Instituto de Física de Cantabria

Large Language Models

Ignacio Heredia
Instituto de Física de Cantabria (CSIC-UC)

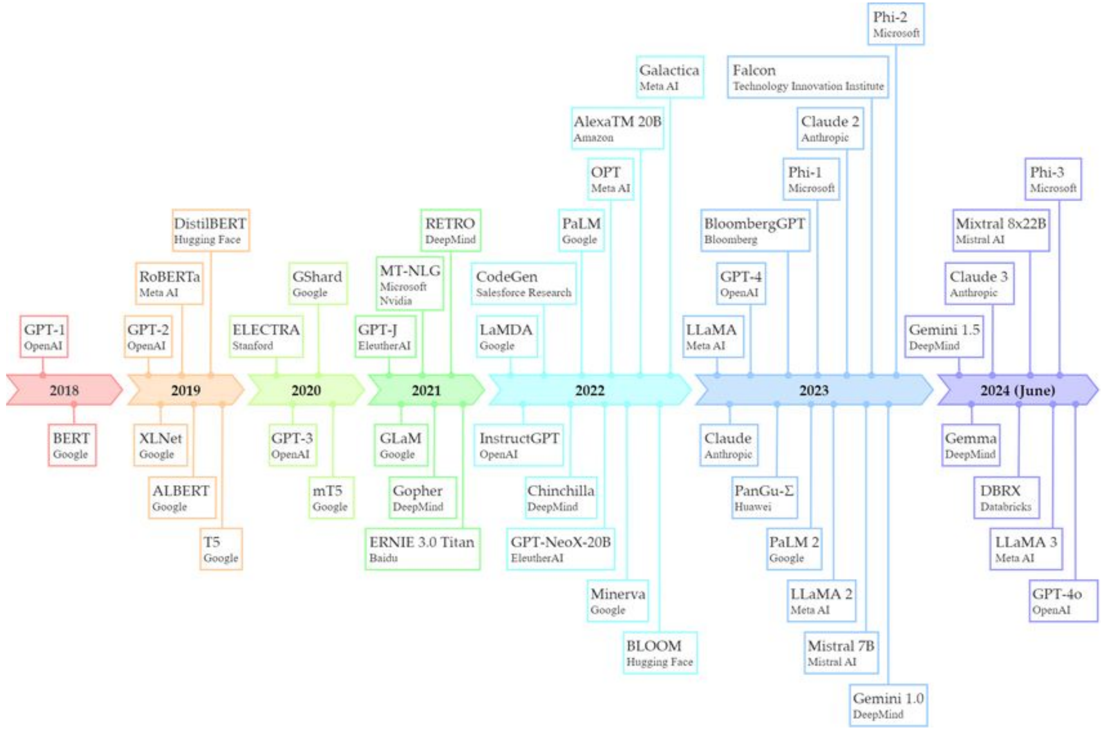
iheredia@ifca.unican.es
IgnacioHeredia

Master Data Science (UIMP-UC)
March 2025

Outline

Introduction

- Large Language Models are the practical application of the **Transformer** architecture to Natural Language Processing: Chat-**GPT** → **G**eneral **P**urpose **T**ransformer
- To make them aligned with human preferences we additionally use **Reinforcement Learning**



Training - Overview

1

Pretraining

Dataset:
100B to >5T tokens

Task: Next-token prediction on unlabeled texts

Output: base model / “foundation model”

Project Gutenberg (PG) is a volunteer effort to digitize and archive cultural works, as well as to "encourage the creation and distribution of eBooks." It was founded in 1971 by American writer Michael S. Hart and is the oldest digital **library**. Most of the items in its collection are the full texts of books or individual stories in the public domain. All files can be accessed for free under an open format layout, available on almost any computer. As of 3 October 2015, Project Gutenberg had reached 50,000 items in its collection of free eBooks.

