

Introduction to Large Language Models

Huihuo Zheng

Argonne Leadership Computing Facility, Argonne National Laboratory

July 31, 2025

Some contents were generated with the help of ChatGPT / Gemini.

Outline



History
of language
model

How LLMs
works



Transformer
Architecture

Training pipeline

- Dataset
- Training
- Evaluation



Retrieval
Augmented
Generation



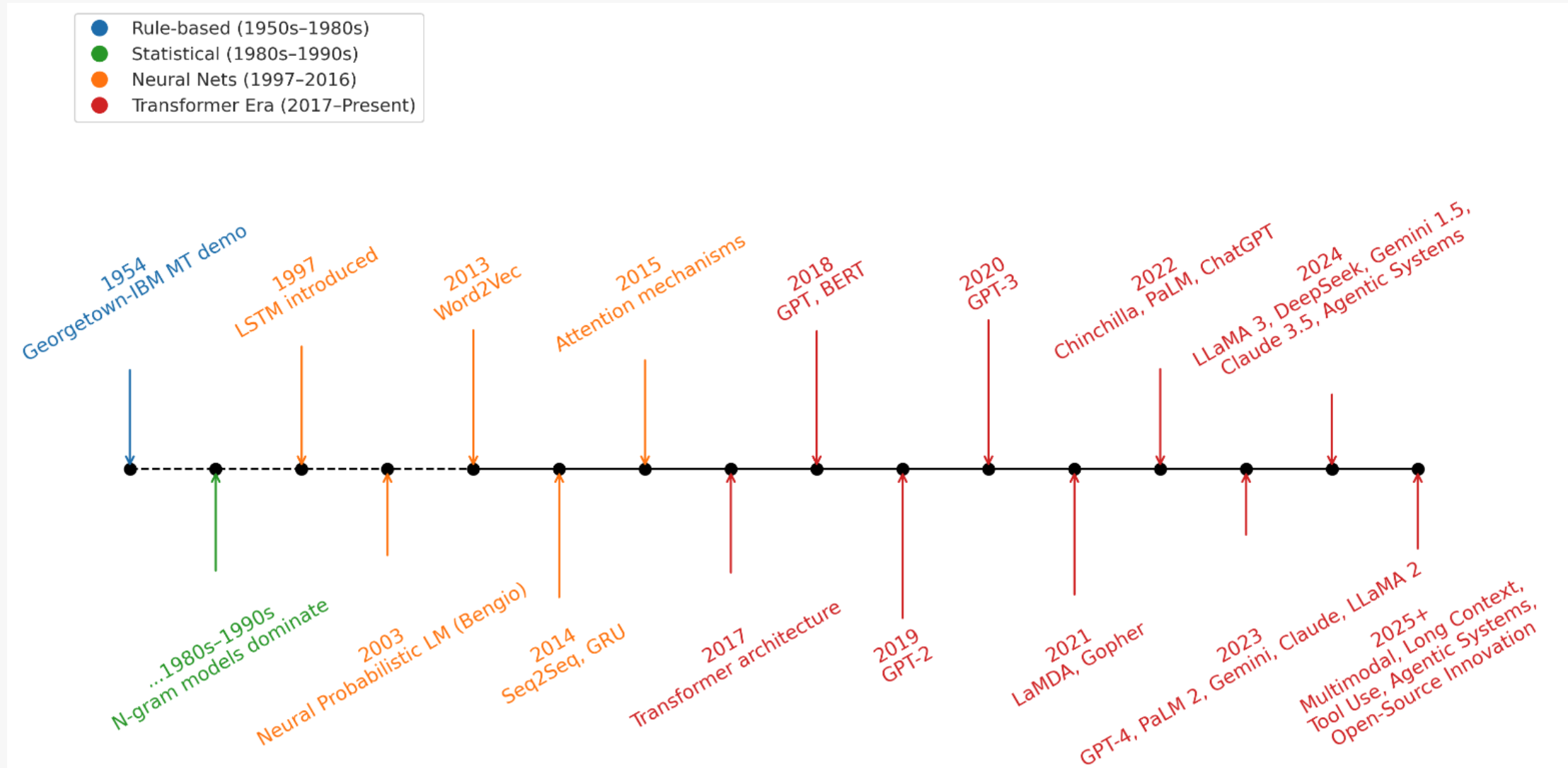
Mult-agent
systems

What Is a large language model?

A **large language model (LLM)** is a language model trained with self-supervised learning on a vast amount of text, designed for natural language processing tasks, especially language generation.



Evolution of language models



The Power of LLMs

Text understanding

- Comprehension
- Classification
- Sentiment analysis

Specialized tasks

- Retrieval augmented QA
- Scientific QA
- Code generation
- Data analysis

Agentic tasks

- Tool Use
- Function calling
- Autonomous Agents

Text Generation

- Paraphrasing
- Translation
- Summarization
- Editing

Reasoning & Planning

- Commonsense reasoning
- Solving Math problems
- Chain of thoughts

The Power of LLMs – GenAI?

Large Language Models for Batteries

Wenhua Zuo¹, Huihuo Zheng², Tanjin He³, Venkatram Vishwanath², Maria K. Y. Chan³, Rick L. Stevens², Gui-Liang Xu^{1,8*}, Khalil Amine^{1,8*}

¹Chemical Sciences and Engineering Division, Argonne National Laboratory, Lemont, IL 60439, USA

²Leadership Computing Facility, Argonne National Laboratory, Lemont, IL 60439, USA

³Center for Nanoscale Materials, Argonne National Laboratory, Lemont, IL 60439, USA

*Corresponding authors. Email: xug@anl.gov, amine@anl.gov

⁸Lead contact

*Correspondence: xug@anl.gov

*Correspondence: amine@anl.gov

Accepted to Joule

The efficacy of language models in tackling a wide range of problems lies in the intimate connection between language and knowledge: Language is not only a communicative medium but also a repository, where collective human knowledge and logical inference are deposited. As a result, LLMs, by accurately capturing and reproducing these patterns implicitly conveyed in language, have a profound capacity to handle various tasks and challenges. *[Quote from Large Language Models for Batteries]*

Models (LLMs) are advanced artificial intelligence (AI) models that use various tools to address diverse tasks. Despite their increasing use in academia and industry, the potential of LLMs is still underutilized.

John 1:1, 3 In the beginning was the Word, and the Word was with God, and the Word was God. ... All things came into being through Him; and apart from Him not one thing came into being which has come into being.

IN
THE
BEGINNING
WAS
THE
WORD

Fundamental principle behind LLMs

Two ways of learning English

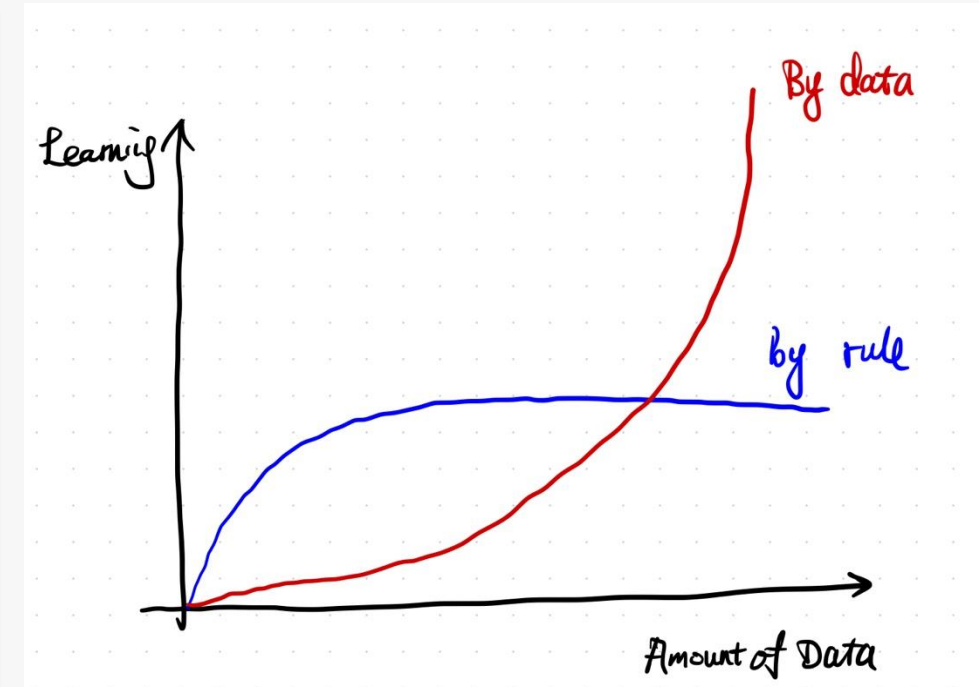
- Through learning grammatic rules (deterministic)
- Through hearing a lot of speaking (data driven)



Follow grammar rules
Past tense, present tense



Follow patterns
I went to school yesterday
He played basketball last weekend.
I will go to UK to see my cousin tomorrow
...



