LLM—

Large Language Models (LLMs) are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. LLMs can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that is coherent and grammatically accurate.

**Learning Objectives**

* Understand the concept of Large Language Models (LLMs) and their importance in natural language processing.
* Know about different types of popular LLMs, such as BERT, GPT-3, and T5.
* Discuss the applications and use cases of Open Source LLMs.
* Hugging Face APIs for LLMs.
* Explore the future implications of LLMs, including their potential impact on job markets, communication, and society as a whole.

This article was published as a part of the [Data Science Blogathon](https://analyticsvidhya.com/blogathon).

Langchain—

Langchain is a framework for developing applications powered by language models. It enables applications that:

Area Context- Awair : connect a language model to sources of context (prompt instructions, few shot examples, content to ground its response in, etc.)

Off- the self chain- a structured assembly of components for accomplishing specific higher-level tasks.

There are two types of language models, which in LangChain are called:

* LLMs: this is a language model which takes a string as input and returns a string
* ChatModels: this is a language model which takes a list of messages as input and returns a message

The input/output for LLMs is simple and easy to understand - a string. But what about ChatModels? The input there is a list of ChatMessages, and the output is a single ChatMessage. A ChatMessage has two required components:

* content: This is the content of the message.
* role: This is the role of the entity from which the ChatMessage is coming from.

LangChain provides several objects to easily distinguish between different roles:

* HumanMessage: A ChatMessage coming from a human/user.
* AIMessage: A ChatMessage coming from an AI/assistant.
* SystemMessage: A ChatMessage coming from the system.
* FunctionMessage: A ChatMessage coming from a function call.

If none of those roles sound right, there is also a ChatMessage class where you can specify the role manually. For more information on how to use these different messages most effectively, see our prompting guide.

Langchain provides a common interface that's shared by both LLMs and ChatModels. However it's useful to understand this difference in order to construct prompts for a given language model.

The standard interface that LangChain provides has two methods:

* predict: Takes in a string, returns a string
* predict\_messages: Takes in a list of messages, returns a message.

Let's see how to work with these different types of models and these different types of inputs. First, let's import an LLM and a ChatModel.

from langchain.llms import OpenAI  
from langchain.chat\_models import ChatOpenAI  
  
llm = OpenAI()  
chat\_model = ChatOpenAI()  
  
llm.predict("hi!")  
>>> "Hi"  
  
chat\_model.predict("hi!")  
>>> "Hi"

Models—

LangChain provides interfaces and integrations for two types of models:

* [LLMs](https://python.langchain.com/docs/modules/model_io/models/llms/): Models that take a text string as input and return a text string
* [Chat models](https://python.langchain.com/docs/modules/model_io/models/chat/): Models that are backed by a language model but take a list of Chat Messages as input and return a Chat Message

## LLMs vs chat models[​](https://python.langchain.com/docs/modules/model_io/models/#llms-vs-chat-models)

LLMs and chat models are subtly but importantly different. LLMs in LangChain refer to pure text completion models. The APIs they wrap take a string prompt as input and output a string completion. OpenAI's GPT-3 is implemented as an LLM. Chat models are often backed by LLMs but tuned specifically for having conversations. And, crucially, their provider APIs use a different interface than pure text completion models. Instead of a single string, they take a list of chat messages as input. Usually these messages are labeled with the speaker (usually one of "System", "AI", and "Human"). And they return an AI chat message as output. GPT-4 and Anthropic's Claude are both implemented as chat models.

To make it possible to swap LLMs and chat models, both implement the Base Language Model interface. This includes common methods "predict", which takes a string and returns a string, and "predict messages", which takes messages and returns a message. If you are using a specific model it's recommended you use the methods specific to that model class (i.e., "predict" for LLMs and "predict messages" for chat models), but if you're creating an application that should work with different types of models the shared interface can be helpful.

Dataset—

LLMs consist of multiple hidden layers of deep neural networks, which extract and train their parameters from a significant amount of data sources.  If you train LLMs with questionable datasets, they will be impacted by performance issues like bias and overfitting. Conversely, training a deep learning model with [high-quality datasets](https://kili-technology.com/platform/explore-and-fix) enables a more accurate and coherent output.

Leading organizations have realized that good language modeling needs more than state-of-the-art machine learning models and training methods. Curating and annotating a diverse training dataset that fairly represents the model’s domain is equally important in implementing neural network artificial intelligence solutions in various industries.

For example, Bloomberg trained a transformer architecture from scratch with decades-worth of carefully curated financial data. The resulting [BloombergGPT](https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/) allows the financial company to empower its clients and perform existing financial-specific NLP tasks faster and with more accuracy. Likewise, HuggingFace has developed a programmer-friendly model [StarCode](https://huggingface.co/bigcode/starcoder),

Text Generation—

Large Language Models (LLMs) are foundational machine learning models that use deep learning algorithms to process and understand natural language. These models are trained on massive amounts of text data to learn patterns and entity relationships in the language. LLMs can perform many types of language tasks, such as translating languages, analyzing sentiments, chatbot conversations, and more. They can understand complex textual data, identify entities and relationships between them, and generate new text that is coherent and grammatically accurate.

Text2 Image Generation—

Text-to-image generation is **a machine learning model that creates an image from a textual description**. The model takes a natural language description as input and produces an image that matches that description.

**Text-to-image generation is important because it:**

* Converts textual descriptions into visual representations
* Enables efficient communication and creative expression
* Simplifies content creation and design processes

Text-to-image models began to be developed in the mid-2010s, as a result of advances in deep neural networks.

**Some text-to-image generation tools include:**

* Text2Image

An AI-based tool that allows users to generate images based on a textual description

* TEXT2IMAGE

An online tool by Ted Davis that renders an abstract image that is the translation of the given input

* Picsart for Business

A Text2Image API that can create endless iterations of unique design for marketing and business materials

* Runway

An AI image generator that can create an image from scratch by entering descriptive text

Text-to-image

LLm Index—

LlamaIndex, previously known as the GPT Index, is a remarkable data framework aimed at helping you build applications with LLMs by providing essential tools that facilitate data ingestion, structuring, retrieval, and integration with various application frameworks. The capabilities offered by LlamaIndex are numerous and highly valuable:

✅ Ingest from different data sources and data formats using Data connectors (Llama Hub).  
✅ Enable document operations such as inserting, deleting, updating, and refreshing the document index.  
✅ Support synthesis over heterogeneous data and multiple documents.  
✅ Use “Router” to pick between different query engines.  
✅ Allow for the hypothetical document embeddings to enhance output quality  
✅ Offer a wide range of integrations with various vector stores, ChatGPT plugins, tracing tools, and LangChain, among others.  
✅ Support the brand new OpenAI function calling API.

These are just a few examples of the extensive capabilities provided by LlamaIndex. In this blog post, we will explore some of the functionalities that I find exceptionally useful with LlamaIndex.