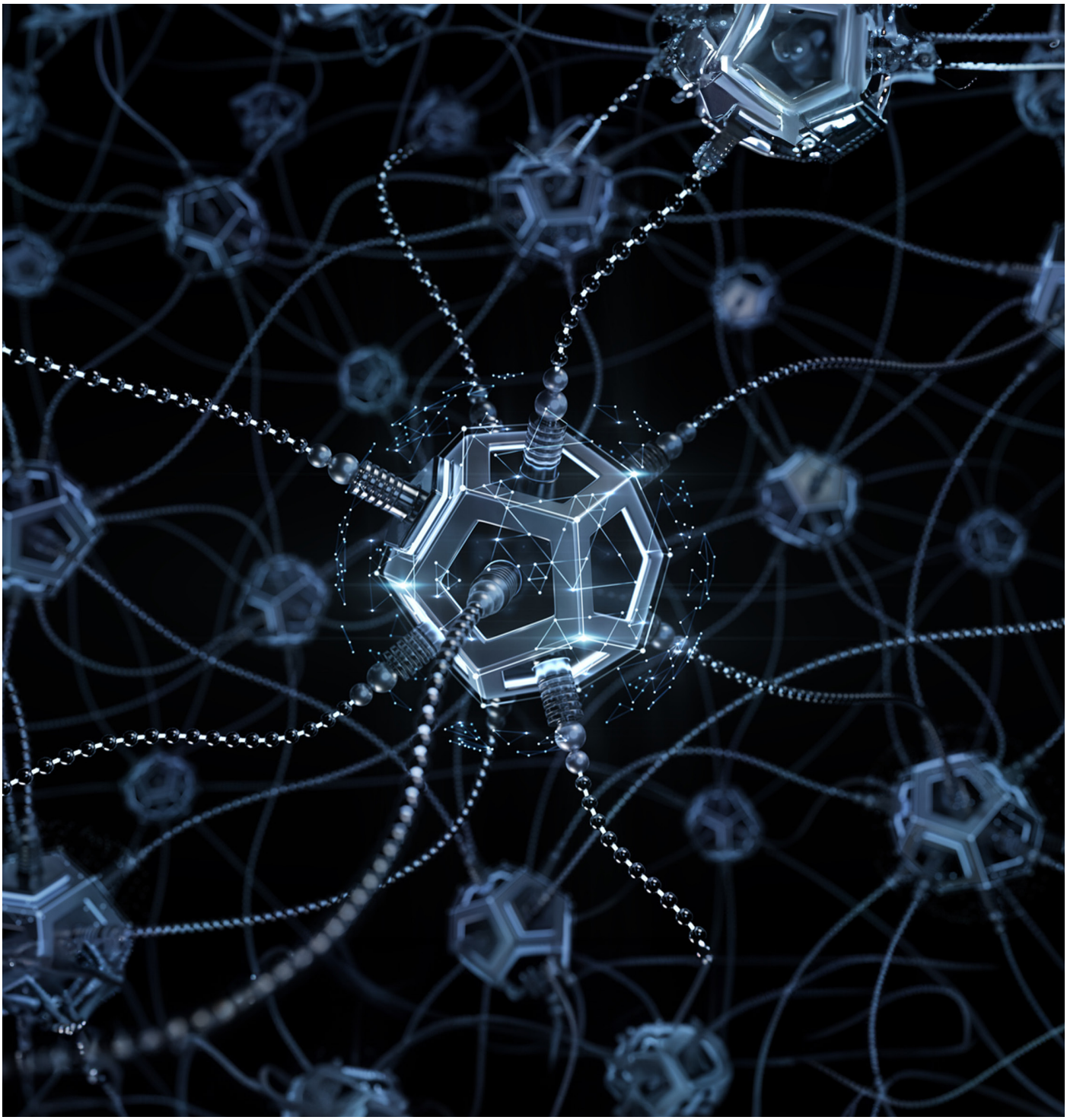
Mphasis
The Next Applied

Building Pipelines and Environments for Large Language Models

Whitepaper by Dr. Archisman Majumdar, Principal – Mphasis NEXT Labs | Amrit R, Lead Data Scientist - Mphasis NEXT Labs | Manami Mandal, Lead Data Scientist - Mphasis NEXT Labs | Dr. Udayaadithya Avadhanam, Principal & VP, Mphasis NEXT Labs | Sai Bharath Sundar, Manager (Data Science), Mphasis NEXT Labs | Biju Mathews, Partner – Mphasis NEXT Labs | Siddharth Shankar, Affiliated to University of Maryland



Contents

1.
Introduction to LLMOps

Generative AI models have gained wide popularity in recent times with the adoption of transformer-based neural network architectures. Generative model's ability to generate new data enables them to go beyond traditional prediction and classification use cases. These models are now used across domains and use cases like chatbots, question answering, fraud detection, protein folding and many more.