2

Supervised finetuning

More next-token prediction

Usually 1k-50k instruction-response pairs

```
{
    "instruction": "Write a limerick about a
                    pelican.”,
    "input": "",
    "output": "There once was a pelican so fine,
               \nHis beak was as colorful as
               sunshine,\nHe would fish all day,\nIn
               a very unique way,\nThis pelican was
               truly divine!\n\n\n"
},
```

```
{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```

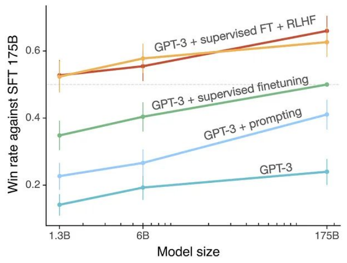
3

Alignment

Align with human preferences

Usually reinforcement learning with human feedback (RLHF)

>50k examples

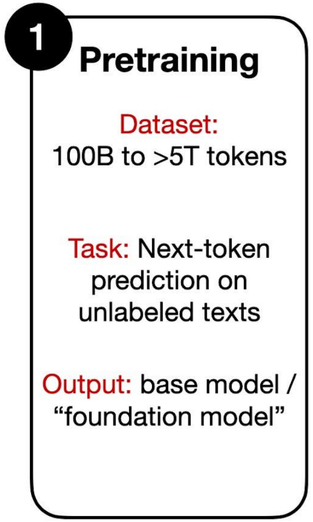


Training - Pretraining

This phase allows the model to learn the grammar and to *acquire its general knowledge* about the world.

It's a **self-supervised** phase where we try to predict the word that comes next. So we take text chunks, remove last word and use as input. Then use last word as a label.

It's often trained in *huge datasets*, in the order of hundred TB of data (eg. FineWeb). Dataset is self.-supervised, so easy to produce (web scraping).

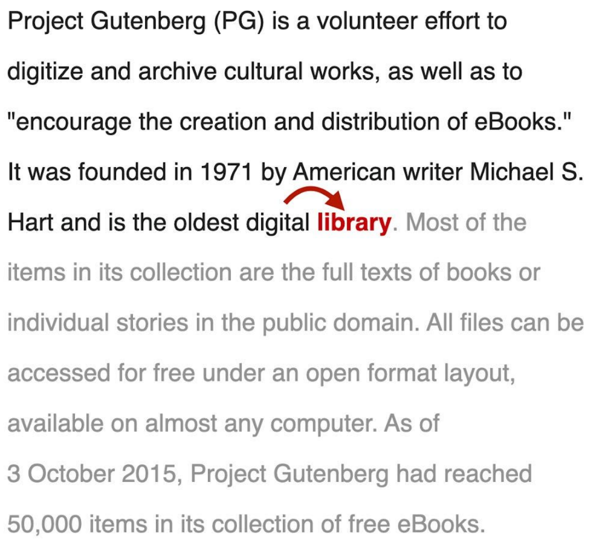


1 **Pretraining**

Dataset:
100B to >5T tokens

Task: Next-token prediction on unlabeled texts

Output: base model / “foundation model”



Project Gutenberg (PG) is a volunteer effort to digitize and archive cultural works, as well as to "encourage the creation and distribution of eBooks." It was founded in 1971 by American writer Michael S. Hart and is the oldest digital **library**. Most of the items in its collection are the full texts of books or individual stories in the public domain. All files can be accessed for free under an open format layout, available on almost any computer. As of 3 October 2015, Project Gutenberg had reached 50,000 items in its collection of free eBooks.

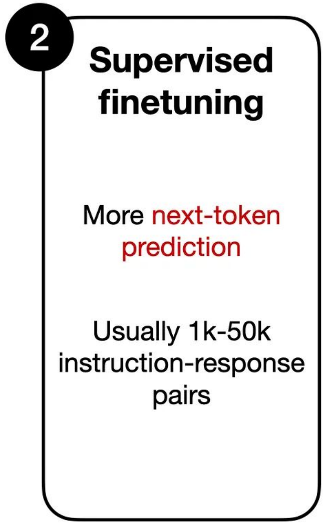
Training - Supervised finetuning

This tunes the model to *follow instructions*.

The **supervised finetuning** stage involves another round of next-token prediction.

However, unlike the preceding pretraining stage, we now work with **instruction-output pairs**:

- The *instruction* is the input given to the model (it is sometimes accompanied by an optional *input* text, depending on the task).
- The *output* represents a desired response similar to what we expect the model to produce.


2
Supervised finetuning
More next-token prediction
Usually 1k-50k instruction-response pairs

```
{
    "instruction": "Write a limerick about a
                    pelican.”,
    "input": "",
    "output": "There once was a pelican so fine,
               \nHis beak was as colorful as
               sunshine,\nHe would fish all day,\nIn
               a very unique way,\nThis pelican was
               truly divine!\n\n\n"
},


{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```

Training - Supervised finetuning

- The dataset is manually produced so it's not very big.
- It's important that it is *high quality* to avoid adding noise to the model,
- Sometimes the input-output pairs can be themselves generated by an LLM to speed update the process (*self-instruct*),
- In particular, small models (low learning capacity) should be trained on data with simple words.

