[Week 5] Tools for Building LLM Applications

ETMI5: Explain to Me in 5

In this section of our course, we explore the essential technologies and tools that facilitate the creation and enhancement of LLM applications. This includes Custom Model Adaptation for bespoke solutions, RAG-based Applications for contextually rich responses, and an extensive range of tools for input processing, development, application management, and output analysis. Through this comprehensive overview, we aim to equip you with the knowledge to leverage both proprietary and open-source models, alongside advanced development, hosting, and monitoring tools.

Types of LLM Applications

LLM applications are gaining momentum, with an increasing number of startups and companies integrating them into their operations for various purposes. These applications can be categorized into three main types, based on how LLMs are utilized

1. **Custom Model Adaptation**: This encompasses both the development of custom models from scratch and fine-tuning pre-existing models. While custom model development demands skilled ML scientists and substantial resources, fine-tuning involves updating pre-trained models with additional data. Though fine-tuning is increasingly accessible due to open-source innovations, it still requires a sophisticated team and may result in unintended consequences. Despite its challenges, both approaches are witnessing rapid adoption across industries.
2. **RAG based Applications**: The Retrieval Augmented Generation (RAG) method, likely the simplest and most widely adopted approach currently, utilizes a foundational model supplemented with contextual information. This involves retrieving embeddings, which represent words or phrases in a multidimensional vector space, from dedicated vector databases. Through the conversion of unstructured data into embeddings and their storage in these databases, RAG enables efficient retrieval of pertinent context during queries. This facilitates natural language comprehension and timely insights extraction without the need for extensive model customization or training. A notable advantage of RAG is its ability to bypass traditional model limitations like context window constraints. Moreover, it offers cost-effectiveness and scalability, catering to diverse developers and organizations. Furthermore, by harnessing embeddings retrieval, RAG effectively addresses concerns regarding data currency and seamlessly integrates into various applications and systems.

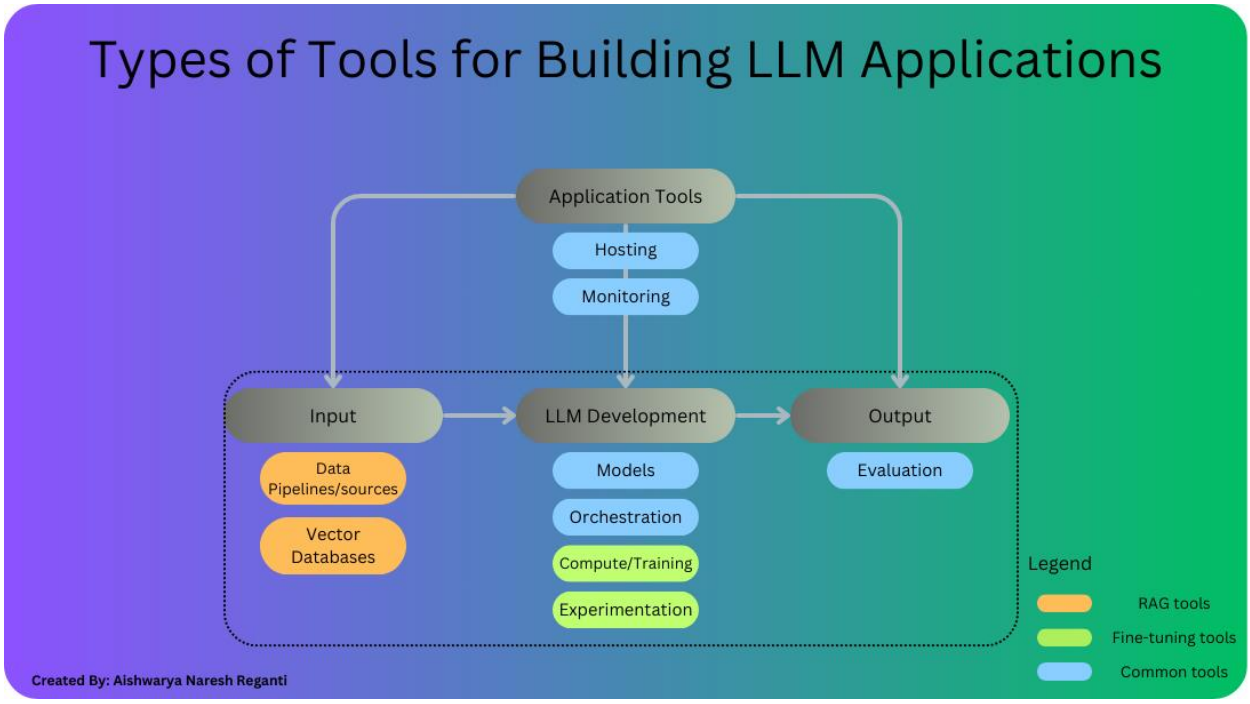
In the previous weeks' content, we covered the distinctions between these methodologies and discussed the criteria for selecting the most appropriate one based on your specific needs. Please review the materials for further details.

