10 essential Python libraries for building LLM applications

Here's a list of 10 Python libraries that help build LLM applications, from RAGs and agents to model implementation and evaluation.

Building applications based on a large language model (LLM) is completely different from using end-user tools like ChatGPT, Claude Code, or the Codex. Those products are convenient for direct use, but when you start building your own LLM system, you'll need much more control over how things work behind the scenes.

This often means working with libraries and frameworks that support a wide range of tasks: from loading open-source models, building retrieval-augmented generation (RAG) pipelines, deploying models via APIs, fine-tuning private data, creating agent-based workflows, to evaluating system performance. The challenge lies in the fact that developing LLM applications is not simply about writing prompts, but about a system of many interconnected components. When all of these are combined into a stable pipeline, complexity can increase rapidly.

In this context, Python libraries act as an 'infrastructure' layer that simplifies the process. Below are 10 exemplary libraries that help you build LLM applications faster and more robustly, whether you're testing local models, building production systems, or developing multi-agent applications.

Transformers

Transformers is a central library in the open-source LLM ecosystem. It's practically the default starting point when you want to load models, tokenize data, run inferences, or fine-tune your own data.

Many popular models like GLM, Minimax, and Qwen are used through Transformers, and many other tools in the LLM ecosystem are also designed to be compatible with it. Transformers' strength lies in eliminating much of the low-level model configuration work. Instead of building everything from scratch, you can work with a unified interface for many different models and tasks, making testing, verification, and deployment significantly easier.

LangChain

LangChain becomes useful when LLM applications are no longer simply about sending a prompt and receiving a response. In practice, LLM systems often need to connect multiple components such as prompts, retrievers, APIs, external tools, and multiple model calls within a single process.

LangChain provides a structure to connect all those components into a complete flow. This is why it's widely used in chatbots, RAG systems, and agent-based applications. Instead of having to manually connect each logic step, LangChain helps manage multi-step processes, connect to external systems, and build applications that are more complex than simply creating text.

LlamaIndex

If LangChain helps connect the components within an application, then LlamaIndex helps connect the application to the data.

This library is particularly useful in RAG systems, where the model needs to retrieve information from documents, PDFs, databases, or other knowledge sources before providing a response. This is crucial because most practical LLM applications cannot rely solely on the model's 'memory'.

By 'grounding' the answers into real data, LlamaIndex makes the results more relevant, up-to-date, and suitable for systems such as internal assistants, enterprise knowledge repositories, or document processing workflows.

vLLM

vLLM is one of the most popular libraries for efficiently implementing open-source LLM models.

It's designed to optimize inference speed, utilize GPU memory more efficiently, and support high throughput, making model operation more realistic rather than just experimental. In practice, good model 'serve' is a crucial part of building LLM applications.

vLLM makes it easier for open-source models to be deployed at scale, handle more requests, and respond faster, so it's often used when transitioning from the testing phase to production.

Unsloth

Unsloth is a popular choice for fine-tuning, especially for small groups or individuals.

This library is notable for techniques such as LoRA and QLoRA, which allow for faster model training or tuning while using less VRAM compared to traditional fine-tuning methods. This significantly reduces costs when demanding model customization is desired.

Instead of requiring large hardware infrastructure, Unsloth allows developers to fine-tune models more realistically on limited resources, which is why it is becoming increasingly popular.

CrewAI

CrewAI is a framework for building multi-agent systems, where each agent takes on a specific role and task.

Instead of having a single model handle all the work, CrewAI allows multiple agents to collaborate, using tools and working in a structured workflow. This reflects the new trend in LLM applications: no longer simple chatbots, but systems that coordinate multiple components.

CrewAI is particularly useful when a problem requires planning, task allocation, or processing by specific roles.

AutoGPT

AutoGPT was one of the first prominent projects to bring the concept of automated agents closer to the community.

It allows for the creation of systems that can be planned, break down goals into multiple steps, and perform actions with minimal user interaction. This is a prime example of an automated workflow agent.

The key to AutoGPT is its ability to execute tasks objectively, manage multiple steps, and automate lengthy processes, rather than relying solely on simple chat-based interaction.

LangGraph

LangGraph is designed for situations requiring more detailed control over how LLM applications operate.

Instead of a linear pipeline, LangGraph allows for the construction of workflows with state, logical branches, memory, and multiple complex processing steps. This is well-suited for advanced agent systems or long-term tasks.

This library helps developers define how data and logic move through the system, track state, and manage processes as complexity increases.

DeepEval

DeepEval is a Python framework specifically designed for testing and evaluating LLM applications.

Instead of simply checking whether the model responds, DeepEval allows measurement of factors such as relevance, hallucination, fidelity, and task completion. This is especially important once the application begins to be used in real-world scenarios.

DeepEval provides a structured approach to evaluating prompts, pipeline RAGs, and workflow agents, making systems more reliable before and after deployment.

OpenAI Python SDK

The OpenAI Python SDK is the fastest way to integrate LLM into your application without having to run the model yourself.

It provides a simple interface for Python developers to work with OpenAI-hosted models, enabling them to quickly build features such as chat, reasoning, image processing, or multimodal experiences.

SDKs' strengths lie in their speed and simplicity. Instead of dealing with infrastructure, scaling, or inference, developers can focus on product logic—and that's why they remain a popular choice for API-based LLM applications.

Building an LLM application is not just about writing a prompt, but about combining many components into a complete system. From loading the model, processing data, deploying, to evaluating, each step has its own level of complexity.

The aforementioned Python libraries simplify each part of that process, while providing a foundation for building stable and scalable applications. As the LLM ecosystem continues to evolve, choosing the right tools will be just as important as choosing the right model.