Generative AI models for natural language use cases are powered by Large Language Models (LLMs). LLMs are transformer-based Deep Learning architectures that harness vast amounts of textual data to develop language and domain understanding. The models are built with an emphasis on generating human-like responses and reasoning. Their ability to understand human languages allows them to serve as powerful tools for information retrieval, natural language processing, language translation and even creative writing.

Using large language models in production environments poses a certain unique set of challenges such as organizing LLMs into agents for sub-tasks, developing robust instructions for each LLM agent, evaluating the correctness of generated response and efficiencies with fine-tuning. Hence, effective usage in a production environment requires appropriate infrastructure and practices focusing on experimentation, deployment, management and monitoring of large language models.

Large Language Model Operations (LLMOps) is a framework of tools and best practices to manage the lifecycle of LLM-powered applications, from development to deployment and maintenance.

The aim is to enable AI capabilities with LLMs by developing better prompts, longer context, faster inference and customized techniques that enable rapid experimentation and innovation with LLMs. Together, these allow data scientists and engineers to collaborate effectively and deliver high-quality solutions.

In this paper, we highlight some of the recent advances, challenges, common frameworks, platforms and tools that are used in a typical LLMOps workflow. We conclude by demonstrating the components of a typical LLMOps environment for production.

2.
Why LLMOps?

LLMOps aims to focus on overcoming the unique challenges and requirements specific to LLM models such as:

- **Infrastructure, scaling and performance:** LLMs require significant computational resources for text data collection. LLMs are often composed of multiple layers and components, such as transformers, recurrent neural networks, attention mechanisms and embeddings. Training these models requires high computational resources and efficient optimization algorithms.

- **The ambiguity of natural languages:** Prompts can be interpreted differently by users and LLMs, leading to silent failures or unexpected outputs. Prompt engineering requires rigorous evaluation, versioning and optimization methods to ensure quality and consistency.
- **Backward and forward compatibility:** LLMs are constantly evolving and being updated, which can affect the behavior and performance of existing prompts. Prompt engineering needs to account for the compatibility issues between different versions of LLMs and ensure smooth transitions. Steps such as model fine-tuning, deployment and retraining could affect the backward-forward compatibility of the model.
- **Cost and latency:** LLMs are expensive to run and can have high latency for real-time applications. Cost and latency analysis for LLMs is difficult due to the variability of inputs and outputs. LLMs also need to be accelerated with techniques such as inference with reference, prompt tuning or distillation.
- **Data privacy and compliance:** Data privacy and compliance are critical issues for large language models like ChatGPT. These models are trained on massive amounts of text data from various sources, which may contain sensitive or personal information.

3.
Typical Stages in an LLMOps Workflow

LLMOps comprise various components/stages involved in the entire lifecycle of a large language model from data ingestion to model deployment and monitoring. Some of these include sourcing the data, cleaning, choosing the right architecture, model training, fine-tuning, monitoring, maintenance and optimization to list some. In this section, we highlight the typical activities in each stage.

3.1 Data Collection, Preparation, Labelling

The data preparation stage for Machine Learning involves several steps: sourcing data, ensuring completeness, adding labels and data transformations to generate features. LLMs require massive amounts of data to train. This can be a challenge to collect, especially if the data is specific to a particular domain or task.

The quality of the data is critical for the performance of large language models. Any bias or inconsistencies in the data can be amplified by the model leading to poor performance. If the data is noisy or inconsistent, it can lead to errors in the model, such as generating incorrect or nonsensical text. Labelling data for such tasks can be an expensive, time-consuming and complex activity. This requires human experts, domain knowledge or specific guidelines. Without Large Language, models can frequently produce results that are unsuited for real-world applications. Thus, reinforcement learning with human feedback has emerged as an area of great importance in the training of large language models.

Only a few companies train benchmark models from scratch since they need large computing budgets, hardware resources and extensive ML knowledge.

A complete discussion on the training of large language models is beyond the scope of this paper.