In the upcoming sections, we'll explore the tool options available for both of these methodologies. There's certainly some overlap between them, which we'll address.

Types of Tools

We can broadly categorize tools into four major groups:

1. **Input Processing Tools**: These are tools designed to ingest data and various inputs for the application.
2. **LLM Development Tools**: These tools facilitate interaction with the Large Language Model, including calling, fine-tuning, conducting experiments, and orchestration.
3. **Output Tools**: These tools are utilized for managing the output from the LLM application, essentially focusing on post-output processes.
4. **Application Tools**: These tools oversee the comprehensive management of the aforementioned three components, including application hosting, monitoring, and more.



tools_1.png

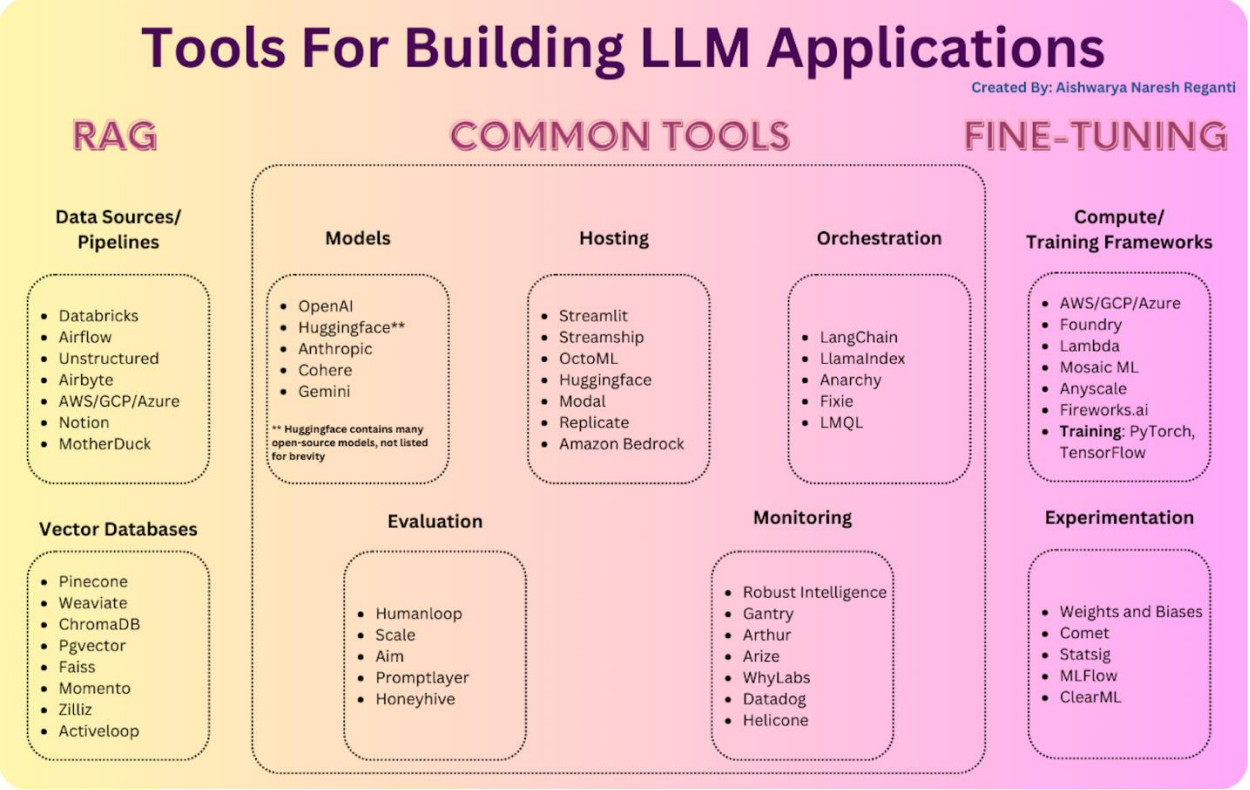
If you're remember from the previous content how RAG operates, an application typically follows these steps:

1. Receives a query from the user (user's input to the application).
2. Utilizes an embedding search to find pertinent data (this involves an embedding LLM, data sources and a vector database for storing data embeddings).
3. Forwards the retrieved documents along with the query to the LLM for processing.

4. Delivers the LLM's output back to the user.

Hosting and monitoring LLM responses are integrated into the overall application architecture, as depicted in the image below. For fine-tuning applications, much of this workflow is maintained. However, there's a need for a framework and computing resources dedicated to model fine-tuning. Additionally, the application may or may not utilize external data, in which case the vector database component might not be necessary. In the figure below, each of these components and their category association is depicted. Now that we know how each of the tools are utilized, let's dig deeper into each of these tool types.

💡 If you're still unsure why each of these tool categories are required, please review the previous weeks' content to understand how RAG and Fine-Tuning applications work



tools_3.png

Summary of tools available to build LLM Apps

Input Processing Tools

1. Data Pipelines/Sources

In LLM applications, the effective management and processing of data are key to boosting performance and functionality. The types of data these applications work with are diverse,

encompassing text documents, PDFs, and structured formats like CSV files or SQL tables. To navigate this variety, a range of data pipelines and source tools are chosen for loading and transforming data.

A. Data Loading and ETL (Extract, Transform, Load) Tools

- **Traditional ETL Tools**: Established ETL solutions are widely used to manage data workflows. **Databricks** is chosen for its robust data processing capabilities, emphasizing machine learning and analytics, while **Apache Airflow** is preferred for its ability to programmatically author, schedule, and monitor workflows.
- **Document Loaders and Orchestration Frameworks**: Applications that predominantly deal with unstructured data often utilize document loaders integrated within orchestration frameworks. Notable examples include:
  - **LangChain**, powered by Unstructured, aids in processing unstructured data for LLM applications.
  - **LlamaIndex**, a component of the Llama Hub ecosystem, offers indexing and retrieval functions for efficient data management.

Further details on LlamaIndex and LangChain will be provided in the orchestration section.

B. Specialized Data-Replication Solutions

Although the existing stack for data management in LLM applications is operational, there is potential for enhancement, especially in developing data-replication solutions specifically tailored for LLM apps. Such innovations could make the integration and operationalization of data more streamlined, improving both efficiency and the scope of possible applications.

Data Loaders for Structured and Unstructured Data

The capability to integrate data from a variety of sources is enabled by data loaders that can handle both structured and unstructured inputs. For instance:

- **Unstructured Data**: Solutions provided by **Unstructured.io** allow for the creation of complex ETL pipelines. These are vital for applications aimed at generating personalized content or conducting semantic searches with data stored in formats like PDFs, documents, and presentations.
- **Structured Data Sources**: Loaders that directly connect to databases and other structured data repositories are used, facilitating seamless data integration and manipulation.

2. Vector Databases

Referring back to the content on RAG, we explored how the most relevant documents are identified through embedding similarity. This is the role where vector databases come into play.

The primary role of a vector database is to store, compare, and retrieve embeddings (i.e., vectors) efficiently, often scaling up to billions. Among the various options available,

Pinecone stands out as a prevalent choice due to its cloud-hosted nature, making it readily accessible and equipped with features that cater to the demands of large enterprises, such as scalability, Single Sign-On, and Service Level Agreements on uptime.

