The LLM Development Stack

Rob Barton and Jerome Henry

Pearson

Course Overview

- Lesson 1: LLM Development Fundamentals
- Lesson 2: Building Applications with LLMs
- Lesson 3: Agentic AI Development Tools
- Lesson 4: Agentic AI Protocols
- Lesson 5: Cloud-Based LLM Platforms
- Lesson 6: LLM Benchmark and Optimization Tools

The LLM Development Stack

Models and Transformers

- Llama
- OpenAI, Claude, etc
- DeepSeek

Agent Orchestration Platforms

- LangGraph
- AutoGen
- CrewAI

Development Environments

- Colab
- VS Code
- Cursor / Windsurf

Agentic Protocols

- APIs
- MCP
- A2A

Local LLM Management Platforms

- Ollama
- LM Studio
- Jan

Cloud Development Platforms

- AWS Bedrock
- Azure AI Studio
- Google Vertex AI

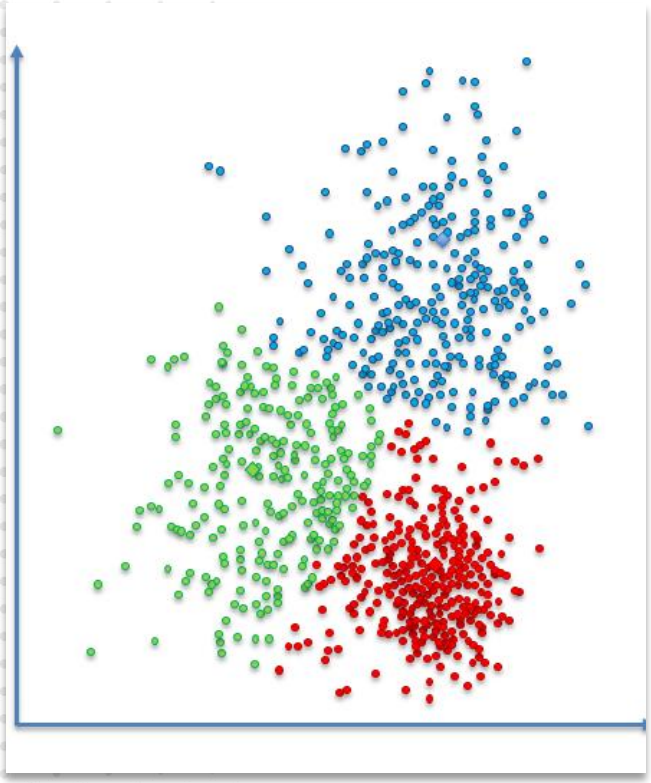
RAG & Agentic Frameworks

- LlamaIndex
- LangGraph
- Haystack

Monitoring and Observability

- LangSmith
- Weights & Biases
- Helicone

Mature | Strong Contenders | Emerging



Lesson 1:
Foundational Tools

Objectives

- Choosing a model: cloud vs local, open vs closed
- Hugging Face for model selection
- Google Colab to run development and experiments
- Tracking progress: Wandb

Models, Transformers, and Development Environments

Models and Transformers

- Llama
- OpenAI, Claude, etc
- DeepSeek

Development Environments

- Colab
- VS Code
- Cursor / Windsurf

Local LLM Management Platforms

- Ollama
- LM Studio
- Jan

RAG & Agentic Frameworks

- LlamaIndex
- LangGraph
- Haystack

Agent Orchestration Platforms

- LangGraph
- AutoGen
- CrewAI

Agentic Protocols

- APIs
- MCP
- A2A

Cloud Development Platforms

- AWS Bedrock
- Azure AI Studio
- Google Vertex AI

Monitoring and Observability

- LangSmith
- Weights & Biases
- Helicone

Mature | Strong Contenders | Emerging

LLM Development Phases

LLM development creates an LLM from scratch, or modifies an LLM, it has 3 components:

Step 1: Model and data selection
Train from scratch? Augment/distill? Which model base? (compromise between task (specialization), size and precision). Which data to train from?

Step 2: Environments and tools selection
Cloud? On prem? Link to data lake? Link between agents?

Step 3: Tracking and Testing
Log experiments to remember outcomes? Evaluate model performances? Optimize?

