LLM ROADMAP

BEGINNER TO ADVANCED

By Youssef Hosni

LLM Roadmap from Absolute Beginner to Advanced

Mastering LLM: A Comprehensive Guide from Novice to Expert

Youssef Hosni

This book is devoted to my father, mother, wife, and my daughter who are my everything and have supported me in every move. My father, I hope you are in a better place.

Youssef

Table of Contents

Introduction 8

Part I: LLM Basics & Architecture 9

Part II: Building & Training LLM From Scratch 11

1. Best Resources on Building Datasets to Trian LLMs 12

1.1. Dataset Hubs 12

1.2. Building an Instruction Dataset 13

1.3. Datasets Preparing & Processing 14

2. Mastering LLM Fine-Tuning: Top Learning Resources 14

2.1. Introduction to LLM Fine-Tuning 15

2.2. Parameter Efficient Fine-Tuning (PEFT) 16

2.3. LLMs Instruction Tuning 17

2.4. LLM Fine-Tuning in Practice 17

3. 14 Free Large Language Models Fine-Tuning Notebooks 18

3.1. Fine-Tuning Large Language Models with LoRA and Hugging Face 18

3.2. Fine-Tuning Llama 2 Model in a Colab Notebook 19

3.3. Guanaco Chatbot Demo with LLaMA-7B Model 20

3.4. PEFT Finetune-Bloom-560m-tagger 20

3.5. Fine-Tuning Meta_OPT-6–1b Model with bnb_peft 21

3.6. Fine-Tuning Falcon-7b with BNB Self-Supervised Training 21

3.7. Fine-Tuning LLaMa2 with QLoRa 22

3.8. Stable Vicuna 1 3B-8bit in Google Colab 23

3.9. GPT-Neo-X 20B bnb2bit Training 24

3.10. MPT-Instruct-30B Model Training 24

3.11. RLHF Training for Custom Dataset for Any Model 25

3.12. Fine-Tuning Microsoft Phi 1.5 On Custom Dataset 25

3.13. Fine-Tuning OpenAI GPT3.5 Turbo 26

3.14. Finetuning Mistral-7b using Autotrain Advanced 26

4. Best Resources to Learn & Understand Evaluating LLMs 27

4.1. Overview of LLM Evaluation Methods 27
4.2. LLM Benchmarking 28
4.3. LLM Evaluation Methods 28
4.4. Evaluating Chatbots 28
4.5. Evaluating RAG Applications 29
4.6. Automated Testing for LLMs 29
5. Overview of LLM Quantization Techniques & Where to Learn Each of Them? 30
5.1. Introduction to Quantization 30
5.2. GGUF 31
5.3. Activation-aware Weight Quantization (AWQ) 32
5.4. Post-Training Model Quantization (PTQ) 33
5.5. Accurate Post-Training Quantization for Generative Pre-trained Transformers (GPTQ) 34
5.6. Quantization-Aware Training (QAT) 35
6. Top Resources to Learn & Understand RLHF & LLM Alignment 36
6.1. What is RLHF? 36
6.2. Important Blogs 37
6.3. Important Videos & Talks 38
6.4. Important Research Papers 39
7. How to Stay Updated with LLM Research & Industry News? 40
7.1. LLM Research Leaders 41
7.2. LLM Industry Leaders 42
7.3. LLM Research & Industry Organizations 45
7.4. LLM Influencers & Content Creators 48
7. 5. Newsletters & Blogs for LLM Research News 49
7.6. Newsletters & Blogs for LLM Industry News 51
Part III: LLMs In Production 53
1. Best Resources to Learn Prompt Engineering 54
1.1. Awesome GPT Prompt Engineering GitHub Repository 54

1.2. Prompt Engineering Guide....55
1.2. Prompt Engineering for LLMs....56
1.4. Maximizing the Potential of LLMs: A Guide to Prompt Engineering....57
1.5. ChatGPT Prompt Engineering for Developers....57
2. 6 Resources to Master Vector Databases & Building a Vector Storage....58
2.1. Vector Databases: from Embeddings to Applications....59
2.2. Building Applications with Vector Databases....60
2.3. The Top 5 Vector Database Blog....61
2.4. LangChain — Text splitters....62
2.5. Sentence Transformers Library....63
3. Top Resources to Master RAG: From Basic Level to Advanced....65
3.1. Retrieval Augmented Generation Basics....65
3.2. More LangChains....66
3.3. Advanced RAG....67
3.4. RAG Evaluation....68
3.5. LLM Agents....69
4. 5 Free Tools to Run Large Language Models (LLM) Locally on Your Laptop....70
4.1. GPT4All....70
4.2. LM Studio....72
4.3. Ollama....72
4.4. LLaMA.cpp....74
4.5. NVIDIA Chat with RTX....77
5. Deploying LLMs: Top Learning & Educational Resources to Get Started....77
5.1. Local Deployment....78
5.2. Demo deployment....78
5.3. Server deployment....79
5.4. Edge deployment....80
6. Getting Started with LLM Inference Optimization: Best Resources....81
6.1. Understanding LLM Inference Optimization....81

6.2. Deep Understanding of LLM Inference Optimization 82
6.3. LLM Inference by Hugging Face 82
6.4. LLM Inference Optimization Libraries & Tools 83
7. What are LLMOps and How to Get Started with It? 83
7.1. Building LLM Applications for Production 84
7.2. Awesome LLMOps 85
7.3. LLMOps 87
7.4. Automated Testing for LLMOps 88
8. Securing LLMs: Best Learning & Educational Resources 89
8.1. List of the 10 most critical vulnerabilities seen in LLM applications 89
8.2. How to hack Google Bard, ChatGPT, or any other chatbot 90
8.3. Prompt Injection Primer for Engineers 90
8.4. Quality and Safety for LLM Applications 91
8.5. Red Teaming LLM Applications 92
8.6. LLM Security: A comprehensive list of resources and papers 93
Part IV: Building Your LLM Portfolio 94
1. 10 Large Language Models Projects Ideas To Build Your Portfolio 95
1.1. Content Generation 95
1.2. Language Translation and Localization 96
1.3. Entertainment and Gaming 98
1.4. Healthcare and Well-being 99
1.5. Social Good 100
1.6. Productivity and Utility 105
2. 10 Guided Large Language Models Projects to Build Your Portfolio 109
2.1. Building a Conversational Chatbot with Langchain and Large Language Models 109
2.2. YouTube Script Writer Assistant Using OpenAI API & Streamlit 110
2.3. Chat with your Document using OpenAI API & Streamlit 111
2.4. Building A Falcon-40B Instruct Chat Web Application using HuggingFace & Gardio 112

2.5. Building GPT Banker Using LLaMA 2 70B 112
2.6. Finetune Llama 2 On Your Local Machine Using HuggingFace Autotrain 112
2.7. Building a Lex Fridman Podcast Summarization App with Whisper Jax, Azure OpenAI, and Langchain 113
2.8. Creating a Veterinary Chatbot using Llama 2: Harnessing Gen AI for Pet Care ...114
2.9. Deploy Llama 2 on AWS SageMaker using DLC (Deep Learning Containers).....115
2.10. Article/Blog Generation App using Llama2, Langchain, and Pexels 115

Introduction

Large Language Models (LLMs) have transformed the landscape of AI, driving advancements in natural language processing and generating human-like text with unprecedented accuracy. It is now becoming an essential skill if you are joining the AI market and most of the open positions will require LLM-related skills.

Whether you're a beginner eager to dip your toes into the fascinating world of LLMs or an advanced practitioner aiming to refine your skills, this comprehensive roadmap will guide you through every stage of mastering these powerful models.

This roadmap is divided into four comprehensive sections. In the first section, you will grasp the foundational concepts of Large Language Models (LLMs) and delve into their core architectures, learning about key components like transformers, attention mechanisms, and tokenization.

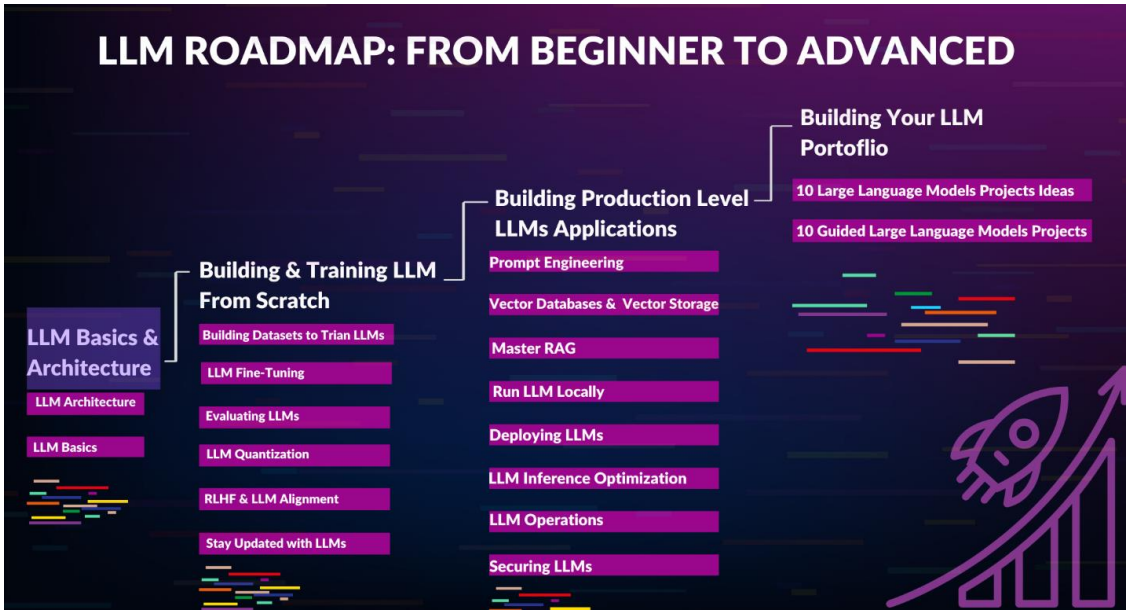
The second section guides you through the complete process of training and fine-tuning an LLM from scratch, starting with data preparation and moving on to training, fine-tuning, evaluating, and aligning the model for optimal performance.

In the third section, you'll focus on developing real-world applications using LLMs, beginning with prompt engineering and progressing to building Retrieval-Augmented Generation (RAG) applications, as well as learning best practices for deploying and optimizing LLMs in production environments.

The final section is dedicated to helping you build your portfolio, offering ten project ideas and ten guided projects to provide a solid starting point for showcasing your expertise to potential employers or clients.

Each section builds upon the previous one, ensuring a smooth and logical progression from basic concepts to advanced applications, ultimately equipping you with the essential knowledge and practical skills needed to master LLMs and leverage their capabilities effectively.

Part I: LLM Basics & Architecture



In the first section, you'll grasp the foundational concepts of Large Language Models (LLMs) and their core architectures, focusing on transformers, attention mechanisms, and tokenization.

Key resources include Andrej Karpathy's "Let's Build the GPT Tokenizer," Jay Alammar's "The Illustrated Transformer" and "The Illustrated GPT-2," and 3Blue1Brown's "Visual Intro to Transformers." You'll also explore "nanoGPT" by Karpathy, "Attention? Attention!" by Lilian Weng, various decoding strategies, Karpathy's "Intro to Large Language Models," and top practical and theoretical courses on LLMs. This section provides a blend of theoretical and useful insights, preparing you for the next sections.

Articles:

1. **The Illustrated Transformer** by Jay Alammar
2. **The Illustrated GPT-2** by Jay Alammar
3. **Attention? Attention!** by Lilian Weng
4. **Decoding Strategies in LLMs** by Maxime Lebonne

YouTube Videos:

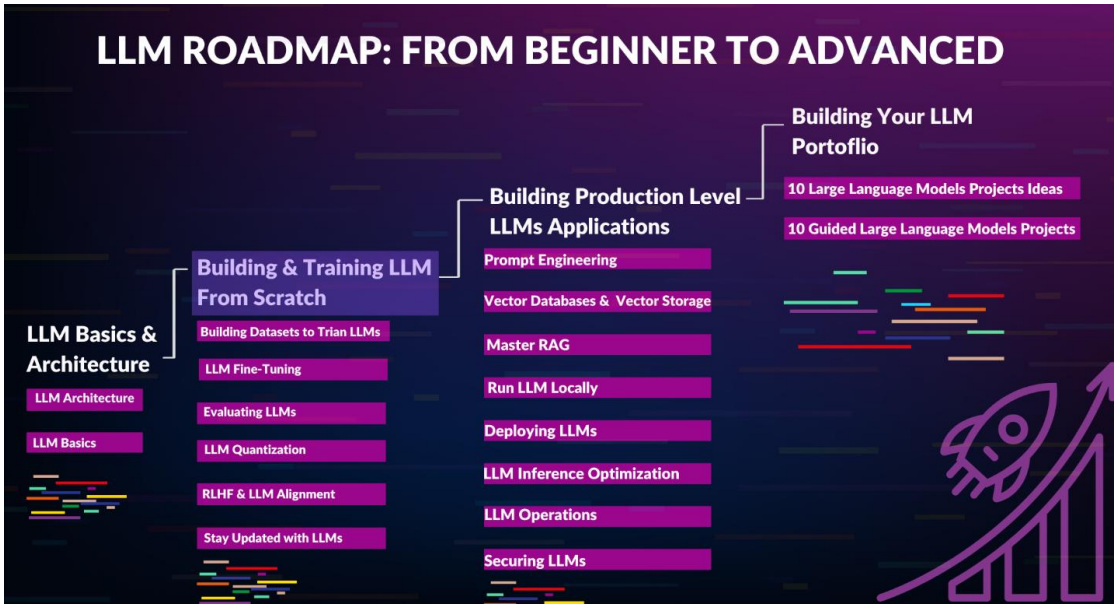
1. **Let's build the GPT Tokenizer by** Andrej Karpathy
2. **nanoGPT** by Andrej Karpathy
3. **Intro to Large Language Models** by Andrej Karpathy

4. **Visual Intro to Transformers** by 3Blue1Brown

Courses:

1. **Generative AI with Large Language Models**
2. **Advanced LLM Application Building**
3. **Full Stack LLM Bootcamp**
4. **Training & Fine-Tuning LLMs for Production**
5. **H2O.ai LLM Learning Path**

Part II: Building & Training LLM From Scratch



The second section guides you through the complete process of training and fine-tuning a Large Language Model (LLM) from scratch, covering every crucial step from data preparation to ensuring optimal model performance. This section begins with the best resources for building datasets to train LLMs, providing comprehensive guidance on collecting, cleaning, and organizing data for effective model training.

Next, you will delve into mastering the fine-tuning process with top learning resources, exploring techniques and strategies to adapt pre-trained models to specific tasks or domains. The section also includes 14 free LLM fine-tuning notebooks, offering practical, hands-on experience with fine-tuning processes.

Evaluating LLMs is a critical aspect covered in this section, with the best resources to learn and understand various evaluation metrics and methodologies, ensuring your model's performance meets the desired standards. Additionally, you will gain an overview of LLM quantization techniques, which help in optimizing models for efficiency and speed, along with resources for mastering each technique.

Understanding Reinforcement Learning from Human Feedback (RLHF) and LLM alignment is another key component, with top resources provided to deepen your knowledge in aligning models with human values and preferences. Finally, this section offers insights on how to stay updated with the latest LLM research and industry news, ensuring you remain at the forefront of advancements in the field.

By the end of this section, you will have a comprehensive understanding of how to build, train, fine-tune, evaluate, and optimize LLMs, equipped with the knowledge and practical skills to develop high-performing models tailored to specific needs.

1. Best Resources on Building Datasets to Trian LLMs

Large language models (LLMs), such as OpenAI's GPT series and Google's Bard, are driving profound technological changes. Recently, with the emergence of open-source large model frameworks like LlaMa and ChatGLM, training an LLM is no longer the exclusive domain of resource-rich companies.

Training LLMs by small organizations or individuals has become an important interest in the open-source community, with some notable works including Alpaca, Vicuna, and Luotuo.

In addition to large model frameworks, large-scale and high-quality training corpora are also essential for training large language models. Currently, relevant open-source corpora in the community are still scattered. Therefore, this section aims to introduce and collect high-quality resources to learn how to build training datasets for LLM applications.

1.1. Dataset Hubs

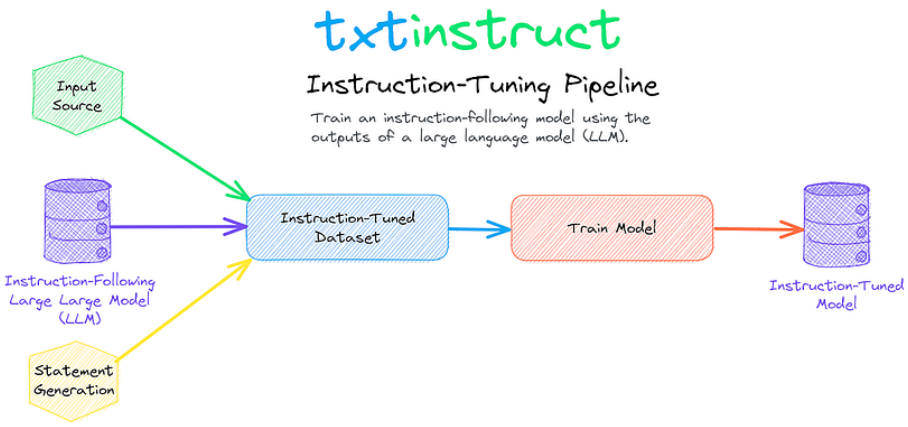
LLM datasets are extensive sets of text used to train large language models. These datasets typically contain texts in multiple languages, topics, and styles, used to train models to predict and generate text related to given input text. They are commonly employed for various natural language processing tasks like machine translation, summarization, question-answering systems, and more. The dataset hubs contain open-source datasets that are pivotal in training or fine-tuning many LLMs that ML engineers use today.

Resources:

1. LLMDataHub: Awesome Datasets for LLM Training

2. How-to-Use HuggingFace's Datasets
3. Open-Sourced Training Datasets for Large Language Models (LLMs)

1.2. Building an Instruction Dataset

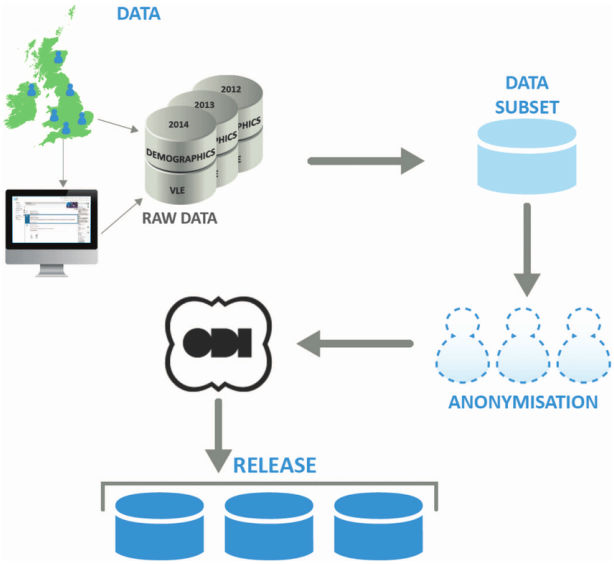


Training a chatbot LLM that can follow human instruction effectively requires access to high-quality datasets that cover a range of conversation domains and styles. In this section, we provide a curated collection of resource datasets specifically designed for building an instruction dataset for instruction-tuning LLM.

Resources:

1. How to Fine-Tune an LLM Part 1: Preparing a Dataset for Instruction Tuning
2. How I created an instruction dataset using GPT 3.5 to fine-tune Llama 2 for news classification
3. Dataset creation for fine-tuning LLM
4. How to Generate Instruction Datasets from Any Documents for LLM Fine-Tuning

1.3. Datasets Preparing & Processing



Enhancing LLM and RAG system's performance depends on efficiently processing diverse unstructured data sources. In this section, you'll learn techniques for representing all sorts of unstructured data, like text, images, and tables, from many different sources and implement them to extend your LLM RAG pipeline to include Excel, Word, PowerPoint, PDF, and EPUB files.

Resources:

1. How to Build an Effective Data Collection and Processing Strategy for LLM Training
2. Preprocessing Unstructured Data for LLM Applications

2. Mastering LLM Fine-Tuning: Top Learning Resources

Large language models (LLMs) have transformed the field of natural language processing with their advanced capabilities and highly sophisticated solutions.

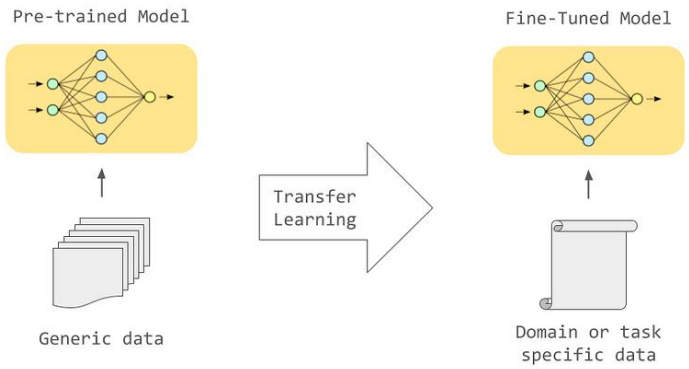
These models, trained on massive datasets of text, perform a wide range of tasks, including text generation, translation, summarization, and question-answering. But while LLMs are powerful tools, they're often incompatible with specific tasks or domains.

Fine-tuning allows users to adapt pre-trained LLMs to more specialized tasks. By fine-tuning a model on a small dataset of task-specific data, you can improve its performance on that task while preserving its general language knowledge.

In this section, we will provide the best learning resource to learn what fine-tuning is, how it works, and how fine-tuning LLMs can significantly improve model performance, reduce training costs, and enable more accurate and context-specific results.

Also, these resources will cover different fine-tuning techniques and applications to show how fine-tuning has become a critical component of LLM-powered solutions.

2.1. Introduction to LLM Fine-Tuning



LLMs are trained on massive datasets of text and can perform a wide range of tasks, including text generation, translation, summarization, and question-answering. But while LLMs are powerful tools, they're often incompatible with specific tasks or domains.

Fine-tuning allows users to adapt pre-trained LLMs to more specialized tasks. By fine-tuning a model on a small dataset of task-specific data, you can improve its performance on that task while preserving its general language knowledge. For example, a Google study found that fine-tuning a pre-trained LLM for sentiment analysis improved its accuracy by 10 percent.

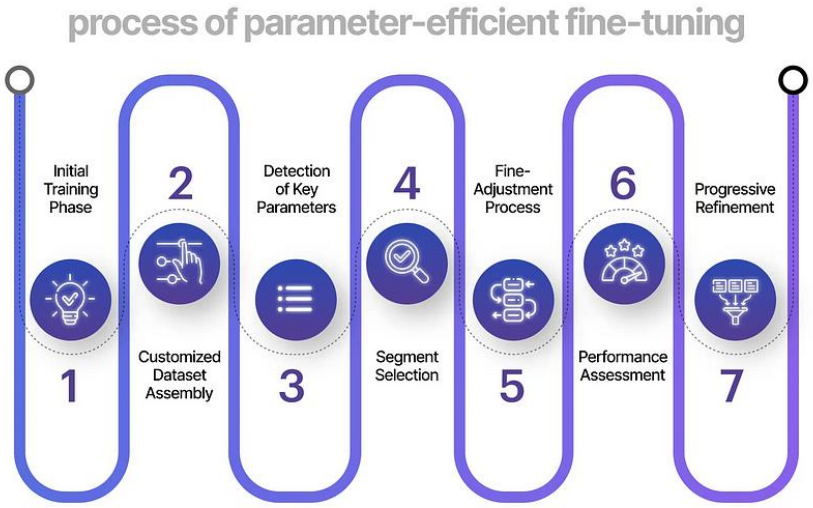
In this section, you will explore how fine-tuning LLMs can significantly improve model performance, reduce training costs, and enable more accurate and context-specific results.

You will also learn the different fine-tuning techniques and applications to show how fine-tuning has become a critical component of LLM-powered solutions.

Learning Resources:

1. The Novice's LLM Training Guide
2. Fine-Tuning LLMs: Overview, Methods, and Best Practices
3. Finetuning Large Language Models

2.2. Parameter Efficient Fine-Tuning (PEFT)



Transfer learning plays a crucial role in the development of large language models such as GPT-3 and BERT. It is an ML technique in which a model trained on a certain task is used as a starting point for a distinct but similar task.

The idea behind transfer learning is that the knowledge gained by a model from solving one problem can be leveraged to help solve another problem. However, with the parameter count of large language models reaching trillions, fine-tuning the entire model has become computationally expensive and often impractical.

In response, the focus has shifted towards in-context learning, where the model is provided with prompts for a given task and returns in-context updates. However, inefficiencies like processing the prompt each time the model makes a prediction and its poor performance at times make it a less favorable choice.

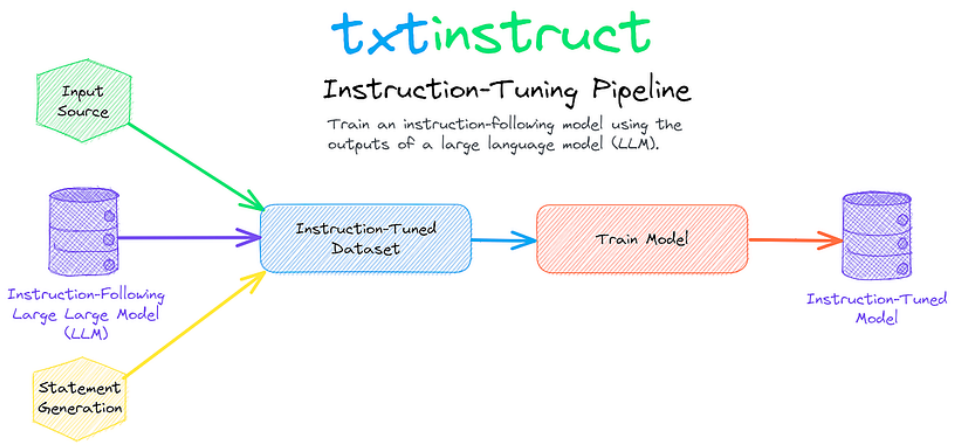
This is where Parameter-efficient Fine-tuning (PEFT) comes in as an alternative paradigm to prompting. PEFT aims to fine-tune only a small subset of the model's parameters, achieving comparable performance to full fine-tuning while significantly reducing computational requirements.

These learning resources will introduce you to the PEFT method in detail, exploring its benefits and how it has become an efficient way to fine-tune LLMs on downstream tasks. Also, it will discuss different methods of the PEFT with practical examples of each.

Learning Resources:

1. Optimizing Pre-trained Models: A Guide to Parameter-Efficient Fine-Tuning (PEFT)
2. QLoRA paper explained (Efficient Finetuning of Quantized LLMs)
3. QLoRA — How to Fine-tune an LLM on a Single GPU (w/ Python Code)
4. LoRA Fine-tuning & Hyperparameters Explained (in Plain English)
5. Fine-Tuning Mistral-7B with LoRA (Low-Rank Adaptation)
6. Finetuning LLMs with LoRA and QLoRA: Insights from Hundreds of Experiments

2.3. LLMs Instruction Tuning



Instruction tuning represents a specialized form of fine-tuning in which a model is trained using pairs of input-output instructions, enabling it to learn specific tasks guided by these instructions.

Learning Resources:

1. Building with Instruction-Tuned LLMs: A Step-by-Step Guide

2.4. LLM Fine-Tuning in Practice

Finally, it's time to put what you have learned into practice and explore different fine-tuning techniques using different datasets for different domains and contexts.

Learning Resources:

1. A Beginner's Guide to LLM Fine-Tuning with Axolotl
2. Fine-Tune Your Own Llama 2 Model in a Colab Notebook
3. 14 Free Large Language Models Fine-Tuning Notebooks

3. 14 Free Large Language Models Fine-Tuning Notebooks

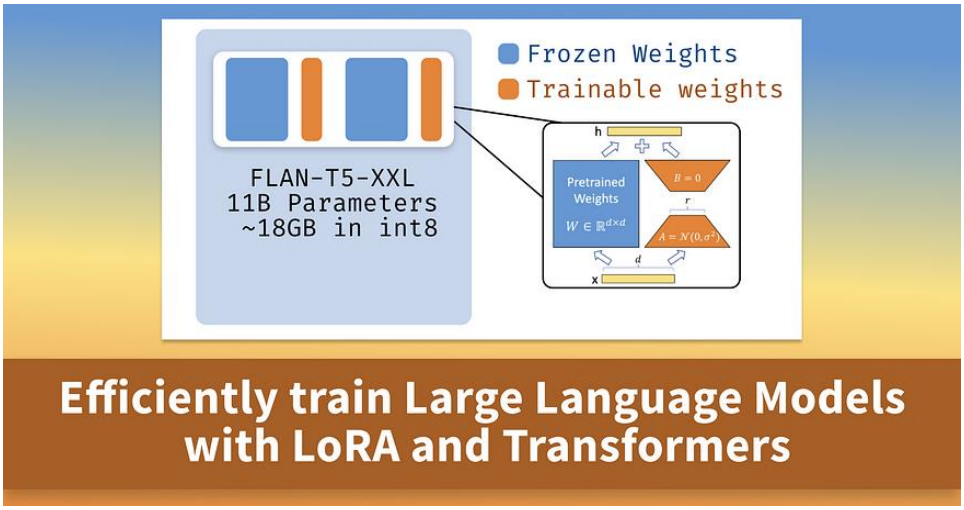
Fine-tuning large language models (LLMs) has become a crucial skill for NLP practitioners, enabling customization and improved performance across various tasks. This article introduces 14 free Colab notebooks that provide hands-on experience in fine-tuning LLMs.

From efficient training methodologies like LoRA and Hugging Face to specialized models such as Llama, Guanaco, and Falcon, each notebook explores unique aspects of the fine-tuning process. Advanced techniques like PEFT Finetune, Bloom-560m-tagger, and Meta_OPT-6–1b_Model offer insights into state-of-the-art approaches.

Whether you're interested in GPT-Neo-X, MPT-Instruct-30B, or Microsoft Phi 15B, these notebooks cover a diverse range of LLMs, making them suitable for both beginners and experienced practitioners. Delve into custom dataset training, self-supervised methods, and RLHF techniques, gaining a comprehensive understanding of fine-tuning.

This section provides a roadmap to navigate these notebooks, making it an essential read for anyone keen on mastering the art of fine-tuning large language models.