The spectrum of vector databases is broad, encompassing:

- **Open Source Systems** like Weaviate, Vespa, and Qdrant: These platforms offer exceptional performance on a single-node basis and can be customized for particular applications, making them favored choices among AI teams with the expertise to develop tailored platforms.
- **Local Vector Management Libraries** such as Chroma and Faiss: Known for their excellent developer experience, these libraries are straightforward to implement for small-scale applications and development experiments. However, they may not serve as complete substitutes for a full-fledged database at scale.
- **OLTP Extensions like pgvector**: This option is suited for those who tend to use Postgres for various database requirements or enterprises that procure most of their data infrastructure from a single cloud provider, offering a viable solution for vector support. The long-term viability of closely integrating vector and scalar workloads remains to be seen.

With the evolution of technology, many open-source vector database providers are venturing into cloud services. Achieving high performance in the cloud, catering to a wide array of use cases, presents a significant challenge. While the immediate future may not witness drastic changes in the offerings available, the long-term landscape is expected to evolve.

LLM Development Tools

1. Models

Developers have a variety of model options to choose from, each with its own set of advantages depending on the project's requirements. The starting point for many is the OpenAI API, with GPT-4 or GPT-4-32k models being popular choices due to their wide-ranging compatibility and minimal need for fine-tuning.

As applications move from development to production, the focus often shifts towards balancing cost and performance.

Beyond proprietary models, there's a growing interest in open-source alternatives, most of which are available on **Huggingface.** Open-source models provide a flexible and cost-effective solution, especially useful in high-volume, consumer-facing applications like search or chat functions. While traditionally seen as lagging behind their proprietary counterparts in terms of accuracy and performance, the gap is closing. Initiatives like Meta's LLaMa models have showcased the potential for open-source models to reach high levels of accuracy, spurring the development of various alternatives aimed at matching or even surpassing proprietary model performance.

The choice between proprietary and open-source models doesn't just hinge on cost. Considerations include the specific needs of the application, such as accuracy, inference speed, customization options, and the potential need for fine-tuning to meet particular requirements. Users may also weigh the benefits of hosting models themselves against using cloud-based solutions, which can simplify deployment but may involve different cost structures and scalability considerations.

💡 Note that many proprietary models cannot be fine-tuned by the application developers.