Power of data driven way

Fundamental principle behind LLMs


LLMs function by pre-training on extensive datasets to learn language patterns

- I got a big present; I am very **happy**
- I will go to UK this weekend; I am very **excited**.
- My mom brought me new shoes; I am very **happy**.
- I got a new car; I am very **happy**.
- I failed my calculus class; it makes me very **sad**.
- My iPad Pro got water damage; I am very **upset**. Should I get a new one?
- ...




I got an A+;
I am very ____


- happy
- pleased
- proud
- sad
- ...

 **You**
What should I fill in the blank?


I got an A+; I am very ____

 **ChatGPT**
I got an A+; I am very proud.


 **You**
What should I fill in the blank?

I got an A+; I am very ____


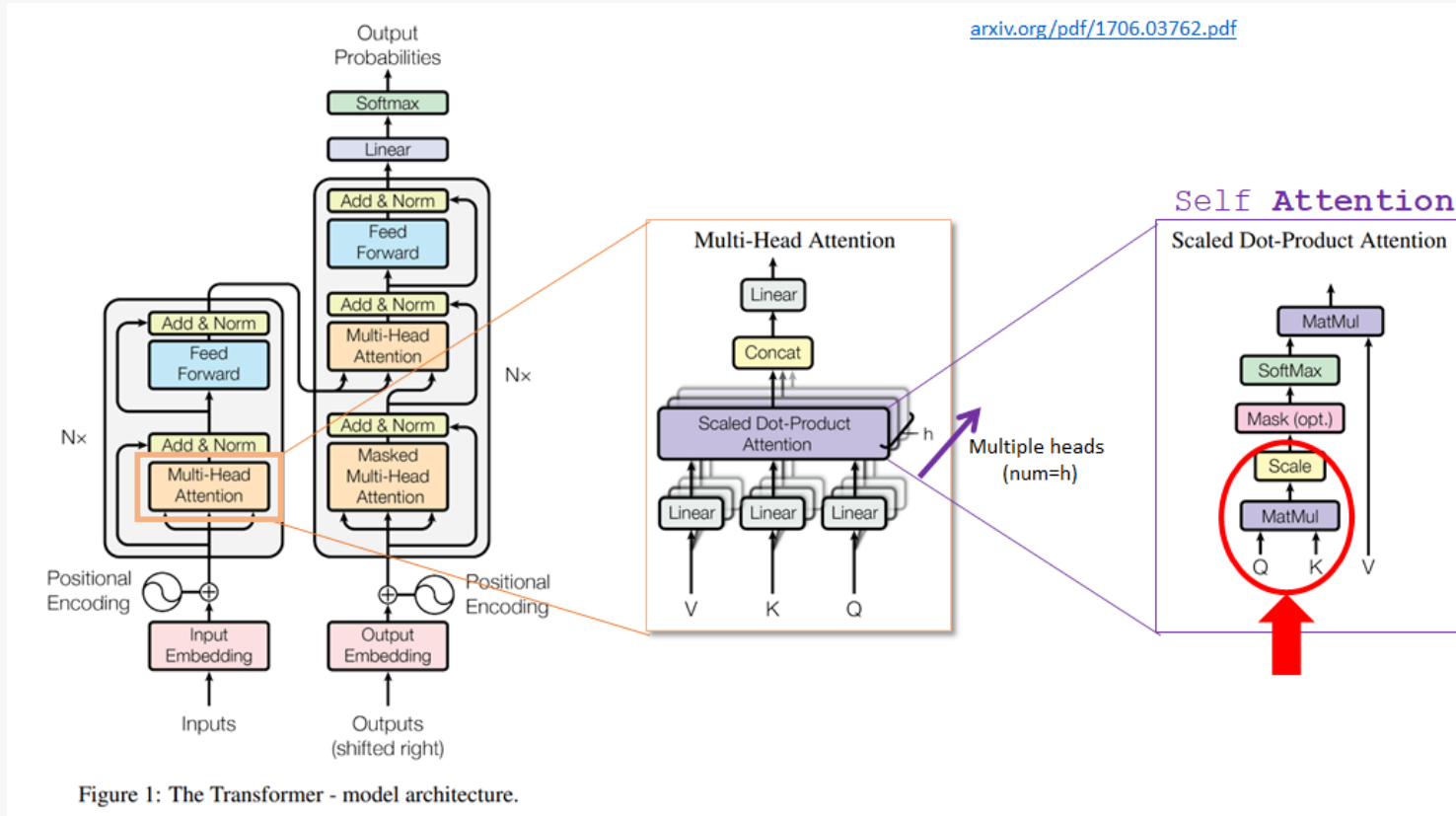
 **ChatGPT**
I got an A+; I am very pleased.
< 3 / 3 >

 **You**
What should I fill in the blank?

I got an A+; I am very ____

 **ChatGPT**
I got an A+; I am very ecstatic.

The Transformer Architecture: A Paradigm Shift

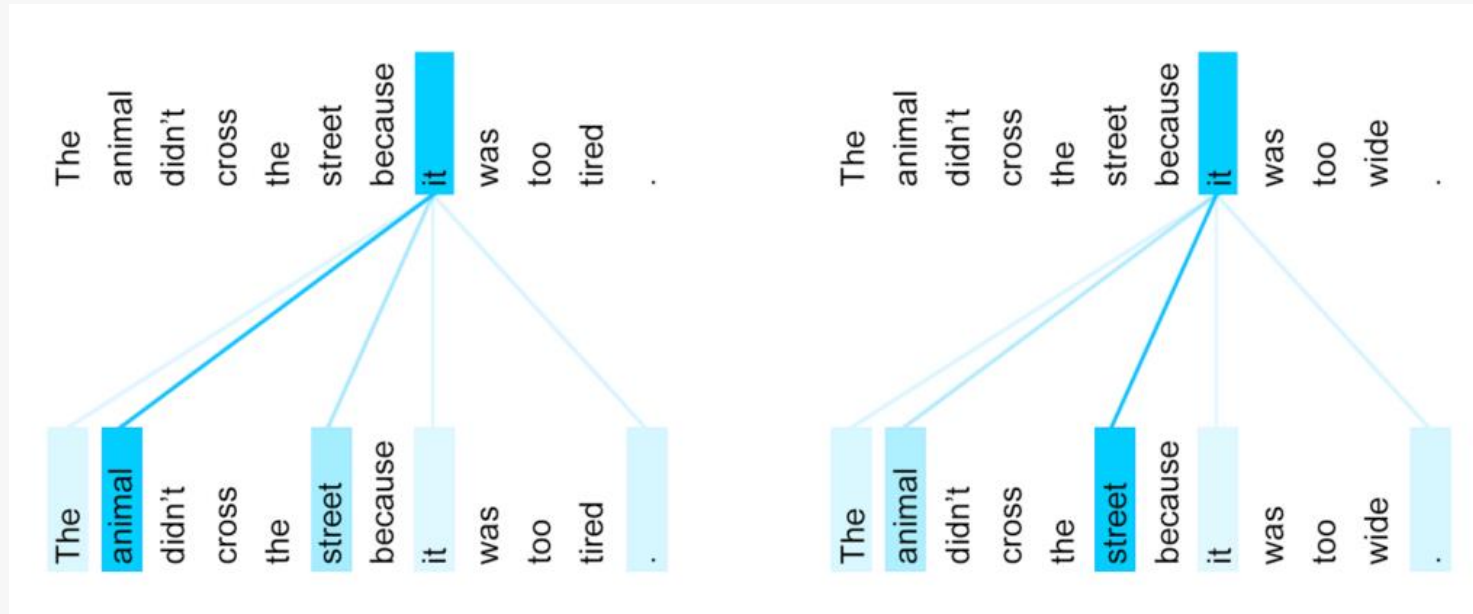


Main breakthroughs

- Captures global context using **self-attention**
- All tokens in a sequence can be processed **in parallel**