3.1. Fine-Tuning Large Language Models with LoRA and Hugging Face



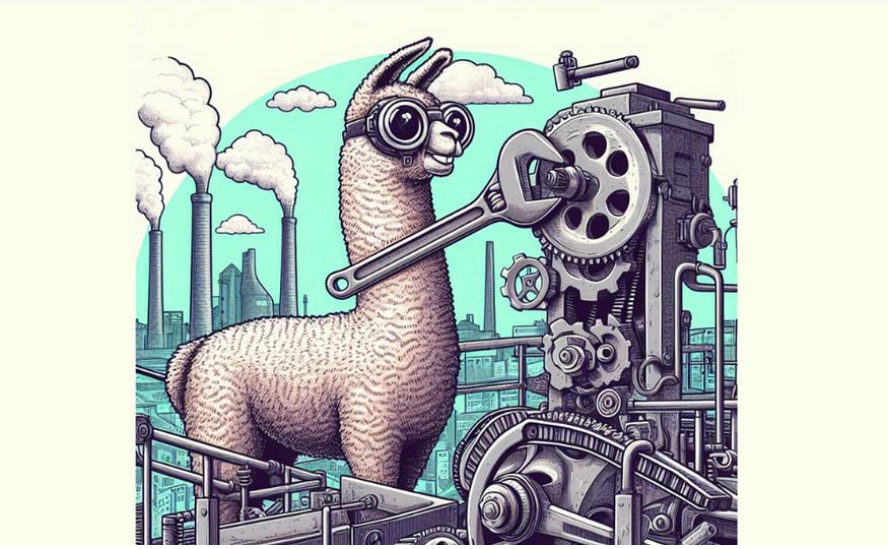
This **colab notebook** outlines efficiently training large language models using LoRA (Layer-wise Recall-oriented Attention) and Hugging Face's libraries. The example focuses on fine-tuning the FLAN-T5-XXL model on the samsum dataset for abstractive summarization tasks. The notebook covers the following:

1. **Setup Development Environment:** The notebook begins by setting up the development environment. It utilizes the PyTorch Deep Learning AMI with pre-

installed CUDA drivers and PyTorch. Additional Hugging Face Libraries, including transformers and datasets, are installed using pip.

2. **Load and Prepare the Dataset**: The notebook uses the samsum dataset, a collection of messenger-like conversations with summaries. The dataset is loaded using the Hugging Face Datasets library. Tokenization and preprocessing steps are demonstrated to convert inputs to token IDs.
3. **Fine-Tune T5 with LoRA and bnb int-8**: This section demonstrates the fine-tuning process using LoRA and bnb int-8 quantization. The FLAN-T5-XXL model is loaded from the Hugging Face Hub, and LoRA configuration is applied to train only a fraction of the model's parameters, achieving significant memory reduction.
4. **Evaluate & Run Inference with LoRA FLAN-T5:** The notebook showcases evaluation and inference steps for the fine-tuned model. The rogue score is used as a metric for summarization quality. A sample summarization is demonstrated, and the model is evaluated against the test set. The results show a significant performance improvement compared to a fully fine-tuned smaller model.
5. **Save Model and Wrap-Up:** The final part covers saving the trained LoRA model and tokenizer results, offering the option to upload them to the Hugging Face Hub. The notebook concludes by briefly comparing resource requirements and costs for training the FLAN-T5-XXL model using traditional methods.

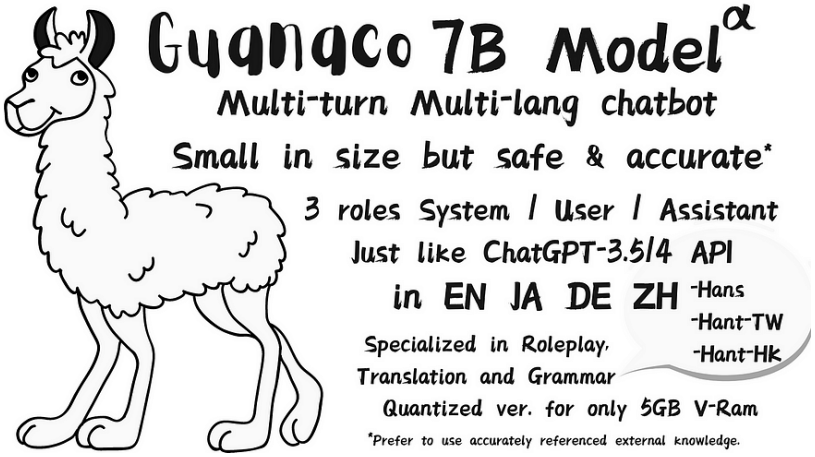
3.2. Fine-Tuning Llama 2 Model in a Colab Notebook



This **colab notebook** provides a comprehensive guide to fine-tuning the Llama 2 language model using Google colab. Llama 2, pre-trained on an extensive dataset of 2 trillion tokens, presents a resource-intensive and time-consuming pretraining process. This tutorial focuses on instruction fine-tuning, exploring techniques like Supervised Fine-Tuning (SFT)

and Reinforcement Learning from Human Feedback (RLHF) to enhance the model's performance.

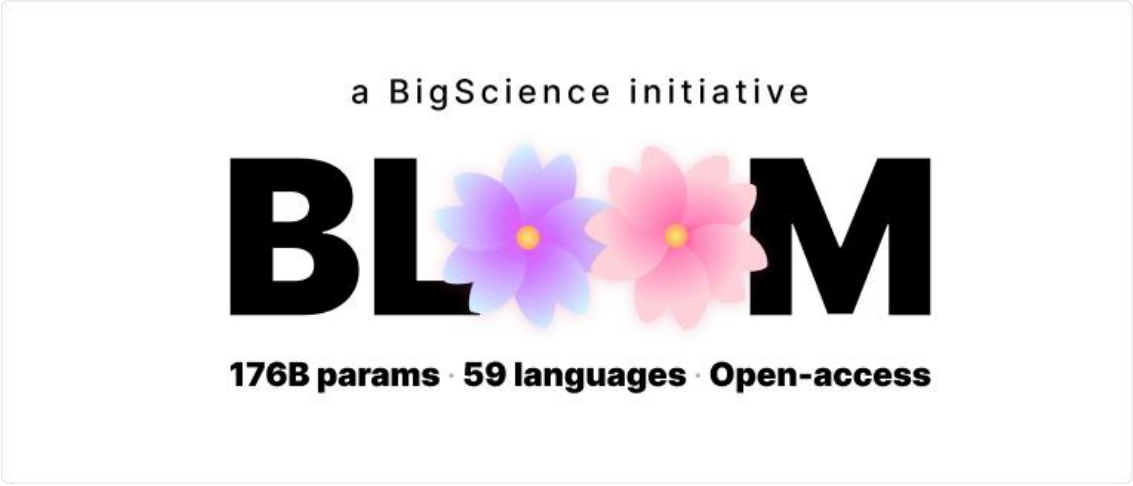
3.3. Guanaco Chatbot Demo with LLaMA-7B Model



This **notebook** demonstrates the implementation of a chatbot named Guanaco, powered by the LLaMA-7B language model. The chatbot interacts with users through a user-friendly Gradio interface, generating responses based on input conversation history. By breaking down the code into various sections, we've explored how the model is loaded, the chatbot's behavior is defined, and the Gradio interface is set up for user interaction.

It's important to note that this demo showcases the capabilities of the LLaMA-7B model and its interaction with users. However, as with any AI-generated content, users should be aware that the model's responses may not always be entirely accurate or appropriate, and they should exercise caution when using the chatbot for factual information or important decisions.

3.4. PEFT Finetune-Bloom-560m-tagger



This **notebook** serves as a comprehensive guide to leveraging Hugging Face's PEFT (Parameter-Efficient Fine-Tuning) and bitsandbytes libraries to fine-tune a LoRa (Localized Randomization) checkpoint. The primary objective is to demonstrate the efficient training of a language model while preserving memory and computation resources.

The notebook provides a step-by-step guide, code snippets, and explanations to facilitate users in understanding and implementing PEFT and LoRa techniques for fine-tuning language models. The emphasis is on resource-efficient training, making it suitable for various natural language processing tasks with reduced computational demands.

3.5. Fine-Tuning Meta_OPT-6–1b Model with bnb_peft

In this **notebook**, we will cover how we can fine-tune large language models using the very recent `peft` library and `bitsandbytes` for loading large models in 8-bit.

The fine-tuning method will rely on a recent method called **(LoRA)**, instead of fine-tuning the entire model you just have to fine-tune these adapters and load them properly inside the model. After fine-tuning the model you can also share your adapters on the 🤗 Hub and load them very easily.

3.6. Fine-Tuning Falcon-7b with BNB Self-Supervised Training

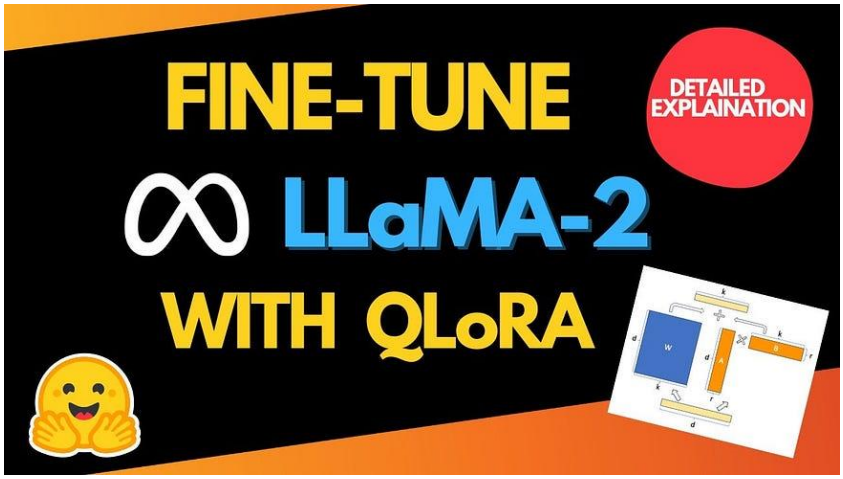
This **Google Colab notebook** provides a step-by-step guide for fine-tuning the advanced Falcon-7b language model on a single Colab instance to create a powerful chatbot.

Leveraging the PEFT library from Hugging Face and QLoRA for memory efficiency, the notebook covers setup, library installations, dataset selection (using the Guanaco dataset), loading the Falcon 7B model, and configuring the trainer with TRL's `SFTTrainer`.

The training process is encapsulated in a series of well-documented code cells, ensuring users can easily follow along. The notebook concludes by pushing the fine-tuned model to

the Hugging Face Model Hub for convenient sharing and integration, with the entire process designed for accessibility and adaptability to diverse use cases.

3.7. Fine-Tuning LLaMa2 with QLoRa



This **Google Colab notebook** provides a step-by-step guide for fine-tuning the Llama-2–7b language model on a single Colab instance, transforming it into a chatbot. Leveraging the PEFT library and QLoRA for memory efficiency, the tutorial covers setup, dataset selection, model loading, and training configuration.

The notebook utilizes 'bitsandbytes' for 4-bit quantization and 'einops' for model loading requirements. Training is configured using `TrainingArguments` with the help of the `SFTTrainer` from TRL library. The fine-tuned model is then pushed to the Hugging Face Model Hub for easy sharing and integration. The guide concludes with an example of generating chatbot responses using the trained Llama-2-7b model.

3.8. Stable Vicuna 1 3B-8bit in Google Colab

This **notebook** introduces and demonstrates the StableVicuna model, a 13-billion parameter Reinforcement Learning from Human Feedback (RLHF) chat model.

Utilizing the Transformers library, the notebook showcases the setup, configuration, and fine-tuning details of the model. It establishes a text generation pipeline with controlled parameters for response quality. The notebook includes functions for standardized prompt creation, and response extraction, and showcases the model's performance through timed evaluations of diverse prompts.

Interactive queries cover a spectrum of topics, illustrating the model's versatility and reasoning capabilities. Overall, the notebook provides a comprehensive exploration of StableVicuna's capabilities in generating human-like responses across various conversational scenarios.

3.9. GPT-Neo-X 20B bnb2bit Training

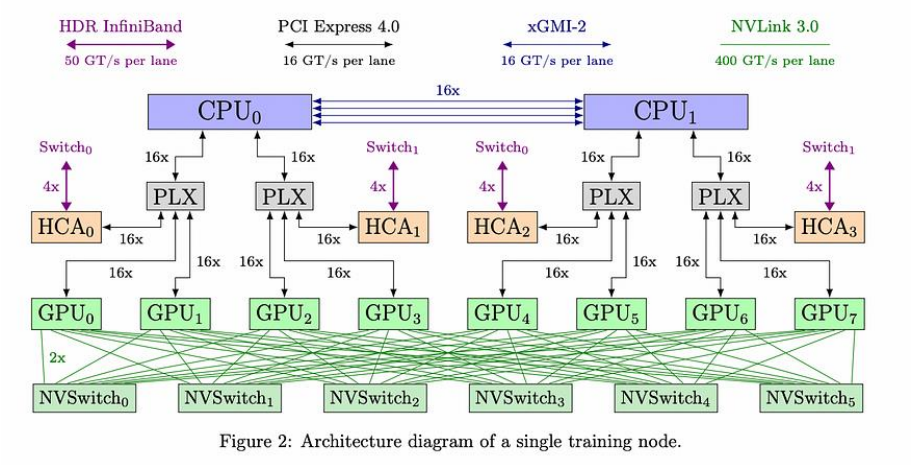


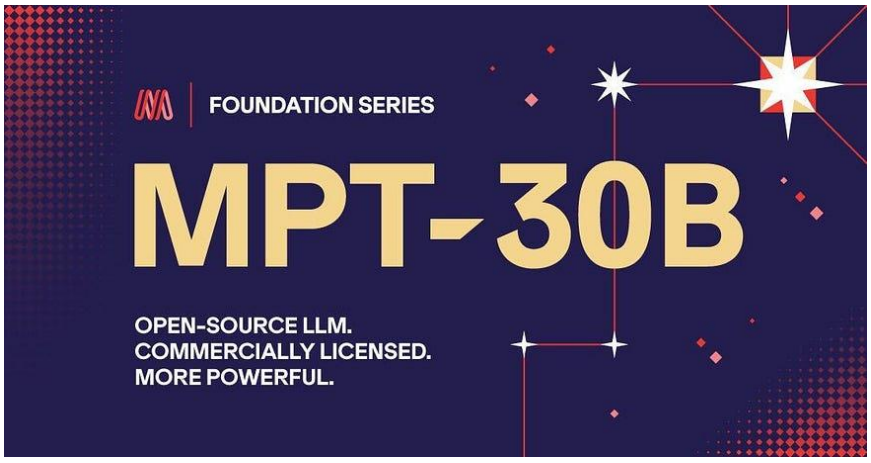
Figure 2: Architecture diagram of a single training node.

This **notebook** introduces the integration of `bitsandbytes` into the `transformers` library, aiming to democratize Large Language Models (LLMs) through 4-bit quantization techniques.

The notebook guides users in loading and training a large 4-bit model (`gpt-neo-x-20b`) using Google Colab and the PEFT library. The tutorial covers proper model loading, and preprocessing for training, and showcases the training process using the `transformers` Trainer class.

The notebook emphasizes accessibility and usability, and the training steps are kept minimal for demonstration, providing a valuable resource for practitioners interested in leveraging 4-bit quantization in language model workflows.

3.10. MPT-Instruct-30B Model Training



This **notebook** provides a detailed guide on training the MPT-30B model for natural language processing tasks. It demonstrates loading the model with specific configurations, including bfloat16 data type and 4-bit quantization. The notebook showcases the model's abilities through prompt-based interactions, emphasizing its proficiency in understanding and responding to various instructions.

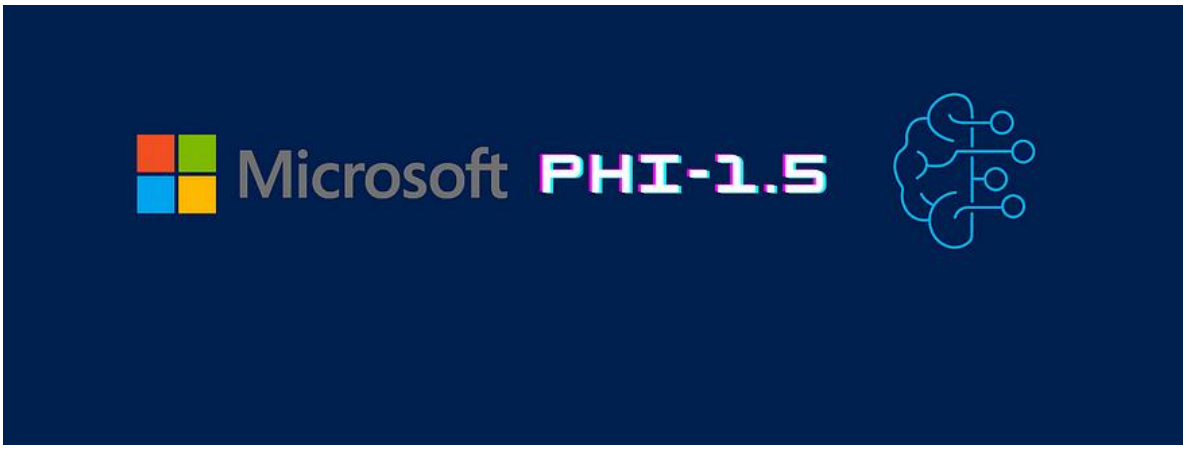
3.11. RLHF Training for Custom Dataset for Any Model

This **notebook** focuses on training and using the "bigcode/tiny_starcoder_py" model for summarization tasks. The notebook imports necessary libraries, loads datasets for training and testing, tokenizes the dataset, and formats it for model input. It then proceeds to train the model using a Reward Trainer, evaluating its performance.

The notebook also introduces a Policy Model using Proximal Policy Optimization (PPO) for further training. The PPO Trainer is configured with specific parameters, and the training loop involves generating responses, sentiment analysis, and PPO training steps. The model is saved after training.

Finally, the notebook includes code for inference using the trained model, generating responses to given prompts. It showcases the functionality of the model and provides insights into its performance on summarization tasks.

3.12. Fine-Tuning Microsoft Phi 1.5 On Custom Dataset

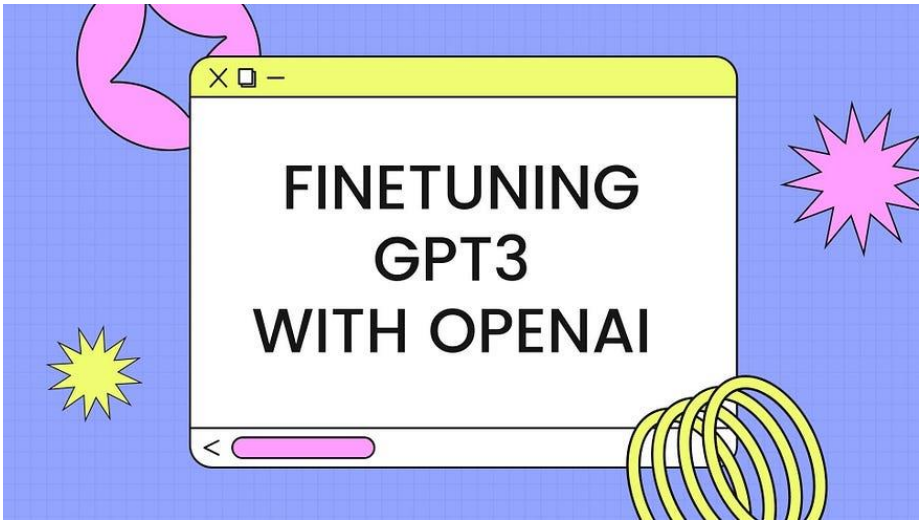


This **notebook** outlines the process of fine-tuning the Microsoft Phi-1.5 language model on a custom dataset using the DialogStudio task. The notebook covers steps such as data loading, preprocessing, model configuration, training setup, and inference.

It uses the trl library for fine-tuning and Hugging Face Hub for model versioning. Key details include the use of 4-bit quantization for the Bits and Bytes model and the incorporation of

the PeftModel for merging the fine-tuned model with the original Phi-1.5 model. The notebook provides references to relevant datasets and models.

3.13. Fine-Tuning OpenAI GPT3.5 Turbo



This **notebook** provides a detailed guide on fine-tuning the GPT-3.5-turbo model using OpenAI's API. The process involves preparing custom datasets, tokenizing conversations, and addressing potential format errors.

The notebook covers considerations for token counting, pricing estimation, and default epoch settings. It includes instructions on uploading data to OpenAI and creating a fine-tuning job with details such as training and validation file IDs.

The notebook concludes with generating responses using the newly fine-tuned model and comparing it with the original GPT-3.5-turbo.

3.14. Finetuning Mistral-7b using Autotrain Advanced

This **notebook** provides a step-by-step guide on fine-tuning the Mistral 7B model using AutoTrain. The process involves setting up the runtime environment, connecting to the Hugging Face hub for model upload, and uploading the dataset.

The notebook includes an overview of the AutoTrain command with detailed explanations of its flags. Users are guided through necessary steps before running the fine-tuning process, such as specifying the project name, and repository ID, and confirming the dataset location.

After completion, the notebook covers information about the inference engine and loads the fine-tuned model using the Peft Model. The notebook concludes with an example of generating text with the newly fine-tuned Mistral 7B model.

4. Best Resources to Learn & Understand Evaluating LLMs

Large language models (LLMs) are gaining increasing popularity in both academia and industry, owing to their unprecedented performance in various applications.

As LLMs continue to play a vital role in both research and daily use, their evaluation becomes increasingly critical, not only at the task level but also at the societal level for a better understanding of their potential risks. Over the past years, significant efforts have been made to examine LLMs from various perspectives.

This article presents a comprehensive set of resources that will help you understand LLM evaluation starting from what to evaluate, where to evaluate, and how to evaluate.

4.1. Overview of LLM Evaluation Methods

With the rapid advancement and integration of large language models (LLMs) in business workflows, ensuring these models are reliable and efficient has become critical. This need underscores the significance of understanding and deploying robust evaluation and benchmarking techniques for successful model implementation.

LLMs are evaluated and benchmarked on various tasks such as language generation, translation, reasoning, summarization, question-answering, and relevance. A representative set of evaluations helps build well-rounded, robust, and secure models across different dimensions and detects any regressions over a period of time.

In this blog, we explore the nuances of evaluation metrics, the significance of LLM benchmarks in quantifying model performance, and the challenges associated with building standardized metrics. We also touch upon the latest trends in benchmarking and provide a comprehensive guide on building effective evaluation protocols.

Educational Resources:

1. **Understanding LLM Evaluation and Benchmarks: A Complete Guide**
2. **Decoding LLM Performance: A Guide to Evaluating LLM Applications**
3. **A Survey on Evaluation of LLMs**
4. **Evaluating and Debugging Generative AI**

4.2. LLM Benchmarking

It is evident that merely training LLMs is not sufficient. Thus, the question arises: How can we confidently assert that LLM 'A' (with 'n' number of parameters) is superior to LLM 'B' (with 'm' parameters)? Or is LLM 'A' more reliable than LLM 'B' based on quantifiable, reasonable observations? There needs to be a standard to benchmark LLMs, ensuring they are ethically reliable and factually performant.

In this section, you will learn about the current evaluation paradigm and understand the terminologies of LLM benchmarking/evaluation. You will also learn about some prominent research on evaluating benchmarking and comparing LLMs on various tasks or scenarios.

Educational Resources:

1. **The Definitive Guide to LLM Benchmarking**

4.3. LLM Evaluation Methods

In the era of artificial intelligence and machine learning, evaluating the performance of models is crucial for their development and improvement. Large Language Models (LLMs) have shown incredible capabilities in generating human-like text, and their application has been extended to code generation.

In this section, you will explore different LLM evaluation methods for different use cases. Starting with traditional ones such as BLEU to code generation evaluation metrics such as HumanEval.

Educational Resources:

2. **BLEU at your own risk by Rachael Tatman**
3. **Perplexity of fixed-length models**
4. **HumanEval: Decoding the LLM Benchmark for Code Generation**

4.4. Evaluating Chatbots

Following the great success of ChatGPT, there has been a proliferation of open-source large language models that are finetuned to follow instructions. These models are capable of

providing valuable assistance in response to users' questions/prompts. Notable examples include Alpaca and Vicuna, based on LLaMA, and OpenAssistant and Dolly, based on Pythia.

Despite the constant release of new models every week, the community faces a challenge in benchmarking these models effectively. Benchmarking LLM assistants is extremely challenging because the problems can be open-ended, and it is very difficult to write a program to automatically evaluate the response quality. In this case, we typically have to resort to human evaluation based on pairwise comparison.

In this section, you will learn about the Elo rating system, which is a widely used rating system in chess and other competitive games. The Elo rating system is promising to provide the desired property mentioned above.

Educational Resources:

1. **Chatbot Arena: Benchmarking LLMs in the Wild with Elo Ratings**
2. **Chatbot Arena Leaderboard**

4.5. Evaluating RAG Applications

Retrieval Augmented Generation (RAG) stands out as one of the most popular use cases of large language models (LLMs). This method facilitates the integration of an LLM with an organization's proprietary data. Therefore it is important to evaluate and track experimenting to improve your RAG pipeline's performance. Also to understand the RAG triad: Context Relevance, Groundedness, and Answer Relevance, which are methods to evaluate the relevance and truthfulness of your LLM's response.

Educational Resources:

1. **Building and Evaluating Advanced RAG Applications**

4.6. Automated Testing for LLMs

When building applications with generative AI, model behavior is less predictable than traditional software. That's why systematic testing can make an even bigger difference in saving you development time and cost.

Continuous integration, a key part of LLMOps, is the practice of making small changes to software in development and thoroughly testing them to catch issues early when they are easier to fix.

With a robust automated testing pipeline, you'll be able to isolate bugs before they accumulate — when they're easier and less costly to fix. Automated testing lets your team focus on building new features so that you can iterate and ship products faster.

Educational Resources:

1. **Automated Testing for LLMOps**

5. Overview of LLM Quantization Techniques & Where to Learn Each of Them?

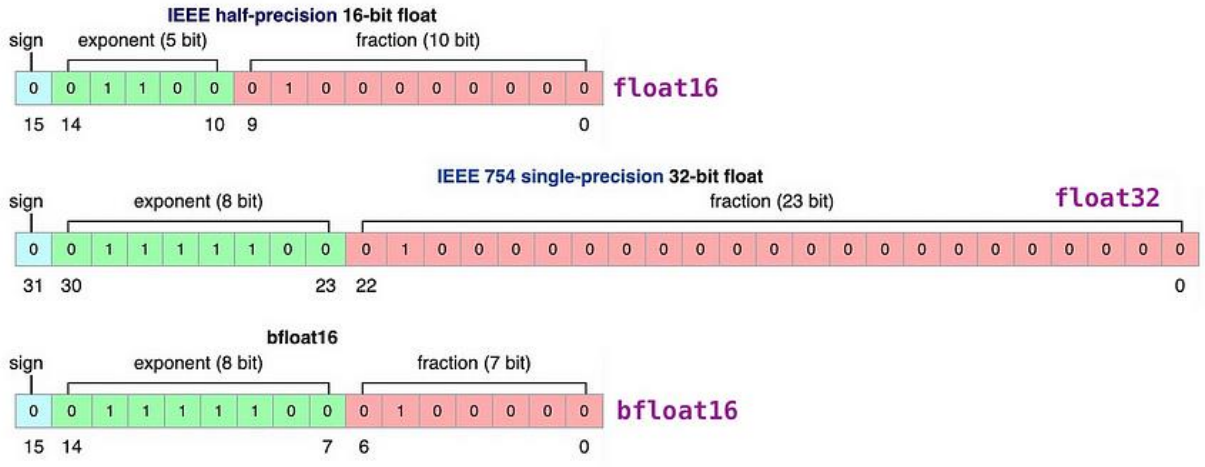
Model Quantization enhances the efficiency of large language models (LLMs) by representing their parameters in low-precision data types. This article presents an overview of LLM quantization techniques and resources for learning each of them.

This section covers different quantization methods, including GGUF, AWQ, PTQ, GPTQ, and QAT, elucidating their mechanisms and applications in LLM optimization.

Each sub-section provides learning resources, including tutorials, specifications, and practical guides, facilitating a deeper understanding of the quantization techniques.

This section serves as a comprehensive guide for individuals interested in exploring LLM quantization, offering insights into various techniques and resources for continued learning and professional development.

5.1. Introduction to Quantization



Model Quantization is a topic that has been gaining popularity recently. The concept of quantization in AI or specifically neural networks, is a technique to represent the weights,

biases, and activations in low-precision data types like 8-bit integer (int8) instead of the usual 32-bit floating point (float32). The two most common quantization cases are `float32 -> float16` and `float32 -> int8`. In this section, you will be introduced to model quantization and what are the main techniques for it:

Learning Resources:

2. **What are Quantized LLMs?**
3. **A Guide to Quantization in LLMs**
4. **Introduction to Weight Quantization**
5. **LLM Quantization | GPTQ | QAT | AWQ | GGUF | GGML | PTQ**
6. **Which Quantization Method is Right for You? (GPTQ vs. GGUF vs. AWQ)**
7. **Model Quantization Methods In TensorFlow Lite**
8. **ExLlamaV2: The Fastest Library to Run LLMs**
9. **Democratizing LLMs: 4-bit Quantization for Optimal LLM Inference**

5.2. GGUF

GGML is a C library focused on machine learning. It was created by Georgi Gerganov, which is what the initials “GG” stand for. This library not only provides foundational elements for machine learning, such as tensors but also a **unique binary format** to distribute LLMs.

This format recently changed to **GGUF**. This new format is designed to be extensible so that new features don't break compatibility with existing models. It also centralizes all the metadata in one file, such as special tokens, RoPE scaling parameters, etc.

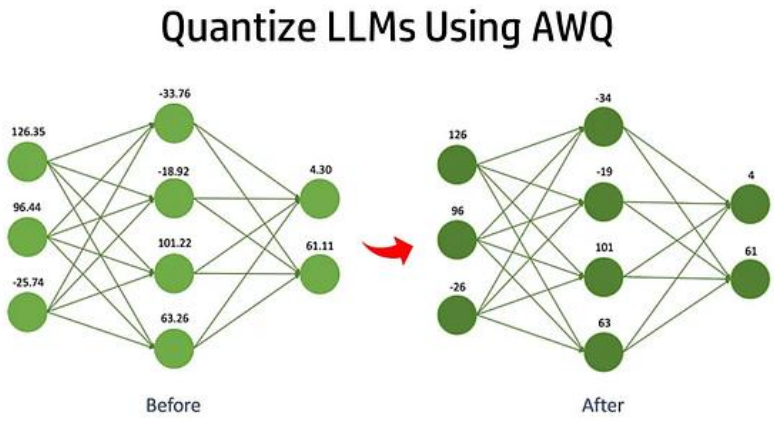
In short, it answers a few historical pain points and should be future-proof. For more information, you can read the specification at this address. GGML was designed to be used in conjunction with the llama.cpp library, also created by Georgi Gerganov.