3.2 Selection of Foundation Models

Since training LLMs from scratch is costly and time-consuming, requiring enormous amounts of data and computational resources, many researchers and practitioners opt to use pre-trained LLMs as foundation models, which can be fine-tuned or adapted to specific downstream applications with less data and resources.

Some popular foundation models include BERT, GPT-3, T5 and XLNet. These models have been shown to be effective for a wide range of tasks including natural language understanding, natural language generation and machine translation.

Different foundation models have different strengths and weaknesses depending on their architecture, pre-training data and learning objectives. It is important to carefully consider the different factors when selecting a foundation model for an LLM project, some points that can be considered are:

- **Performance of model on benchmark tasks**

  Several public benchmarks offer comprehensive evaluations of large language models across various tasks and datasets. Some examples of these are HELM, GLUE, SuperGLUE, Big-Bench and MMLU.

- **Size and intricacy of the model**

  However, many of the models require expensive and high-performance hardware to run and deploy. Models like LLaMA and ALPACA are built on principles of community-driven innovation and wide accessibility, contributing to the democratization of AI technology. Increasingly, open-source LLaMA-inspired models, such as Vicuna, a tuned version of LLaMA that matches GPT-4 performance, Koala from Berkeley AI Research Institute and ColossalChat from UC Berkeley, which is a ChatGPT-like model, are becoming available to run on less expensive hardware.

- **Availability of domain-specific pre-trained models**

  Domain-specific models are foundational models that are specialized for tasks that can be selected as per use cases. Some examples are BloombergGPT which is trained on a wide range of financial data for the financial industry, Med-PaLM2 and BioMedLM which are aligned to the medical domain to answer medical questions more accurately and safely. Galactica which is an LLM fine-tuned with scientific knowledge.

- **Licensing terms of the foundation model**

  One of the key factors that influence the choice of an LLM is the licensing and availability of the model. Many organizations can opt for a managed model service provided by various vendors where for a flat fee they get a fully managed deployment service and technical support with only the option to fine-tune. Another option is to use an open-source model and manage the deployment. This enables the flexibility of fine-tuning and retraining.

- **Ethical and social implications**

  We discuss the ethical and social implications of model selection in section 3.6 Security, Privacy, Governance and Ethical Considerations.

3.3 Using Large Language Models - Prompting and Fine-tuning

Large Language models have been developed as powerful models for obtaining labels on text data which can complete a variety of tasks without any additional training or examples, i.e., with Zero-Shot Learning (ZSL). The goal of zero-shot learning is to develop the capacity to forecast outcomes without the use of training samples; to do this, the machine must be able to also identify objects belonging to classes that were not previously taught.

The foundation of zero-shot learning is the transmission of knowledge that is already present in the examples provided during training. Contextual embeddings in LLMs can identify the semantic concepts within the text making it helpful for ZSL. The key to this is robust pre-training using varied prompt patterns and techniques.

Prompting refers to the bundling of queries intended to gain information from the model in the form of natural language processing. Prompt templates come as a useful way to generate and handle prompts. A prompt template usually consists of:

- Instructions for the language model
- A set of a few shot examples to help the language model generate a better response (an optional requirement in the case of ZSL)
- And a question for the language model

The quality of the prompt used as input for training LLMs is important as LLMs can practice zero-shot or few-shot learning just with contextual prompts without optimizing parameters. A collection of such templates and prompts can be stored in the form of a prompt library. These libraries include prompts and templates of various categories such as editing, SEO, content creation, etc.

For more complicated tasks, where zero-shot prompting may not suffice, LLMs can be given a few examples to enable in-context learning. In the few-shot learning scenario, the LLM sees a few examples of input-output pairs and solves new tasks.

Few-shot or zero-shot prompting is not always sufficient in customizing model behavior for complex tasks. In such scenarios, tuning of the models may be required to improve the performance of the model to deliver specific and optimal outputs. There are two primary ways in which a model can be tuned: Fine-tuning and Parameter-efficient Fine-tuning.

- **Fine-tuning:** In this method, the model is pre-trained on a generic dataset and after that, it is copied and retrained using the learned weights. The challenge of fine-tuning an LLM model is that since it is a large model, it takes an exceptionally long time to update every weight. Therefore, parameter-efficient tuning is preferred more in the case of LLMs.
- **Parameter Efficient Fine-Tuning (PEFT)**: It includes training only a small subset of parameters that are identified as the most important ones out of the lot. These parameters could either be new or existent. With PEFT, most parameters of the pre-trained model are unchanged which makes it less likely to fall into Catastrophic Forgetting compared to full fine-tuning which revises the weights of all the parameters making the model more prone to Catastrophic Forgetting (a phenomenon in which the model loses its generalization ability on a task after being trained on a new task because of overriding of the weights that have been learned in the past, thus affecting model performance for the tasks learned previously).