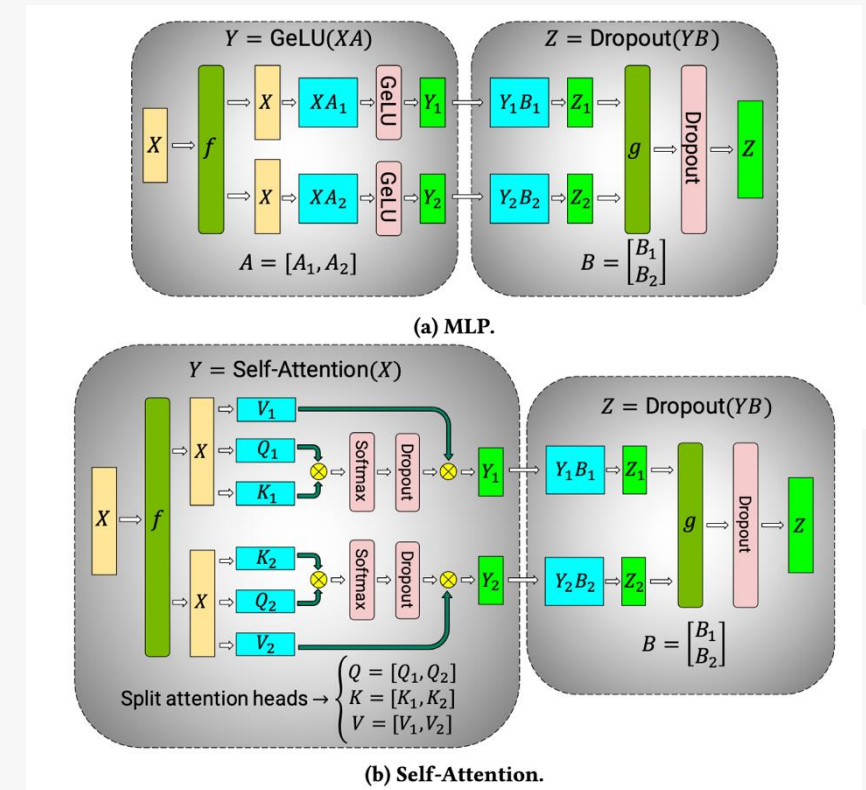
Attention Is All You Need, [Ashish Vaswani](#), et al, arXiv:1706.03762

Self-attention, multihead attention



Self-attention allows a model to weigh the importance of **each word** in a sequence relative to the **others**, enabling it to capture contextual relationships.

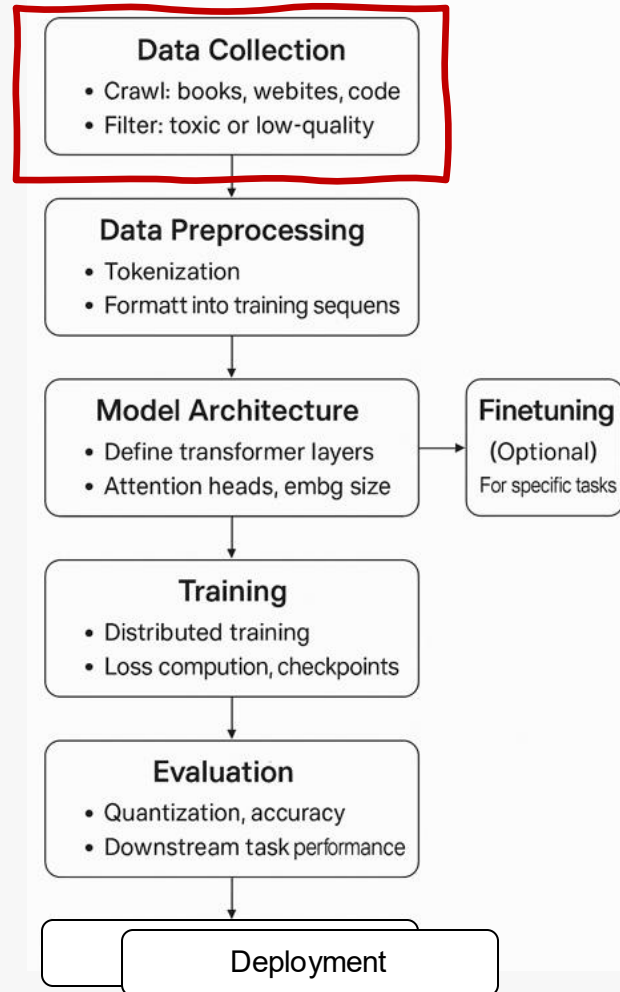
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Main operations involved

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Deepak Narayanan et al, arXiv:2104.04473

Training pipeline – Data Collection



Blending of
different corpora

Web crawls: *Common Crawl, CCNet, C4* (used by T5, PaLM)