The library is written in C/C++ for efficient inference of Llama models. It can load GGML models and **run them on a CPU**. Originally, this was the main difference with GPTQ models, which are loaded and run on a GPU. However, you can now offload some layers of your LLM to the GPU with llama.cpp. To give you an example, there are 35 layers for a 7b parameter model. This drastically speeds up inference and allows you to run LLMs that don't fit in your VRAM.

Learning Resources:

1. **Quantize Llama models with GGUF and llama.cpp**
2. **How to Quantize an LLM with GGUF or AWQ**

5.3. Activation-aware Weight Quantization (AWQ)



AWQ takes the concept of weight quantization to the next level by considering the activations of the model during the quantization process. In traditional weight quantization, the weights are quantized independently of the data they process. In AWQ, the quantization process takes into account the actual data distribution in the activations produced by the model during inference.

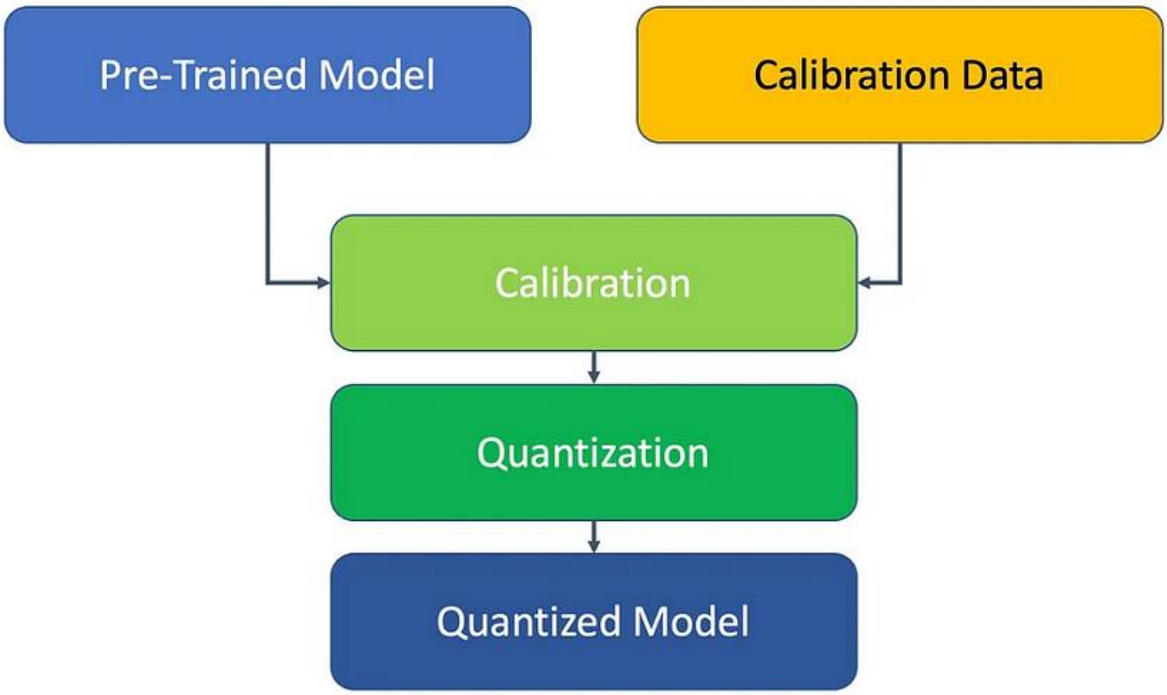
Here's how AWQ works:

- **Collect Activation Statistics**: During a calibration phase, a subset of the data is used to collect statistics on the activations produced by the model. This involves running the model on this data and recording the range of values and the distribution of activations.
- **Searching Weight Quantization Parameters:** Weights are quantized by taking the activation statistics into account. Concretely, we perform a space search for quantization parameters (e.g., scales and zero points), to minimize the distortions incurred by quantization on output activations. As a result, the quantized weights can be accurately represented with fewer bits.
- **Quantizing:** With the quantization parameters in place, the model weights are quantized using a reduced number of bits.

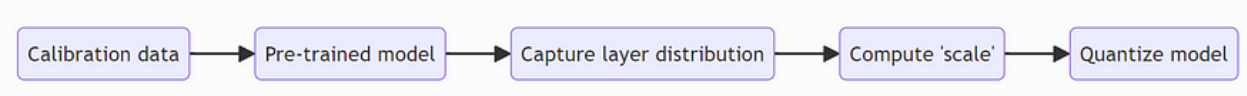
Learning Resources:

1. **AWQ for LLM Quantization**
2. **Understanding Activation-Aware Weight Quantization (AWQ): Boosting Inference Serving Efficiency in LLMs**
3. **How to Quantize an LLM with GGUF or AWQ**

5.4. Post-Training Model Quantization (PTQ)



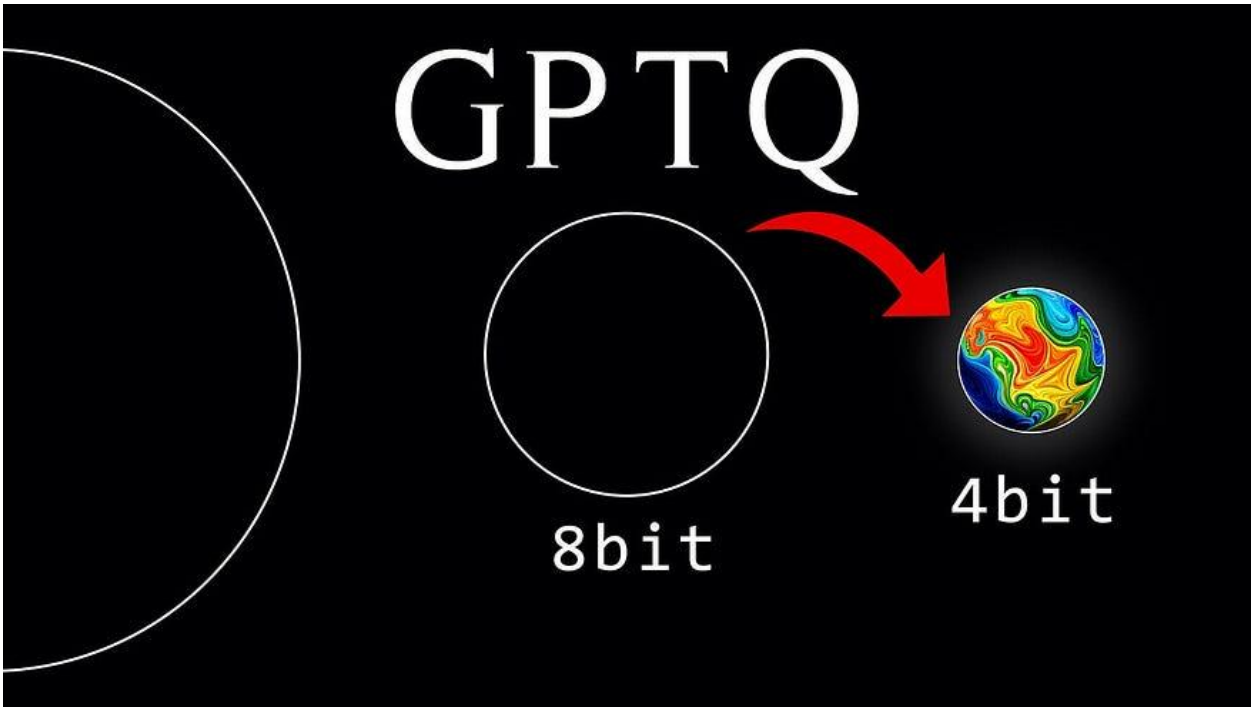
Post Training Quantization computes the scale after the network has been trained. A representative dataset is used to capture the distribution of activations for each activation tensor, then this distribution data is used to compute the scale value for each tensor. Each weight distribution is used to compute the weight scale.



Learning Resources:

1. **Post training Quantization**
2. **Diving deeper into Quantization Realm: Post-Training Magic (PTQ)**
3. **Post Training Quantization with OpenVINO Toolkit**

5.5. Accurate Post-Training Quantization for Generative Pre-trained Transformers (GPTQ)



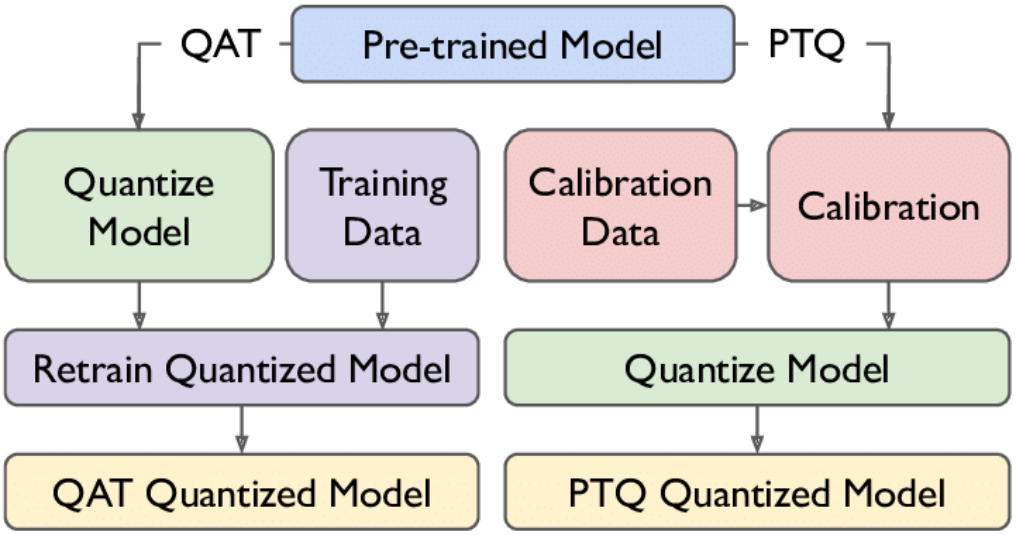
GPTQ is a post-training quantization (PTQ) method to make the model smaller with a calibration dataset. The idea behind GPTQ is very simple: it quantizes each weight by finding a compressed version of that weight, that will yield a minimum mean squared error. The GPTQ algorithm requires calibrating the quantized weights of the model by making inferences on the quantized model.

The effectiveness of quantization greatly depends on samples for evaluating and refining their quality. These samples serve as a basis for comparing the outputs of the original and quantized models. By using a higher number of samples, the potential for precise and impactful comparisons increases, subsequently enhancing the quality of quantization.

Learning Resources:

1. WTH is LLM quantization? 4bit GPTQ?
2. LLM Quantization w/ QLoRA, GPTQ and Llamacpp, LLama 2
3. Optimize open LLMs using GPTQ and Hugging Face Optimum
4. 4-bit Quantization with GPTQ

5.6. Quantization-Aware Training (QAT)



Quantization Aware Training (QAT) aims at computing scale factors during training. Once the network is fully trained, Quantize (Q) and Dequantize (DQ) nodes are inserted into the graph following a specific set of rules.

The network is then further trained for a few epochs in a process called Fine-Tuning. Q/DQ nodes simulate quantization loss and add it to the training loss during fine-tuning, making the network more resilient to quantization. In other words, QAT can better preserve accuracy when compared to PTQ.

Learning Resources:

1. Quantization aware Training — Concepts

2. Quantization Aware Training (QAT)
3. Quantization-aware training comprehensive guide

6. Top Resources to Learn & Understand RLHF & LLM Alignment

Reinforcement Learning from Human Feedback (RLHF) has become one of the main building blocks of building chatbots and large language applications. This section aims to provide you with a curated list of top resources that will help you learn and understand RLHF in depth. Whether you are a researcher, developer, or simply curious about this exciting field, this compilation of blogs, videos, talks, and research papers will serve as a comprehensive guide on RLHF.

6.1. What is RLHF?

Reinforcement Learning from Human Feedback (RLHF) is a branch of machine learning that combines reinforcement learning (RL) algorithms with human guidance or feedback to improve the learning process. In RLHF, instead of relying solely on an environmental reward signal, the learning agent interacts with human experts who provide feedback or demonstrations to guide the learning process.

The primary motivation behind RLHF is to enable machines to learn complex tasks more efficiently and effectively by leveraging human expertise. While RL algorithms can learn from trial and error, they can require a large number of interactions with the environment to achieve desirable performance. By incorporating human feedback, RLHF aims to reduce the number of interactions needed and accelerate the learning process.

RLHF is used in the development of chatbots to enhance their performance and improve their ability to interact with users. Here are some reasons why RLHF is used in developing chatbots:

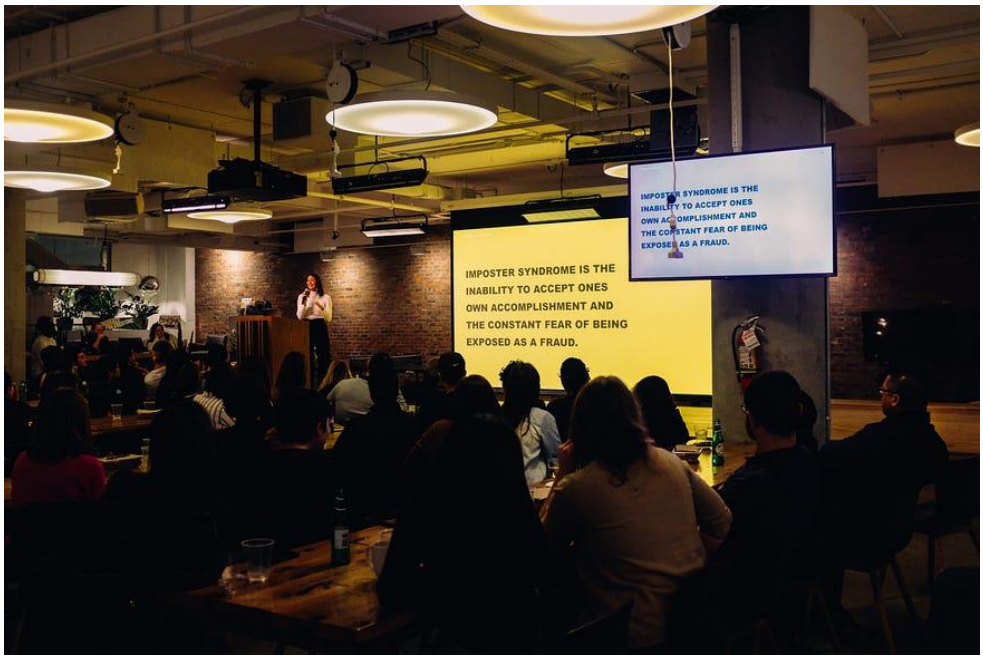
1. **User Satisfaction**: Chatbots aim to provide helpful and engaging conversations with users. By incorporating RLHF, chatbots can learn from human feedback and adapt their responses based on user preferences, leading to more satisfying interactions. Human feedback helps the chatbot understand what kind of responses are desirable and how to improve over time.
2. **Rapid Learning**: Training chatbots solely through traditional methods, such as rule-based systems or supervised learning, can be time-consuming and limited in their ability to handle diverse user inputs. RLHF enables chatbots to learn directly from interactions with human experts, reducing the reliance on large amounts of pre-existing data. This accelerates the learning process and allows chatbots to quickly adapt to new situations.

3. **Handling Uncertainty**: Chatbot conversations can often involve ambiguity and uncertainty. RLHF provides a mechanism for chatbots to seek clarification and guidance from human experts when faced with challenging or unfamiliar user inputs. This helps the chatbot make more informed decisions and provide accurate responses, even in uncertain situations.
4. **Personalization**: Chatbots that can understand and adapt to individual user preferences can deliver a more personalized experience. RLHF allows chatbots to learn from user feedback and tailor their responses based on individual preferences, improving user satisfaction and engagement. This personalization enhances the chatbot's ability to understand and fulfill user needs.
5. **Continuous Improvement:** Chatbots can benefit from continuous learning and improvement based on ongoing user interactions. RLHF enables chatbots to receive feedback from users in real-time, helping them refine their responses and behavior. This iterative learning process allows chatbots to continually enhance their performance and adapt to evolving user requirements.
6. **Ethical Considerations**: Chatbots that rely solely on pre-existing data may inherit biases or produce inappropriate responses. RLHF provides an opportunity to incorporate human guidance and ensure that the chatbot's behavior aligns with ethical standards. Human experts can help shape the chatbot's responses, ensuring they are fair, unbiased, and respectful.

6.2. Important Blogs

1. **How RLHF actually works by Nathan Lambert:** This article will build the intuition behind RLHF and how and why it works in a simple and straightforward way.
2. **RLHF: Reinforcement Learning from Human Feedback by Chip Huyen:** how exactly does RLHF work? Why does it work? This post will discuss the answers to those questions with a focus on theoretical details.
3. **StackLLaMA: A hands-on guide to train LLaMA with RLHF by HuggingFace:** In this blog post, we show all the steps involved in training a **LlaMa model** to answer questions on **Stack Exchange** with RLHF through a combination of Supervised Fine-tuning (SFT), Reward/preference modeling (RM) and Reinforcement Learning from Human Feedback (RLHF)

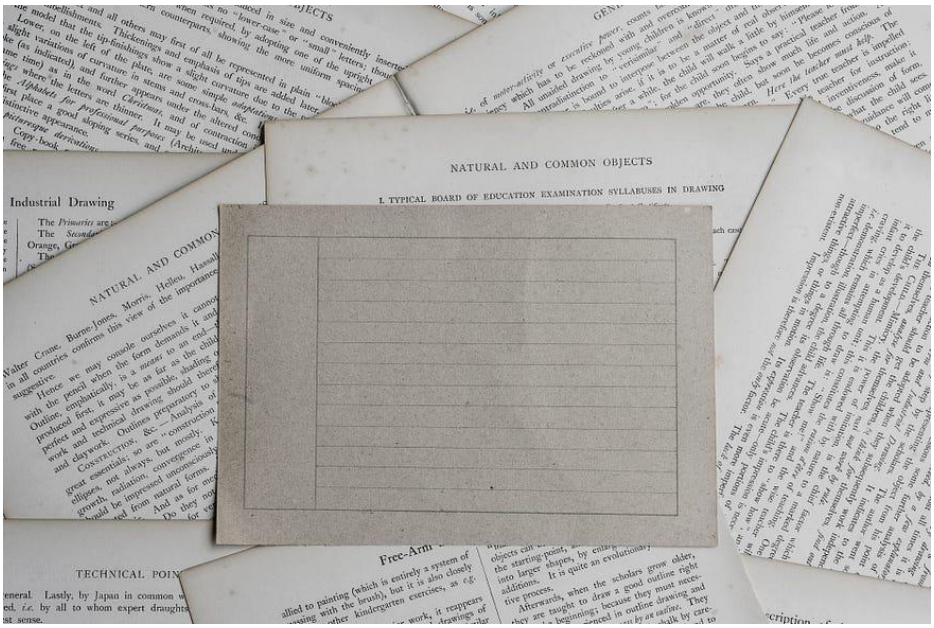
6.3. Important Videos & Talks



1. **Reinforcement Learning from Human Feedback: From Zero to chatGPT:** In this talk, speakers will cover the basics of Reinforcement Learning from Human Feedback (RLHF) and how this technology is being used to enable state-of-the-art ML tools like ChatGPT. Most of the talk will be an overview of the interconnected ML models and cover the basics of Natural Language Processing and RL that one needs to understand how RLHF is used on large language models. It will conclude with an open question in RLHF.
2. **State of GPT by Andrej Karpathy:** In this talk, you will learn about the training pipeline of GPT assistants like ChatGPT, from tokenization to pretraining, supervised finetuning, and Reinforcement Learning from Human Feedback (RLHF). In addition to

that you will dive deeper into practical techniques and mental models for the effective use of these models, including prompting strategies, finetuning, the rapidly growing ecosystem of tools, and their future extensions.

6.4. Important Research Papers



1. **Deep reinforcement learning from human preferences by OpenAI**: In this work, authors explore goals defined in terms of (non-expert) human preferences between pairs of trajectory segments. They show that this approach can effectively solve complex RL tasks without access to the reward function, including Atari games and simulated robot locomotion while providing feedback on less than one percent of our agent's interactions with the environment. This reduces the cost of human oversight far enough that it can be practically applied to state-of-the-art RL systems. To demonstrate the flexibility of this approach, they show that we can successfully train complex novel behaviors with about an hour of human time. These behaviors and environments are considerably more complex than any that have been previously learned from human feedback.
2. **Learning to summarize from human feedback by OpenAI:** In this work, authors show that it is possible to significantly improve summary quality by training a model to optimize for human preferences. They collect a large, high-quality dataset of human comparisons between summaries, train a model to predict the human-preferred summary, and use that model as a reward function to fine-tune a summarization policy using reinforcement learning. They apply the method to a version of the TL;DR dataset of Reddit posts and find that our models significantly

outperform both human reference summaries and much larger models fine-tuned with supervised learning alone. The models also transfer to CNN/DM news articles, producing summaries nearly as good as the human reference without any news-specific fine-tuning. They conduct extensive analyses to understand our human feedback dataset and fine-tuned models. They establish that our reward model generalizes to new datasets, and that optimizing our reward model results in better summaries than optimizing ROUGE according to humans.

3. **Training language models to follow instructions with human feedback by OpenAI:** In this paper, the authors show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning them with human feedback. Starting with labeler-written prompts and prompts submitted through the OpenAI API, they collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. They then collect a dataset of rankings of model outputs, which they use to fine-tune this supervised model further using reinforcement learning from human feedback. We call the resulting models InstructGPT. In human evaluations on the prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, the results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

7. How to Stay Updated with LLM Research & Industry News?

In the rapidly evolving landscape of Large Language Models (LLMs), staying abreast of the latest research breakthroughs and industry developments is paramount for professionals and enthusiasts alike. This blog serves as a comprehensive guide, presenting a curated list of resources tailored to cater to this need.

This section begins by spotlighting the pioneers and thought leaders in LLM research, providing insights into their work and contributions to the field. It then transitions to examining key players driving innovation and application within the LLM industry. Additionally, it explores prominent organizations dedicated to advancing LLM research and fostering collaboration within the community.

Furthermore, the section identifies influential individuals and content creators shaping discourse and disseminating valuable insights across various platforms. It delves into the

realm of newsletters and blogs, highlighting essential sources for staying updated on the latest trends and developments in LLM research and industry.

Whether one is an aspiring researcher, industry practitioner, or simply intrigued by the capabilities of LLMs, this blog equips readers with the essential resources to remain informed and engaged in this dynamic domain.

7.1. LLM Research Leaders

1.1. Yann LeCun

Yann is also one of the deep learning pioneers and he is currently a Chief AI Scientist at Meta AI. He publishes the research output at Meta AI in addition to his opinions on AI research.

2.2. Andrej Karpathy

Andrej is one of the leading AI researchers and currently, he is latest position was at OpenAI before that, he was the director of AI at Tesla. By following Andrej you will stay updated with recent AI research and also important opinions about the AI advances and research.

1.3. Sebastian Raschka

Sebastian is a machine learning researcher and author of Machine Learning with Pytorch and Scikit-Learn. Sebastian shares very insightful tweets about AI, deep learning, and Pytorch.

7.2. LLM Industry Leaders

2.1. Andrew Ng

Andrew Ng, a prominent figure in the AI community, shares research, publications, projects, courses, and industry news. He engages in discussions, offers guidance, and provides valuable perspectives on AI ethics and applications. Following Andrew Ng on Twitter keeps you informed about the latest developments and connects you with the AI community.

2.2. Demis Hassabis

Demis is the co-founder and the CEO of DeepMind. Following him will keep you updated on the new research output and new projects from DeepMind.

2.3. Thomas Wolf

Thomas is the Co-founder of HuggingFace. He shares the new projects and important models of HuggingFace in addition to important AI news.

2.4. Oriol Vinyals

Oriol is the VP of research and deep learning lead at Google DeepMind. He shares updates about research projects and research output at Google DeepMind.

2.5. Sam Altman

Sam is the co-founder of OpenAI. His tweets are mainly about AI ethics, regulations, and also recent outputs from OpenAI.

2.6. Abubakar Abid

Abubakar is the founder of **Gradio** which was recently acquired by HuggingFace. He shares important updates on Gardio in addition to demos on Gardio. Also, he is the founder of the Fatima fellowship and he shares important announcements related to it.

2.8. Jay Alammar

Jay is the director of engineering at Cohere. Jay is well known for his illustrated NLP blogs in which he simplifies complex concepts using illustrations.

7.3. LLM Research & Industry Organizations

7.3.1. NeurIPS Conference

NeurIPS, a prestigious conference in the ML and AI community, offers valuable updates, keynote announcements, registration details, submission deadlines, and highlights from past conferences. The account also shares research papers, workshop announcements, and tutorials. By following NeurIPS, you join a global network of leading researchers, practitioners, and industry professionals, gaining knowledge and networking opportunities. Stay tuned to NeurIPS on Twitter for the latest trends and breakthroughs in ML research. Prepare for an exciting journey through cutting-edge ML research.

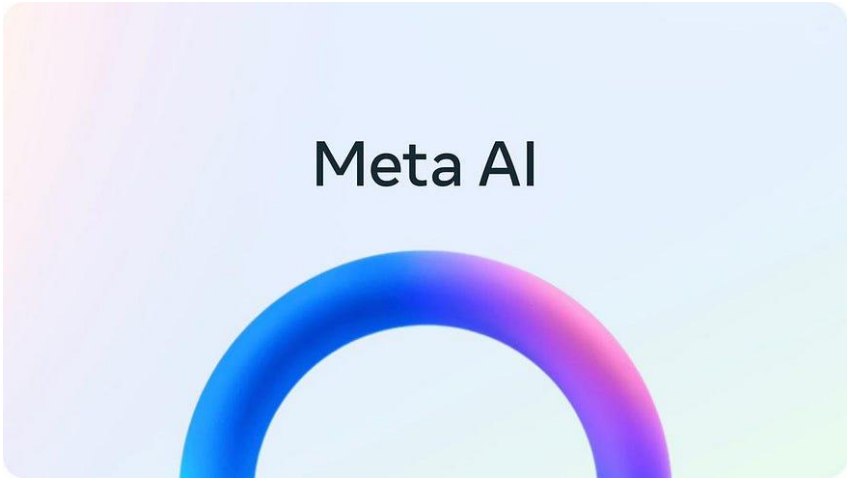
7.3.2. Google DeepMind

DeepMind, a subsidiary of Google, is known for its cutting-edge advancements in AI. By following their account, you gain access to updates on research findings, publications, and breakthroughs. DeepMind's work in reinforcement learning, neural networks, and deep learning is highlighted, along with applications in various domains. Their Twitter account also features events, talks, and panel discussions with their researchers. Following DeepMind on Twitter keeps you connected to the forefront of AI research and innovation.

7.3.3. Google AI

Google AI shares updates on AI projects, research papers, technology advancements, and practical applications across domains. Google AI's Twitter account also features highlights from conferences, events, and talks by their researchers. By following Google AI, you stay informed about cutting-edge developments and engage with diverse AI applications. Stay tuned to their account for valuable insights and inspiration in the field of AI.

7.3.4. Meta AI



This is the official account for Meta AI. By following it you will stay updated with the new research output by Meta AI.

7.3.5. Hugging Face

Hugging Face is a company known for its state-of-the-art models and NLP tools. By following their account, you gain access to model releases, library improvements, NLP research advancements, tips, tutorials, and community engagement. Stay connected to cutting-edge developments and leverage their tools for your own NLP projects and research.

7.3.6. OpenAI

OpenAI is a leading research organization focused on safe and beneficial AI development. By following their account, you gain access to updates on research projects, new AI models/tools, insights from team members, and announcements. The account engages

with the AI community, discusses ethical implications, and promotes responsible AI development. Stay informed about AI trends and discussions by following OpenAI on Twitter.

7.4. LLM Influencers & Content Creators

4.1. **Omar Sanseviero:** Omar Sanseviero is a prominent figure in the machine learning community, currently serving as a machine learning advocate at Hugging Face. He plays a significant role at the intersection of open source, product development, research, and community engagement. He shares valuable insights on LLM development and recent news.

4.2. **Rohan Paul:** Rohan is known for his educational content and resources on machine learning. He has a YouTube channel, "Rohan-Paul-AI," where he shares tutorials and insights on topics like LLM, GANs (Generative Adversarial Networks), and other deep learning models.

4.3. **Rowan Cheung:** Rowan Cheung is the founder of “The Rundown AI,” a popular newsletter that provides updates and insights on the latest developments in artificial intelligence. He covers a wide range of AI topics, including new technologies, industry trends, and significant advancements from leading tech companies like Microsoft and Google.

4.4. **AK**: AK is a machine learning engineer at Gardio. He is well known for publishing summaries of recent machine learning and AI papers on a daily basis.

4.5. **Yannic Kilcher:** Yannic is the co-founder of DeepJudge and he runs a YouTube channel under his name. He is very active in making videos for in-depth explanations of recent important research papers and important AI news.

4.6. **Jeremy Howard**: Jeremy Howard is a renowned data scientist, entrepreneur, and educator. He is known for his contributions to the field of AI, particularly in deep learning, and for founding Fast ai. Jeremy shares important opinions about AI and deep learning in addition to news and updates related to Fast ai.

4.7. **Santiago:** Santiago was previously a director of computer vision at Levatas and is currently managing his own machine-learning school. Santiago shares valuable technical and career tips that will help you build both better careers and better projects.

4.8. **Elvis:** Elvis is a former machine learning engineer at Meta working on the **Paper with Code** project and currently he is focusing on DIAR an open-source educational platform. He

shares very important summaries of recent machine learning papers in addition to important AI news and also DIAR recent projects.