2

Supervised finetuning

More next-token prediction

Usually 1k-50k instruction-response pairs

```
{
    "instruction": "Write a limerick about a
                    pelican.”,
    "input": "",
    "output": "There once was a pelican so fine,
               \nHis beak was as colorful as
               sunshine,\nHe would fish all day,\nIn
               a very unique way,\nThis pelican was
               truly divine!\n\n\n"
},
```

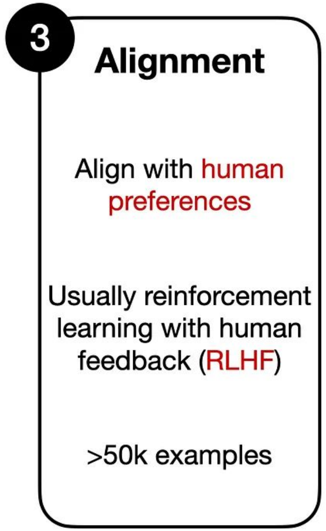
```
{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```

Training - Alignment

This tunes the model to be useful as a chatbot, to *align with human preferences*

For this is uses **Reinforcement Learning with Human Feedback** (RLHF) which basically can be divided in three steps:

1. Supervised finetuning of the pretrained model
2. Creating a reward model
3. Finetuning via proximal policy optimization

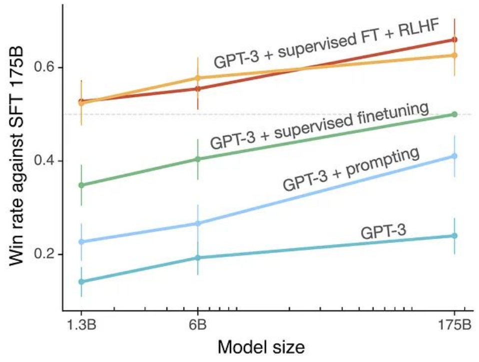


3 **Alignment**

Align with human preferences

Usually reinforcement learning with human feedback (RLHF)

>50k examples

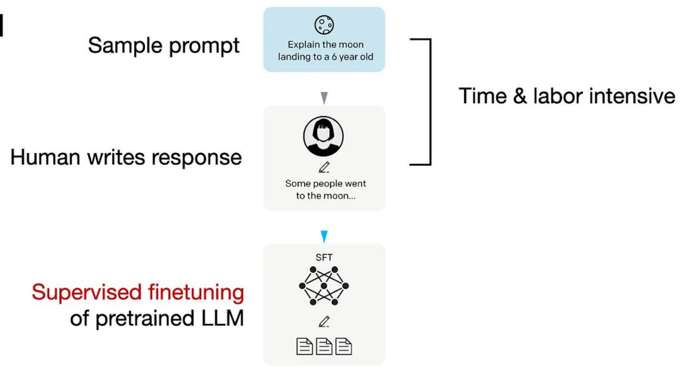


Each step further increases the overall performance.

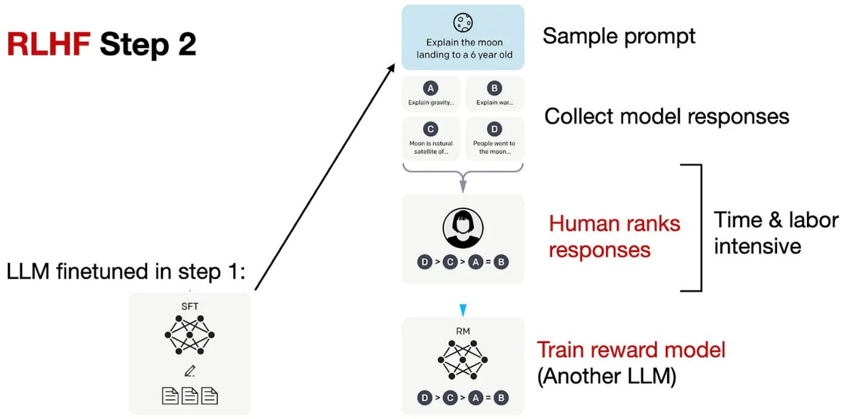
(*) can be merged with the previous step but listed because it is an integral part of the RLHF workflow

Training - RLHF

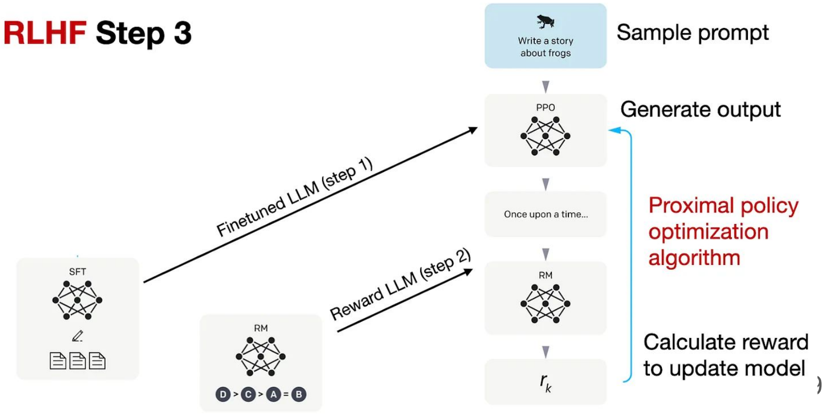
RLHF Step 1


Sample prompt
Explain the moon landing to a 6 year old
Time & labor intensive
Human writes response
Some people went to the moon...