Books: *Books1 & Books2* (used by GPTs), *The Pile: Books3*

Wikipedia: High-quality curated text, multilingual

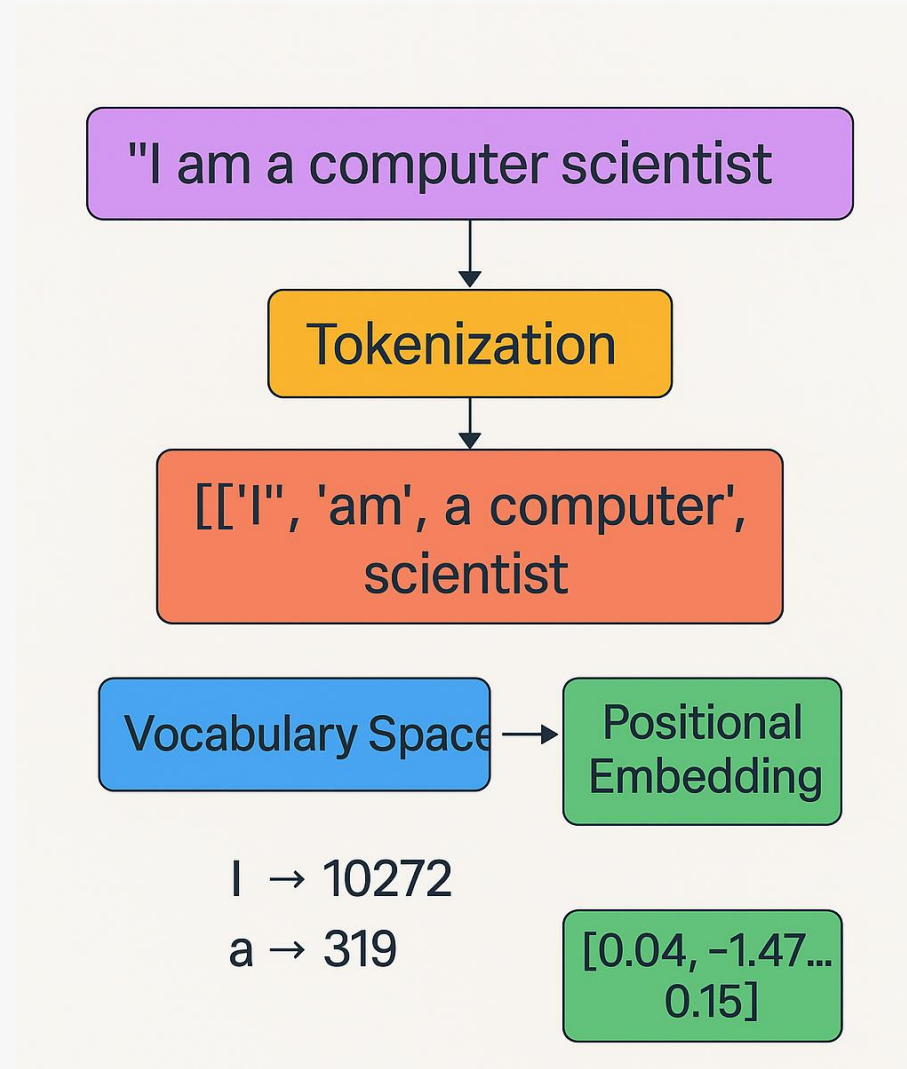
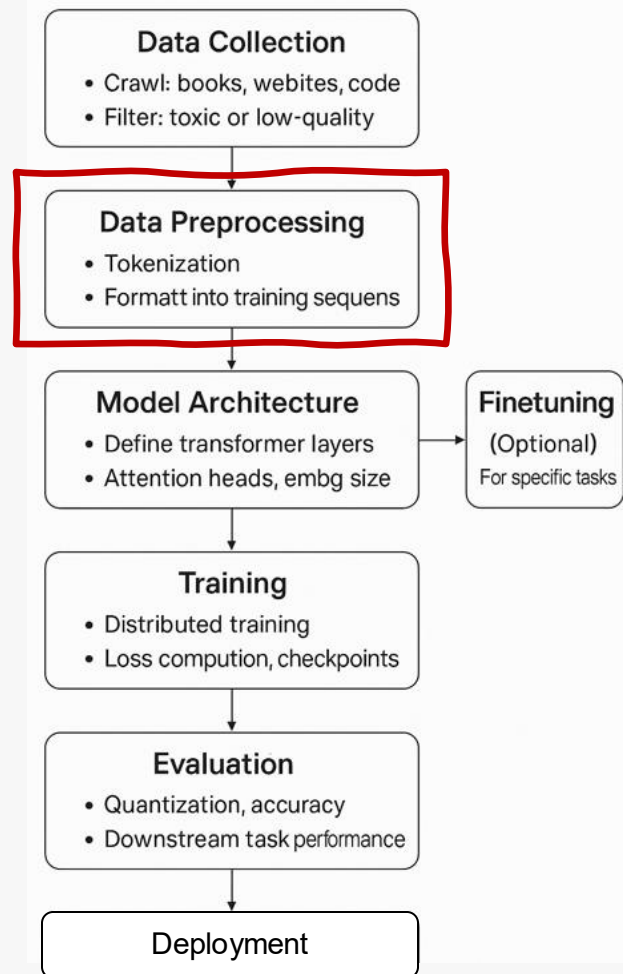
Code: *GitHub* data via *The Stack*, *CodeSearchNet*

Scientific papers: *arXiv, PubMed, S2ORC*

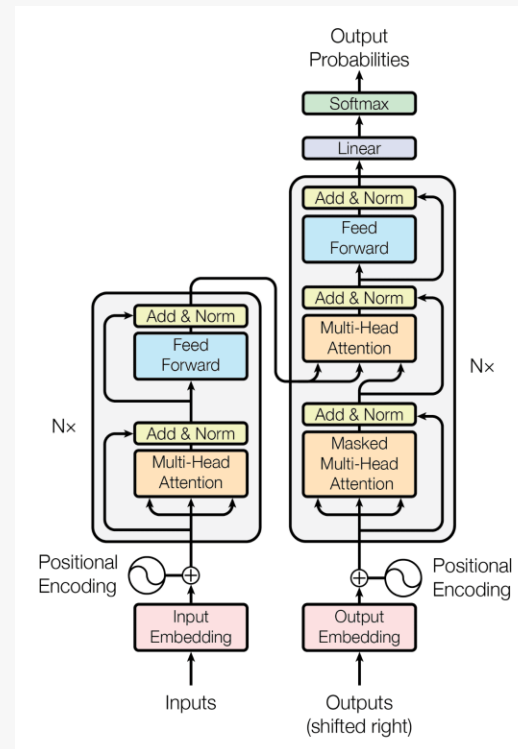
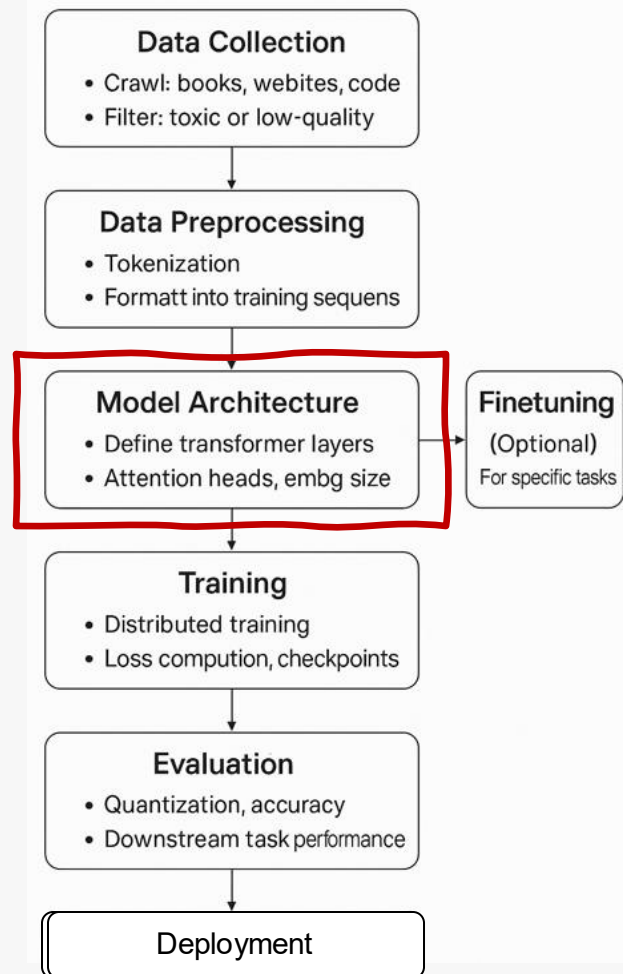
Dialog & QA: *OpenWebText, Reddit, StackExchange, Quora*

Multilingual corpora: *CCMatrix, OPUS, mC4*

Training pipeline – Data Preprocessing



Training pipeline – Model Architecture



Encoder-only (BERT)

- Pre-training: Masked Language Modeling (MLM)
- Great for classification tasks, but hard to do generation