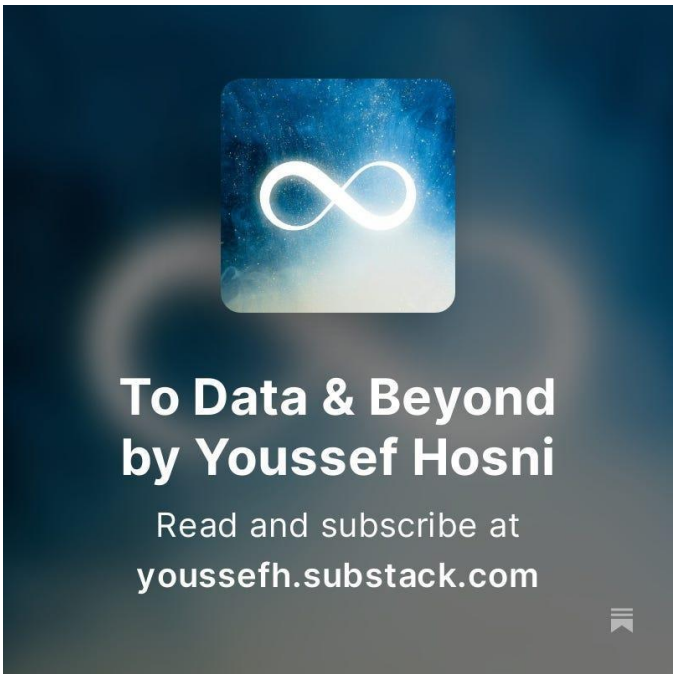
4.9. **Sanyam Bhutani:** Sanyam is a senior data scientist at Weights and Biases and he is also one of the world kaggle grandmasters. He is very active on Twitter and shares summarization of important research papers on a daily basis in addition to important AI news.

4.10. **Harrison Kinsley:** Harrison runs one of the best AI & Programming YouTube channels under the name of Sentdex. He covers different topics on his YouTube channel including explaining recent papers, news, and also in-depth explanations for different concepts.

4.11. **Lilian Weng:** Lilian is working on AI safety at OpenAI. Her posts are mainly about LLM politics and safety at openAI and also new announcements related to OpenAI work.

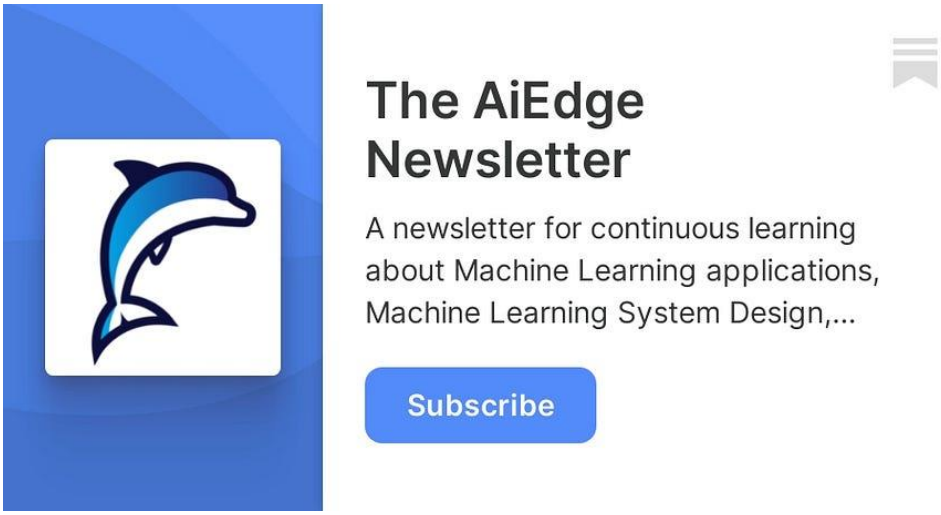
7. 5. Newsletters & Blogs for LLM Research News

7.5.1. To Data & Beyond Newsletter



The "**To Data & Beyond**" newsletter by Youssef Hosni is an excellent resource for staying updated with the latest research and developments in large language models (LLMs). It offers in-depth analysis, summaries of recent research papers, and discussions on trends in data science and machine learning. The newsletter aims to provide valuable insights for both professionals and enthusiasts in the field, making complex topics more accessible.

7.5.2. The AiEdge Newsletter



The AiEdge Newsletter, curated by Damien Benveniste, is an excellent source for staying updated with the latest research and developments in large language models (LLMs). The newsletter covers a wide array of topics including machine learning applications, system design, MLOps, and the latest techniques and news in the field. It aims to make complex machine-learning concepts accessible to a broader audience, often diving into subjects not typically covered in mainstream sources.

7.5.3. Ahead of AI

Ahead of AI newsletter authored by Sebastian Raschka, is a highly regarded newsletter that provides in-depth coverage of the latest research and developments in AI, particularly focusing on machine learning and large language models (LLMs). With over a decade of experience in AI and a passion for education, Raschka curates content that is valuable for both researchers and practitioners aiming to stay ahead in the rapidly evolving AI field.

7.5.4. Daily Papers by Hugging Face

Hugging Face's daily paper initiative focuses on providing the community with access to significant papers, facilitating discussions around them, and offering a centralized place to explore these papers and related artifacts like models, datasets, and demos.

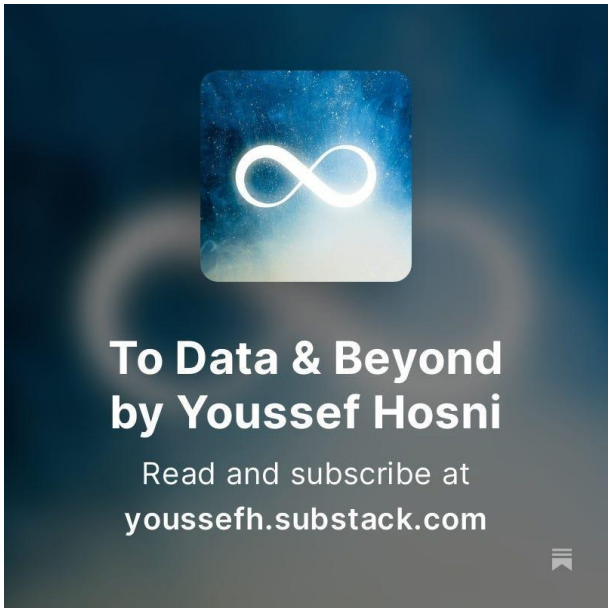
7.6. Newsletters & Blogs for LLM Industry News

7.6.1. The Rundown

The Rundown newsletter gives you access to the latest AI news and learn how to apply it in 5 minutes. By simply following this newsletter, you'll be able to:

- Keep up with the rapid pace of AI
- Learn how to use AI to automate your work
- Discover the best AI tools and job opportunities
- Gain expert insights on how AI will reshape the future of work

7.6.2. To Data & Beyond Newsletter

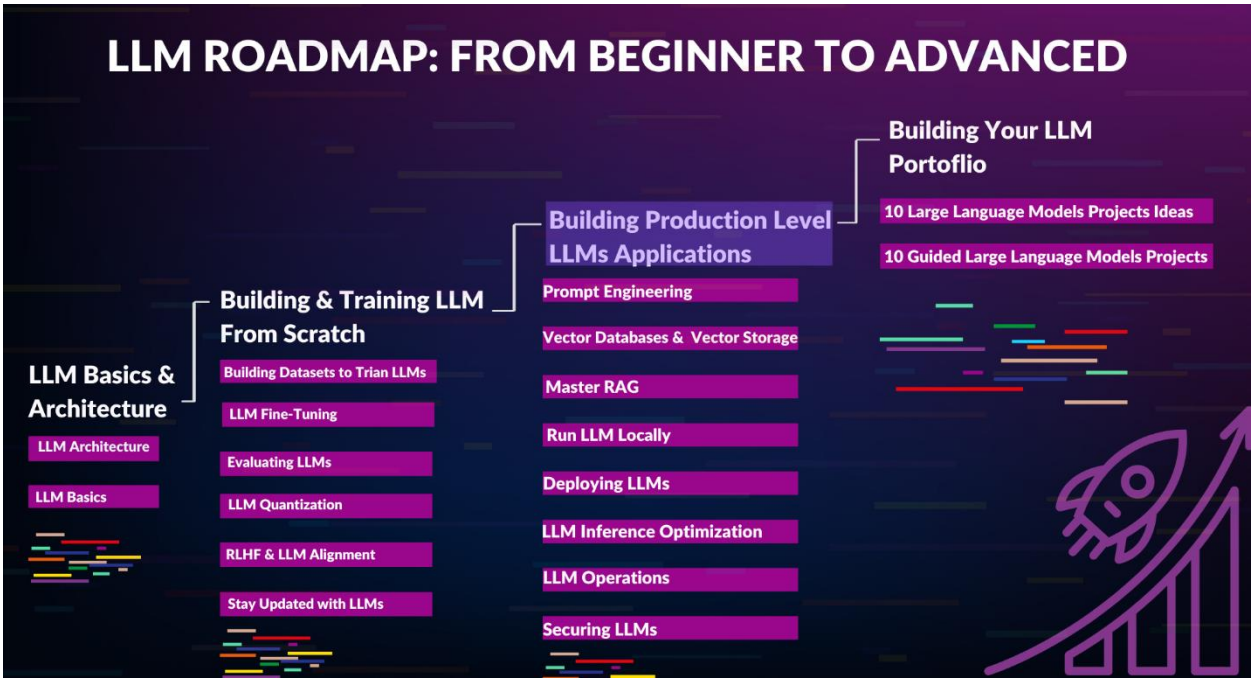


The "**To Data & Beyond**" newsletter by Youssef Hosni is also an excellent resource for staying updated with the latest developments in LLMs in the industry. It offers high-quality articles on the recent developments in LLMs in addition to hands-on tutorials for advanced topics.

7.6.3. KDNuggets

KDnuggets is a highly reputable source for staying updated with industry news related to large language models (LLMs) and other advancements in data science, machine learning, and AI. It offers a wealth of resources, including articles, tutorials, opinions, and industry news, making it a comprehensive platform for professionals and enthusiasts in these fields.

Part III: LLMs In Production



In the third section, you'll focus on developing real-world applications using Large Language Models (LLMs), equipping yourself with practical skills and knowledge to deploy these models effectively. This section begins with mastering prompt engineering, using the best resources to craft prompts that elicit desired responses from LLMs, a foundational skill for various applications.

You'll then delve into vector databases and building vector storage, learning from top resources that explain how to efficiently store and retrieve embeddings, which are crucial for enhancing LLM capabilities. This leads to mastering Retrieval-Augmented Generation (RAG), where you'll explore resources ranging from basic to advanced levels, enabling you to integrate external information retrieval with generation tasks for more accurate and relevant outputs.

To facilitate hands-on experimentation, this section also highlights five free tools to run LLMs locally on your laptop, providing a practical approach to testing and refining your models without needing extensive computational resources. Following this, you'll find top learning and educational resources for deploying LLMs, guiding you through the process of setting up models in production environments.

Optimizing LLM inference is another critical area covered, with the best resources to help you enhance the efficiency and speed of model predictions. You will also learn about LLMOps, a practice focusing on the operational aspects of managing and maintaining LLMs, including monitoring, versioning, and scaling, with resources to get you started.

Finally, securing LLMs is addressed, providing the best learning and educational resources to ensure your models are robust against various threats and vulnerabilities. By the end of this section, you will have a thorough understanding of developing, deploying, and optimizing LLM applications in real-world settings, equipped with the practical skills to handle various challenges in production environments.

1. Best Resources to Learn Prompt Engineering

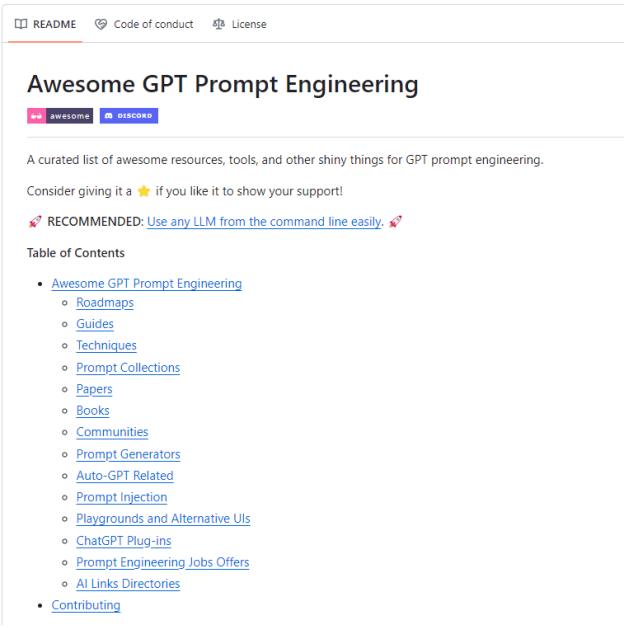
Prompt engineering, an emerging field, focuses on the development and refinement of prompts to enhance the utilization of language models (LMs) across diverse applications and research domains.

Proficiency in prompt engineering contributes to a deeper comprehension of the strengths and weaknesses inherent in large language models (LLMs). Researchers leverage prompt engineering to enhance the performance of LLMs in tasks ranging from commonplace to intricate, including question-answering and arithmetic reasoning. Meanwhile, developers employ prompt engineering to craft resilient and efficient prompting techniques that seamlessly interface with LLMs and other tools.

Motivated by the high interest in developing with LLMs, in this section, I will share five resources that will help you understand prompt engineering and learn how to write better prompts.

1.1. Awesome GPT Prompt Engineering GitHub Repository

The **Awesome GPT Prompt Engineering** is a curated list of awesome resources, tools, and other shiny things for GPT Prompt Engineering.



README Code of conduct License

Awesome GPT Prompt Engineering

awesome DISCORD

A curated list of awesome resources, tools, and other shiny things for GPT prompt engineering.

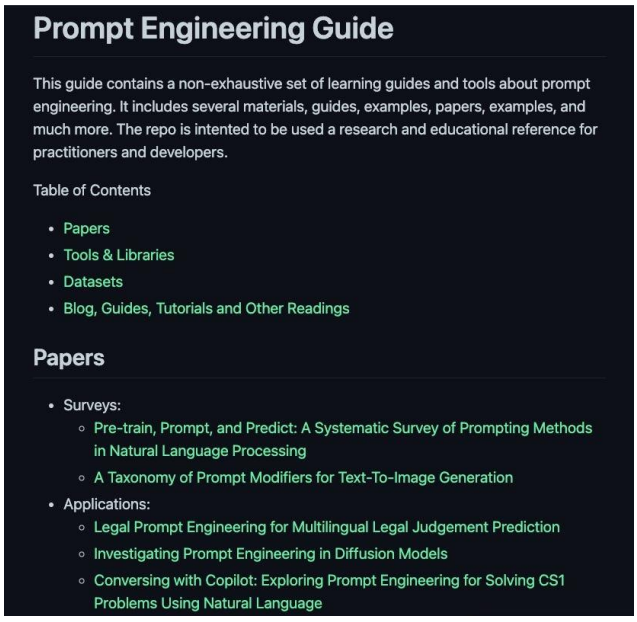
Consider giving it a ⭐ if you like it to show your support!

🚀 RECOMMENDED: Use any LLM from the command line easily. 🚀

Table of Contents

- Awesome GPT Prompt Engineering
  - Roadmaps
  - Guides
  - Techniques
  - Prompt Collections
  - Papers
  - Books
  - Communities
  - Prompt Generators
  - Auto-GPT Related
  - Prompt Injection
  - Playgrounds and Alternative UIs
  - ChatGPT Plug-ins
  - Prompt Engineering Jobs Offers
  - AI Links Directories
- Contributing

1.2. Prompt Engineering Guide



Prompt Engineering Guide

This guide contains a non-exhaustive set of learning guides and tools about prompt engineering. It includes several materials, guides, examples, papers, examples, and much more. The repo is intented to be used a research and educational reference for practitioners and developers.

Table of Contents

- Papers
- Tools & Libraries
- Datasets
- Blog, Guides, Tutorials and Other Readings

Papers

- Surveys:
  - Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing
  - A Taxonomy of Prompt Modifiers for Text-To-Image Generation
- Applications:
  - Legal Prompt Engineering for Multilingual Legal Judgement Prediction
  - Investigating Prompt Engineering in Diffusion Models
  - Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language

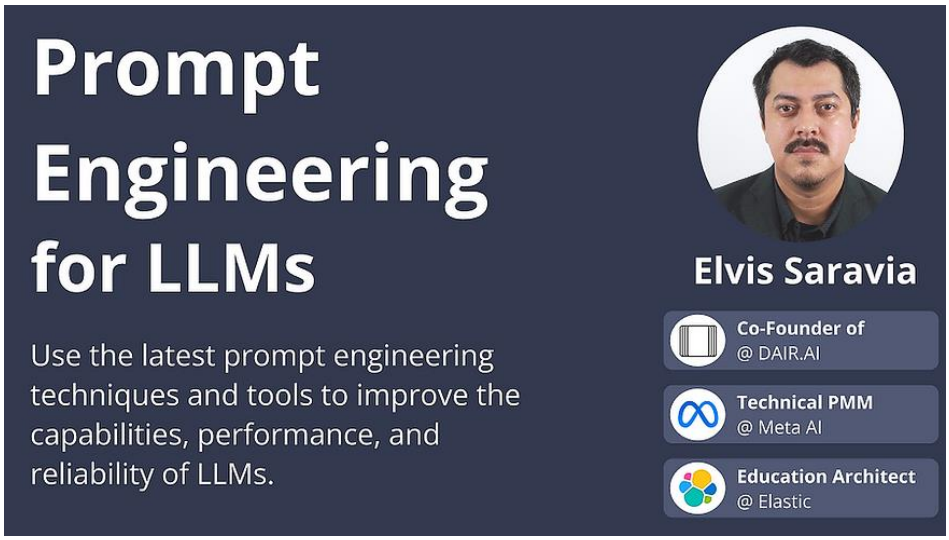
Prompt engineering is a relatively new discipline for developing and optimizing prompts to efficiently use language models (LMs) for various applications and research topics. Prompt engineering skills help better understand the capabilities and limitations of large language models (LLMs). Researchers use prompt engineering to improve the capacity of LLMs on a wide range of common and complex tasks such as question answering and arithmetic reasoning. Developers use prompt engineering to design robust and effective prompting techniques that interface with LLMs and other tools.

Motivated by the high interest in developing with LLMs, Elvis Saravia has created this **Prompt Engineering Guide** that contains all the latest papers, learning guides, lectures, references, and tools related to prompt engineering for LLMs.

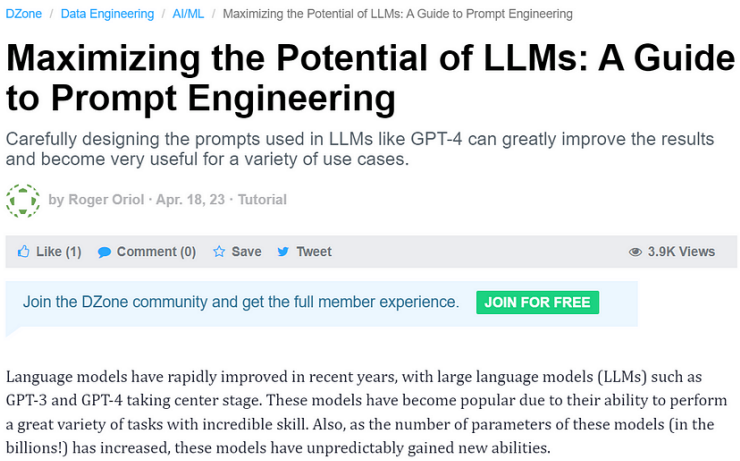
1.2. Prompt Engineering for LLMs

Prompt Engineering for LLMs is a 4-day hands-on course that teaches how to efficiently and effectively use LLMs. It covers the best and latest prompting techniques that you can apply to a variety of use cases that range from building long article summarizers to prompt injection detectors to LLM-powered evaluators.

Be ready to dive deep and acquire advanced skills for effectively prompting and building with LLMs. By the end of the course, the goal is NOT to teach you the "10 best prompts". The goal is for you to learn how to apply advanced prompting techniques to help you grow in your career, build personal projects, or build advanced LLM-powered products.



1.4. Maximizing the Potential of LLMs: A Guide to Prompt Engineering



DZone / Data Engineering / AI/ML / Maximizing the Potential of LLMs: A Guide to Prompt Engineering

Maximizing the Potential of LLMs: A Guide to Prompt Engineering

Carefully designing the prompts used in LLMs like GPT-4 can greatly improve the results and become very useful for a variety of use cases.

by Roger Oriol · Apr. 18, 23 · Tutorial

Like (1) Comment (0) Save Tweet 3.9K Views

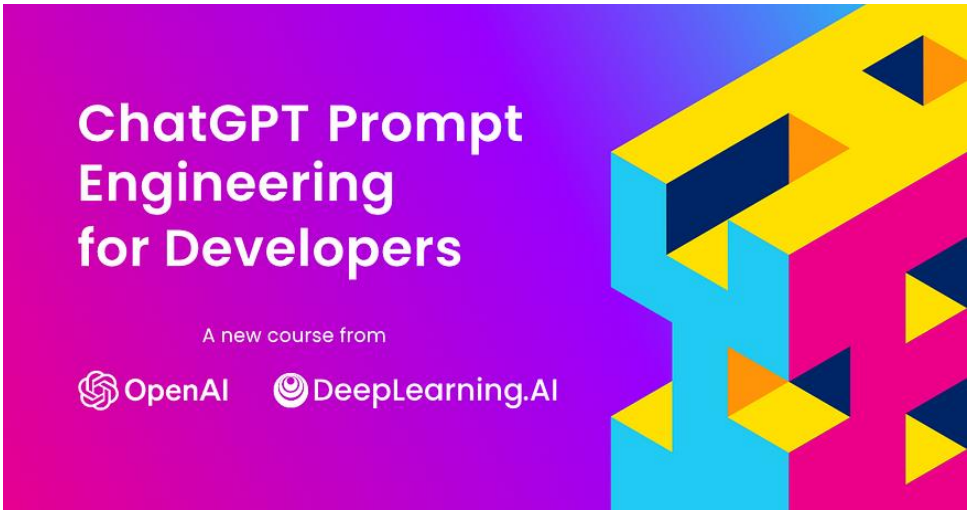
Join the DZone community and get the full member experience. JOIN FOR FREE

Language models have rapidly improved in recent years, with large language models (LLMs) such as GPT-3 and GPT-4 taking center stage. These models have become popular due to their ability to perform a great variety of tasks with incredible skill. Also, as the number of parameters of these models (in the billions!) has increased, these models have unpredictably gained new abilities.

Language models have rapidly improved in recent years, with large language models (LLMs) such as GPT-3 and GPT-4 taking center stage. These models have become popular due to their ability to perform a great variety of tasks with incredible skill. Also, as the number of parameters of these models (in the billions!) has increased, these models have unpredictably gained new abilities.

In this **article**, you will explore LLMs, the tasks they can perform, their shortcomings, and various prompt engineering strategies.

1.5. ChatGPT Prompt Engineering for Developers



In **ChatGPT Prompt Engineering for Developers**, you will learn how to use a large language model (LLM) to quickly build new and powerful applications. Using the OpenAI API, you'll be able to quickly build capabilities that teaches you to innovate and create value in ways that were cost-prohibitive, highly technical, or simply impossible before now. This short course taught by Isa Fulford (OpenAI) and Andrew Ng (DeepLearning.AI) will describe how LLMs work, provide best practices for prompt engineering, and show how LLM APIs can be used in applications for a variety of tasks, including:

- Summarizing (e.g., summarizing user reviews for brevity)
- Inferring (e.g., sentiment classification, topic extraction
- Transforming text (e.g., translation, spelling & grammar correction)
- Expanding (e.g., automatically writing emails)

In addition, you'll learn two key principles for writing effective prompts, how to systematically engineer good prompts, and learn to build a custom chatbot. All concepts are illustrated with numerous examples, which you can play with directly in our Jupyter Notebook environment to get hands-on experience with prompt engineering.

2. 6 Resources to Master Vector Databases & Building a Vector Storage

Vector databases, essential components in various fields like natural language processing and image recognition, serve as pivotal tools for organizing and retrieving information efficiently.

Understanding vector databases is crucial due to their significant role in enabling advanced AI applications such as semantic search, retrieval augmented generation (RAG), and recommender systems.

This section provides a comprehensive overview of resources aimed at mastering vector databases and building vector storage solutions. It covers fundamental concepts, practical applications, and an array of tools and libraries essential for working with vector databases.

Through tutorials, blog recommendations, and tools like LangChain and Sentence Transformers library, readers gain insights and hands-on experience to leverage vector databases effectively in their AI projects. Additionally, the article highlights the importance of staying updated with emerging technologies and offers avenues for further learning and community engagement.

2.1. Vector Databases: from Embeddings to Applications

Vector databases play a pivotal role across various fields, such as natural language processing, image recognition, recommender systems, and semantic search, and have gained more importance with the growing adoption of LLMs.

These databases are exceptionally valuable as they provide LLMs with access to real-time proprietary data, enabling the development of Retrieval Augmented Generation (RAG) applications.

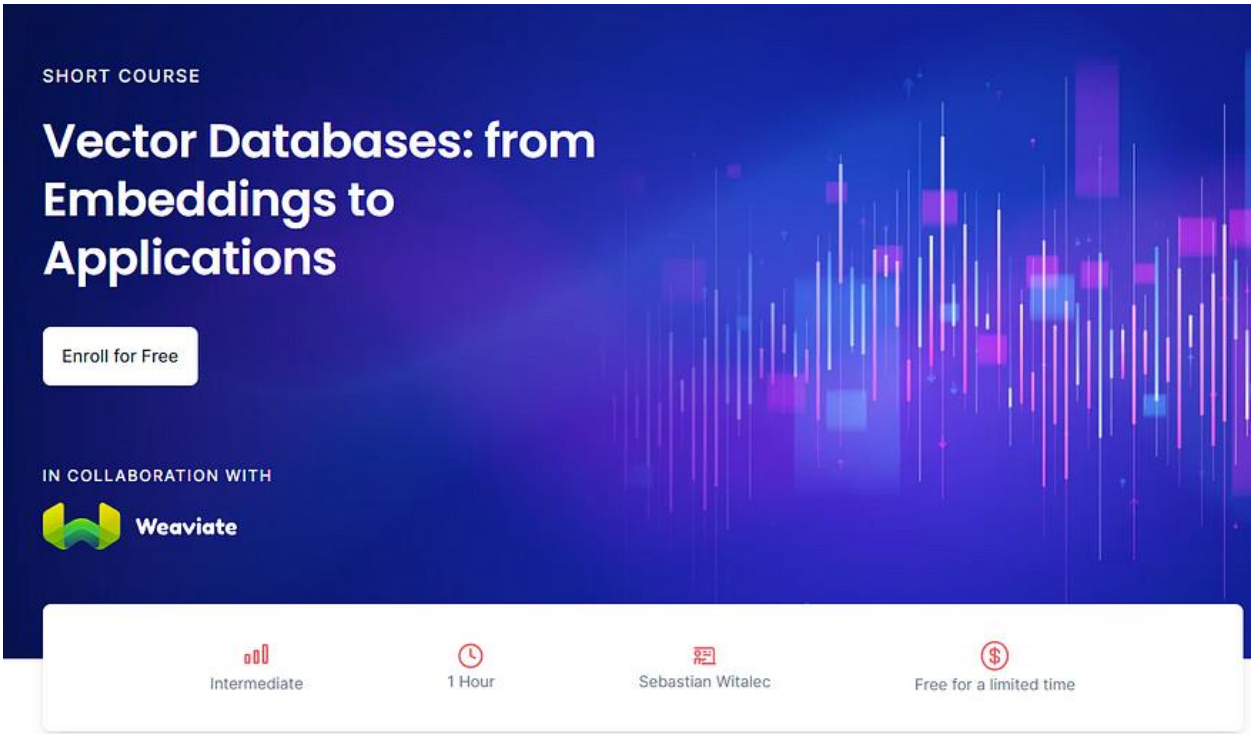
At their core, vector databases rely on the use of embeddings to capture the meaning of data gauge the similarity between different pairs of vectors, and sift through extensive datasets, identifying the most similar vectors.

This **course** will help you gain the knowledge to make informed decisions about when to apply vector databases to your applications. You'll explore:

How to use vector databases and LLMs to gain deeper insights into your data.

Build labs that show how to form embeddings and use several search techniques to find similar embeddings.

Explore algorithms for fast searches through vast datasets and build applications ranging from RAG to multilingual search.



2.2. Building Applications with Vector Databases

Vector databases use embeddings to capture the meaning of data, gauge the similarity between different pairs of vectors, and navigate large datasets to identify the most similar vectors.

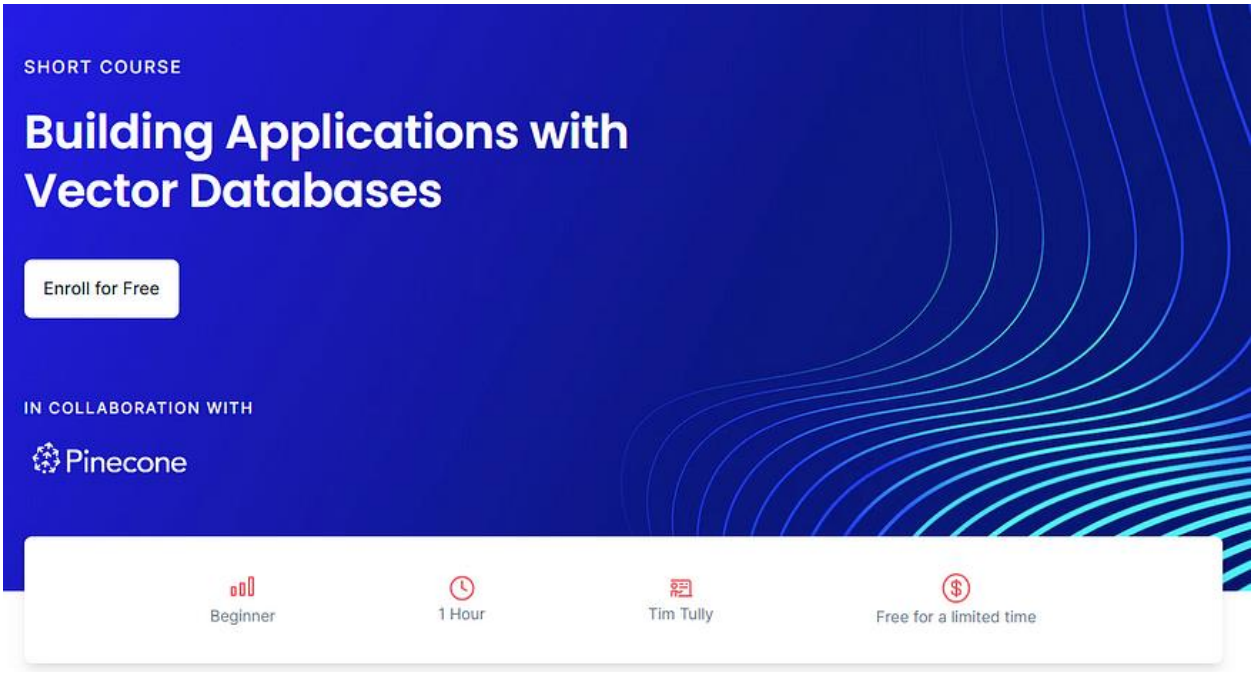
In the context of large language models, the primary use of vector databases is retrieval augmented generation (RAG), where text embeddings are stored and retrieved for specific queries.