Private-Source vs. Open-Source LLMs

	Closed-Source	Open-Source
Examples	ChatGPT, Google Gemini, Claude	BERT, LLaMA, Mistral, etc.
Fine Tuning Capability	Limited to none (RAG only)	Depends on the LLM, but generally possible
Embodiment	Public-Cloud (aaS model)	Private cloud, on-prem
Security	Difficult to control	Easy (you control the model)
Typical Model Size	Huge	Smaller, but can be more focused
Price	aaS pricing model	Generally capital cost only

Business Lens: Closed vs. Open-Source Models

1. **Cost:** OpenAI claims GPT-4 cost them over $100 million to develop. This cost means the models will likely stay under the control of the organization who developed it (forcing users to send data to the service either via the web interface or API)
2. **Privacy and Security:** Commercially available LLMs are something of a “black box” for users. The users have no control over how they were trained, bias, etc.
3. **Training Gap:** Private-source LLMs are hard to keep up to date, and can typically be 1-2 years old
4. **Fine Tuning:** Open-Source models can generally be fine-tuned, allowing them to be smaller for a specific purpose

Finding Open-Source Models – Hugging Face

- What is Hugging Face?
- An AI company which has become the world's largest repository of open-source transformers

Hugging Face Open Source LLM Repository

huggingface.co/models

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs | Solutions | Pricing

Tasks | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name

Multimodal

- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering

Models 626,177 | Filter by name | new Full-text search | Sort: Trending

- meta-llama/Meta-Llama-3-8B — Text Generation • Updated 5 days ago • 505k • 2.75k
- apple/OpenELM — Updated 5 days ago • 774
- microsoft/Phi-3-mini-4k-instruct — Text Generation • Updated about 5 hours ago • 88.9k • 391
- shenzhi-wang/Llama3-8B-Chinese-Chat — Text Generation • Updated 30 minutes ago • 3.57k • 265
- microsoft/Phi-3-mini-4k-instruct-gguf — Text Generation • Updated 2 days ago • 51.7k • 234
- apple/OpenELM-3B-Instruct — Text Generation • Updated 5 days ago • 3.4k • 200
- gradientai/Llama-3-8B-Instruct-262k — Text Generation • Updated 3 days ago • 6.93k • 148
- microsoft/Phi-3-mini-128k-instruct — Text Generation • Updated 2 days ago • 103k • 909
- meta-llama/Meta-Llama-3-8B-Instruct — Text Generation • Updated 5 days ago • 703k • 1.54k
- ByteDance/Hyper-SD — Text-to-Image • Updated about 16 hours ago • 34.5k • 272
- Snowflake/snowflake-arctic-instruct — Text Generation • Updated 2 days ago • 1.87k • 261
- meta-llama/Meta-Llama-3-70B-Instruct — Text Generation • Updated 5 days ago • 88.9k • 788
- meta-llama/Meta-Llama-3-70B — Text Generation • Updated 5 days ago • 296k • 510
- cognitivecomputations/dolphin-2.9-llama3-8b — Text Generation • Updated 7 days ago • 14.7k • 231

Model Cards

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs | Solutions | Pricing

cardiffnlp / **twitter-roberta-base-sentiment-latest** | like 399

Text Classification | Transformers | PyTorch | TensorFlow | tweet_eval | English | roberta | Inference Endpoints | arxiv:2202.03829

Model card | Files and versions | Community 25 | Train | Deploy | Use in Transformers

Edit model card

Twitter-roBERTa-base for Sentiment Analysis - UPDATED (2022)

This is a RoBERTa-base model trained on ~124M tweets from January 2018 to December 2021, and finetuned for sentiment analysis with the TweetEval benchmark. The original Twitter-based RoBERTa model can be found here and the original reference paper is TweetEval. This model is suitable for English.

- Reference Paper: TimeLMs paper.
- Git Repo: TimeLMs official repository.

Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

This sentiment analysis model has been integrated into TweetNLP. You can access the demo here.

Example Pipeline

Downloads last month
3,662,004

Inference API

Text Classification | Examples

Covid cases are increasing fast!

Compute

Computation time on cpu: cached

negative	0.724
neutral	0.229
positive	0.048

JSON Output | Maximize

Transformers Library

Open-source Python library that provides:

- **pretrained models (**(e.g., BERT, GPT, T5, LLaMA, etc.)
- Tools for **tokenization**, **model training**, **fine-tuning**, and **inference**
- APIs to download and manage models from the Hugging Face **Model Hub**

Not only for Natural Language Processing (NLP) but also for related tasks such as vision and audio processing using **transformer-based architectures**

LLM Development Components

LLM development creates an LLM from scratch, or modifies an LLM, it has 3 components:

Step 1: Model and data selection
Each model is a compromise between task (specialization), size and precision.