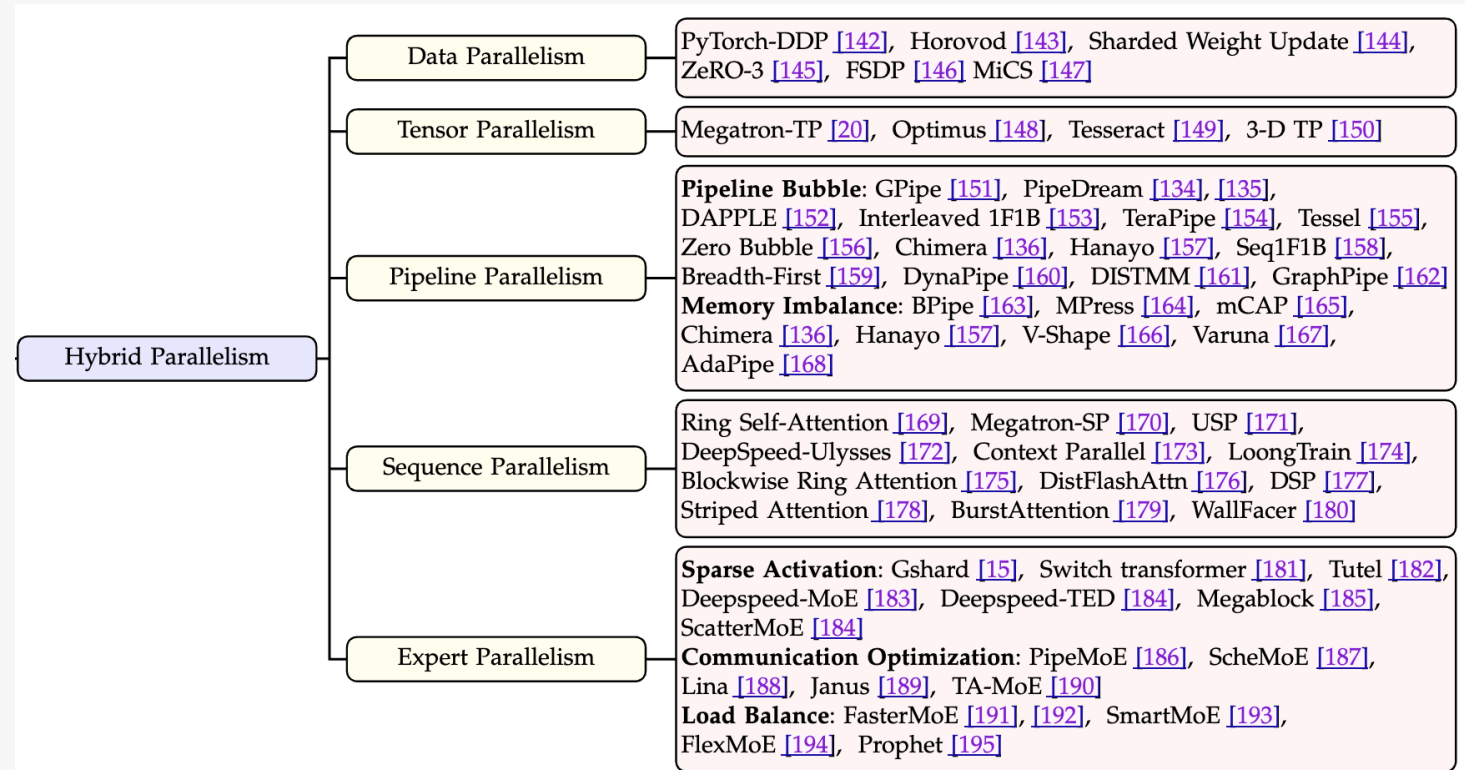
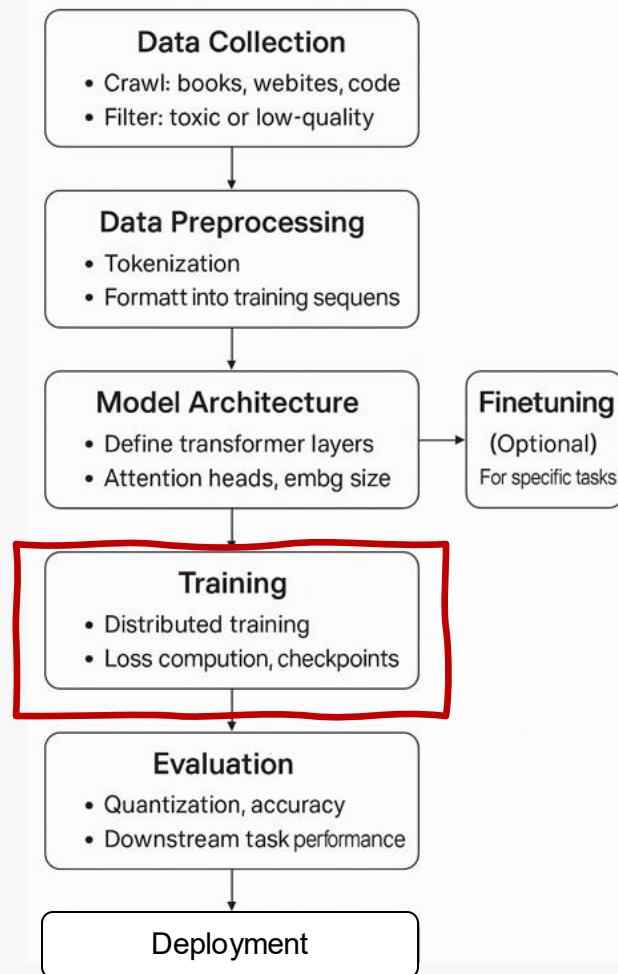
Decoder-only (GPT)

- Pre-training: Auto-regressive Language Modeling
- Stable training, faster convergence
- Better generalization after pre-training

Encoder-decoder (T0/T5)

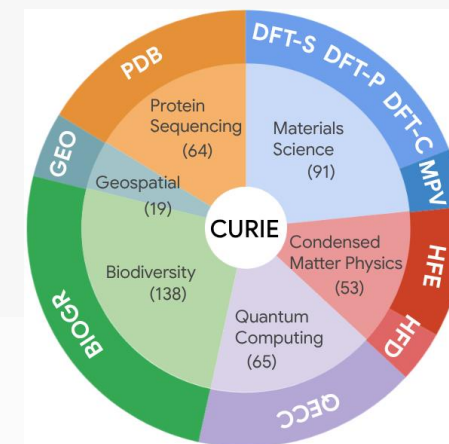
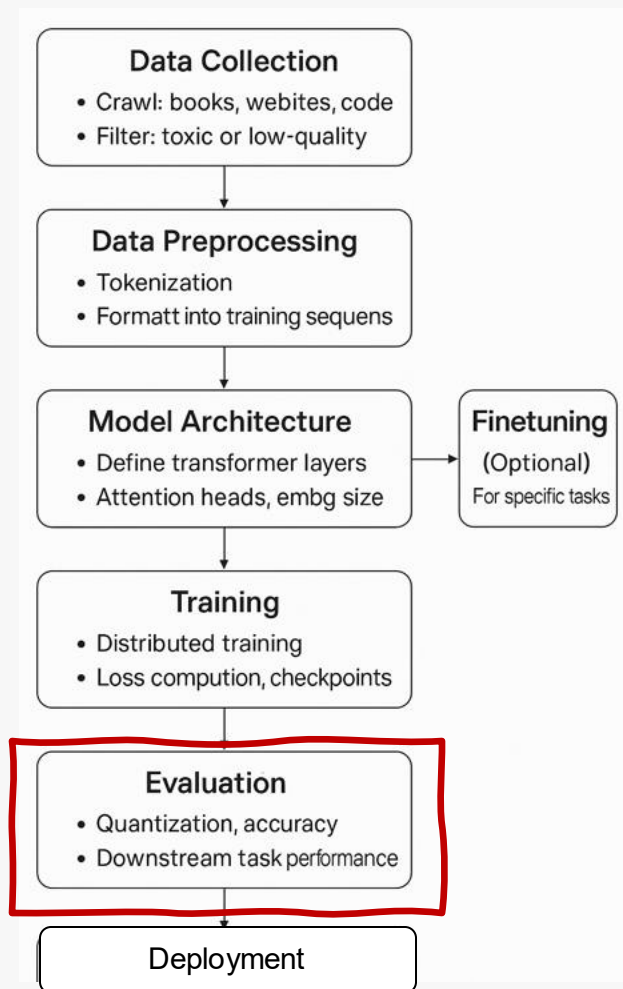
- Pre-training: Masked Span Prediction
- Good for tasks like MT, summarization

Training pipeline – Distributed Training



Efficient Training of Large Language Models on Distributed Infrastructures: A Survey, Jiangfei Duan et al, arXiv:2407.20018

Training pipeline



Evaluation Benchmarks

General Language Understanding

GLUE GLUE
SuperGLUE

Math & STEM

HumanEval
MBPP
CodeEval
BIG-bench Hard

Knowledge & Reasoning

MMLU
ARC **OpenBookQA**
BoolQ

Coding & Alignment

MT-Bench
AlpacaEval
Helpful, Honest, Harmless (HHH)