However, the versatility of vector databases extends beyond RAG and makes it possible to build a wide range of applications quickly with minimal coding.

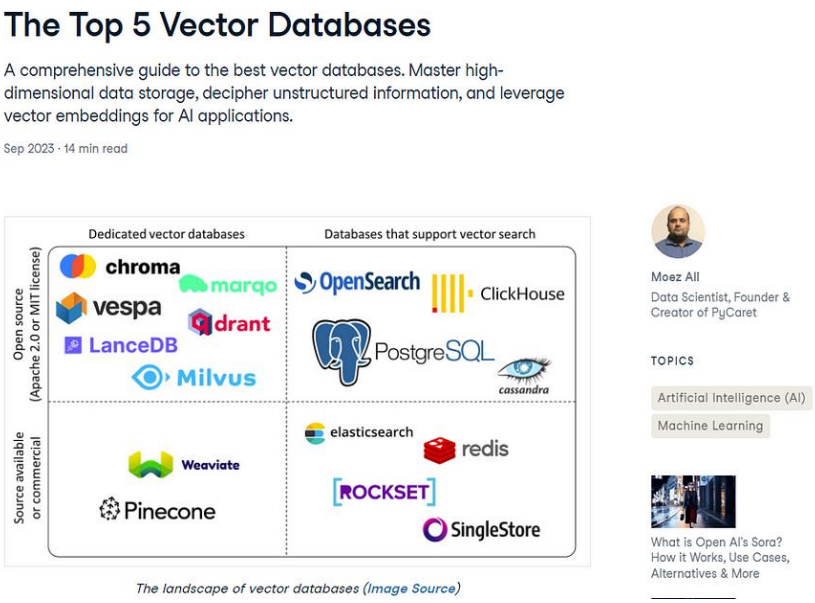
In this **course**, you'll explore the implementation of six applications using vector databases:

- **Semantic Search:** Create a search tool that goes beyond keyword matching, focusing on the meaning of content for efficient text-based searches on a user Q/A dataset.
- **RAG:** Enhance your LLM applications by incorporating content from sources the model wasn't trained on, like answering questions using the Wikipedia dataset.
- **Recommender System:** Develop a system that combines semantic search and RAG to recommend topics, and demonstrate it with a news article dataset.
- **Hybrid Search:** Build an application that finds items using both images and descriptive text, using an eCommerce dataset as an example.
- **Facial Similarity:** Create an app to compare facial features, using a database of public figures to determine the likeness between them.
- **Anomaly Detection:** Learn how to build an anomaly detection app that identifies unusual patterns in network communication logs.

After taking this **course**, you'll be equipped with new ideas for building applications with any vector database.



2.3. The Top 5 Vector Database Blog



The landscape of vector databases (Image Source)

This **blog** by **Moez Ali** **is a comprehensive guide** to the best vector databases. It will help you master high-dimensional data storage, decipher unstructured information, and leverage vector embeddings for AI applications.

This blog covers:

- What is a Vector Database?
- How Does a Vector Database Work?
- Examples of Vector Database
- Features of a Good Vector Database
- 5 of the Best Vector Databases in 2023
- The Rise of AI and the Impact of Vector Database

2.4. LangChain — Text splitters

LangChain — Text splitters is a list of different text splitters implemented in LangChain. Once you've loaded documents, you'll often want to transform them to better suit your application.

The simplest example is you may want to split a long document into smaller chunks that can fit into your model's context window. LangChain has several built-in document transformers that make it easy to split, combine, filter, and otherwise manipulate documents.

When you want to deal with long pieces of text, it is necessary to split up that text into chunks. As simple as this sounds, there is a lot of potential complexity here. Ideally, you want to keep the semantically related pieces of text together. What "semantically related" means could depend on the type of text. This notebook showcases several ways to do that.

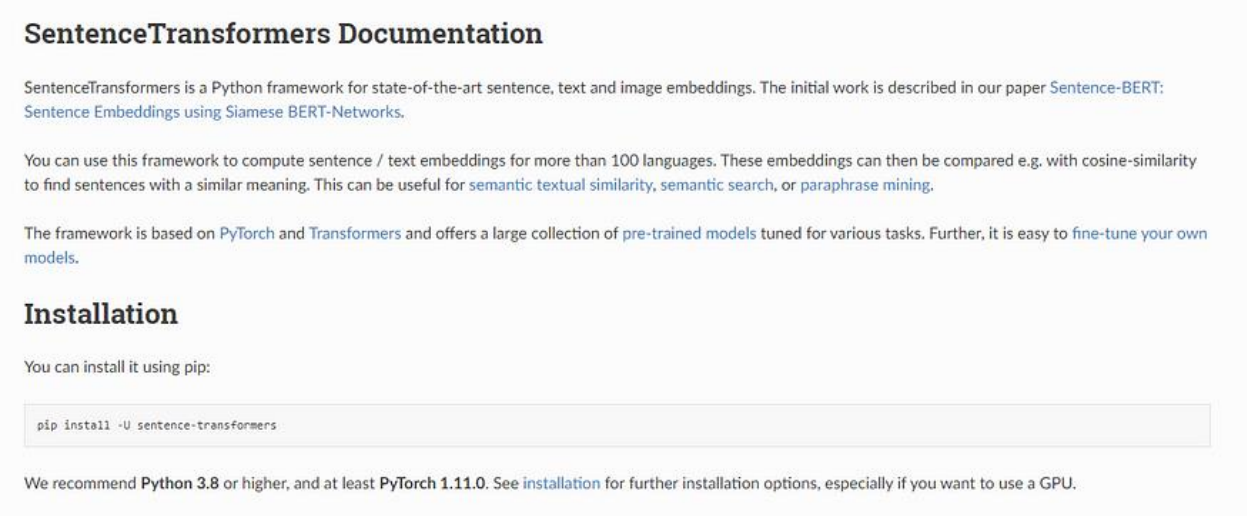
Name	Splits On	Adds Metadata	Description
Recursive	A list of user defined characters		Recursively splits text. Splitting text recursively serves the purpose of trying to keep related pieces of text next to each other. This is the recommended way to start splitting text.
HTML	HTML specific characters	✅	Splits text based on HTML-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the HTML)
Markdown	Markdown specific characters	✅	Splits text based on Markdown-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the Markdown)
Code	Code (Python, JS) specific characters		Splits text based on characters specific to coding languages. 15 different languages are available to choose from.
Token	Tokens		Splits text on tokens. There exist a few different ways to measure tokens.
Character	A user defined character		Splits text based on a user defined character. One of the simpler methods.
[Experimental] Semantic Chunker	Sentences		First splits on sentences. Then combines ones next to each other if they are semantically similar enough. Taken from Greg Kamradt

2.5. Sentence Transformers Library

Sentence Transformers library is a popular Python framework for state-of-the-art sentence, text, and image embeddings. The initial work is described in our paper Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

You can use this framework to compute sentence/text embeddings for more than 100 languages. These embeddings can then be compared e.g. with cosine-similarity to find sentences with a similar meaning. This can be useful for semantic textual similarity, semantic search, or paraphrase mining.

The framework is based on PyTorch and Transformers and offers a large collection of pre-trained models tuned for various tasks. Further, it is easy to fine-tune your models.



SentenceTransformers Documentation

SentenceTransformers is a Python framework for state-of-the-art sentence, text and image embeddings. The initial work is described in our paper Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

You can use this framework to compute sentence / text embeddings for more than 100 languages. These embeddings can then be compared e.g. with cosine-similarity to find sentences with a similar meaning. This can be useful for semantic textual similarity, semantic search, or paraphrase mining.

The framework is based on PyTorch and Transformers and offers a large collection of pre-trained models tuned for various tasks. Further, it is easy to fine-tune your own models.

Installation

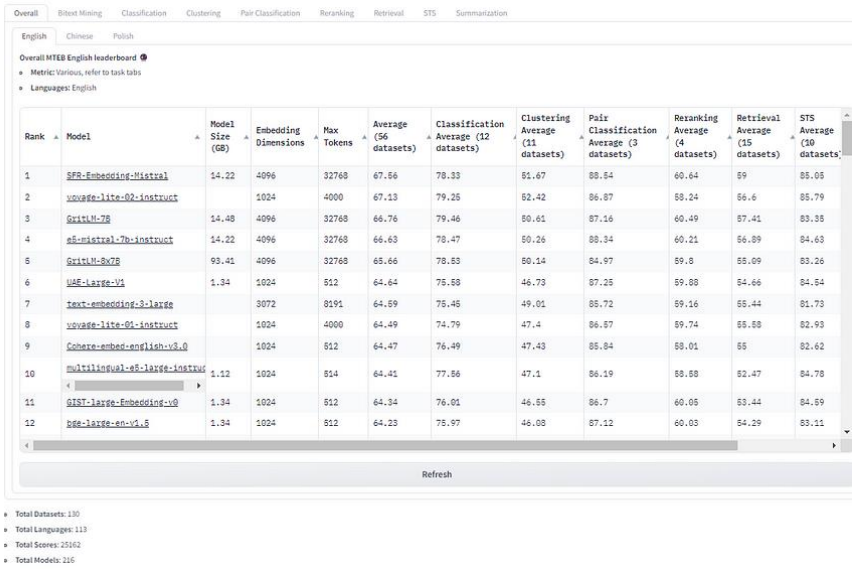
You can install it using pip:

```
pip install -U sentence-transformers
```

We recommend **Python 3.8** or higher, and at least **PyTorch 1.11.0**. See installation for further installation options, especially if you want to use a GPU.

2.6. MTEB Leaderboard

MTEB Leaderboard is a leaderboard for embedding models so you can compare different embedding models to use.



Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization

English | Chinese | Polish

Overall MTEB English leaderboard

- Metric: Various, refer to task tabs
- Languages: English

Rank	Model	Model Size (GB)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)	Reranking Average (4 datasets)	Retrieval Average (15 datasets)	STS Average (10 datasets)
1	SFR-Embedding-Mistral	14.22	4096	32768	67.56	78.33	51.67	88.54	60.64	59	85.05
2	voyage-lite-02-instruct		1024	4000	67.13	79.25	52.42	86.87	58.24	56.6	85.79
3	GritLM-7B	14.48	4096	32768	66.76	79.46	50.61	87.16	60.49	57.41	83.35
4	e5-mistral-7b-instruct	14.22	4096	32768	66.63	78.47	50.26	88.34	60.21	56.89	84.63
5	GritLM-8x7B	93.41	4096	32768	65.66	78.53	50.14	84.97	59.8	55.09	83.26
6	UAE-Large-V1	1.34	1024	512	64.64	75.58	46.73	87.25	59.88	54.66	84.54
7	text-embedding-3-large		3072	8191	64.59	75.45	49.01	85.72	59.16	55.44	81.73
8	voyage-lite-01-instruct		1024	4000	64.49	74.79	47.4	86.57	59.74	55.58	82.93
9	Cohere-embed-english-v3.0		1024	512	64.47	76.49	47.43	85.84	58.01	55	82.62
10	multilingual-e5-large-instruc	1.12	1024	514	64.41	77.56	47.1	86.19	58.58	52.47	84.78
11	GIST-large-Embedding-v0	1.34	1024	512	64.34	76.01	46.55	86.7	60.05	53.44	84.59
12	bge-large-en-v1.5	1.34	1024	512	64.23	75.97	46.08	87.12	60.03	54.29	83.11

Refresh

- Total Datasets: 130
- Total Languages: 113
- Total Scores: 25162
- Total Models: 216

3. Top Resources to Master RAG: From Basic Level to Advanced

Retrieval Augmented Generation (RAG) stands at the forefront of natural language processing, blending retrieval, and generative models to produce contextually relevant text. Understanding its significance and learning its intricacies are crucial for navigating the evolving landscape of AI.

This section serves as a comprehensive resource, offering a structured journey from RAG fundamentals to advanced techniques. It begins by elucidating the essence of RAG, highlighting its applications and importance in various domains.

Subsequently, it delves into mastering LangChain, exploring query construction and SQL interactions. Advanced RAG concepts are then unveiled, covering self-querying retrieval, hybrid search strategies, and more. Evaluating RAG systems becomes seamless with insights into metrics and techniques provided. Additionally, this section introduces the role of Large Language Model (LLM) agents in bolstering RAG applications.

By offering an array of learning resources and practical insights, this section equips readers with the knowledge and skills necessary to excel in harnessing RAG's potential in AI applications. Whether novice or expert, this guide propels individuals towards becoming adept RAG practitioners, shaping the future of natural language understanding and generation.

3.1. Retrieval Augmented Generation Basics

Learning Retrieval Augmented Generation (RAG) Basics entails grasping the fundamental concepts of merging retrieval and generation models in natural language processing. RAG involves understanding how retrieval mechanisms can enhance generative models by leveraging pre-existing knowledge bases to inform text generation.

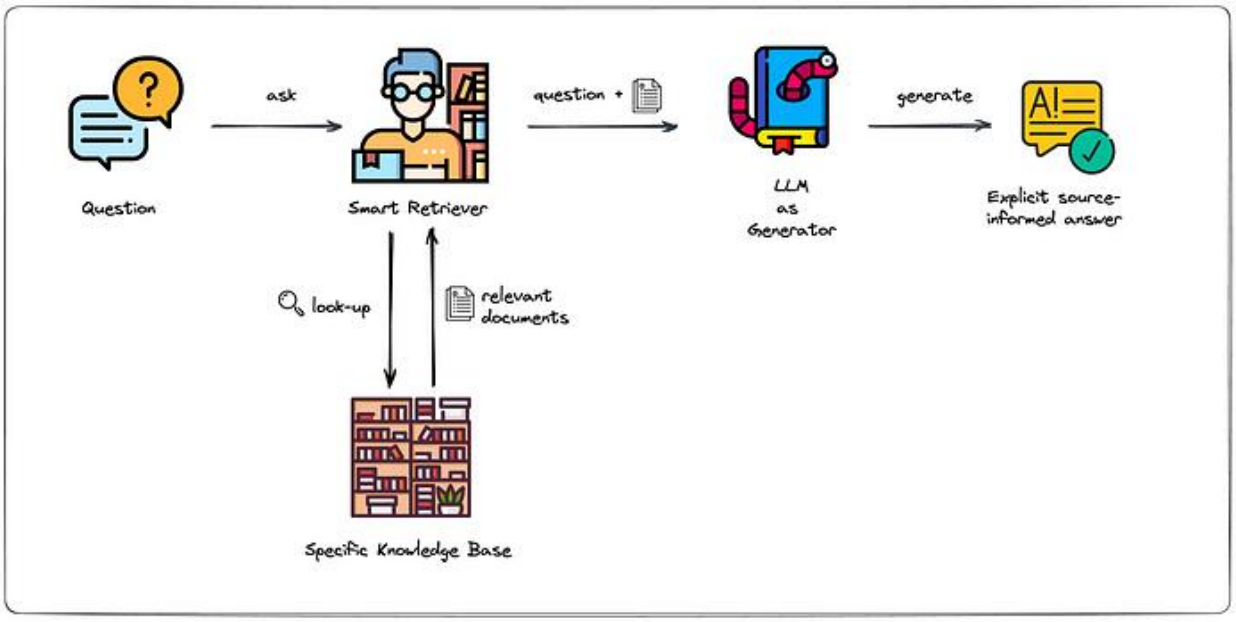
By acquiring knowledge of how to integrate retrieval techniques into the generation process, one can harness the power of structured information to produce more contextually relevant and coherent text outputs.

Mastering RAG basics is crucial for advancing capabilities in various applications such as question answering, summarization, and content generation, as it represents a significant advancement in the field of language modeling.

Learning Resources:

- **What Is Retrieval-Augmented Generation, aka RAG?:** A Comprehensive article by Nivida that introduces RAG and how it works.

- **RAG Applications with Llama-Index:** This tutorial introduces you to Llma-Index and how to build production-ready RAG applications.
- **Building RAG Applications with LangChain:** A compressive list of articles that walk you step by step through building RAG applications using LangChain.
- **LangChain — OpenAI's RAG:** Overview of the RAG strategies employed by OpenAI, including post-processing.



3.2. More LangChains

This step involves providing a comprehensive set of articles to deepen the understanding of LangChain concepts. It serves as a foundation for exploring the intricacies of LangChain technology.

Key topics covered include LangChain Query Construction, which offers insights into effective query construction methods for information retrieval. Additionally, a tutorial on LangChain SQL guides readers on interfacing with SQL databases using Large Language Models (LLMs), covering text-to-SQL conversion and introducing an optional SQL agent for seamless interaction.

These resources aim to impart a holistic understanding of LangChain fundamentals and practical applications, enabling readers to tackle complex language-based tasks more proficiently.

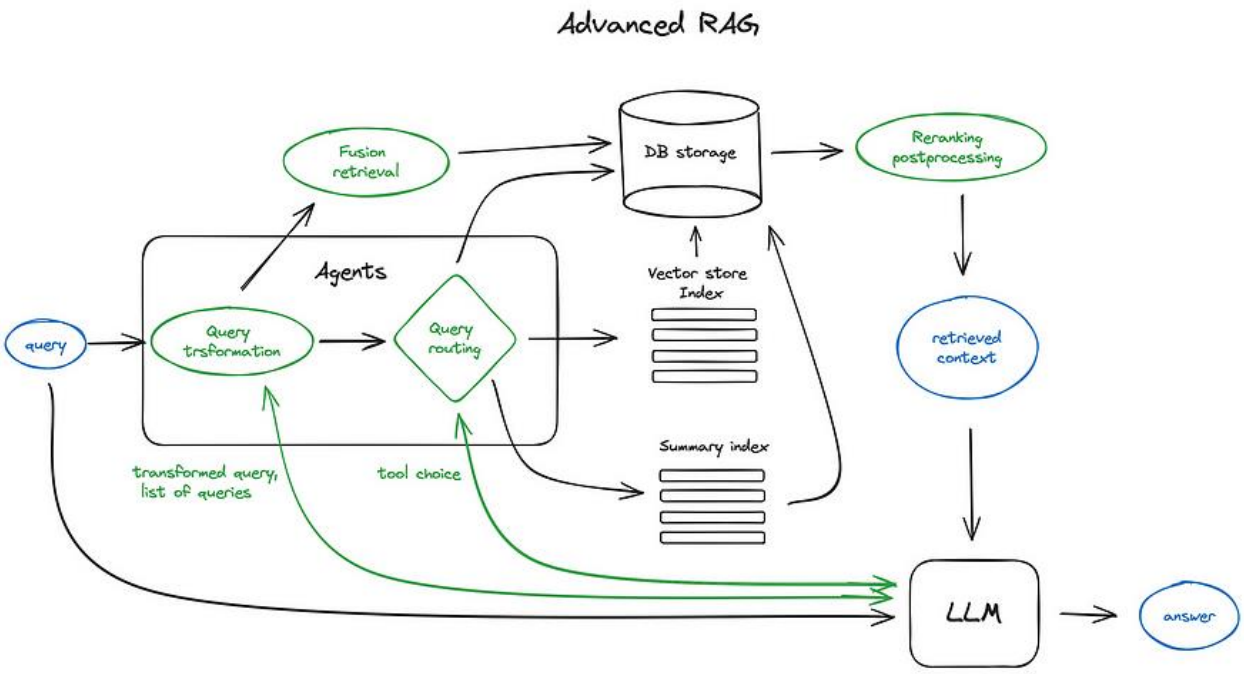
Learning Resources:

- **Basics of LangChain:** A compressive set of articles that introduced you to the most important LangChain concepts.
- **LangChain — Query Construction:** Blog post about different types of query construction.
- **LangChain — SQL:** Tutorial on how to interact with SQL databases with LLMs, involving Text-to-SQL and an optional SQL agent.

3.3. Advanced RAG

This learning step offers a range of learning resources aimed at enhancing skills in Retrieval Augmented Generation. Deep Learning.ai provides a short course titled "**Advanced Retrieval for AI with Chroma**," focusing on identifying and improving queries through large language models (LLMs) and embedding fine-tuning with user feedback.

Additionally, Sam Witteveen's advanced RAG tutorial series covers various advanced topics, including Self Querying Retrieval, Parent Document Retriever, Hybrid Search with BM25 & Ensembles, Contextual Compressors & Filters, HyDE (Hypothetical Document Embeddings), and RAG Fusion. These resources provide learners with in-depth knowledge and practical techniques to excel in the realm of Retrieval Augmented Generation.



Learning Resources:

- **Advanced Retrieval for AI with Chroma:** This is a short course provided by Deep Leanring.ai in which you will learn how to recognize when queries are producing poor

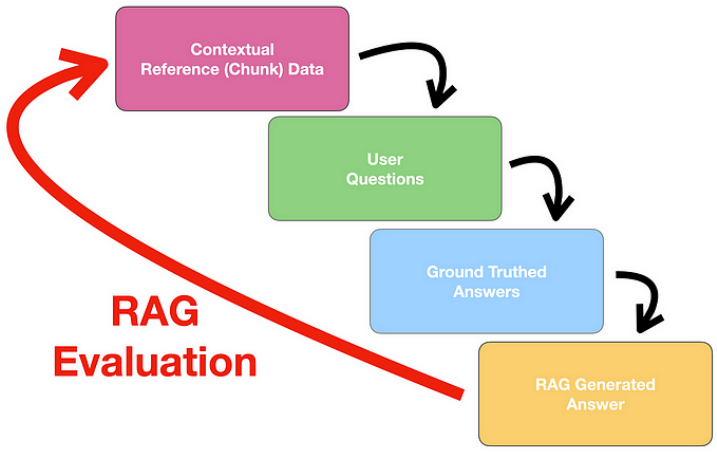
results, how to use a large language model (LLM) to improve your queries, and to fine-tune your embeddings with user feedback.

- **Advanced RAG by Sam Witteveen:** Throughout this advanced RAG series of tutorials, you will learn:
  1. **Self Querying Retrieval**
  2. **Parent Document Retriever**
  3. **Hybrid Search BM25 & Ensembles**
  4. **Contextual Compressors & Filters**
  5. **HyDE — Hypothetical Document Embeddings**
  6. **Advanced RAG 06 — RAG Fusion**

3.4. RAG Evaluation

The combined resources of "RAG Pipeline — Metrics" and "Building and Evaluating Advanced RAG Applications" offer learners a comprehensive understanding of evaluating Retrieval Augmented Generation (RAG) systems.

The former introduces essential evaluation metrics, while the latter delves into advanced retrieval methods, evaluation techniques, and the RAG triad for assessing relevance and accuracy. This holistic approach equips learners with the skills to effectively evaluate and optimize RAG systems in various applications.



Learning Resources:

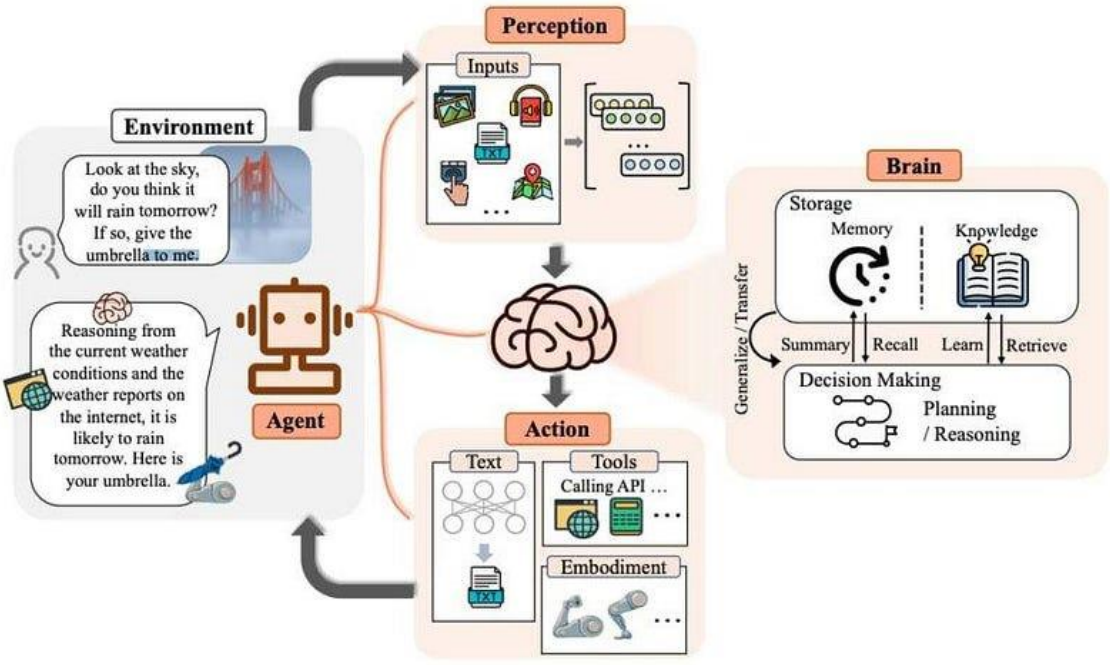
- **RAG Pipeline — Metrics:** This article introduces you to the main metrics used to evaluate RAG pipelines.

- **Building and Evaluating Advanced RAG Applications:** This course delves into advanced retrieval methods, including sentence-window retrieval and auto-merging retrieval, which outperform the baseline RAG pipeline. It also covers evaluation techniques and experiment tracking to iteratively enhance the RAG pipeline's performance. Furthermore, the course explores the RAG triad — Context Relevance, Groundedness, and Answer Relevance — as methods for assessing the relevance and accuracy of responses generated by large language models (LLMs).

3.5. LLM Agents

Learning to utilize LLM agents is essential for building advanced RAG applications. These agents, powered by models like GPT, enable advanced natural language understanding and generation capabilities, crucial for creating contextually relevant responses.

By integrating retrieval mechanisms with generative models, LLM agents enhance the quality and relevance of generated content, facilitating more accurate and insightful responses to user queries.



Learning Resources:

- **Pinecone — LLM agents:** This article by Pinecone introduced agents and tools with different types.
- **LLM-Powered Autonomous Agents:** This article gives you a more in-depth exploration of the theoretical concepts behind LLM-powered autonomous agents.

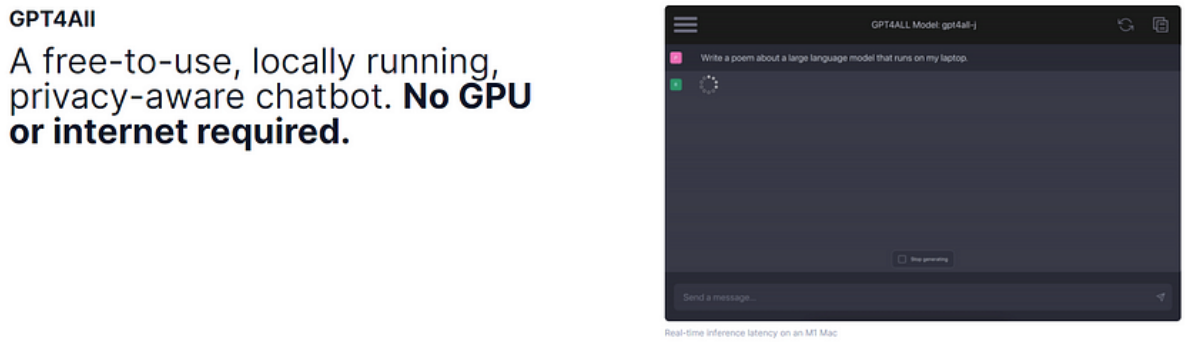
4. 5 Free Tools to Run Large Language Models (LLM) Locally on Your Laptop

While accessing LLM-based chatbots online is simple with just an internet connection and a good browser, it comes with potential privacy risks. For example, OpenAI stores your interactions and metadata to improve their models, raising concerns for privacy-conscious users. Opting to use these models locally provides a solution for those seeking greater control over their data.

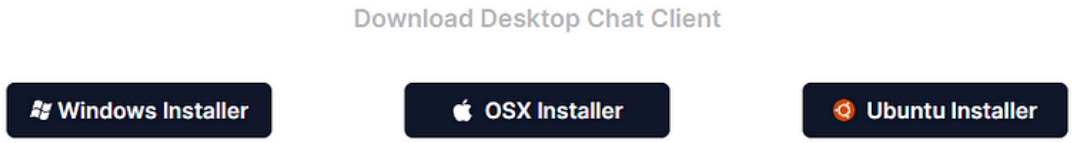
In this section, we'll explore five methods to utilize large language models (LLMs) locally. Compatible across major operating systems, these tools can be swiftly downloaded and installed.

With locally run LLMs, you retain control over model selection and can easily access models from the Hugging Face hub. Moreover, granting access to project folders enables context-aware responses.

4.1. GPT4All

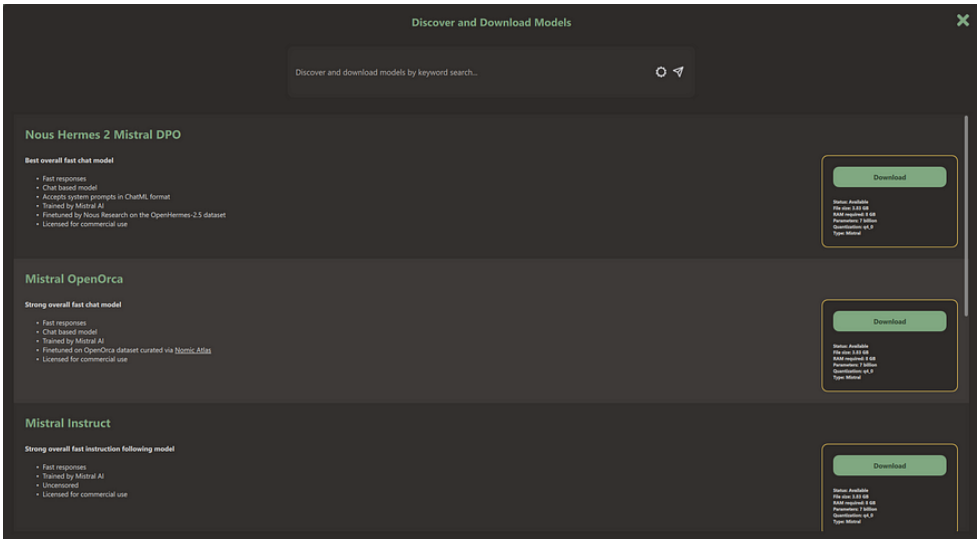


GPT4All is open-source software that enables you to use the state-of-the-art open-source LLM on your local machine with ease and in simple steps. To get started simply download **GPT4ALL** from the website and install it on your system.