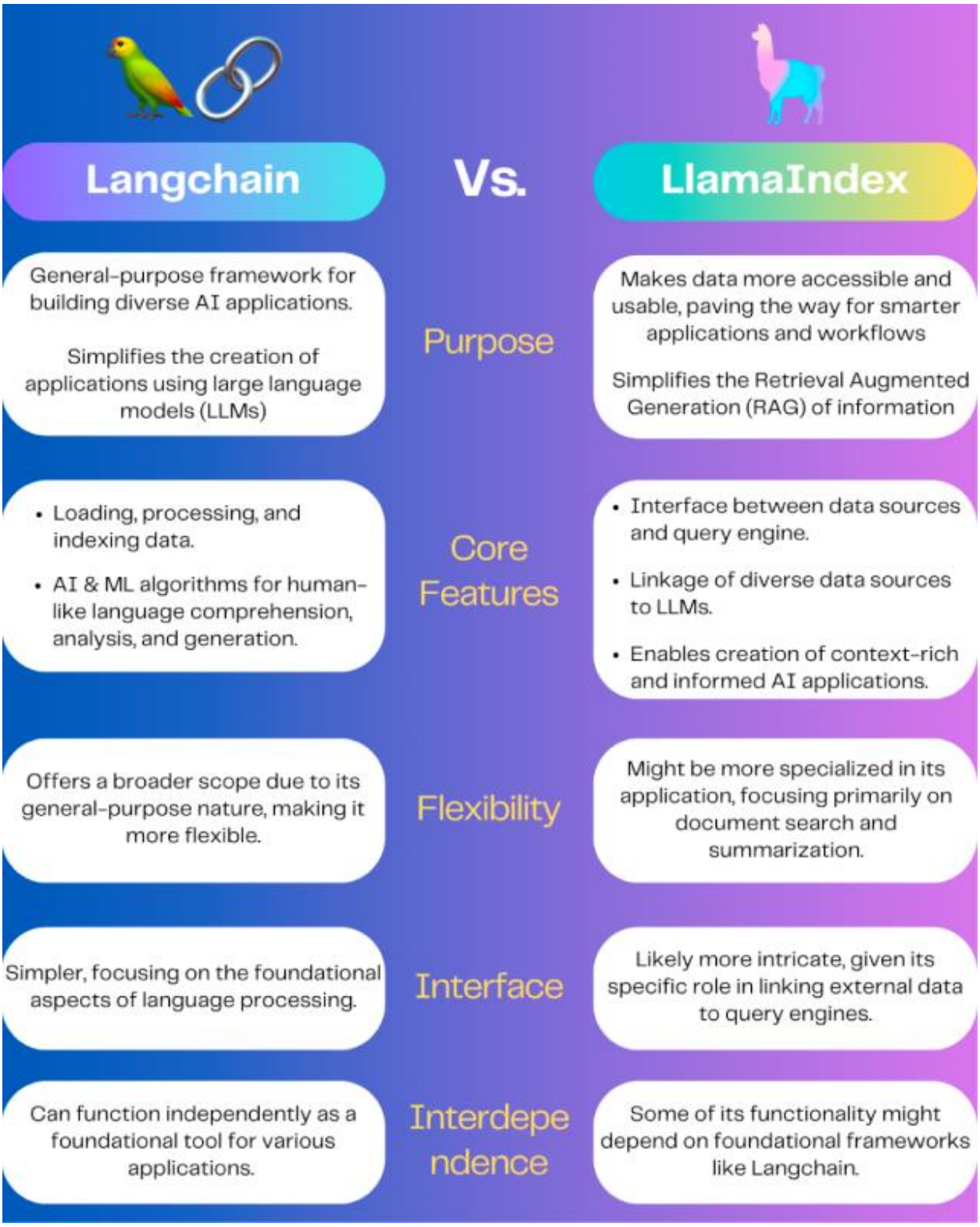
2. Orchestration

Orchestration tools in the context of LLM applications are software frameworks designed to streamline and manage complex processes involving multiple components and interactions with LLMs. Here's a breakdown of what these tools do:

1. **Automate Prompt Engineering**: Orchestration tools automate the creation and management of prompts, which are queries or instructions sent to LLMs. These tools use advanced strategies to construct prompts that effectively communicate the task at hand to the model, improving the relevance and accuracy of the model's responses.
2. **Integrate External Data**: They facilitate the incorporation of external data into prompts, enhancing the model's responses with context that it wasn't originally trained on. This could involve pulling information from databases, web services, or other data sources to provide the LLM with the most current or relevant data for generating its responses.
3. **Manage API Interactions**: Orchestration tools handle the complexities of interfacing with LLM APIs, including making calls to the model, managing API keys, and handling the data returned by the model. This allows developers to focus on higher-level application logic rather than the intricacies of API communication.
4. **Prompt Chaining and Memory Management**: They enable prompt chaining, where the output of one LLM interaction is used as input for another, allowing for more sophisticated dialogues or data processing sequences. Additionally, they can maintain a "memory" of previous interactions, helping the model build on past responses for more coherent and contextually relevant outputs.
5. **Simplify Application Development**: By abstracting away the complexity of working directly with LLMs, orchestration tools make it easier for developers to build applications. They provide templates and frameworks for common use cases like chatbots, content generation, and information retrieval, speeding up the development process.
6. **Avoid Vendor Lock-in**: These tools often design their systems to be model-agnostic, meaning they can work with different LLMs from various providers. This flexibility allows developers to switch between models as needed without rewriting large portions of their application code.

Frameworks like **LangChain** and **LlamaIndex** work by simplifying complex processes such as prompt chaining, interfacing with external APIs, integrating contextual data from vector

databases, and maintaining consistency across multiple LLM interactions. They offer templates for a wide range of applications, making them particularly popular among hobbyists and startups eager to launch their applications quickly, with LangChain leading in usage.