Supervised finetuning of pretrained LLM
SFT

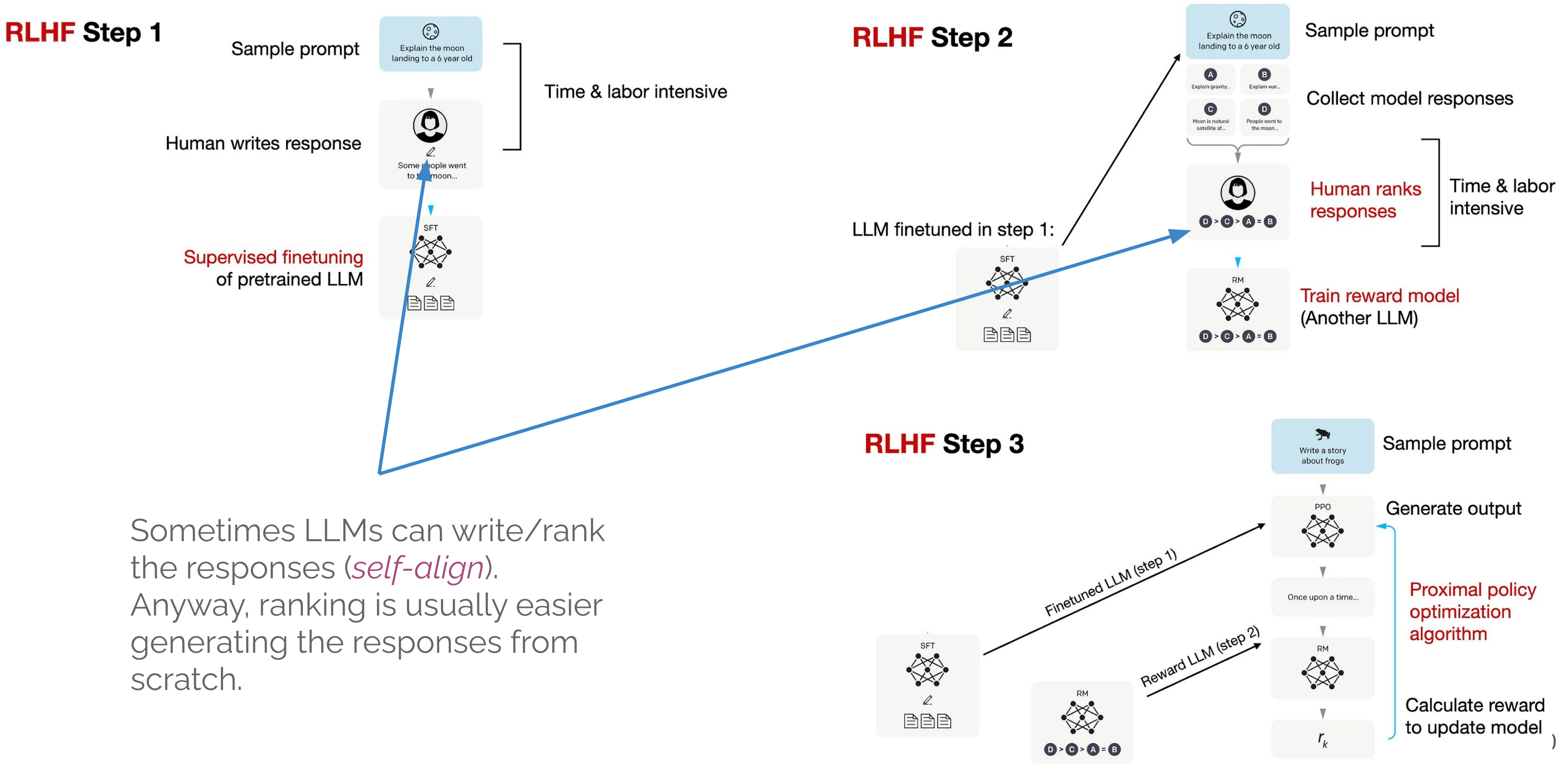
RLHF Step 2



RLHF Step 3



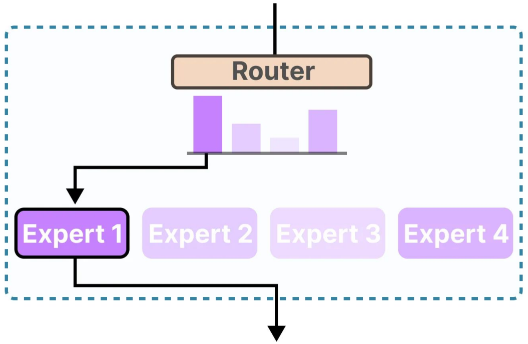
Training - RLHF

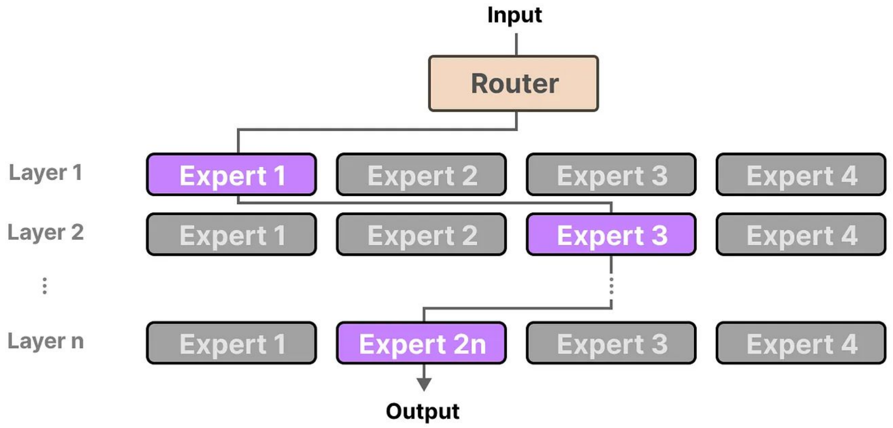


Sometimes LLMs can write/rank the responses (*self-align*).
Anyway, ranking is usually easier generating the responses from scratch.

Concepts - Mixture of Experts

- LLMs are usually very consuming to deploy. Instead of training a giant LLM that can perform all the tasks, we can have a set of smaller LLMs that are **specialized in different tasks**.
- We have a router module that can redirect the question to the correct expert.
- Then at inference time we **only activate a subset** of the network.
- For example, DeepSeek-R1 has 671B parameters but only activates 37B, which is much more **energy efficient** than comparable models (like Open AI o1).

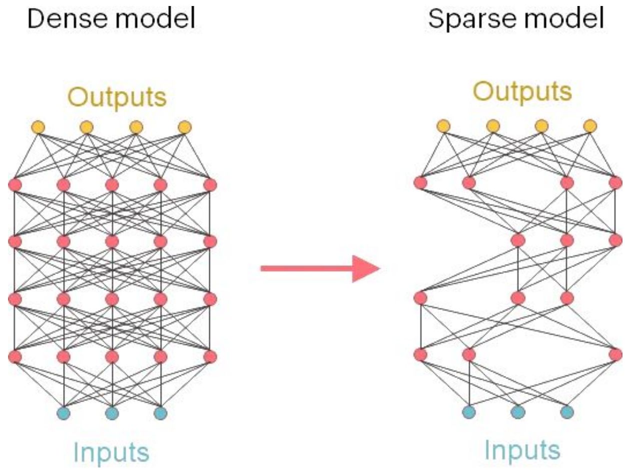


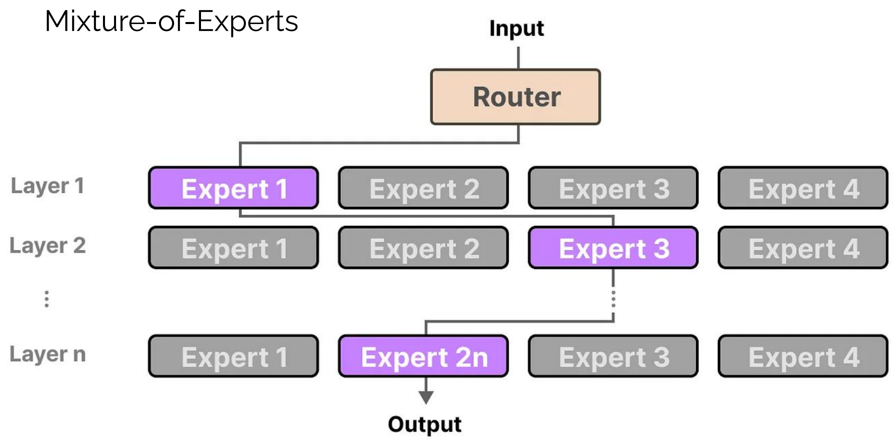


Concepts - Mixture of Experts

The concept of MoE is similar to **sparse models** but:

- Sparsity is a general principle of selective activation. MoE is a specific implementation of sparsity in model architecture.
- MoE *structured* approach allows for easy addition or modification of experts, making the system more adaptable.
- General sparsity doesn't necessarily imply *specialization* of activated parts.





Concepts - Model distillation

- LLMs are sometimes too big to deploy in a given infrastructure