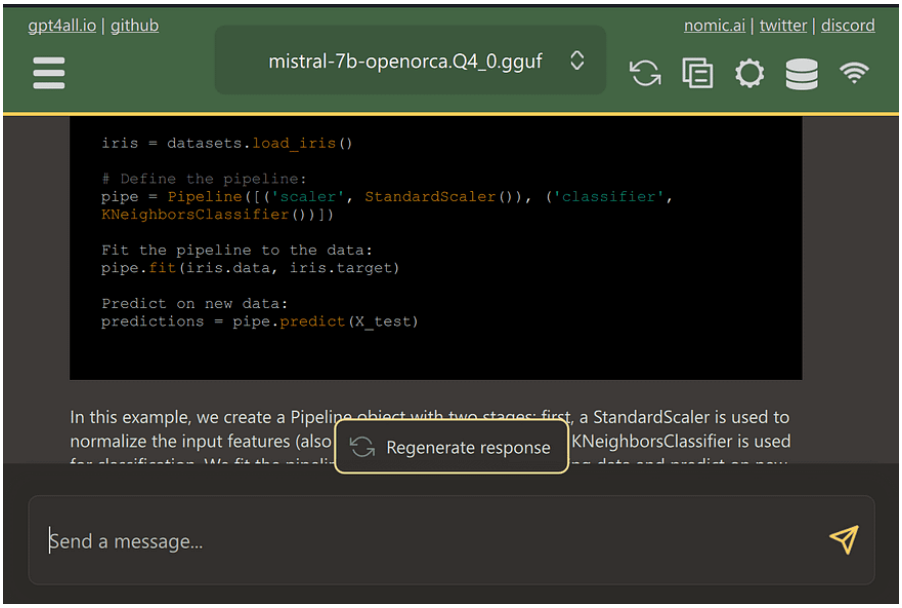


After downloading and installing, you should be able to find the application in the directory you specified in the installer. You will find a desktop icon for GPT4All after installation.

Next, choose the model from the panel that suits your needs and start using it. If you have CUDA (Nvidia GPU) installed, GPT4ALL will automatically start using your GPU to generate quick responses of up to 30 tokens per second.



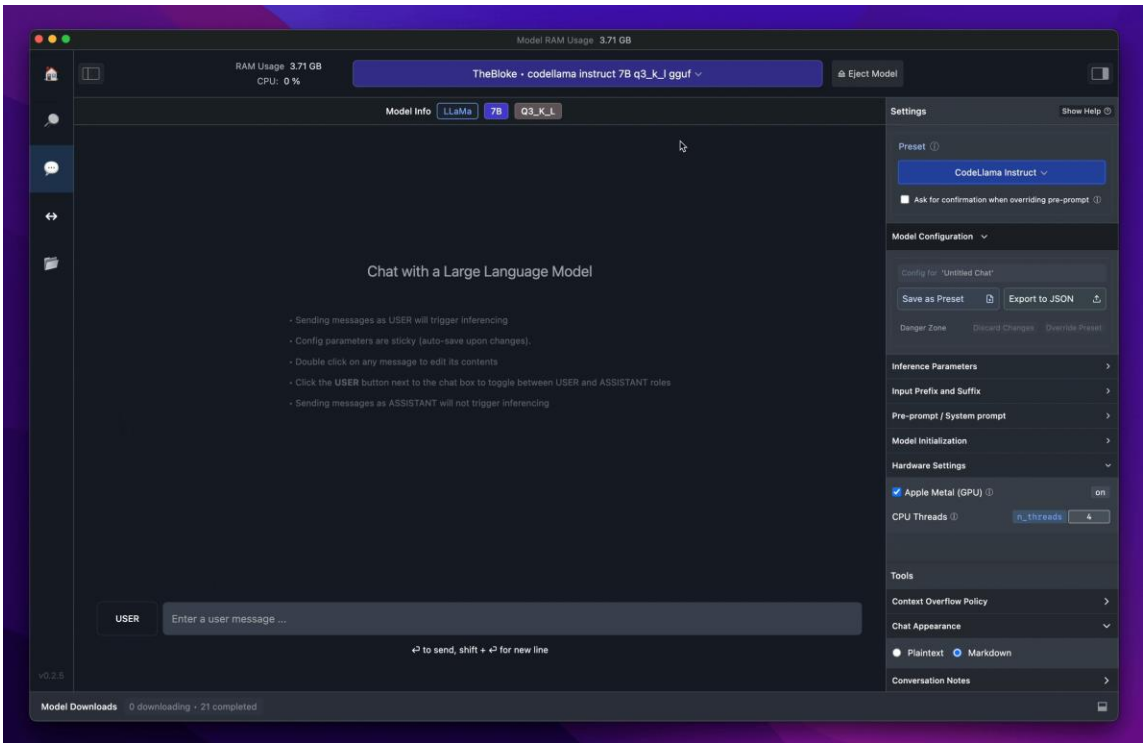
You can provide access to multiple folders containing important documents and code, and GPT4ALL will generate responses using Retrieval-Augmented Generation. GPT4ALL is user-friendly, fast, and popular among the AI community.



4.2. LM Studio

LM Studio is an easy-to-use desktop app for experimenting with local and open-source Large Language Models (LLMs). **With LM Studio, you can:**

- Run LLMs on your laptop, entirely offline.
- Use models through the in-app Chat UI or an OpenAI-compatible local server.
- Download any compatible model files from Hugging Face repositories.
- Discover new & noteworthy LLMs on the app's home page.



LM Studio offers several advantages over GPT4ALL. The user interface is excellent, and you can install any model from Hugging Face Hub with a few clicks.

Additionally, it provides GPU offloading and other options that are not available in GPT4ALL. However, LM Studio is a closed source, and it doesn't have the option to generate context-aware responses by reading project files.

4.3. Ollama

Ollama provides a **lightweight and user-friendly** way to set up and run various open-source LLMs on your computer. This eliminates the need for complex configurations or relying on external servers, making it ideal for various purposes:

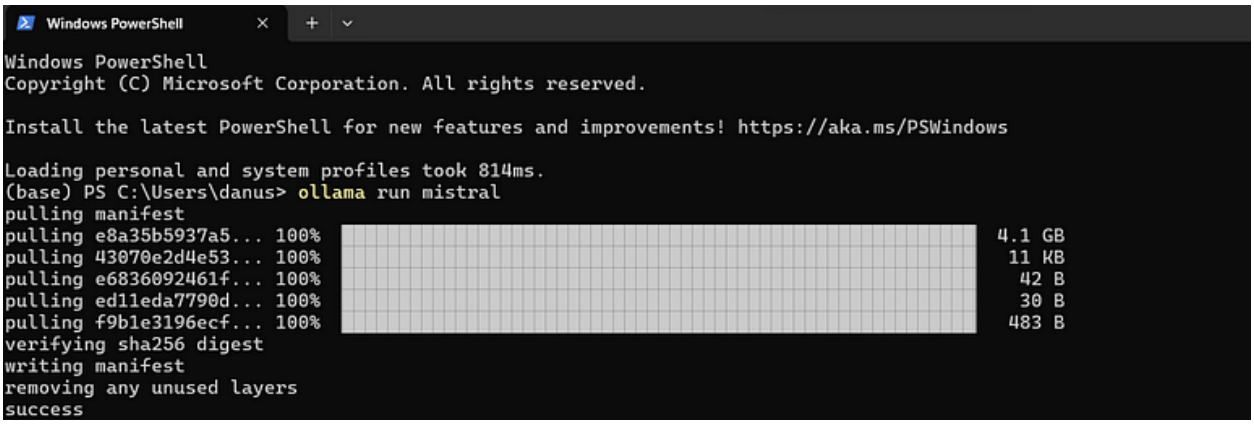
- **Development:** It allows developers to **experiment and iterate quickly** on LLM projects without needing to deploy them to the cloud.
- **Research:** Researchers can use Ollama to study LLM behavior in a **controlled environment**, facilitating in-depth analysis.
- **Privacy:** Running LLMs locally ensures that your data **never leaves your machine**, which is crucial for sensitive information.

Ollama comes with a pre-built library of **trained language models**, such as:

- **Llama 2:** A large language model capable of various tasks like text generation, translation, and question answering.
- **Mistral:** A factual language model trained on a massive dataset of text and code.
- **Gemma:** A conversational language model designed for engaging dialogue.
- **LLaVA:** A robust model trained for both chat and instruction use cases.

This library allows you to **easily integrate these pre-trained models** into your applications, eliminating the need to train them from scratch, and saving time and resources. Ollama accelerates running models using NVIDIA GPUs as well as modern CPU instruction sets such as AVX and AVX2 if available. No configuration or virtualization is required!

In the example below we can download mistral LLM:



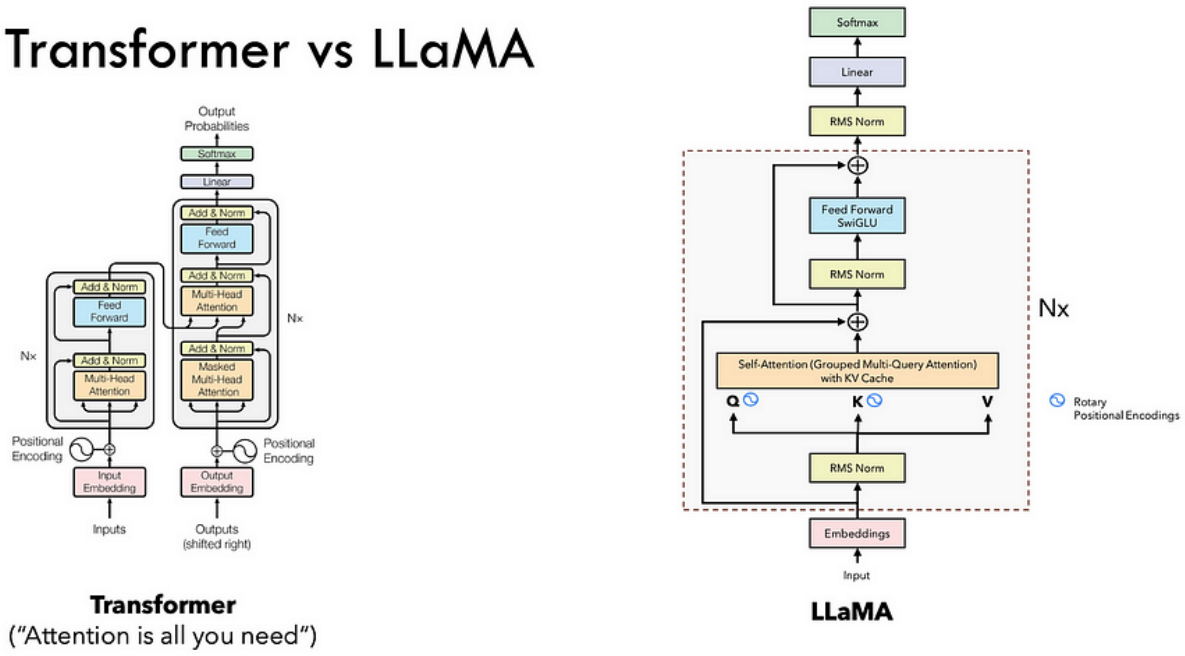
After the model is downloaded, we can start asking it and chatting with it:

```
>>> Generate a poem for friendship
 In the quiet halls of memory's keep,
Beyond the veil of fleeting time,
A bond as deep and true as sleep,
Lies the echo of a friend so prime.

Through laughter shared and tears unshed,
We journeyed side by side, entwined,
Our hearts like threads of crimson red,
A friendship woven in life's tapestry designed.

In whispered words and glances cast,
The language of the heart we spoke,
And though our worlds were vast and past,
Together we forged a bond that wouldn't break or choke.

We danced beneath the moon's soft glow,
Laughed in the sun's warm, golden light,
Through storms and trials, through joy and woe,
Our friendship stood as beacon bright.

A mirror reflecting souls so dear,
In moments sweet and moments trying,
With every year that we hold dear,
Our bond grew stronger, deepening, lying.

Oh, friendship, pure and true, a gift beyond measure,
An eternal flame that keeps our spirits warm,
A bond unbroken, through the sands of time's treasure,
A testament to love, a precious form.
```

4.4. LLaMA.cpp

LLaMa.cpp was developed by Georgi Gerganov. It implements the Meta's LLaMa architecture in efficient C/C++, and it is one of the most dynamic open-source communities around the LLM inference with more than 390 contributors, 43000+ stars on the official GitHub repository, and 930+ releases.

Llama.cpp's backbone is the original Llama model, which is also based on the transformer architecture. The authors of Llama leverage various improvements that were subsequently proposed and used different models such as PaLM.



You can install it using the following command:

```
pip install llama-cpp-python
```

Once it is installed you can import it using the following command:

```
from llama_cpp import Llama
```

The Llama class imported above is the main constructor leveraged when using Llama.cpp, and it takes several parameters and is not limited to the ones below. The complete list of parameters is provided in the **official documentation**:

- **model_path**: The path to the Llama model file being used
- **prompt**: The input prompt to the model. This text is tokenized and passed to the model.
- **device**: The device to use for running the Llama model; such a device can be either CPU or GPU.

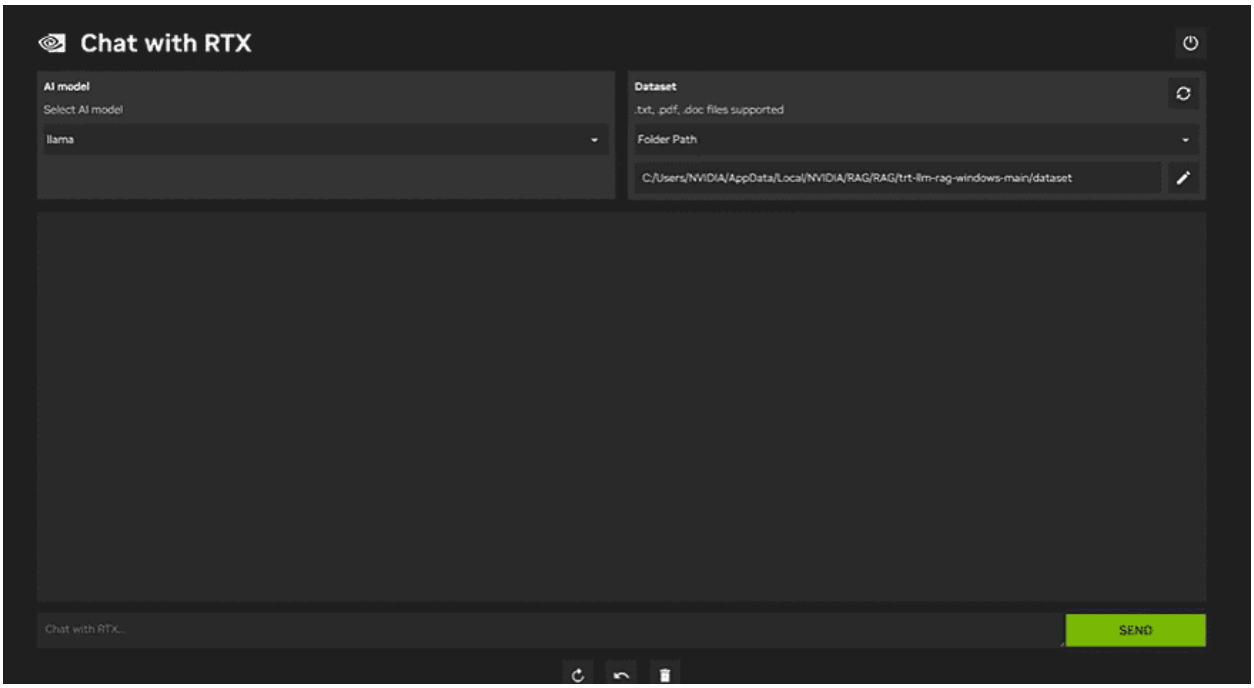
- **max_tokens**: The maximum number of tokens to be generated in the model's response.
- **stop:** A list of strings that will cause the model generation process to stop.
- **temperature:** This value ranges between 0 and 1. The lower the value, the more deterministic the result. On the other hand, a higher value leads to more randomness, hence more diverse and creative output.
- **top_p**: This is used to control the diversity of the predictions, meaning that it selects the most probable tokens whose cumulative probability exceeds a given threshold. Starting from zero, a higher value increases the chance of finding a better output but requires additional computations.
- **echo**: A boolean used to determine whether the model includes the original prompt at the beginning (True) or does not include it (False)

For instance, let's consider that we want to use a large language model called <MY_AWESOME_MODEL> stored in the current working directory, the instantiation process will look like this:

```
# Instanciate the model
my_aweseome_llama_model = Llama(model_path="./MY_AWESOME_MODEL")

prompt = "This is a prompt"
max_tokens = 100
temperature = 0.3
top_p = 0.1
echo = True
stop = ["Q", "\n"]


# Define the parameters
model_output = my_aweseome_llama_model(
       prompt,
       max_tokens=max_tokens,
       temperature=temperature,
       top_p=top_p,
       echo=echo,
       stop=stop,
   )
final_result = model_output["choices"][0]["text"].strip()
```

4.5. NVIDIA Chat with RTX

ChatRTX is a demo app that lets you personalize a GPT large language model (LLM) connected to your own content — docs, notes, or other data. Leveraging retrieval-augmented generation (RAG), TensorRT-LLM, and RTX acceleration, you can query a custom chatbot to quickly get contextually relevant answers. Because it all runs locally on your Windows RTX PC or workstation, you'll get fast and secure results.



With Chat with RTX, you can run LLaMA and Mistral models locally on your laptop. It's a fast and efficient application that can even learn from documents you provide or YouTube videos. However, it's important to note that Chat with RTX relies on TensorRTX-LLM, which is only supported on 30 series GPUs or newer.

5. Deploying LLMs: Top Learning & Educational Resources to Get Started

Deploying Large Language Models (LLMs) is pivotal in leveraging their capabilities across various applications, from enhancing user experiences to addressing privacy concerns. There are four distinct deployment techniques: local, demo, server, and edge deployment.

In this article, you will be provided with a selection of learning resources for each deployment technique, equipping readers with the knowledge needed to navigate and implement these techniques effectively.

The section begins by emphasizing the significance of local deployment, offering insights into five free tools for running LLMs locally on personal devices. It then delves into demo

deployment, where readers can learn to build interactive applications with minimal coding experience. Server deployment is subsequently addressed, guiding readers through the process of deploying LLMs using cloud platforms and containers. Finally, the article elucidates edge deployment, highlighting the benefits of embedding LLMs directly into real-world systems and providing resources for implementing this approach.

This comprehensive resource is tailored for individuals seeking to harness the power of LLMs across diverse deployment scenarios. Whether you are a novice exploring local deployment or an experienced developer venturing into edge computing, this article offers invaluable learning materials to accelerate your journey. Designed to empower AI enthusiasts, developers, and researchers alike, this article serves as a roadmap for deploying LLMs effectively and efficiently.

5.1. Local Deployment

While accessing LLM-based chatbots online is simple with just an internet connection and a good browser, it comes with potential privacy risks. For example, OpenAI stores your interactions and metadata to improve their models, raising concerns for privacy-conscious users. Opting to use these models locally provides a solution for those seeking greater control over their data.

In this section, you will explore five methods to utilize large language models (LLMs) locally. Compatible across major operating systems, these tools can be swiftly downloaded and installed.

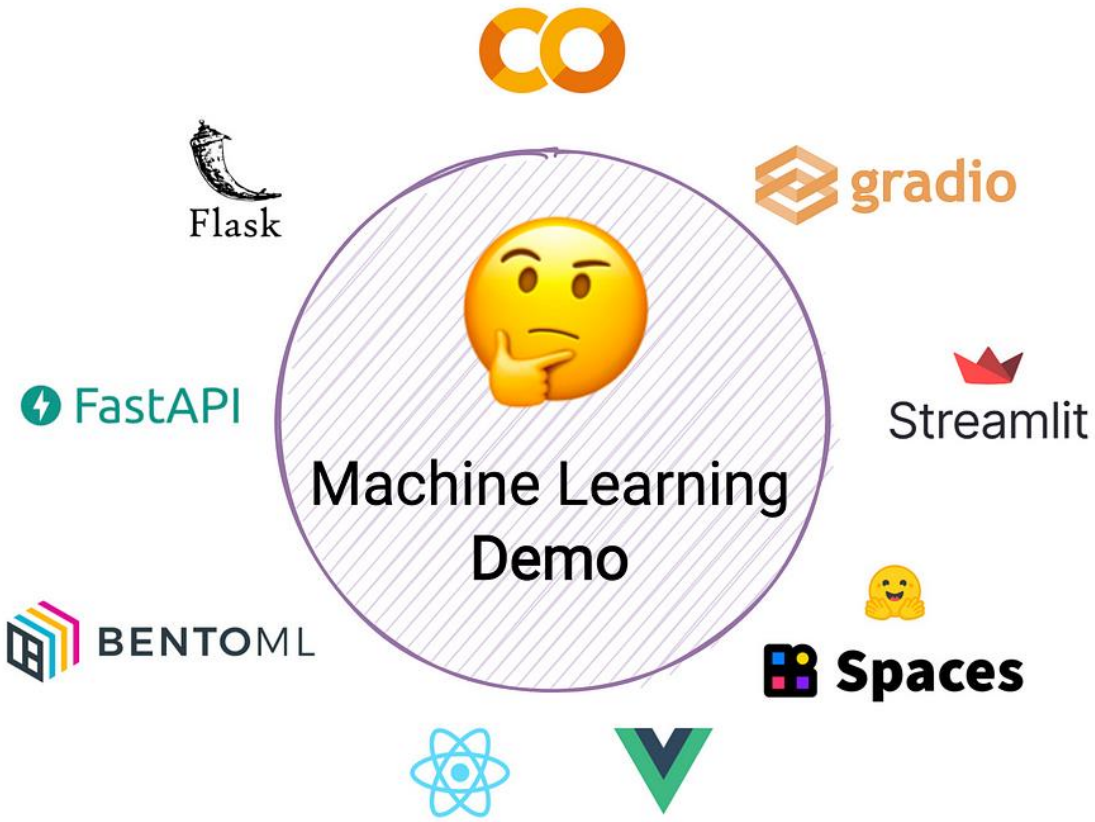
With locally-run LLMs, you retain control over model selection and can easily access models from the HuggingFace hub. Moreover, granting access to project folders enables context-aware responses.

Learning Resources:

- **5 Free Tools to Run Large Language Models (LLM) Locally on Your Laptop**

5.2. Demo deployment

In this section, you will learn how you can with just a few lines of code, create a user-friendly app (usable for non-coders) to take input text then apply different tasks with an open-source large language model, and display the output. By the end of the course, you'll gain the practical knowledge to rapidly build interactive apps and demos to validate your project and ship faster.



Learning Resources:

- **Streamlit — Build a basic LLM app**
- **Building Generative AI Applications with Gradio**
- **Serving an LLM Application as an API Endpoint using FastAPI in Python**
- **Deploying ML Models in 60 Minutes using Python, Flask & Render | Step-by-Step Tutorial**

5.3. Server deployment

Deploying Large Language Models (LLMs) is a step towards enhancing user experience. But, knowing where to start and which aspects to consider before LLM deployment is essential. LLMs have been instrumental in powering everything from machine translation to content creation to virtual assistants and chatbots.

In this section, you will learn how to deploy LLMs using Amazon bedrock, amazon Sagamaker, and Hugging Face inference container.

Learning Resources:

- **Serverless LLM apps with Amazon Bedrock**
- **HF LLM Inference Container**
- **Philschmid blog**
- **Build Customize and Deploy LLMs at scale on Azure with NVIDIA NeMo | DISFP08**

5.4. Edge deployment

Running LLMs on the edge is of great importance. By embedding LLMs directly into real-world systems such as the copilot services (coding, smart reply, and office) on laptops, in-car entertainment systems, vision-language assistants in robots or spaceship control interfaces, users can access instant responses and services without relying on a stable internet connection.

Moreover, this approach alleviates the inconvenience of queuing delays often associated with cloud services. As such, running LLMs on the edge enhances user experience and addresses privacy concerns, as sensitive data remains localized and reduces the risk of potential breaches.

In this learning section, you will learn how to deploy LLMs on edge devices using **MLC**, **TinyChat**, and **NVIDIA IGX Orin Developer Kit**.

Learning Resources:

- **Bringing Open Large Language Models to Consumer Devices**
- **Bringing Hardware Accelerated Language Models to Android Devices**
- **TinyChat: Large Language Model on the Edge**
- **Deploy Large Language Models at the Edge with NVIDIA IGX Orin Developer Kit**
- **Bringing GenAI Offline: running SLMs like Phi-2/Phi-3 and Whisper Models on Mobile Devices**

6. Getting Started with LLM Inference Optimization: Best Resources

Combining layers in transformer models makes them bigger and better at understanding language tasks. But making these big models costs a lot to train and they need a lot of memory and computer power to use afterward.

The most popular Large Language Models (LLM) today such as ChatGPT have billions of settings and sometimes they have to handle long pieces of text, which makes them even more expensive to use.

For example, RAG pipelines require putting large amounts of information into the input of the model, greatly increasing the amount of processing work the LLM has to do.

In this section, you will be provided with a comprehensive list of resources to delve into the foremost challenges encountered in LLM inference and proffer practical solutions.

6.1. Understanding LLM Inference Optimization

Stacking transformer layers to create large models results in better accuracies, few-shot learning capabilities, and even near-human emergent abilities on a wide range of language tasks.

These foundation models are expensive to train, and they can be memory- and compute-intensive during inference (a recurring cost). The most popular large language models (LLMs) today can reach tens to hundreds of billions of parameters in size and, depending

on the use case may require ingesting long inputs (or contexts), which can also add expense.

For example, RAG pipelines require putting large amounts of information into the input of the model, greatly increasing the amount of processing work the LLM has to do.

In this section, you will explore the most pressing challenges in LLM inference, along with some practical solutions. It would be best if you had a basic understanding of transformer architecture and the attention mechanism in general. It is essential to grasp the intricacies of LLM inference, which we will address in the next section.

Learning Resources:

- **Mastering LLM Techniques: Inference Optimization by Nvidia**
- **LLM Inference by Databricks**

6.2. Deep Understanding of LLM Inference Optimization

Open-source LLMs are great for conversational applications, but they can be difficult to scale in production and delivery latency and throughput that are incompatible with your cost-performance objectives.

In this section, you will zoom in on optimizing LLM inference, and study key mechanisms that help reduce latency and increase throughput: the KV cache, continuous batching, and speculative decoding, including the state-of-the-art Medusa approach.

Learning Resources:

- **Deep Dive: Optimizing LLM inference**

6.3. LLM Inference by Hugging Face

Unlike CPUs, GPUs are the standard choice of hardware for machine learning because they are optimized for memory bandwidth and parallelism.

To keep up with the larger sizes of modern models or to run these large models on existing and older hardware, there are several optimizations you can use to speed up GPU inference.

In this section, you'll learn how to use FlashAttention-2 (a more memory-efficient attention mechanism), BetterTransformer (a PyTorch native fastpath execution), and bitsandbytes to quantize your model to a lower precision. Finally, learn how to use 🤗 Optimum to accelerate inference with ONNX Runtime on Nvidia and AMD GPUs.

Learning Resources:

- **GPU Inference by Hugging Face**
- **Optimizing LLMs for Speed and Memory by Hugging Face**
- **Assisted Generation by Hugging Face**

6.4. LLM Inference Optimization Libraries & Tools

In the last section, you will explore and compare the features and capabilities of leading LLM libraries for inference Optimization. In addition to exploring you will explore how to optimize OpenAI in inference and finally, you will explore the future trends in this field.

Learning Resources:

- **LLMs at Scale: Comparing Top Inference Optimization Libraries**
- **Large language model inference optimizations on AMD GPUs**
- **Accelerate Large Language Model (LLM) Inference on Your Local PC**
- **OpenAI Latency Optimization**
- **Inference Optimization Strategies for Large Language Models: Current Trends and Future Outlook**

7. What are LLMOps and How to Get Started with It?

LLMOps is primarily focused on enhancing operational capabilities and establishing the necessary infrastructure for refining existing foundational models and seamlessly integrating these optimized models into products.

Although LLMOps may not seem groundbreaking to most observers within the MLOps community, it serves as a specialized subset within the broader MLOps domain. A more specific definition can elucidate the intricate requirements involved in fine-tuning and deploying these models effectively.

Foundational models, such as GPT-3 with its massive 175 billion parameters, demand substantial amounts of data and compute resources for training. While fine-tuning these models may not require the same scale of data or computational power, it remains a significant task that necessitates robust infrastructure capable of parallel processing and handling large datasets.

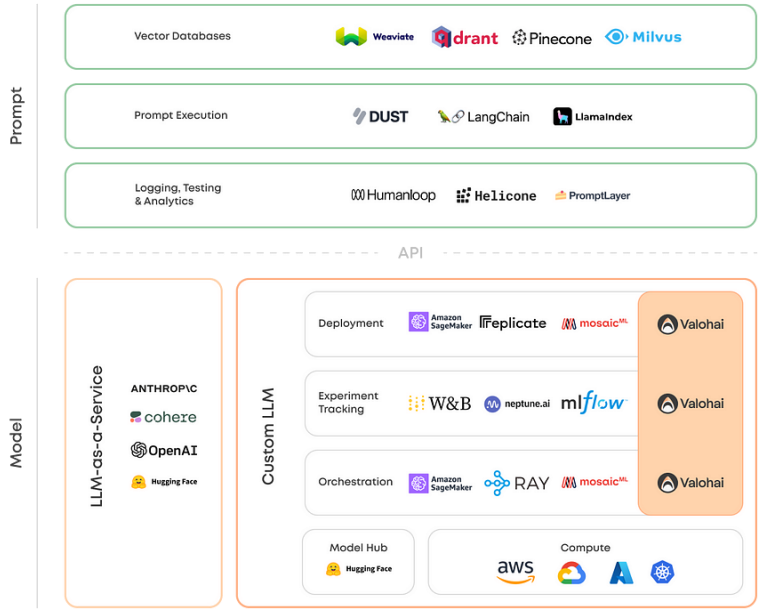
This article delves into essential resources to help initiate your journey into LLMOps, providing valuable insights and guidance for getting started effectively.