Two of the most common ways of applying PEFT are using adapters, which are submodules inserted in the specific layers in the model architecture and via Low-Rank Adaption (LoRA) which is also like adapters in functionality.

Instruction Fine-tuning or Instruction Tuning is also a well-known method of tuning pre-trained models. Fine-tuned Language Net (FLAN) is a pre-trained model that is tuned based on instructions. FLAN's instruction tuning phase requires less updates than the model's pre-training phase, which requires a significant amount of computation. Creating a dataset of instructions from scratch to fine-tune the model would take many resources. Hence, existing templates are used to convert the dataset into instruction format. Research shows that model scale is important for the model's ability to benefit from instruction tuning. The FLAN approach reduces performance at smaller sizes, and it is only at higher scales that the model can generalize from the training data's instructions to new tasks. This may be because too small models lack the parameters necessary to carry out a wide range of tasks.

3.4 Evaluation of Prompts and Models and Version Control

Evaluating the output of large language models can be a challenge because they often produce outputs that are not directly comparable to the ground truth or human expectations. While evaluating a typical discriminative model, the performance can be measured by metrics such as accuracy, precision, recall or F1-score because they usually produce outputs that are discrete, categorical or numerical. LLM evaluation, however, is multi-dimensional, hence a comprehensive evaluation framework of the models and data (prompts) is necessary to achieve optimal results. Performance evaluation of working models of LLM becomes necessary as it is tested on metrics to conclude how well the model is performing on a varied range of inputs such as accuracy, coherence, fluency and subject relevance. It is frequently required to combine different methodologies to fully assess a language model's performance. Following are the five commonly used evaluation methods for LLMs:

- **Perplexity:** It gives results in figurative/quantitative format to determine the level of performance of the model. The lower the perplexity, the better the performance. It calculates how much the model is surprised by seeing new data. Perplexity is usually used just to determine how extensively a model has learned the training set. Other metrics like BLEU and ROUGE are used on the test set to measure performance.
- **BLEU (Bilingual Evaluation Understudy):** This metric is commonly used in machine translation tasks. The generated output is compared with one or more reference translations and then the similarity between them is calculated. The range of scores for BLEU is from 0 to 1 with scores nearer to 1 suggesting better performance.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** It is used to determine the quality of summaries. It compares the generated summary with one or more reference summaries. It calculates precision, recall and F-1 score. ROUGE scores around fifty are considered excellent results. ROUGE is available in three different forms: rouge-n, which is the most common and finds n-gram overlap; rouge-l, which locates the Longest Common Subsequence; and rouge-s, which focuses on skip grams.

- **METEOR Score:** It measures the quality of generated text based on the alignment between the generated text and the reference text. It is a harmonic mean of unigram precision and recall, with recall weighted higher than precision. This metric produces a good correlation with human judgement at the sentence or segment level.
- **BERTScore:** The pre-trained contextual embeddings from BERT are used by BERTScore to compare words in candidate and reference sentences based on their cosine similarity. It has been demonstrated that it correlates with human judgment in the evaluation of sentences and systems. BERTScore computes precision, recall and F1 measures, which can be useful for evaluating different language generation tasks as well.

Evaluation metrics for multimodal LLMs: In recent experiments, the Multimodal Large Language Model (MLLM), which is built on the potent LLM, has shown extraordinary emergent skills including the ability to create poetry based on images. The various benchmarks for evaluation of such models are under ongoing research but here are a few prominent metrics more widely used:

- **Caption Hallucination Assessment with Image Relevance (CHAIR):** It is a well-liked statistic for assessing object hallucination in activities requiring picture captioning. CHAIR determines the percentage of things that are in the image, but not the caption based on the actual objects in the image. The two variations, CHAIRi (instance-level) and CHAIRs (sentence-level), which assess the degree of hallucination at the object instance level and phrase level, respectively, are often used in existing work.
- **Retrieval evaluation:** Consider the task of retrieving images from sentence queries. Given a test sentence, we compute the model perplexity conditioned on each test image and rank each image accordingly.
- **Benchmark evaluation:** It is another frequently used technique using which functionality and performance of models are evaluated. LLM benchmarking can be conducted with respect to various tasks. It is used to perform task-specific evaluation or assess the model's output for fairness, coherence, relevance which is known as bias and fairness evaluation. The AI2 Reasoning Challenge (ARC) and WinoGrande, for instance, are two well-liked benchmarks for evaluating LLMs like OpenAI's GPT-4, Databricks' Dolly and Facebook's LLaMA. Another important evaluation benchmark is Google's BIG (Beyond the Imitation Game) Bench. One of the project's goals is to see how the performance on the tasks is changing with the model size, with the size ranging by many orders of magnitude. BIG bench currently consists of 204 tasks.