- One possible strategy to reduce their size is **model distillation**, where a smaller LLM is trained on the outputs of the original model. The underlying idea is that the model knowledge is transferred to the smaller model.
- For example, the original DeepSeek-R1 671B has been distilled into other, smaller, open-source models:
  - **Llama3.1** 8B,
  - **Llama 3.3** 70B,
  - **Qwen2.5**: 1.5B, 7B, 14B, 32B

Concepts - Quantization

Quantization is the process of “compressing” a model’s weights by changing them to **lower-precision** representations. Typically this goes from a 32bit float, to around *4bits*, which is important for *low-memory systems*. These are then dynamically cast to Bfloat16 at runtime for inference to perform matrix multiplication (**dequantization**).

⚠️ Quantization saves space but makes *inference slower* due to the dynamic cast and *loses precision*, making models worse.

Choosing a small model or quantized a big one?

- All things equal, *larger model quantized is usually better* than smaller model at full precision. The idea is that a bigger model quantized is still retaining the multiple associations between tokens, but at a lower precision. Smaller models just lose these associations.
- ⚠️ *“All things equal”*: take into account that new smaller models are better than old bigger models

Concepts - Quantization

Quantization methods

- Most common quantization methods: **GGUF, GPTQ, AWQ**
- Moderns quantization methods does not quantize all layers uniformly; instead, *some layers are quantized more than others*. Leaving most commonly used layers untouched, we improve overall precision and improve inference time (reducing dequantization time).

How to read quantization formats?

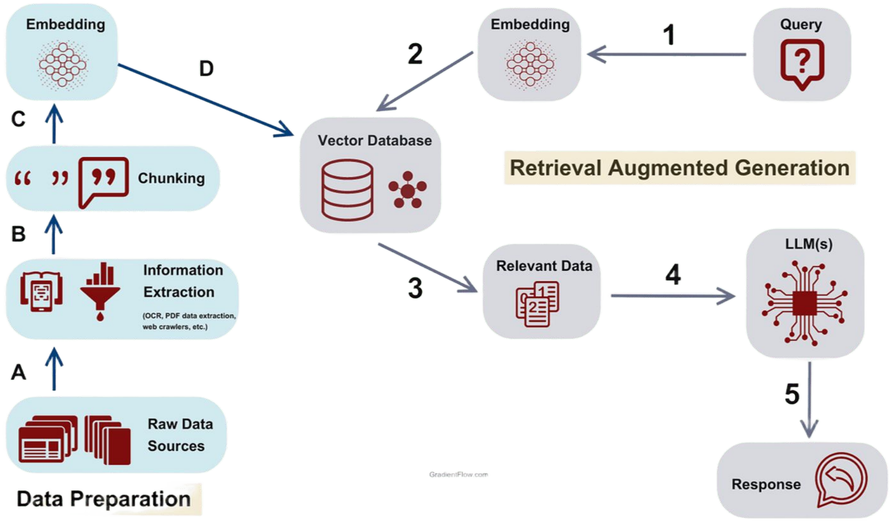
- Ollama uses mainly GGUF formats, vLLM support mainly AWQ.
- By default it usually uses `**Q4_K_M**`
    - **Q4**: uses 4 bits
    - **K**: uses K-quantization, which is a subclass of GGUF quantization
    - **M**: the type of K-quantization used

Concepts - RAG

- **Retrieval Augmented Generation** (RAG) is a technique that enhances the capabilities of LLMs by incorporating external information retrieval.
- **Benefits**:
  - Improves accuracy and *reliability* of AI-generated responses
  - Allows access to *updated information*, addressing the issue of outdated training data
  - Enables *domain-specific knowledge* integration without retraining the entire model (*model finetuning*)
- **(!)** RAG doesn't prevent hallucinations though it can mitigate them.

Concepts - RAG

- **Indexing**: Documents are preprocessed and converted into a LLM-compatible embeddings, using an *embeddings model*. The embeddings are then saved into a *vector database*. Documents are often split into smaller segments to provide more focused context.
- **Retrieval**: When a user query is received, the system employs various methods to find the most relevant documents (ranked):
    - **Vector search** (semantic search): Compares the query's vector representation to document vectors using similarity measures like *cosine similarity*