Step 2: Environments
Use commands to point the model(s) to the right data and generate the training

Step 3: Experiment Tracking
Log your experiments to remember outcomes.

Google Colab: Code + Cloud + GPUs

- LLM development is used by researchers, startups, and hobbyists
- Fast iteration is key when working with LLMs.
- Tools must be accessible, flexible, and well-integrated
- Google Colab:
  - free (or “can be free”)
  - integrated with Google drive (to store data) and/or Gituhub
  - features a simple Jupyter interface

Google Colab Alternatives

- Your own Jupyter server (with GPU local or on another machine)
- Rented virtual environments (e.g. RunPod)
- Cloud-based environments e.g., AWS, Vertex (more advanced)

LLM Development Components

LLM development creates an LLM from scratch, or modifies an LLM, it has 3 components:

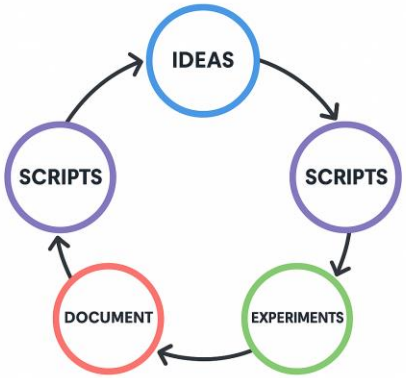
Step 1: Model and data selection
Each model is a compromise between task (specialization), size and precision.

Step 2: Environments
Use commands to point the model(s) to the right data and generate the training

Step 3: Experiment Tracking
Log your experiments to remember outcomes.

Keeping Track

- LLM development is an iterative process
- It is difficult to remember what (and why) you did several weeks/months ago
- Hard if you work alone, a nightmare when working in teams (“final2.ipynb” syndrome)
- A tracking system allows you reduce investigative wasted time



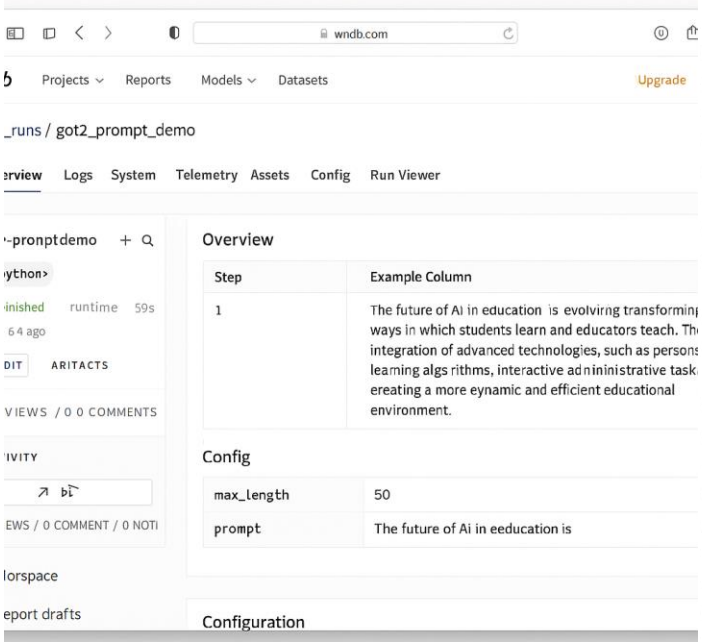
Tracking Solutions

Non exhaustive list of main players

Feature	**W&B** (Beginner)	**ClearML** (Mid-level)	**Aim** (Open source, minimal UI)
UI & Dashboards	Polished, cloud UI	Very detailed	Lightweight & modern
Ease of use	Easiest (Colab-friendly)	Medium (more config)	Easy CLI, but less guidance
Offline use	✅ (via wandb offline)	✅	✅
Collab features	Share links, teams	Team support	Git-style project mgmt
Best for	Prototyping & sharing	Larger pipelines	Minimalist open-source workflows

Tracking Solutions

- Wandb (Weights and Biases, W&B)
  - Free and beginner-friendly (integrates well with Colab & Jupyter)
- Track:
  - prompts
  - parameters (e.g., temperature, model)
  - generated outputs
  - Visualize results across multiple runs
  - Share, resume, or compare experiments with a single link



Google Colab with Hugging Face

Pearson

DEMO

- Colab easily integrates with Hugging Face APIs, so you can call a Hugging Face model from Colab… and check the experiments from Wandb