Langchain
Vs.
LlamaIndex
General-purpose framework for building diverse AI applications.
Simplifies the creation of applications using large language models (LLMs)
Purpose
Makes data more accessible and usable, paving the way for smarter applications and workflows
Simplifies the Retrieval Augmented Generation (RAG) of information
• Loading, processing, and indexing data.
• AI & ML algorithms for human-like language comprehension, analysis, and generation.
Core Features
• Interface between data sources and query engine.
• Linkage of diverse data sources to LLMs.
• Enables creation of context-rich and informed AI applications.
Offers a broader scope due to its general-purpose nature, making it more flexible.
Flexibility
Might be more specialized in its application, focusing primarily on document search and summarization.
Simpler, focusing on the foundational aspects of language processing.
Interface
Likely more intricate, given its specific role in linking external data to query engines.
Can function independently as a foundational tool for various applications.
Interdependence
Some of its functionality might depend on foundational frameworks like Langchain.

tools_2.png

Image Source: https://stackoverflow.com/questions/76990736/differences-between-langchain-llamaindex

Retrieval-augmented generation techniques, which personalize model outputs by embedding specific data within prompts, demonstrate how personalization can be achieved without altering the model's weights through fine-tuning. Tools like LangChain and LlamaIndex offer structures for weaving data into the model's context, facilitating this process.

The availability of language model APIs democratizes access to powerful models, extending their use beyond specialized machine learning teams to the broader developer community. This expansion is likely to spur the development of more developer-oriented tools. LangChain, for instance, assists developers in overcoming common challenges by abstracting complexities such as model integration, data connection, and avoiding vendor lock-in. Its utility ranges from prototyping to full-scale production use, indicating a significant shift towards more accessible and versatile tooling in the LLM application development ecosystem.

3. Compute/Training Frameworks

Compute and training frameworks play essential roles in the development and deployment of LLM applications, particularly when it comes to fine-tuning models to suit specific needs or developing entirely new models. These frameworks and services provide the necessary infrastructure and tools required for handling the substantial computational demands of working with LLMs.

Compute Frameworks

Compute frameworks and cloud services offer scalable resources needed to run LLM applications efficiently. Examples include:

- **Cloud Providers**: Services like **AWS** **(Amazon Web Services)** provide a wide range of computing resources, including GPU and CPU instances, which are critical for both training and inference phases of LLM applications. These platforms offer flexibility and scalability, allowing developers to adjust resources according to their project's requirements.
- **LLM Infrastructure Companies**: Companies like **Fireworks.ai** and **Anyscale** specialize in providing infrastructure solutions tailored for LLMs. These services are designed to optimize the performance of LLM applications, offering specialized hardware and software configurations that can significantly reduce training and inference times.

Training Frameworks

For the development and fine-tuning of LLMs, deep learning frameworks are used. These include:

- **PyTorch**: A popular choice among researchers and developers for training LLMs due to its flexibility, ease of use, and dynamic computational graph. PyTorch

supports a wide range of LLM architectures and provides tools for efficient model training and fine-tuning.
- **TensorFlow**: Another widely used framework that offers robust support for LLM training and deployment. TensorFlow is known for its scalability and is suited for both research prototypes and production deployments.

💡 Note that LLM API applications, such as those leveraging RAG, typically do not require direct access to computational resources for training since they use pre-trained models provided via an API. In these cases, the focus is more on integrating the API into the application and possibly using orchestration tools to manage interactions with the model.

4. Experimentation Tools

Experimentation tools are pivotal for LLM applications, as they facilitate the exploration and optimization of hyperparameters, fine-tuning techniques, and the models themselves. These tools help track and manage the multitude of experiments that are part of developing and refining LLM applications, enabling more systematic and data-driven approaches to model improvement.

💡 It's important to note that the mentioned tools are primarily beneficial for scenarios involving the fine-tuning or training of models, where experimentation is key. If you're working on applications, these tools might not hold the same level of utility since the LLM operates as a black box. In such cases, the LLM's inner workings and training processes are managed externally, and the focus shifts towards optimizing the use of the model through APIs rather than directly manipulating its training or fine-tuning parameters.