    - **Full-text** (keyword search): Uses techniques like TF-IDF and BM25 to match keywords



- **Inference**: The documents are feed to the context of the LLM that produces the answer. LLM with longer context windows allow for more comprehensive responses.

Concepts - Chain of Thought

- In Chain-of-Thought you allow the model to iteratively go over previous responses before spitting out the final response.
- It improves the results in complex tasks that need multi-step. But this computation adds extra latency at inference time.

Training

- To train CoT you can give the model the input, the derivation and the answers.
- In the case of DeepSeek-R1, they are training only with the answers. This is better because you don't need to build an expensive dataset.
- Their Reinforcement Learning workflow rewards both:
  - having outputted the right answer,
  - having a outputted a monolog in the correct format (`<think>`) (small reward),

→ Over the time the internal monologue improves

Software resources

- Deploy LLMs: ollama, vllm
- UI: OpenWebUI , LibreChat , LobeChat , Perplexica
- Create workflows: LangChain, Llama-index
- Leaderboards:
  - Chatbot Arena
  - Humanity Last Exam
  - Massive Text Embedding Benchmark (MTEB)
  - Scale SEAL Leaderboard
  - ARG-AGI
  - LiveBench
  - LMSYS Leatherboard

Instituto de Física de Cantabria

Questions

Ignacio Heredia
Instituto de Física de Cantabria (CSIC-UC)

iheredia@ifca.unican.es

IgnacioHeredia

Image sources