Dialogue & Alignment

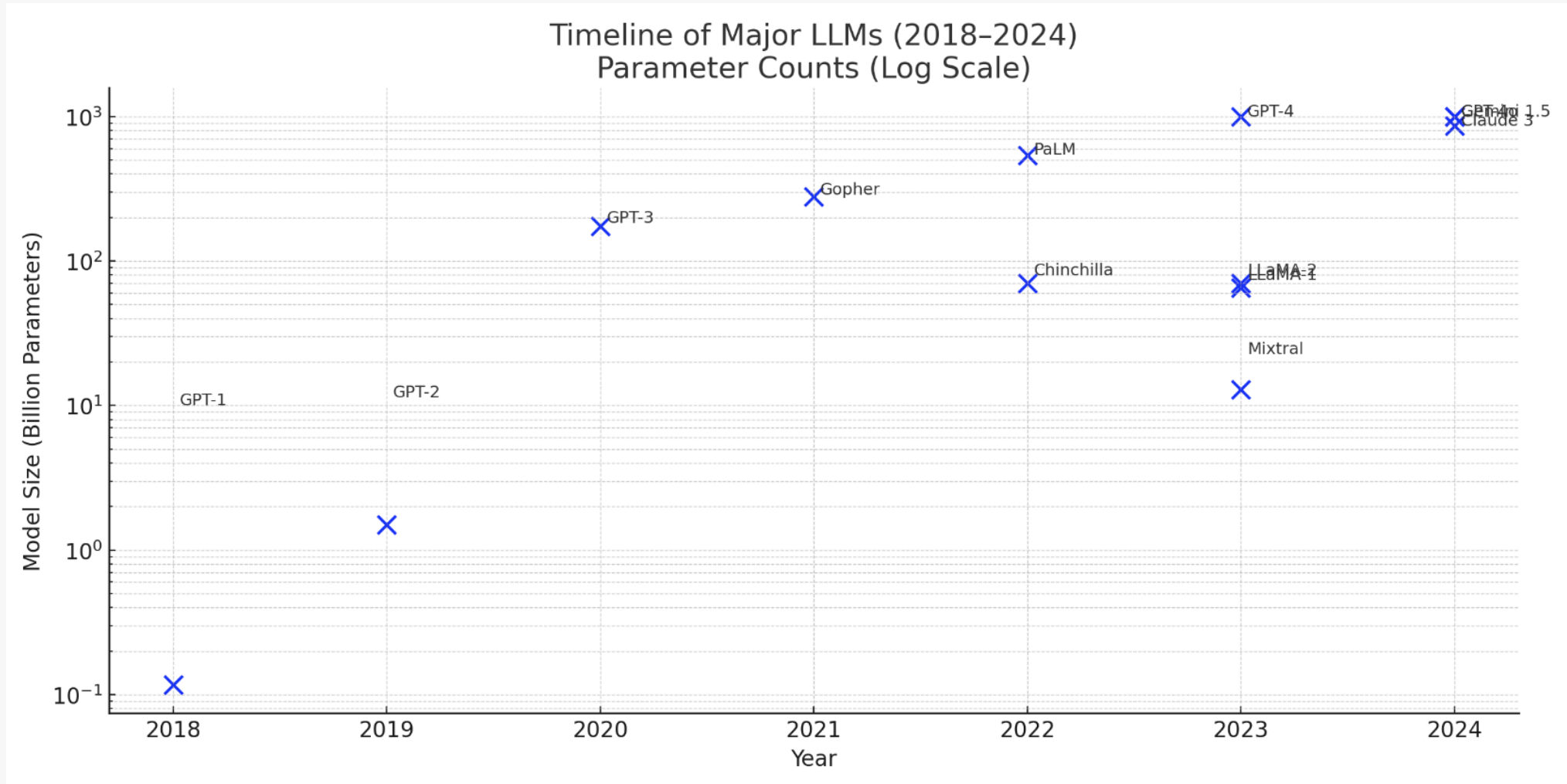
MT-Bench **AlpacaEval**
AlpacaEval Helpful, Honest, Harmless (HHH)
ToxiGen

Multilingual & Multimodal

XGLUE XTREME
FLORES **MMMU**
MathVista **MathVista**

<https://research.google/blog/evaluating-progress-of-llms-on-scientific-problem-solving/>

Why do we need large models?



Why do we need large models?



Emergent abilities are skills or behaviors that **do not appear in smaller models** but **suddenly emerge** once a model crosses a certain **scale threshold**.



Examples of Emergent Abilities

- Chain-of-thought reasoning
- In-context learning (e.g., few-shot learning without fine-tuning)
- Multistep mathematical reasoning
- Code synthesis
- Tool use (e.g., calculators, search engines)

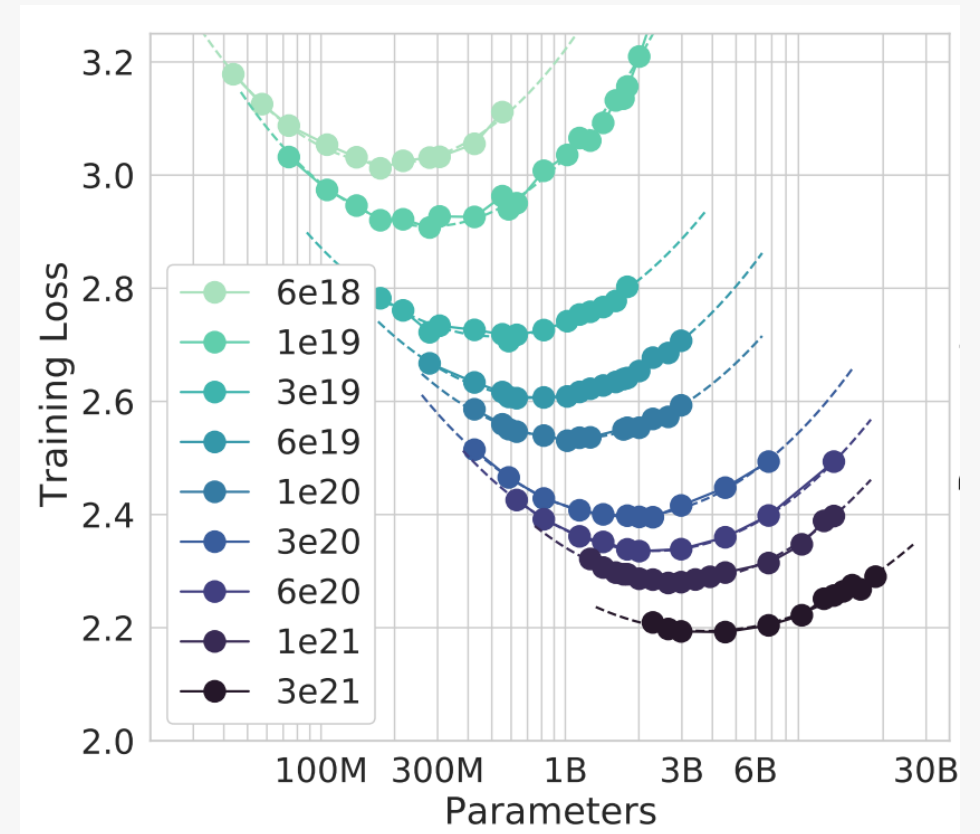
Emergent Abilities of Large Language Models, Jason Wei et al, arXiv:2206.07682

Scaling Laws & Trends

Model performance improves predictably with increased: **Model size** (parameters); **Dataset size** (tokens); **Compute** (FLOPs) (**DeepMind, 2022**)

Chinchilla Scaling

- Overtraining large models on small datasets is inefficient.
- **Optimal tradeoff**: Reduce model size, increase training tokens.
- Chinchilla (70B params, 1.4T tokens) > Gopher (280B, 300B tokens)



Training Compute-Optimal Large Language Models, DeepMind,
arXiv:2203.15556

Hallucination in LLM

LLM generates confident but false or unverifiable information.

Query: Please provide a submission script on ALCF Aurora to run a python job on 1024 nodes.