LLMOps consists of two parts:

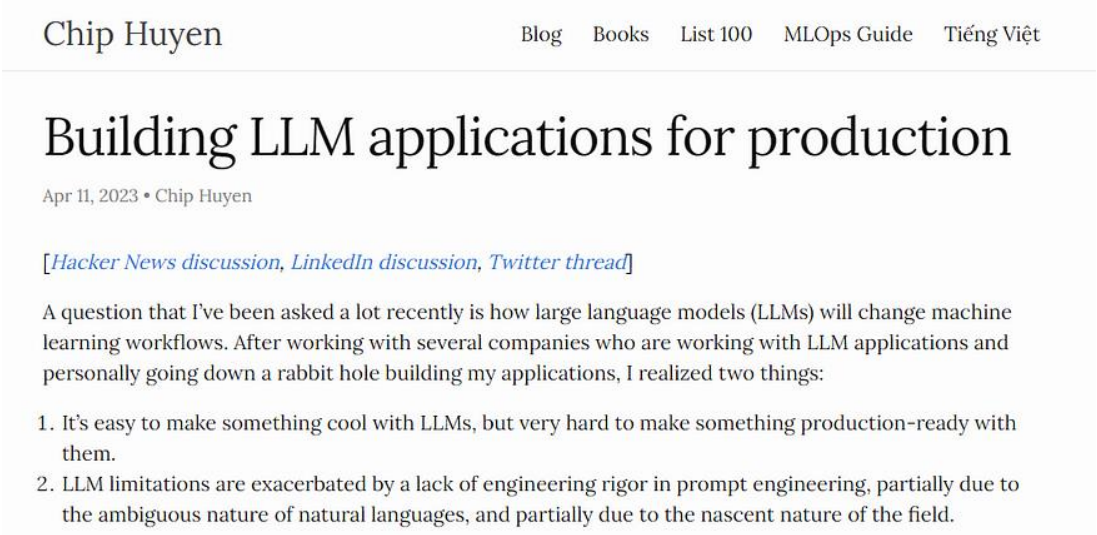
1. **Large Language Models**: LLM-as-a-Service is where a vendor offers the LLM as an API on their infrastructure. This is how primarily closed-source models are delivered.

Custom LLM stack is a broader category of tools necessary for fine-tuning and deploying proprietary solutions built on top of open-source models.

2. **Prompt Engineering tools**: enable in-context learning instead of fine-tuning at lower costs and without using sensitive data. Vector Databases retrieve contextually relevant information for certain prompts. Prompt Execution enables optimizing and improving the model output based on managing prompt templates to building chain-like sequences of relevant prompts. Prompt Logging, Testing, and Analytics ... Let's just say it's an emerging space that has no categories yet.



7.1. Building LLM Applications for Production



Chip Huyen

Blog Books List 100 MLOps Guide Tiếng Việt

Building LLM applications for production

Apr 11, 2023 • Chip Huyen

[*Hacker News discussion*, *LinkedIn discussion*, *Twitter thread*]

A question that I've been asked a lot recently is how large language models (LLMs) will change machine learning workflows. After working with several companies who are working with LLM applications and personally going down a rabbit hole building my applications, I realized two things:

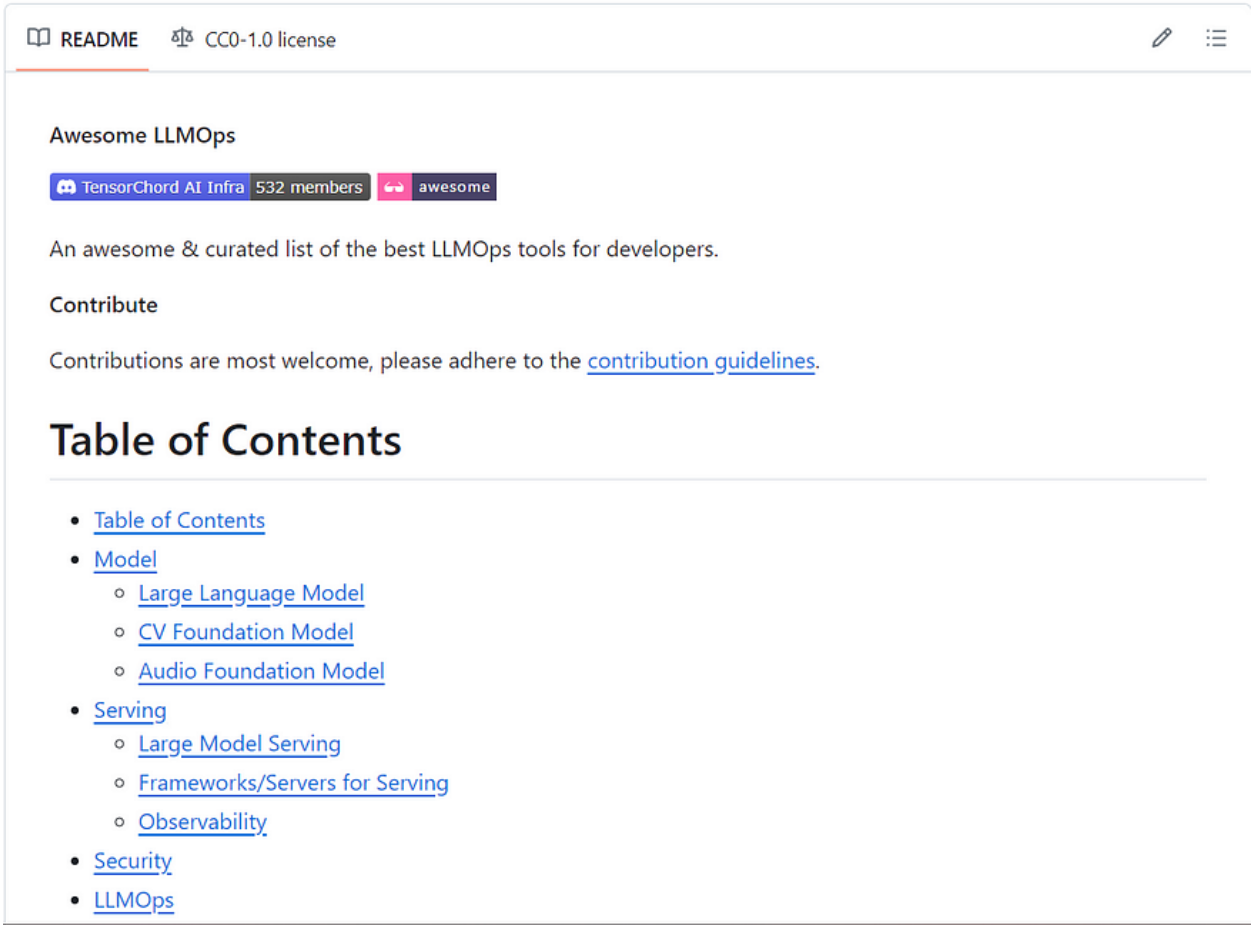
1. It's easy to make something cool with LLMs, but very hard to make something production-ready with them.
2. LLM limitations are exacerbated by a lack of engineering rigor in prompt engineering, partially due to the ambiguous nature of natural languages, and partially due to the nascent nature of the field.

It's easy to build LLMs, but very hard to make something production-ready with them. This **article** by Chip Huyen covers how to put LLMs into production.

This post consists of three parts:

- Part 1 discusses the key challenges of productionizing LLM applications and the solutions that I've seen.
- Part 2 discusses how to compose multiple tasks with control flows (e.g. if statement, for loop) and incorporate tools (e.g. SQL executor, bash, web browsers, third-party APIs) for more complex and powerful applications.
- Part 3 covers some of the promising use cases I've seen companies building on top of LLMs and how to construct them from smaller tasks.

7.2. Awesome LLMOps


README CC0-1.0 license

Awesome LLMOps

TensorChord AI Infra 532 members awesome

An awesome & curated list of the best LLMOps tools for developers.

Contribute

Contributions are most welcome, please adhere to the contribution guidelines.

Table of Contents

- Table of Contents
- Model
  - Large Language Model
  - CV Foundation Model
  - Audio Foundation Model
- Serving
  - Large Model Serving
  - Frameworks/Servers for Serving
  - Observability
- Security
- LLMOps

This **GitHub repo** contains a curated list of the best LLMOps resources and tools for developers. Here is the content:

1. Model

- Large Language Model
- CV Foundation Model
- Audio Foundation Model

2. Serving

- Large Model Serving
- Frameworks/Servers for Serving
- Observability

3. Security

4. LLMOps

5. Search

- Vector search

6. Code AI

7. Training

- IDEs and Workspaces
- Foundation Model Fine Tuning
- Frameworks for Training
- Experiment Tracking
- Visualization
- Model Editing

8. Data

- Data Management
- Data Storage
- Data Tracking
- Feature Engineering
- Data/Feature enrichment

9. Large Scale Deployment

- ML Platforms
- Workflow
- Scheduling
- Model Management

10. Performance

- ML Compiler
- Profiling

11. AutoML

12. Optimizations

13. Federated ML

14. Awesome Lists

7.3. LLMOps

In this **course**, you'll go through the LLMOps pipeline of pre-processing training data for supervised instruction tuning, and adapt a supervised tuning pipeline to train and deploy a custom LLM.

This is useful in creating an LLM workflow for your specific application. For example, create a question-answer chatbot tailored to answer Python coding questions, which you'll do in this course.

Through the course, you'll go through key steps of creating the LLMOps pipeline:

- Retrieve and transform training data for supervised fine-tuning of an LLM.
- Version your data and tuned models to track your tuning experiments.
- Configure an open source supervised tuning pipeline and then execute that pipeline to train and then deploy a tuned LLM.
- Output and study safety scores to responsibly monitor and filter your LLM application's behavior.
- Try out the tuned and deployed LLM yourself in the classroom!

- Tools you'll practice with include BigQuery data warehouse, the open-source Kubeflow Pipelines, and Google Cloud.

7.4. Automated Testing for LLMOps

In this **course**, you will learn how to create a continuous integration (CI) workflow to evaluate your LLM applications at every change for faster, safer, and more efficient application development.

When building applications with generative AI, model behavior is less predictable than traditional software. That's why systematic testing can make an even bigger difference in saving you development time and cost.

Continuous integration, a key part of LLMOps, is the practice of making small changes to software in development and thoroughly testing them to catch issues early when they are easier to fix. With a robust automated testing pipeline, you'll be able to isolate bugs before they accumulate — when they're easier and less costly to fix. Automated testing lets your team focus on building new features so that you can iterate and ship products faster.

After completing this course, you will be able to:

- Write robust LLM evaluations to cover common problems like hallucinations, data drift, and harmful or offensive output.
- Build a continuous integration (CI) workflow to automatically evaluate every change to your application.
- Orchestrate your CI workflow to run specific evaluations at different stages of development.

8. Securing LLMs: Best Learning & Educational Resources

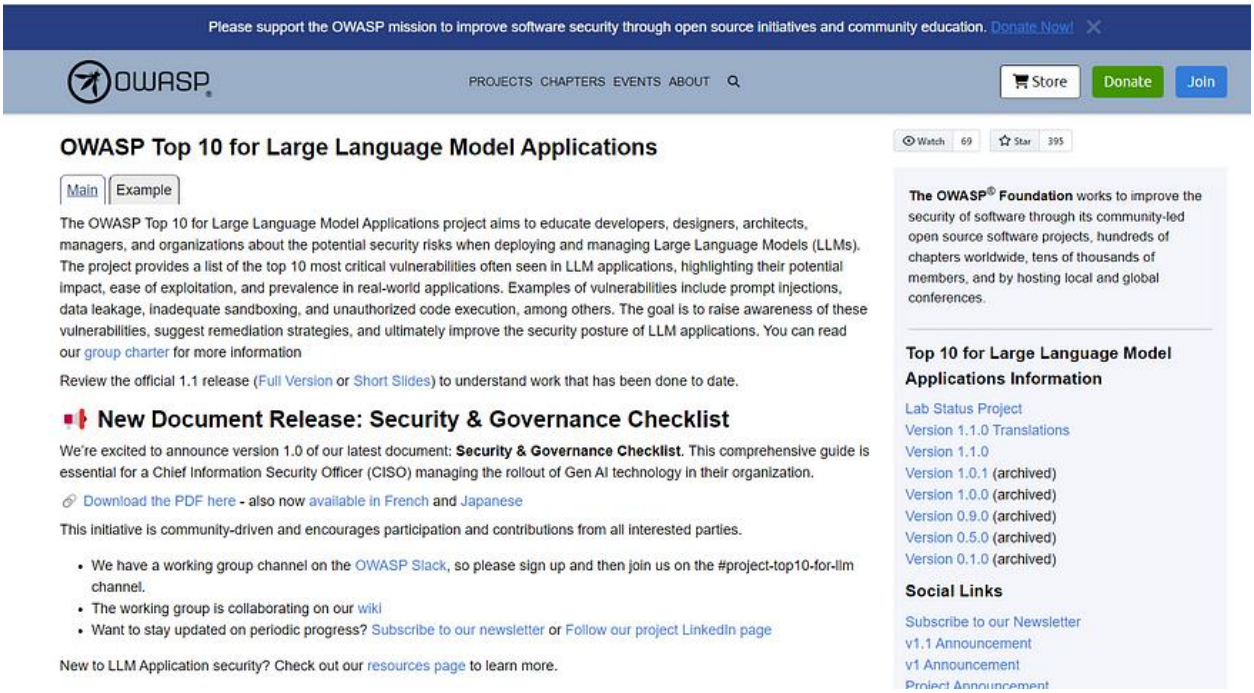
Large Language Models (LLMs) represent a revolutionary advancement in artificial intelligence, yet their deployment introduces significant security challenges.

This article provides a comprehensive resource for developers, engineers, architects, and managers seeking to fortify their understanding and defenses against potential vulnerabilities in LLM applications.

This resource is indispensable for anyone involved in the development, deployment, or management of LLM applications. By arming readers with knowledge and practical tools, it empowers them to safeguard their systems against potential threats.

Whether you're a seasoned developer or a novice explorer in the realm of AI, this article equips you to navigate the intricate landscape of LLM security with confidence and competence.

8.1. List of the 10 most critical vulnerabilities seen in LLM applications

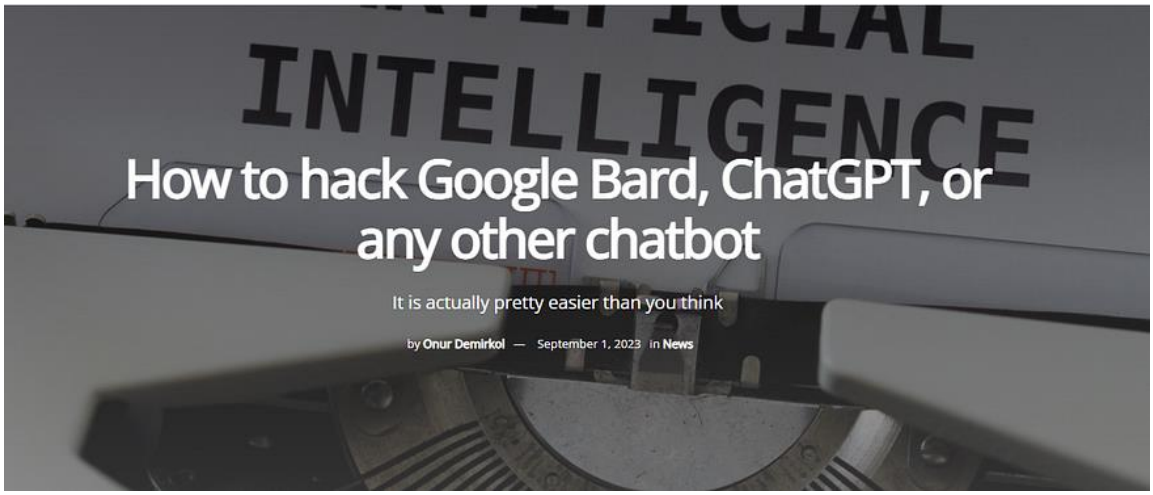


This article aims to educate developers, designers, architects, managers, and organizations about the potential security risks when deploying and managing Large Language Models (LLMs).

The **article** lists the top 10 most critical vulnerabilities often seen in LLM applications, highlighting their potential impact, ease of exploitation, and prevalence in real-world applications.

Examples of vulnerabilities include prompt injections, data leakage, inadequate sandboxing, and unauthorized code execution, among others. The goal is to raise awareness of these vulnerabilities, suggest remediation strategies, and ultimately improve the security posture of LLM applications.

8.2. How to hack Google Bard, ChatGPT, or any other chatbot



Google Bard, ChatGPT, Bing, and all those chatbots have their own security systems, but they are, of course, not invulnerable. This **article** introduces you to how to hack Google and all these other huge tech companies and get the idea behind LLM Attacks.

8.3. Prompt Injection Primer for Engineers

Prompt injection is the highest profile vulnerability in AI-powered features and applications. It's also one of the most misunderstood. The impact varies greatly depending on who will use the feature, what data is accessible, and what functionality is exposed to the LLM. This **guide** aims to assist developers in creating secure AI-powered applications and features by helping them understand the actual risks of prompt injection.

PIPE - Prompt Injection Primer for Engineers

Bringing clarity to questions about Prompt Injection Security

Table of Contents

1. Introduction
2. Background
3. Do I need to worry about prompt injection?
4. Flowchart
5. Attack Scenarios
6. New Vectors for Traditional Web Vulnerabilities
7. Mitigations
9. Multi-modal Considerations
10. Hacking on AI-Powered Apps
11. Conclusion

Note: If you'd rather read the PDF, click here!

8.4. Quality and Safety for LLM Applications

It's always crucial to address and monitor safety and quality concerns in your applications. Building LLM applications poses special challenges.

In this **course**, you'll explore new metrics and best practices to monitor your LLM systems and ensure safety and quality. You'll learn how to:

- Identify hallucinations with methods like SelfCheckGPT
- Detect jailbreaks (prompts that attempt to manipulate LLM responses) using sentiment analysis and implicit toxicity detection models.
- Identify data leakage using entity recognition and vector similarity analysis.
- Build your own monitoring system to evaluate app safety and security over time.

Upon completing the course, you'll have the ability to identify common security concerns in LLM-based applications and be able to customize your safety and security evaluation tools to the LLM that you're using for your application.

8.5. Red Teaming LLM Applications

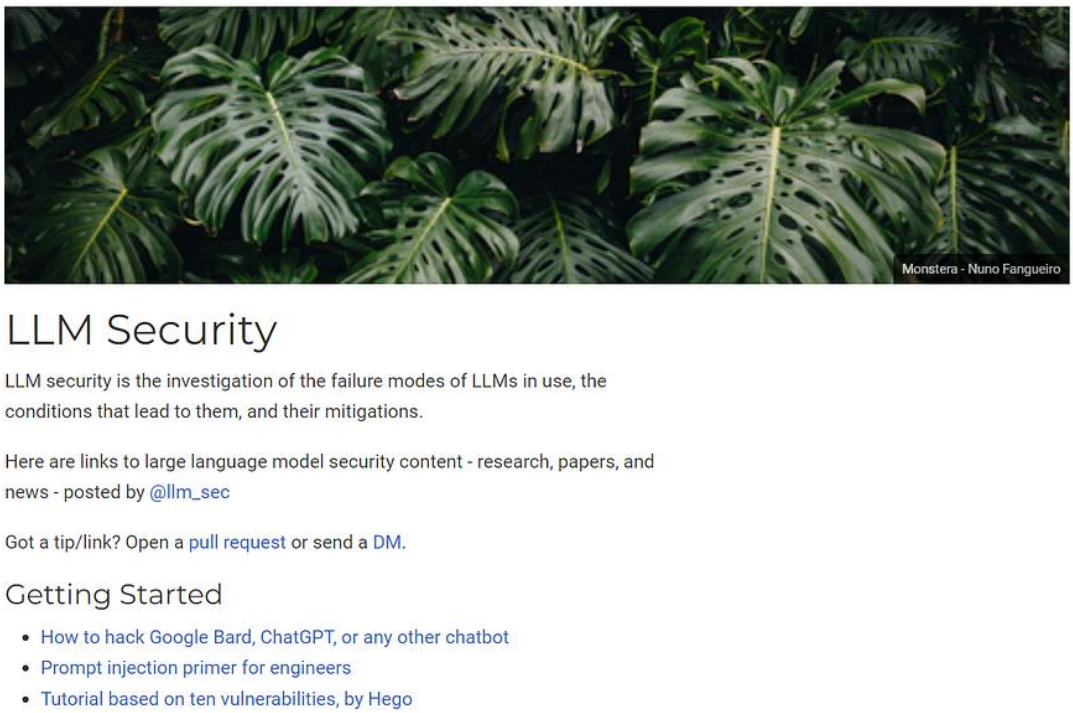
Learn how to test and find vulnerabilities in your LLM applications to make them safer. In this **course**, you'll attack various chatbot applications using prompt injections to see how the system reacts and understand security failures. LLM failures can lead to legal liability, reputational damage, and costly service disruptions. This course helps you mitigate these risks proactively. Learn industry-proven red teaming techniques to proactively test, attack, and improve the robustness of your LLM applications.

In this course:

- Explore the nuances of LLM performance evaluation, and understand the differences between benchmarking foundation models and testing LLM applications.
- Get an overview of fundamental LLM application vulnerabilities and how they affect real-world deployments.
- Gain hands-on experience with both manual and automated LLM red-teaming methods.
- See a full demonstration of red-teaming assessment, and apply the concepts and techniques covered throughout the course.

After completing this course, you will have a fundamental understanding of how to experiment with LLM vulnerability identification and evaluation on your own applications.

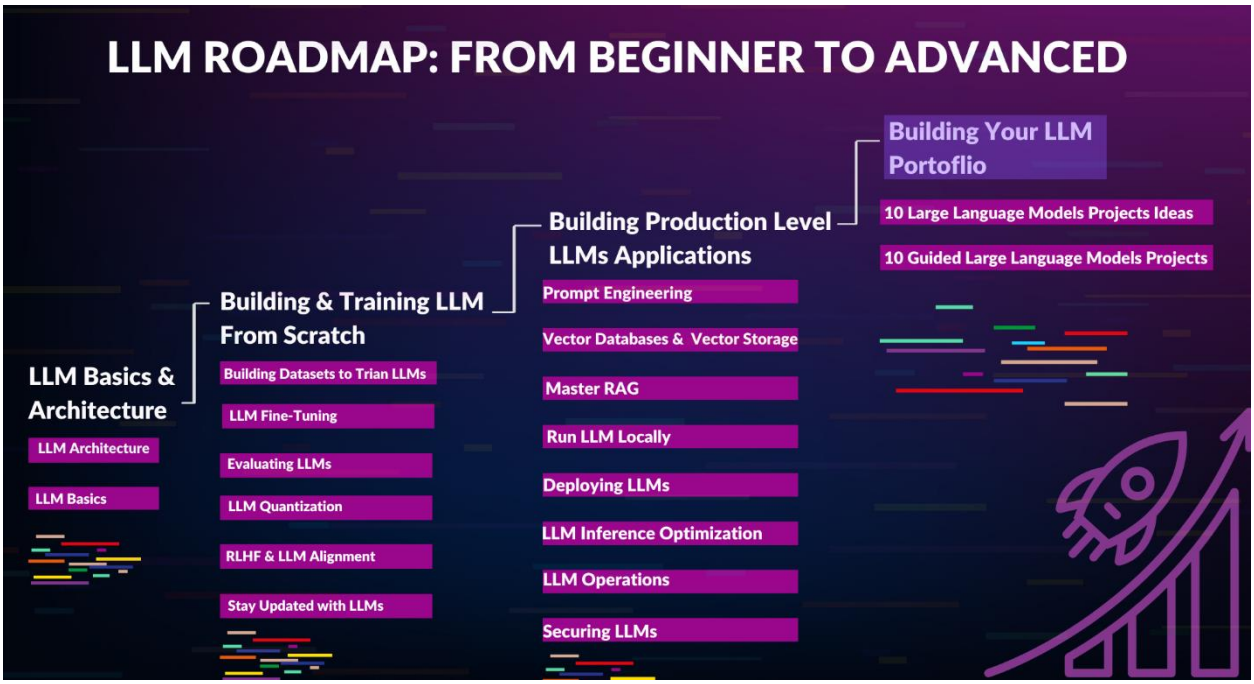
8.6. LLM Security: A comprehensive list of resources and papers



LM security is the investigation of the failure modes of LLMs in use, the conditions that lead to them, and their mitigations. It contains a curated list of links to large language model security content — research, papers, and news — posted by **llm_sec**

Part IV: Building Your LLM Portfolio

The final section is dedicated to helping you build your portfolio, offering ten project ideas and ten guided projects to provide a solid starting point for showcasing your expertise to potential employers or clients.



1. 10 Large Language Models Projects Ideas To Build Your Portfolio

In the dynamic landscape of today's technology, large language models (LLMs) have emerged as transformative tools, capable of understanding and generating human-like text. Harnessing their power can open up a realm of possibilities for building innovative applications that not only demonstrate your expertise but also leave a lasting impact.

In this article, we present an enticing array of 10 project ideas centered around LLMs, each designed to help you construct end-to-end applications and curate a captivating portfolio. From content generation that sparks creativity to healthcare solutions that promote well-being, and from language translation with cultural sensitivity to tackling social issues, we explore diverse avenues that let your skills shine.

Whether you're an aspiring developer looking to bolster your portfolio or a seasoned enthusiast seeking to broaden your horizons, this article is your gateway to exploring the remarkable world of LLM-driven projects. Let's embark on this journey of innovation, learning, and showcasing the potential of large language models.

1.1. Content Generation

Large Language Models (LLMs) have gained significant recognition for their remarkable generative capabilities, enabling them to produce diverse and contextually relevant text. In the following section, we'll delve into project ideas that leverage these capabilities for distinct use cases.

1.1.1. Academic Motivation Letter Generation

The main and the most used feature of LLMs' is their ability to **generate** coherent bodies of text. Although there are a lot of discussions in the media and in the research communities about its usage, this technology is already being widely adapted in **copywriting** or **programming** to increase productivity.

When you are applying to an educational program whether it's a bachelor's, master's, or even doctoral degree one of the main requirements is to write a motivational letter. Although you can build this project by only engineering the prompt and filling in the relevant information about the role or using ChatGPT for it, this is a great small project to practice prompt engineering and using prompt templates.

Project Guide:

- **Model Selection:** Choose a suitable pre-trained LLM-based model, like GPT-3 or similar, with capabilities for text generation and summarization.
- **Develop User Interface:** Design a user-friendly web or desktop interface for easy content input and summarization output.
- **Prompt Design:** Ultize the prompt to generate content for an academic motivational letter

Technology Stack:

- **LLM-based Model**: Select a powerful language model such as GPT-3.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Create a responsive UI using **HTML**, **CSS**, and **JavaScript**.

1.2. Language Translation and Localization

1.2.1. Cultural Context Integration

Language translation often goes beyond mere word-for-word conversion; it involves understanding cultural nuances, idiomatic expressions, and context to provide accurate and culturally sensitive translations. This project aims to enhance traditional language translation systems by incorporating cultural context understanding into the translation process. The goal is to create translations that not only convey the literal meaning but also capture the intended cultural implications and nuances of the original text.

Project Guide:

1. Research and Understanding:

- Gain insights into various cultures, idiomatic expressions, and cultural nuances that affect language usage and interpretation.

- Understand the challenges of translating idiomatic expressions and cultural-specific terms.

2. Data Collection and Analysis:

- Collect bilingual datasets that include both literal and culturally nuanced translations.
- Analyze the data to identify recurring idiomatic expressions and cultural context cues.

3. Model Selection:

- Choose a suitable machine translation model as the base, such as a neural machine translation model.
- Explore pre-trained language models that have demonstrated proficiency in understanding context and generating culturally sensitive content.

4. Cultural Context Annotation:

- Develop a system for annotating cultural context cues and idiomatic expressions in the training data.
- Annotate instances where a literal translation might not capture the cultural meaning accurately.

5. Model Training and Fine-Tuning:

- Train the selected model using the annotated dataset.
- Fine-tune the model to account for cultural variations and idiomatic expressions.

6. Evaluation Metrics:

- Define evaluation metrics that consider both the accuracy of translations and the preservation of cultural context.
- Develop a test dataset containing texts with varying degrees of cultural subtleties.

7. Testing and Iteration:

- Test the model's translations against the evaluation metrics and the test dataset.
- Iterate on the model architecture and fine-tuning process to improve translation quality.

8. Integration and Deployment:

- Integrate the enhanced translation model into an accessible platform, such as a web application or mobile app.
- Allow users to input text for translation and receive culturally sensitive translations.

Tech Stack:

- **Python**: For coding the translation model, data processing, and annotation tools.
- **Machine Translation Libraries**: Libraries like `transformers` for fine-tuning pre-trained language models for translation tasks.
- **Web Framework (optional):** If you're creating a web application, frameworks like Flask or Django can be used for building the user interface.
- **Database (optional):** If user data is collected, a database may be employed to store translations and improve the system over time.
- **Natural Language Processing (NLP) Tools:** NLP libraries like NLTK or spaCy for text analysis and processing.
- **Annotation Tools:** Custom or existing annotation tools for marking cultural context cues in the training data.

1.3. Entertainment and Gaming

1.3.1. Character Creation

The Fictional Character Generator is designed to automate the process of creating rich and diverse characters for various forms of storytelling, including literature, games, and

screenplays. The generator produces not only basic traits but also delves into detailed backstories, personalities, motivations, and quirks that make characters feel authentic and engaging.

Project Guide:

- **Design and Planning**: Define scope, traits, and information to generate.
- **Data Collection**: Gather data from existing characters in various media.
- **Model Choice:** Select a generative model like GPT-3.5 for character descriptions.
- **Input and Output:** Create a user interface for inputting preferences and displaying character profiles.
- **Trait Synthesis:** Develop algorithms for combining traits creatively.
- **Backstory and Personality:** Design prompts for generating character histories and personalities.
- **User Customization:** Allow users to customize generated characters.
- **Quality Evaluation:** Develop metrics to measure the quality and uniqueness of characters.

Tech Stack:

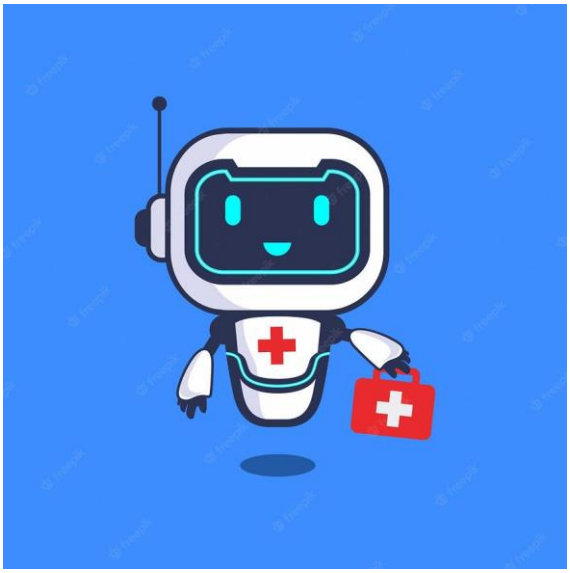
- Backend: Python for development and data processing.
- Web Framework: Flask or Django for user-friendly web application creation.
- Database (optional): For saving and modifying characters.
- NLP Libraries: spaCy for text processing.
- Generative Models: Utilize GPT-3.5 for text generation.
- Frontend: HTML, CSS, and JavaScript for interactive user interfaces.