3.5 Deployment and Monitoring

The optimal deployment strategy for each large language model can vary according to the use cases. Following are some of the key features of the LLM deployment:

- **Hassle-free deployment:** Models can be deployed as production-ready APIs easily with a few steps rather than handling MLOps infrastructure.
- **Cost efficiency:** Automatic scaling benefits through scaling down the infrastructure when the endpoints are not in use, reducing the overall cost.
- **Optimization:** Utilizing proprietary transformer code, Flash Attention Power from Text Generation Inference and Paged Attention to enable high throughput with minimal latency.

- **Security:** For improved data security and compliance, deploy models at secure offline endpoints that can only be accessed through direct VPC (Virtual Private Cloud) connections are SOC2 Type 2 certified, and have BAA and GDPR (General Data Protection Regulation) data processing agreements.

One way to facilitate the deployment of these models is to apply some optimization techniques after training. These techniques reduce the model size, which lowers the infrastructure costs and increases the inference speed. Some examples of these techniques are model distillation, quantization and pruning.

- **Distillation:** This process involves training a separate inference-optimized model using the training-optimized model, where knowledge transfer happens between the two. Model distillation allows the inference-optimized model to keep almost all the quality of the training-optimized model. Stanford Alpaca: An instruction-following LLaMA Model, this paper shows the fine-tuning of a smaller open-source language model (LLaMA-7B, the seven billion parameter version of LLaMA) on examples generated by a larger language model (text-davinci-003 – 175 billion parameters). For fine-tuning, they used 52K instructions, which they input into text-davinci-003 to obtain outputs, which are then used to fine-tune LLaMA-7B. This costs under $500 to generate. The training process for fine-tuning costs under $100.
- **Quantization:** This is a common compression operation that is used to reduce the memory footprint of the model and helps improve the inference performance.
- **Pruning:** This technique is used to reduce the size and computational complexity of an LLM by removing redundant or unnecessary model parameters. Pruning improves the efficiency of LLM inference without sacrificing accuracy. Pruning can operate at two different levels: weight pruning and neuron pruning. Weight pruning eliminates individual connections between neurons, while neuron pruning removes entire neurons or units from the network.

Distributed inference: Models served in production environments are subject to multiple parallel requests for inference. Distributed hardware can be optimally leveraged for inference. In a multi-GPU setup, this involves identifying the number of available GPUs and processing incoming requests parallelly, i.e., assigning a request to a GPU. This can be either real-time or batch (where batches are split and assigned) requests.

The distributed inference is not only applicable to multi-request parallelization, but also to cases where a single request requires to be split into multiple parallel tasks where each task is accomplished by an LLM in a distributed setup.

Model monitoring helps maintain the integrity of the model's performance. E.g., in Vertex AI, the model is monitored post-deployment based on prediction input data for feature skew and drift. TensorFlow Data Validation (TFDV) is used in model monitoring to compute the distributions and distance scores for detecting training-serving skew and prediction drift. As LLMs are used in more applications, it is important to keep an eye on their behavior and put safety measures in place to avoid possible problems like poisonous prompts and replies or the existence of sensitive material. To make sure their behavior adheres to the specified norms and rules, large language models like GPT-3 are routinely monitored after deployment using a combination of automated and human techniques.

Model monitoring helps maintain the integrity of the model's performance. E.g., in Vertex AI, the model is monitored post-deployment based on prediction input data for feature skew and drift. Feature skew occurs when there is a difference between the feature data distribution in production and training. Prediction drift takes place when feature data distribution in production changes significantly over time. There is a specific threshold for both skew and drift and once it's crossed, the model monitor sends an email alert to the admin. TensorFlow Data Validation (TFDV) is an example tool used in model monitoring to compute the distributions and distance scores for detecting training-serving skew and prediction drift.

3.6 Security, Privacy, Governance and Ethical Considerations

Although Language Model Systems (LLMs) provide remarkable abilities to generate natural language texts based on different inputs like keywords, prompts or queries, they also bring forth significant challenges in terms of security, privacy, governance and ethics.