Answer from GPT-4 Turbo:

```
1 #!/bin/bash
2 #COBALT -n 1024           # Number of nodes
3 #COBALT -t 30             # Walltime in minutes
4 #COBALT -q default        # Queue name
5 #COBALT -A your_project_account # Project allocation
6 #COBALT -O my_python_job   # Output file prefix
7
8 # Load necessary modules
9 module load conda/2023-10
10 conda activate myenv
11
12 # Optional: Load Intel oneAPI, if needed
13 # module load intel-oneapi-mpi
14
15 # Path to your Python script
16 SCRIPT=your_script.py
17
18 # Launch the job using mpirun
19 mpirun -np 1024 python $SCRIPT
```

Answers that are completely incorrect and are out of the trained information

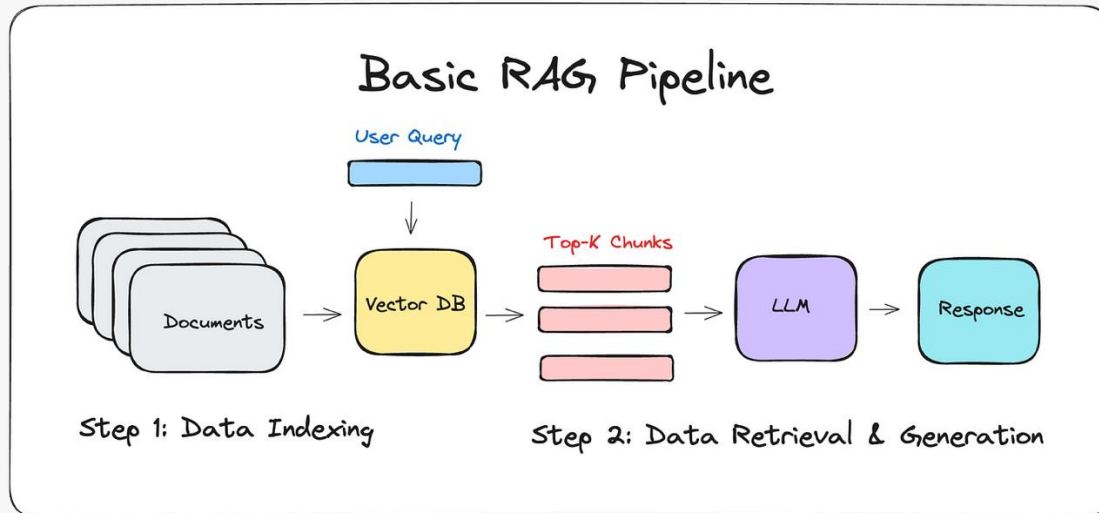
- Outdated training dataset
 - COBALT for Aurora
- Unable to understand the specific context
 - Wrong conda module
 - Wrong suggestion of oneapi module
 - Number of nodes
 - Slurm instead of PBS

A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions ; arXiv:2311.05232

Detecting hallucinations in large language models using semantic entropy; Nature volume 630, pages625–630 (2024)

Hallucination is Inevitable: An Innate Limitation of Large Language Models; arXiv:2401.11817

Retrieval Augmented Generation



RAG is a technique where a language model retrieves relevant external documents before generating an answer, grounding its output in factual, up-to-date information.

A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models, arXiv:2405.06211

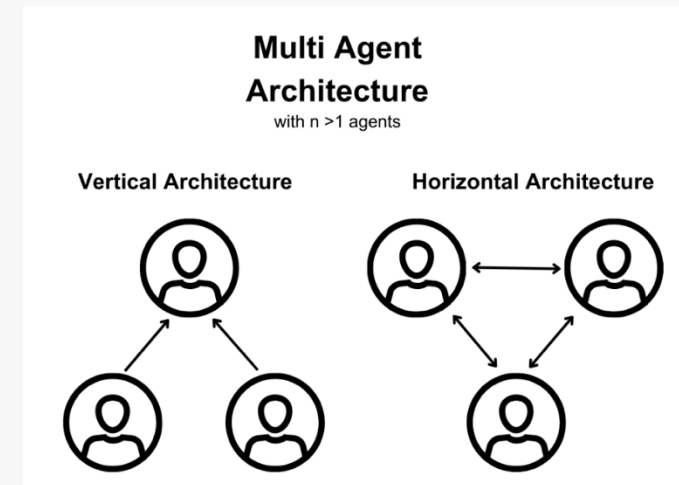
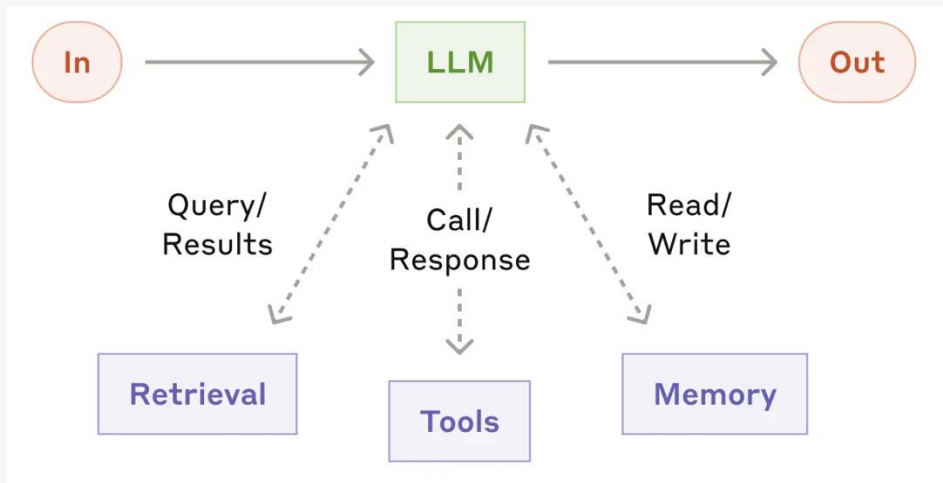
Query: Please provide a submission script on ALCF Aurora to run a python job on 1024 nodes




```
1 #!/bin/bash -l
2 #PBS -l select=1024           # selecting 1024 Nodes
3 #PBS -l place=scatter
4 #PBS ...[Other PBS settings are omitted]
5
6 NNODES=`wc -l < $PBS_NODEFILE`
7 NRANKS_PER_NODE=12
8 let NRANKS=${NNODES}*${NRANKS_PER_NODE}
9
10 module use /soft/modulefiles
11 module load frameworks
12 # ...[oneCCL and Slingshot environment settings are omitted]
13 mpiexec -np ${NRANKS} -ppn ${NRANKS_PER_NODE} \
14 --cpu-bind ${CPU_BIND} \
15 python path/to/application.py
16 ---
17 References:
18 - [1] https://github.com/argonne-lcf/user-guides/blob/main/docs/aurora/data-science/frameworks/tensorflow.md
```

AskALCF: A RAG empowered ChatBot for ALCF user support

Multi-Agent LLM System

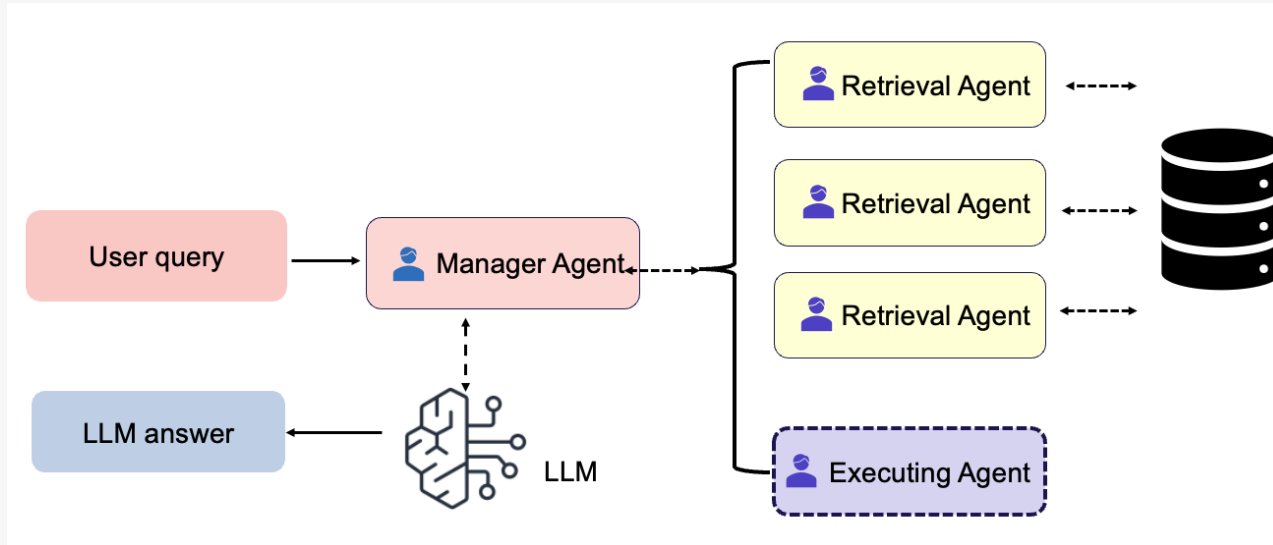
A **multi-agent system** is a collection of AI agents—often LLMs—that interact, collaborate, or compete to solve complex tasks **beyond the capability of a single model**.