1.4. Healthcare and Well-being

The medical field can benefit greatly from LLMs by utilizing them for medical applications. By training an LLM on extensive medical literature and patient data, you can develop a system that assists healthcare professionals in various medical tasks.

1.4. Medical Bot: Medical Diagnosis and Treatment Recommendations

In this project, you will develop a medical chatbot to diagnose the patient's disease based on his medical reports and history and recommend treatment for his case.



Project Guide:

- **Medical Data Collection:** Gather a diverse collection of patient diagnoses, medical literature, research papers, and patient data to train the LLM.
- **LLM Training:** Train the LLM on the medical dataset collected to learn disease patterns, treatment options, and relevant medical knowledge.
- **Diagnosis Assistance:** Implement the LLM to assist healthcare professionals in diagnosing diseases based on patient symptoms and medical history.
- **Treatment Recommendations:** Utilize the LLM to suggest appropriate treatment options for diagnosed conditions, considering medical guidelines and patient specifics.

Technology Stack:

- **LLM-based Model**: Utilize powerful language models like GPT-3 or similar for medical diagnosis and treatment recommendations.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Design a secure and user-friendly web-based interface using HTML, CSS, and JavaScript.

1.5. Social Good

1.5.1. Access to Information

Build a tool that converts complex information (legal documents, medical literature) into easily understandable language. The goal of this project is to leverage the capabilities of a large language model, such as GPT-3.5, to develop a tool that converts complex information from various domains (e.g., legal documents, medical literature) into easily understandable language.

This tool aims to bridge the gap between technical, specialized content and laypeople who may not have expertise in their respective field. By simplifying complex information, this tool can democratize access to critical knowledge and empower individuals to make informed decisions.

Project Guide:

- ***Understanding the Domain***: Before starting the technical development, it's important to understand the specific domains you'll be working with, such as legal documents and medical literature. Familiarize yourself with common terminologies, complex sentence structures, and the overall tone of the content.
- ***Data Collection and Preprocessing***: Gather a diverse set of text data from the chosen domains. This could include legal contracts, medical research papers, or other relevant documents. Preprocess the data by cleaning the text, removing unnecessary formatting, and segmenting the content into smaller units like paragraphs or sentences.

- ***Choosing the Language Model***: Select a suitable language model for this task. GPT-3.5 is a powerful option due to its ability to generate human-like text and handle various writing styles. You might need access to the OpenAI API to integrate the model into your tool.
- ***Fine-tuning (Optional)***: Depending on your specific requirements, you might consider fine-tuning the language model on a smaller dataset related to your chosen domains. Fine-tuning can help the model adapt better to the specific language patterns and terminologies used in legal and medical contexts.
- ***Developing the Tool***: The core of your project involves creating an application or tool that takes in complex text as input and generates simplified, easily understandable output. The tool could be a web application, a mobile app, or a command-line tool. Integrate the selected language model into your application's backend.
- ***User Interface (UI):*** Design a user-friendly interface that allows users to input complex text and receive the simplified output. The UI should be intuitive and straightforward, requiring minimal effort from the user's side.
- ***Text Simplification Strategies***: Implement text simplification strategies within the tool. These might include sentence restructuring, paraphrasing, definition provision for technical terms, and breaking down complex concepts into simpler explanations.
- ***Testing and Iteration***: Thoroughly test your tool with a variety of input texts from the chosen domains. Collect feedback from users and iterate on the tool's performance and user experience. This iterative process will help you refine the tool's output quality.
- ***Privacy and Security***: Ensure that the tool handles sensitive information (e.g., medical records) with the utmost care. Implement data encryption, user authentication, and other security measures to protect users' privacy.
- ***Documentation and Deployment***: Prepare comprehensive documentation that explains how to use the tool, its features, and any limitations. If applicable, deploy the tool to a web server or app store so that users can access it easily.

1.5.2. Fake News Detection

The objective of this project is to harness the capabilities of a large language model, such as GPT-3.5, to create a tool that can effectively detect fake news. With the proliferation of misinformation on the internet, the ability to automatically identify unreliable or deceptive information is crucial for maintaining informed decision-making and a well-informed public.

Project Guide:

- ***Data Collection:*** Collect a diverse dataset comprising both genuine news articles and fake news samples. The dataset should span different topics and domains to ensure the tool's effectiveness across various contexts. Ensure that the dataset is labeled, indicating whether each piece of content is genuine or fake.
- ***Data Preprocessing***: Clean and preprocess the collected data. Remove any irrelevant information, correct inconsistencies, and standardize the formatting of the text. This step is crucial for ensuring the quality of the training data.
- ***Selecting a Language Model***: Choose an appropriate language model for this task. GPT-3.5 is well-suited due to its understanding of context and the ability to generate coherent text. You may require access to the OpenAI API to integrate the model into your tool.
- ***Training and Fine-tuning***: Depending on the availability of a labeled dataset, you might consider fine-tuning the language model on your fake news detection task. Fine-tuning allows the model to learn specific patterns related to fake news and genuine news articles.

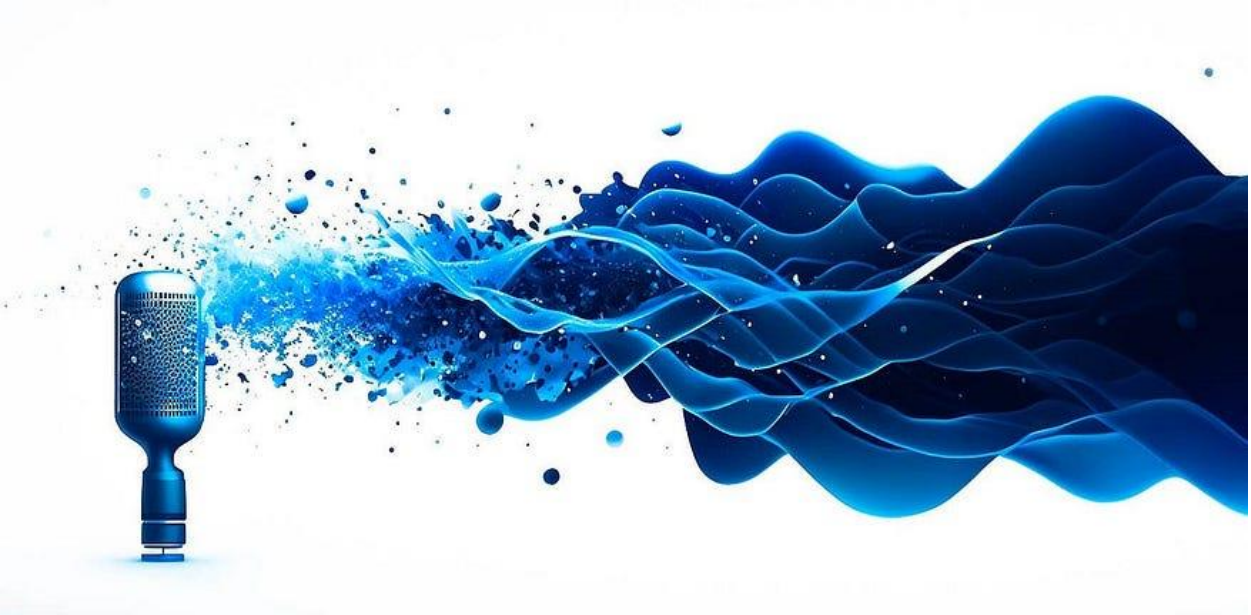
- ***Feature Extraction***: Extract relevant features from the text, such as linguistic patterns, sentiment, and writing style. These features will be used as inputs to the fake news detection model.
- ***Developing the Detection Tool***: Build an application that takes news text as input and provides a likelihood score indicating the probability of the input being fake. This could be a web application, browser extension, or API. Integrate the language model and any additional machine-learning components.
- ***Model Evaluation***: Assess the performance of your fake news detection tool using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Use a separate validation dataset that the model has not seen during training.
- ***Feedback Loop and Iteration***: Continuously improve the tool's performance by collecting user feedback and iteratively enhancing the model's accuracy and reliability.
- ***User Interface and Interpretability***: Design a user interface that facilitates easy input of news text and clearly presents the tool's output. It might also be valuable to provide explanations for the model's decisions, increasing user trust and transparency.
- ***Scaling and Deployment***: Prepare the tool for deployment on a platform of your choice. Ensure that the deployment environment is secure, and consider scalability to handle a potentially large user base.

Technology Stack:

- **LLM-based Model**: Utilize powerful language models like GPT-3 or similar for medical diagnosis and treatment recommendations.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Design a secure and user-friendly web-based interface using HTML, CSS, and JavaScript.

1.6. Productivity and Utility

1.6.1. Podcast summarize



Summarization is one of the popular applications of LLMs. Using LLMs we not only can summarize text but also audio and videos. Due to the large amount of content generated by people and AI nowadays this become an important productivity tool.

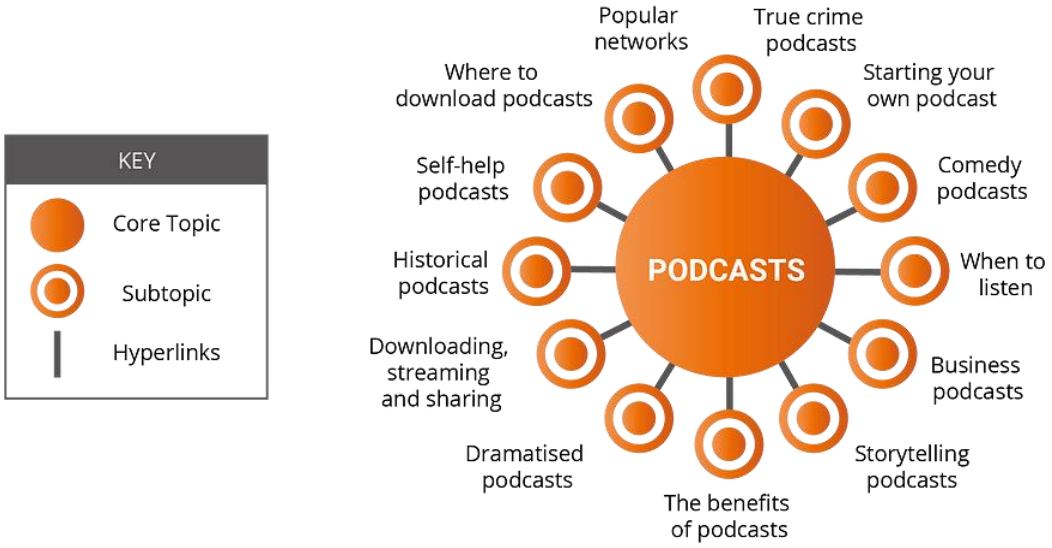
Project Guide:

- Download the video or podcast transcript and load it into documents
- Split long documents into chunks
- Summarize the transcript with an LLM
- Wrap it all in a user-friendly command line interface or even a web application.

Technology Stack:

- **LLM-based Model**: Utilize powerful language models like GPT-3 or similar for medical diagnosis and treatment recommendations.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Design a secure and user-friendly web-based interface using HTML, CSS, and JavaScript.

1.6.2 Topic clustering of social media posts and podcast episodes



Another application that you can use LLMs to increase your productivity is to build an application to cluster social media posts and podcast episodes into clusters.

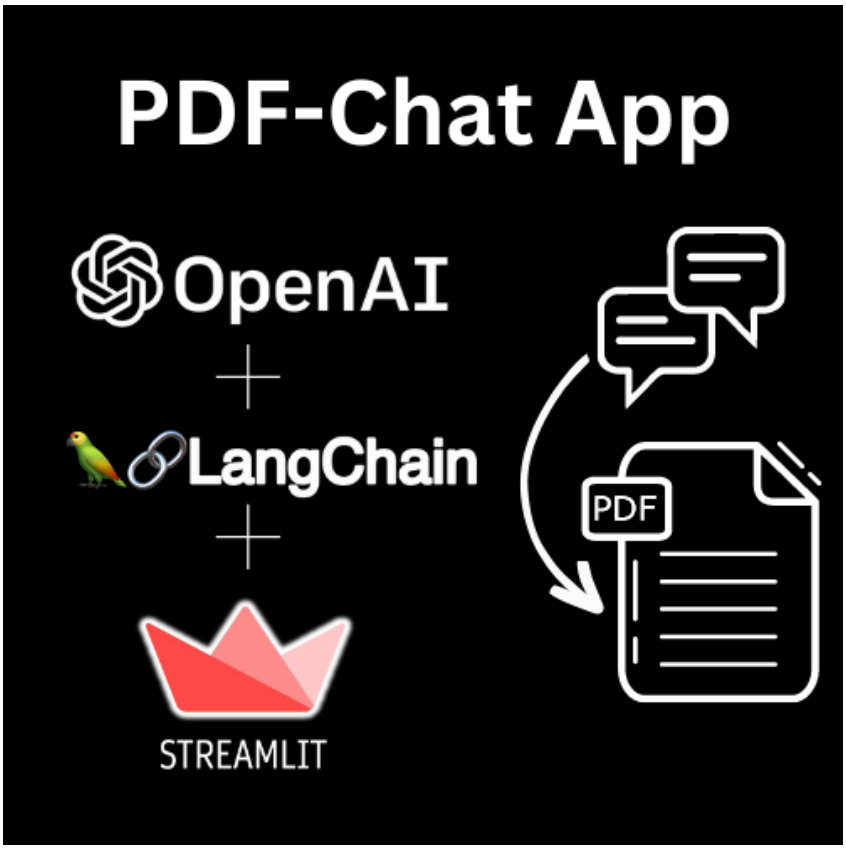
Project Guide:

- Load content into documents
- Split long documents into chunks
- Generate and store the embeddings from the documents with an embeddings model
- Use the embeddings as the input for a clustering algorithm

Technology Stack:

- **LLM-based Model**: Utilize powerful language models like GPT-3 or similar for medical diagnosis and treatment recommendations.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Design a secure and user-friendly web-based interface using HTML, CSS, and JavaScript.

1.6.3. Question answering over documents



The "**Question Answering over Documents**" project involves building a system that can read and comprehend textual documents, and then answer questions related to the content of those documents. This project is a practical application of natural language processing and information retrieval, with various potential use cases, from educational platforms to search engines.

Project Guide:

- Load content into documents
- Split long documents into chunks
- Generate and store the embeddings from the documents with an embeddings model
- Use the embeddings as the input for a clustering algorithm

Tech Stack:

- **Python**: For coding and data preprocessing.
- **Natural Language Processing Libraries**: Libraries like spaCy for text processing and analysis.

- **Deep Learning Frameworks**: TensorFlow or PyTorch for training and fine-tuning models.
- **Pre-trained Models**: Utilize models like BERT, T5, or other transformer-based architectures for document understanding.
- **Web Framework (optional)**: Flask or Django for creating a user-friendly web application.
- **Frontend Technologies (optional):** HTML, CSS, and JavaScript for building an interactive user interface.

1.6.4. YouTube Questions and answering

Similar to the previous project you can build a question-and-answer application for YouTube videos. It is similar to the previous project instead of the fact that you will have to convert the video first into text.

Project Guide:

- Load content into documents
- Split long documents into chunks
- Generate and store the embeddings from the documents with an embeddings model
- Use the embeddings as the input for a clustering algorithm

Technology Stack:

- **LLM-based Model**: Utilize powerful language models like GPT-3 or similar for medical diagnosis and treatment recommendations.
- **Backend**: Python or Node.js for handling API requests and responses.
- **Frontend**: Design a secure and user-friendly web-based interface using HTML, CSS, and JavaScript.

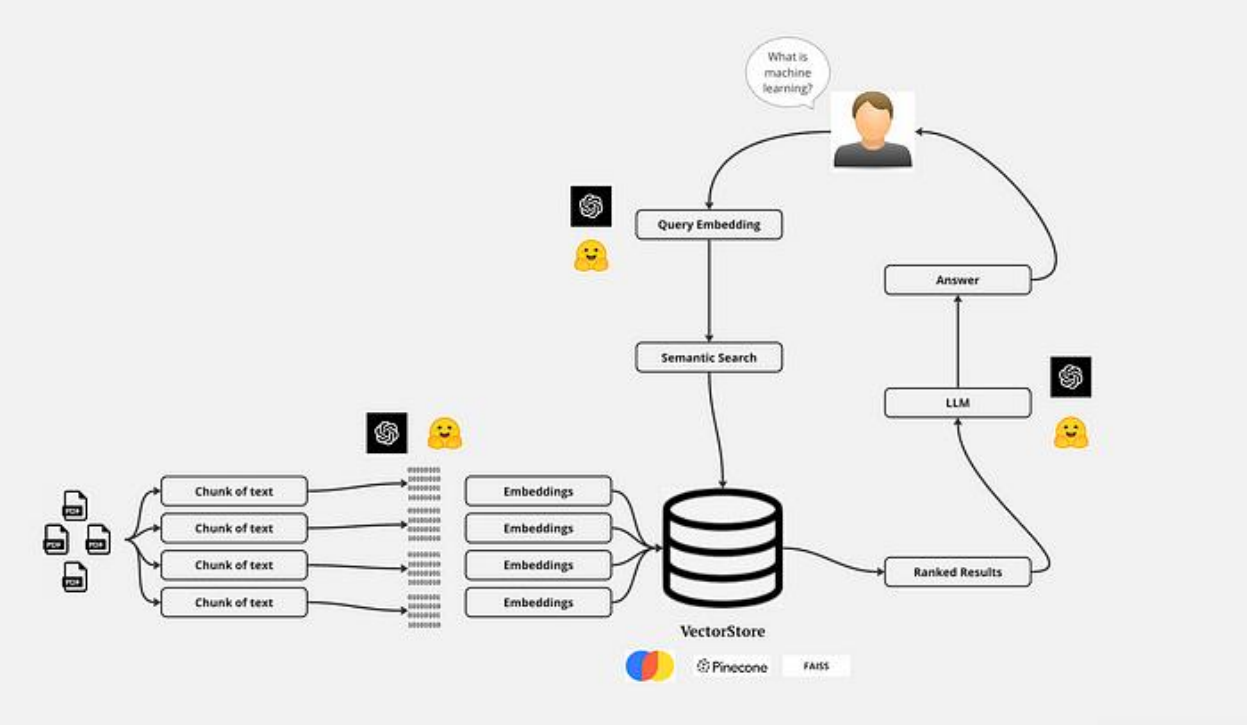
2. 10 Guided Large Language Models Projects to Build Your Portfolio

In the competitive world of AI and data science showcasing your skills through a portfolio is essential. And what better way to start than following a strong guided project?

This article is your gateway to elevating your professional profile. From crafting conversational chatbots to summarizing podcasts and even delving into pet care AI, this comprehensive guide covers it all.

Whether you're a budding developer looking to expand your horizons or a seasoned pro eager to stay at the forefront of AI innovation, this article is tailored just for you. Dive in and discover how to supercharge your portfolio with these 10 guided projects that not only demonstrate your expertise but also keep you ahead in the ever-evolving world of LLMs.

2.1. Building a Conversational Chatbot with Langchain and Large Language Models



In this **guided project**, we will walk through the process of building a conversational chatbot using Langchain to connect the user's data to Large Language Models such as GPT 3.5 and HuggingFace Instructor X1.

The chatbot will allow users to ask questions about multiple uploaded PDF documents and provide relevant answers using conversational retrieval techniques. We'll leverage the power of Artificial Intelligence building a Streamlit application that can be customized for the user's needs.

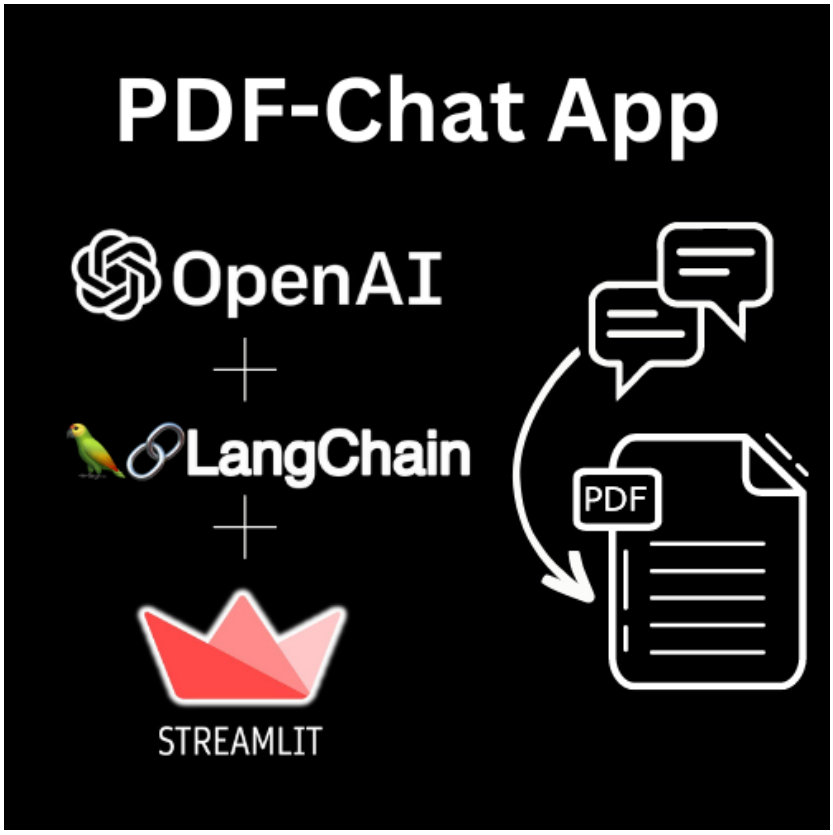
2.2. YouTube Script Writer Assistant Using OpenAI API & Streamlit

In this **guided project**, we will build a YouTube script-generating tool using LangChain and Streamlit. The application's interface is designed using Streamlit, which provides a user-friendly experience for the creators. The step-by-step guide provided in this article will help you set up your working environment, build the application, and run it effortlessly.

The application's prompt templates, and chat history storage capabilities make it easy for creators to customize the generated scripts to their specific needs. Moreover, the language model generates not only the script but also the video title, making the tool a comprehensive solution for YouTube creators.

With LangChain and Streamlit's automated script-writing application, you can take your YouTube content to the next level, saving time and effort in the process. Follow the guide provided in this article to set up the application and create engaging and informative scripts for your videos.

2.3. Chat with your Document using OpenAI API & Streamlit



Throughout this **guided project**, we will walk you through the step-by-step process of building the PDF-Chat app. We will start by setting up the development environment, ensuring that you have all the necessary tools and dependencies installed. Then, we will dive into designing the user interface using Streamlit, a powerful Python library for creating interactive web applications.

Next, we will explore the integration of LangChain and the OpenAI API to handle PDF file processing and generate meaningful answers to user queries. You will learn how to set up the OpenAI API, load and process PDF files, and effectively handle prompts to generate accurate and context-aware responses.

Finally, we will bring everything together and showcase a live demo of the PDF-Chat app. You will see how users can effortlessly engage in conversations with their PDF documents, making information retrieval faster and more interactive than ever before.

2.4. Building A Falcon-40B Instruct Chat Web Application using HuggingFace & Gardio

In this **guided project**, you will build a chat application based on the Falcon-40B Instruct open-source LLM using HuggingFace and Gardio.

2.5. Building GPT Banker Using LLaMA 2 70B

In this **guided project** you will build a banker chatbot by fine-tuning LLaMA 2 70B open source LLM.

2.6. Finetune Llama 2 On Your Local Machine Using HuggingFace Autotrain

In this **guided project**, you will be shown the easiest way to fine-tune the **Llama-2 model** on your own data using the auto-train-advanced package from HuggingFace.

2.7. Building a Lex Fridman Podcast Summarization App with Whisper Jax, Azure OpenAI, and Langchain

In this **guided project**, we demonstrate how to create the Lex Fridman Podcast Summarization App using Whisper Jax for rapid transcription, Langchain for dynamic prompt templates, and the Azure OpenAI GPT-3.5 Turbo Model as a robust language powerhouse.

By combining these cutting-edge technologies, you will learn how to automatically transcribe and summarize podcast episodes accurately and efficiently, elevating the way we interact with audio content through AI-driven summarization. You can then post the content on LinkedIn as the prompt is set for LinkedIn.

2.8. Creating a Veterinary Chatbot using Llama 2: Harnessing Gen AI for Pet Care

In this **guided project**, you will be guided through the fascinating process of building your very own chatbot for veterinary doctors. We'll delve into the powerful world of Generative AI, utilizing cutting-edge open-source tools and the LLM model to make it happen.

- **Data Sources:** We'll tap into a comprehensive encyclopedia of dog pet care, equipping our chatbot with a wealth of knowledge to assist pet owners. We'll also explore vet case studies, enriching our bot with real-life scenarios for informed decision-making.
- **Key Ingredients:** We'll employ the Multilingual E5 Large Embeddings Model to create embeddings for our documents. The FAISS vector store will help us efficiently manage our data. For the brainpower behind our chatbot, we've harnessed the remarkable Llama 2 by Meta AI, a formidable large language model.
- **Tools of the Trade:** The backend will be powered by FastAPI, ensuring seamless communication between the user and the chatbot. Langchain will serve as the trusty framework for prompts, chains, and more.

2.9. Deploy Llama 2 on AWS SageMaker using DLC (Deep Learning Containers)

In this **guided project**, you will learn how to effortlessly deploy the Llama2 large language model on AWS SageMaker using Deep Learning Containers (DLC). We'll walk through each step, from accessing pre-built DLC images to configuring SageMaker for Llama2 deployment, designed to make the process smooth and understandable, whether you're new to Generative AI or experienced in the field.

2.10. Article/Blog Generation App using Llama2, Langchain, and Pexels

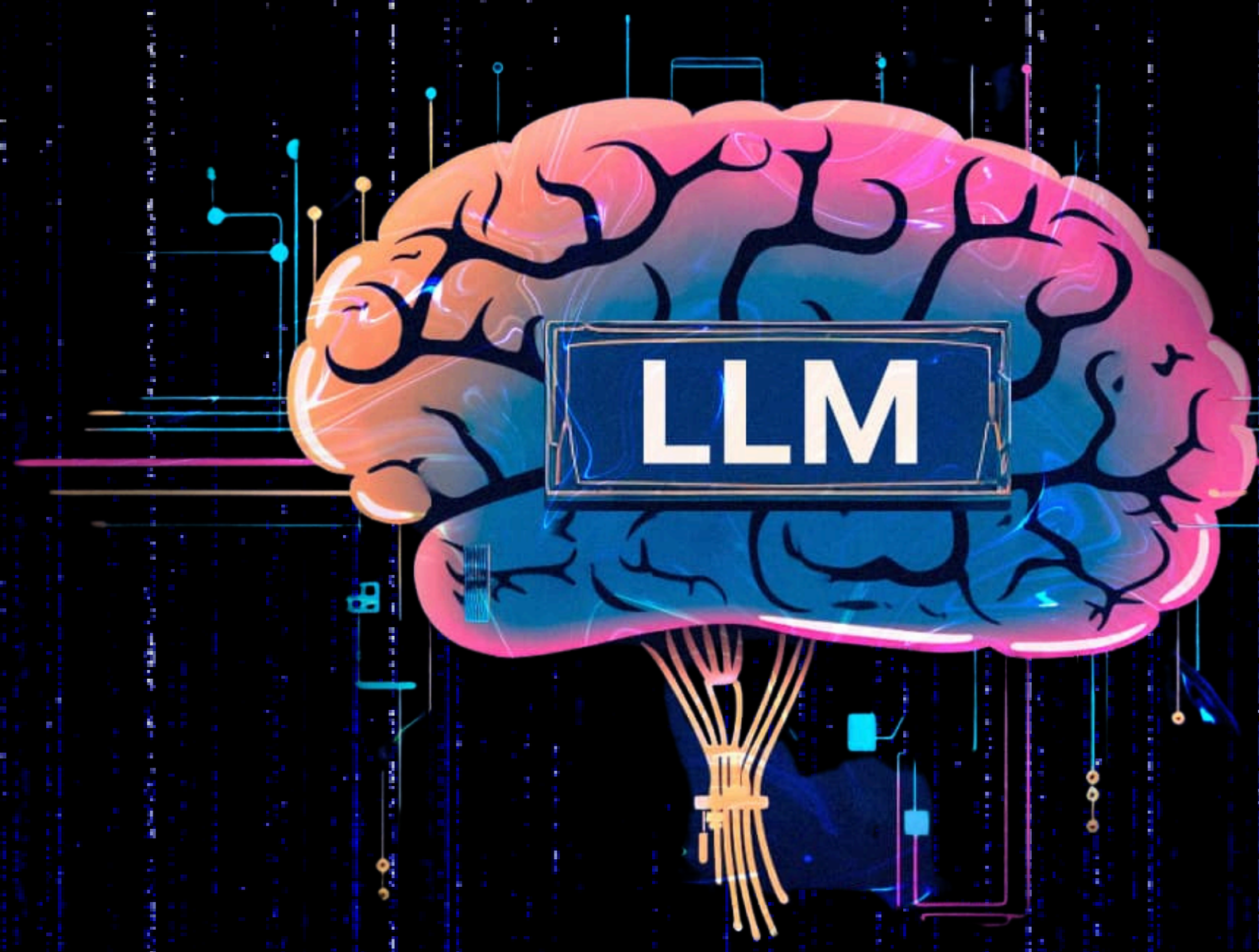
In this **tutorial**, you will be guided through the process of developing an LLM-powered Article/Blog Generation App, which leverages the advanced capabilities of Llama2, a cutting-edge open-source large language model.

You'll discover how to effortlessly generate high-quality content while integrating stunning images from Pexels via an API. The user-friendly Streamlit app interface ensures a seamless experience, and you can even export your work as a DOCX file for further customization. Say goodbye to writer's block and hello to a new era of content creation!

Afterword

Thanks for purchasing and reading my book!

- If you have any questions, feedback, or praise, you can reach me on this **email**
- I would be happy if you could connect with me on **LinkedIn**
- If you liked my writings, make sure to follow me on **Medium**
- Subscribe to my newsletter to never miss any of my **writings**



Youssef is a Senior Data Scientist and machine learning researcher. Youssef is passionate about data and believes in AI's power to improve people's lives. Youssef is on a mission to transfer his passion to others and guide them into this vast field through writing, teaching, and mentoring.