LLMs can be vulnerable to various forms of attacks including adversarial examples and manipulation. Malicious actors might exploit these vulnerabilities to generate misleading or harmful content, spread misinformation or engage in other nefarious activities. They can also potentially leak sensitive information from training data or could inadvertently memorize and reproduce private or personally-identifiable details, raising privacy issues for individuals whose data is used in the training process.

Attacks using prompt injection have emerged as a key vulnerability that affects AI models. Prompt injection attacks can be particularly dangerous for Large Language Models (LLMs) that use prompt-based learning. These attacks can be executed in different forms depending on the system. One method of attack is altering or introducing harmful content into prompts to take advantage of the system. These exploits could make use of real vulnerabilities, modify system behavior or trick users. Apart from this, a quick injection attack can be used to cause an unanticipated reaction from LLM-based technologies leading to getting illegal access, influencing outcomes or going around security precautions. For instance, quick injection attacks frequently try to steal data in the context of language models. A language model's past instructions may be revealed through prompt injection, and in some situations, the model may not be able to carry out those instructions. This enables a malevolent user to circumvent restrictions on what the model is permitted to perform and may potentially reveal sensitive data.

The uncontrolled deployment of LLMs could lead to the generation of biased, offensive or discriminatory content. Proper governance mechanisms are necessary to ensure that the outputs of these models align with ethical standards and do not perpetuate harmful biases. LLMs can inadvertently generate content that promotes hate speech, violence or misinformation.

Since these models are trained on large and diverse datasets, inherent biases present in the training data can lead to biased outputs and since many users may not fully understand how the model arrives at specific conclusions, making it difficult to assess the reliability and credibility of the generated content as well.

Generating content with LLMs could raise concerns about intellectual property rights. Determining ownership and authorship of generated texts can be challenging, especially when the model incorporates a vast corpus of publicly available data.

4.
Setting Up LLMOps Pipelines

A typical LLMOps pipelines need three environments (an environment is a set of isolated compute and associated infrastructure):

- Sandbox environment
- Runner environment
- Production environment

User roles are set up accordingly to manage access to the desired environments.

Sandbox environment:
Every LLM initiative starts in a sandbox environment. A sandbox environment can be considered equivalent to a development environment in any software development lifecycle. However, unlike a developer's workstation which is generally used as the development environment, LLMs (or other Deep Learning exercises) require significant computing, multiple experimentation cycles and collaboration to get to the right recipe. Hence, the sandbox environment itself is hosted on the cloud with LLM hosting capabilities.

As is crucial to any Machine Learning project, quality data is a prerequisite. Hence, setting up a data foundation with tools for data labeling, data loading and feature storing is the zeroth step.

As discussed in section 3.3 Using Large Language Models - Prompting and Fine-tuning, the first step in an LLM customizing/building experiment starts with zero-shot/few-shot in-context learning evaluation. This means using a pre-trained model from the LLM hub, providing instructions and examples for the task at hand as a prompt and evaluating the results. To use a pre-trained model, one must first deploy the model for inference within the sandbox environment. Frameworks such as Langchain provide interfaces (prompts, chaining, connectors, etc.) for commonly performed tasks. Fine-tuning a prompt (which is debated to be more of an art) can be streamlined with techniques like re-prompting and auto-prompting.

A purely in-context approach may not be suitable for all use cases. To adapt the underlying pre-trained LLMs for domain-specific context and task model fine-tuning is the next step in the process. With data being the pre-requisite, the focus for fine-tuning is: how efficiently can the model be fine tuned?, i.e., minimizing time, dataset size and mix, computing resources while maximizing model performance on test data.

The result of model fine-tuning and fixing the prompt template is a new model entry to the LLM model hub.

Runner environment:
The runner environment ensures successful build and deployment to a target environment.
A successful build requires:

- Code checks for vulnerabilities and best practices (such as type match, exception handling)
- Code performance optimization

A successful deployment requires:

- The model serves as a REST endpoint for real-time inference OR as a task for job-based/batch inference
- The endpoint can cater to multiple requests
- Logging is enabled

It is integral for CI/CD setup and robust provisioning.

Production environment:

The production environment hosts the models that integrate the rest of the processes for a use case. The distribution of data keeps varying over time. To maintain optimal model performance, tracking the input and output in production is required. Logging and getting the feedback to the sandbox environment for further experimentation is part of this pipeline.