The below are some experimentation tools

- **Experiment Tracking**: Tools like **Weights & Biases** provide platforms for tracking experiments, including changes in hyperparameters, model architectures, and performance metrics over time. This facilitates a more organized approach to experimentation, helping developers to identify the most effective configurations.
- **Model Development and Hosting**: Platforms like **Hugging Face** and **MLFlow** offer ecosystems for developing, sharing, and deploying ML models, including custom LLMs. These services simplify access to model repositories (model hubs), computing resources, and deployment capabilities, streamlining the development cycle.
- **Performance Evaluation**: Tools like **Statsig** offer capabilities for evaluating model performance in a live production environment, allowing developers to conduct A/B tests and gather real-world feedback on model behavior.

Application Tools

1. Hosting

Developers leveraging open-source models have a range of hosting services at their disposal. Innovations from companies like OctoML have expanded hosting capabilities beyond traditional server setups, enabling deployment on edge devices and directly within browsers. This shift not only enhances privacy and security but also serves to reduce

latency and costs. Hosting platforms like Replicate are incorporating tools designed to simplify the integration and utilization of these models for software developers, reflecting a belief in the potential of smaller, finely tuned models to achieve top-tier accuracy within specific domains.

Beyond the LLM components, the static elements of LLM applications—essentially, everything excluding the model itself—also require hosting solutions. Common choices include platforms like Vercel and services provided by major cloud providers. Yet, the landscape is evolving with the emergence of startups like Steamship and Streamlit, which offer end-to-end hosting solutions tailored for LLM applications, indicating a broadening of hosting options to support the diverse needs of developers.

2. Monitoring

Monitoring and observability tools are essential for maintaining and improving applications, especially after deployment in production. These tools enable developers to track key metrics such as the model's performance, cost, latency, and overall behavior. Insights gained from these metrics are invaluable for guiding the iteration of prompts and further experimentation with models, ensuring that the application remains efficient, cost-effective, and aligned with user needs.

One notable development in this area is the launch of **LangKit by WhyLabs**. LangKit is specifically designed to offer developers enhanced visibility into the quality of model outputs.

Some other examples:

Gantry offers a holistic approach to understanding model performance by tracking inputs and outputs alongside relevant metadata and user feedback. It assists in uncovering how models function in real-world scenarios, identifying errors, and spotting underperforming cohorts or use cases.

Helicone is designed to offer actionable insights into application performance with minimal setup. It enables real-time monitoring of model interactions, helping developers understand how their models are performing across different metrics. By logging inputs, outputs, and enriching them with metadata and user feedback, Helicone provides a comprehensive view of model behavior.

Output Tools

1. Evaluation

When developing applications with LLMs, developers often navigate a complex balance among model performance, inference cost, and latency. Strategies to enhance one aspect, such as iterating on prompts, fine-tuning the model, or switching model providers, can impact the others. Given the probabilistic nature of LLMs and the variability in tasks they perform, assessing performance becomes a critical challenge. To aid in this process, a range of evaluation tools have been developed. These tools assist in refining prompts, tracking

experimentation, and monitoring model performance, both offline and online. Here's an overview of the types of tools available:

For those looking to optimize the interaction with LLMs, No Code / Low Code prompt engineering tools are invaluable. They allow developers and prompt engineers to experiment with different prompts and compare outputs across various models without deep coding requirements. Some examples of such tools include Humanloop, PromptLayer etc.

Once deployed, it's important to continually monitor an LLM application's performance in the real world. Performance monitoring tools offer insights into how well the model is performing against key metrics, identify potential degradation over time, and highlight areas for improvement. These tools can alert developers to issues that may affect user experience or operational costs, enabling timely adjustments to maintain or enhance the application's effectiveness. Some performance monitoring tools include Honeyhive and Scale AI.

The infographic below provides a summary of the tools available for each component of the LLM application process.

Read/Watch These Resources (Optional)

1. https://www.secopsolution.com/blog/top-10-llm-tools-in-2024
2. https://www.sequoiacap.com/article/llm-stack-perspective/
3. https://www.codesmith.io/blog/introducing-the-emerging-llm-tech-stack
4. https://stackshare.io/index/llm-tools