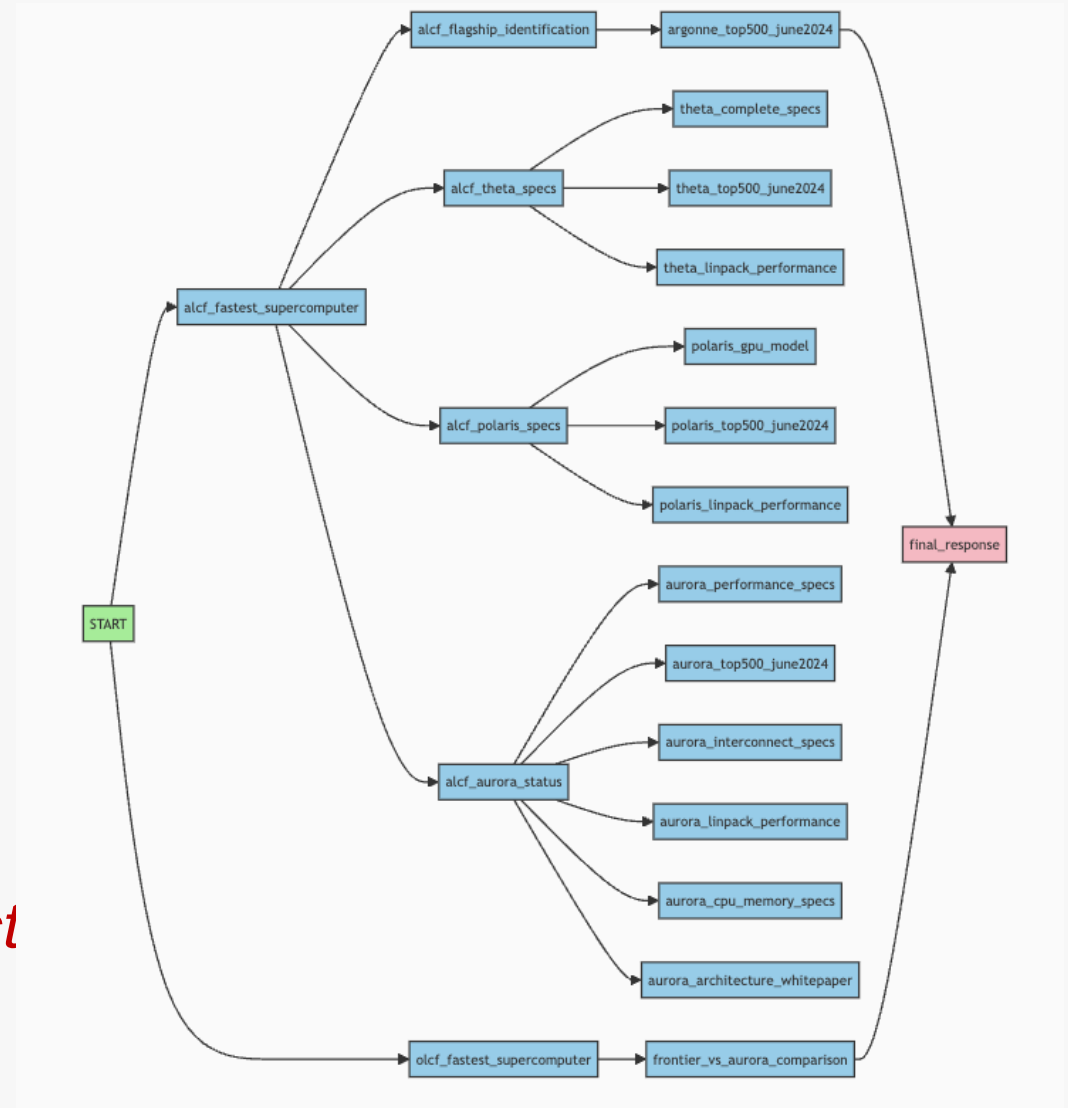
-  All agents have agent personas where they are assigned a role, understand the purpose of their tools, and how to leverage them effectively
-  Most agent implementations have reasoning, planning, and tool calling abilities 

The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey, Tula Masterman et al, arxiv:2404.11584
Large Language Model based Multi-Agents: A Survey of Progress and Challenges, Taicheng Guo et al, arXiv:2402.01680

Multi-Agent RAG system



What are the differences between the fastest supercomputers at ALCF and OLCF?



Reasoning trace from the multi-agent RAG

LLM trends over time

Exponential Growth in Model Size

- GPT-2 (1.5B, 2019) → GPT-3 (175B, 2020) — GPT-4 (est. >1T, 2023)

Rise of Open LLMs

- LLaMA, Falcon, Mistral, Mixtral, Gemma, DeepSeek, etc.

Shifts in Architecture

- Dense → Mixture of Experts (MoE): Switch, Mixtral

Efficient Fine-tuning: LoRA, QLoRA
Multi-agent collaboration

- Retrieval-Augmented Generation (RAG)
- Energy & Compute Efficiency

Hands on example

MiniLLM example

https://github.com/argonne-lcf/ATPESC_MachineLearning/blob/master/02_intro_to_LLMs/03_languagemodels.ipynb

AskALCF ChatBot: <https://anl.box.com/s/cap42vxtk91tk4k0xjk5qckq1bqztpjn>

Access AskALCF

The ChatBot is currently hosted on Crux and accessible via SSH tunneling. To access, please set up the following in your config file in your ~/.ssh/config file on your local laptop

Host askalcf-user

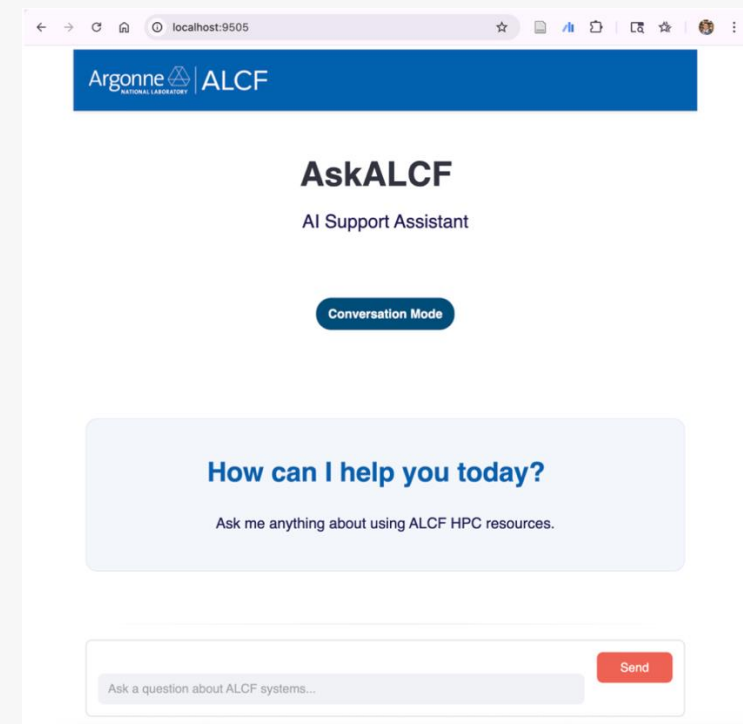
User "username"

HostName crux-uan-0001

ProxyJump username@crux.alcf.anl.gov

LocalForward 9505 localhost:9505

Then do `ssh askalcf-user` and open a browser and type below: <http://localhost:9505>



Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.