LLMs are pre-trained on datasets that can inherently contain various biases. The models can end up generating results that include these biases. LLMs are also observed to be highly sensitive to the input prompts. This can result in hallucinated results that are incorrect or sub-optimal. The sensitivity of the input can also result in a threat of model attacks.

Hence, the focus in a production environment is to control the quality of input data (content filtering), ensure guard rails on model results and monitor for data drifts (part of the observability stack).

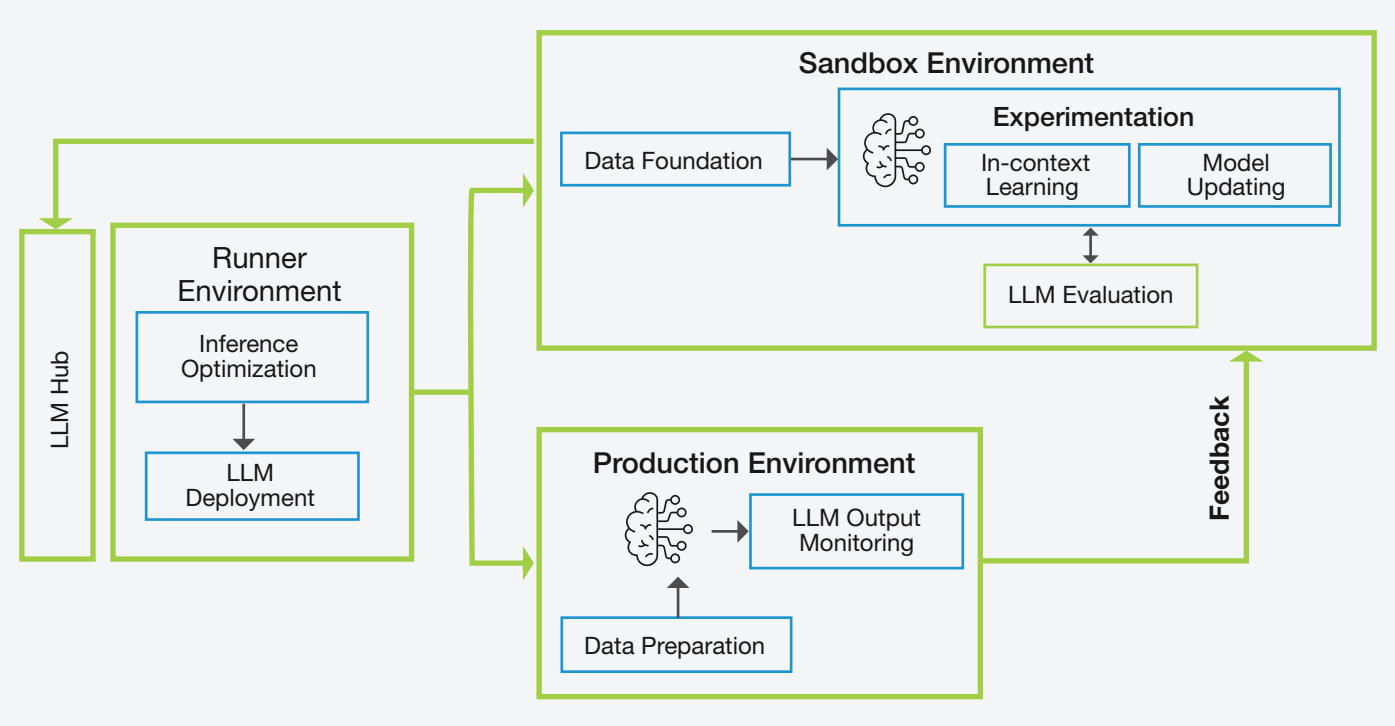


Figure 1: LLMOps setup and pipeline

5.
Conclusions

Some benefits of using LLMOps frameworks and platforms are:

- **Improved quality and reliability:** LLMOps ensure that LLMs are tested and validated before deployment, minimizing errors and bugs. LLMOps also monitor and troubleshoot LLMs in production, ensuring high availability and reliability.
- **Enhanced scalability and efficiency:** LLMOps optimize the use of computational resources such as GPUs (Graphics Processing Unit) and TPUs (Tensor Processing Unit) for LLMs, enabling them to handle large volumes of data and provide real-time predictions. LLMOps also leverages cloud services and distributed systems for LLMs, enabling them to scale up or down as needed. Solutions like Ray simplifies the process of deploying large-scale AI models.
- **Increased innovation and collaboration:** LLMOps enable data scientists and engineers to experiment with new ideas and concepts related to LLMs, such as feature engineering, model architecture and hyperparameters. LLMOps also facilitates communication and coordination among different stakeholders involved in LLM development and deployment.
- **Faster time to market:** LLMOps automate and streamline the processes of building, testing and deploying LLMs, reducing the time and effort required to deliver LLM-powered solutions.
- **LLM agents:** Agents can help enhance the utility and versatility of LLMs by expanding their input modalities and action spaces, allowing them to handle complex tasks that involve text, images, audio, tools and embodiment. This can improve the sociability and trustworthiness of LLMs by exhibiting natural language communication proficiency, cooperation and negotiation abilities and explainable decision-making processes.

LLMOps is an emerging field that aims to address these challenges and provide best practices and solutions for building and managing large language models. LLMOps can help improve the efficiency, reliability, scalability and reproducibility of NLP projects.

6.
References

google/BIG-bench: Beyond the Imitation Game collaborative benchmark for measuring and extrapolating the capabilities of language models (github.com)

Holistic Evaluation of Language Models (HELM) (stanford.edu)

Introducing FLAN: More generalizable Language Models with Instruction Fine-Tuning

Better Language Models Without Massive Compute

Deploying Large NLP Models: Infrastructure Cost Optimization (neptune.ai)

Building LLM applications for production (huyenchip.com)

AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback

Safeguarding and Monitoring Large Language Model (LLM) Applications

Optimizing Pre-Trained Models: A Guide to Parameter-Efficient Fine-Tuning (Peft)

Understanding LLMOps: Large Language Model Operations | Articles – Weights & Biases (wandb.ai)

How Ray Solves Generative AI and LLM Infrastructure Challenges (anyscale.com)

Papineni, Kishore, et al. "Bleu: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002.

Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." Text summarization branches out. 2004.

METEOR – Wikipedia

Zhang, Tianyi, et al. "Bertscore: Evaluating text generation with bert." arXiv preprint arXiv:1904.09675 (2019).

Rohrbach, Anna, et al. "Object hallucination in image captioning." arXiv preprint arXiv:1809.02156 (2018).

Large Language Model Evaluation in 2023: 5 Methods (aimultiple.com)

Wei, Jason, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. "Finetuned language models are zero-shot learners." *arXiv preprint arXiv:2109.01652* (2021).([2109.01652] Finetuned Language Models Are Zero-Shot Learners (arxiv.org))

Yang, N., “Inference with Reference: Lossless Acceleration of Large Language Models”, arXiv e-prints, 2023. doi:10.48550/arXiv.2304.04487.([2304.04487] Inference with Reference: Lossless Acceleration of Large Language Models (arxiv.org))

Deploying Large NLP Models: Infrastructure Cost Optimization (neptune.ai)

Train 175+ billion parameter NLP models with model parallel additions and Hugging Face on Amazon SageMaker | AWS Machine Learning BlogLangchain (Introduction | Langchain))

About Mphasis

Mphasis’ purpose is to be the *“Driver in the Driverless Car”* for Global Enterprises by applying next-generation design, architecture and engineering services, to deliver scalable and sustainable software and technology solutions. Customer centricity is foundational to Mphasis, and is reflected in the Mphasis’ Front2Back™ Transformation approach. Front2Back™ uses the exponential power of cloud and cognitive to provide hyper-personalized ($C = X2C^{2}_{TM} = 1$) digital experience to clients and their end customers. Mphasis’ Service Transformation approach helps ‘shrink the core’ through the application of digital technologies across legacy environments within an enterprise, enabling businesses to stay ahead in a changing world. Mphasis’ core reference architectures and tools, speed and innovation with domain expertise and specialization, combined with an integrated sustainability and purpose-led approach across its operations and solutions are key to building strong relationships with marquee clients. Click here to know more. (BSE: 526299; NSE: MPHASIS)

For more information, contact: marketinginfo.m@mphasis.com

USA
Mphasis Corporation
41 Madison Avenue
35th Floor, New York
New York 10010, USA
Tel: +1 (212) 686 6655

UK
Mphasis UK Limited
1 Ropemaker Street, London
EC2Y 9HT, United Kingdom
T : +44 020 7153 1327

INDIA
Mphasis Limited
Bagmane World Technology Center
Marathahalli Ring Road
Doddanakundhi Village, Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

www.mphasis.com

WR 27/11/23 US LETTER BASIL8451