Advanced Human Language Technologies

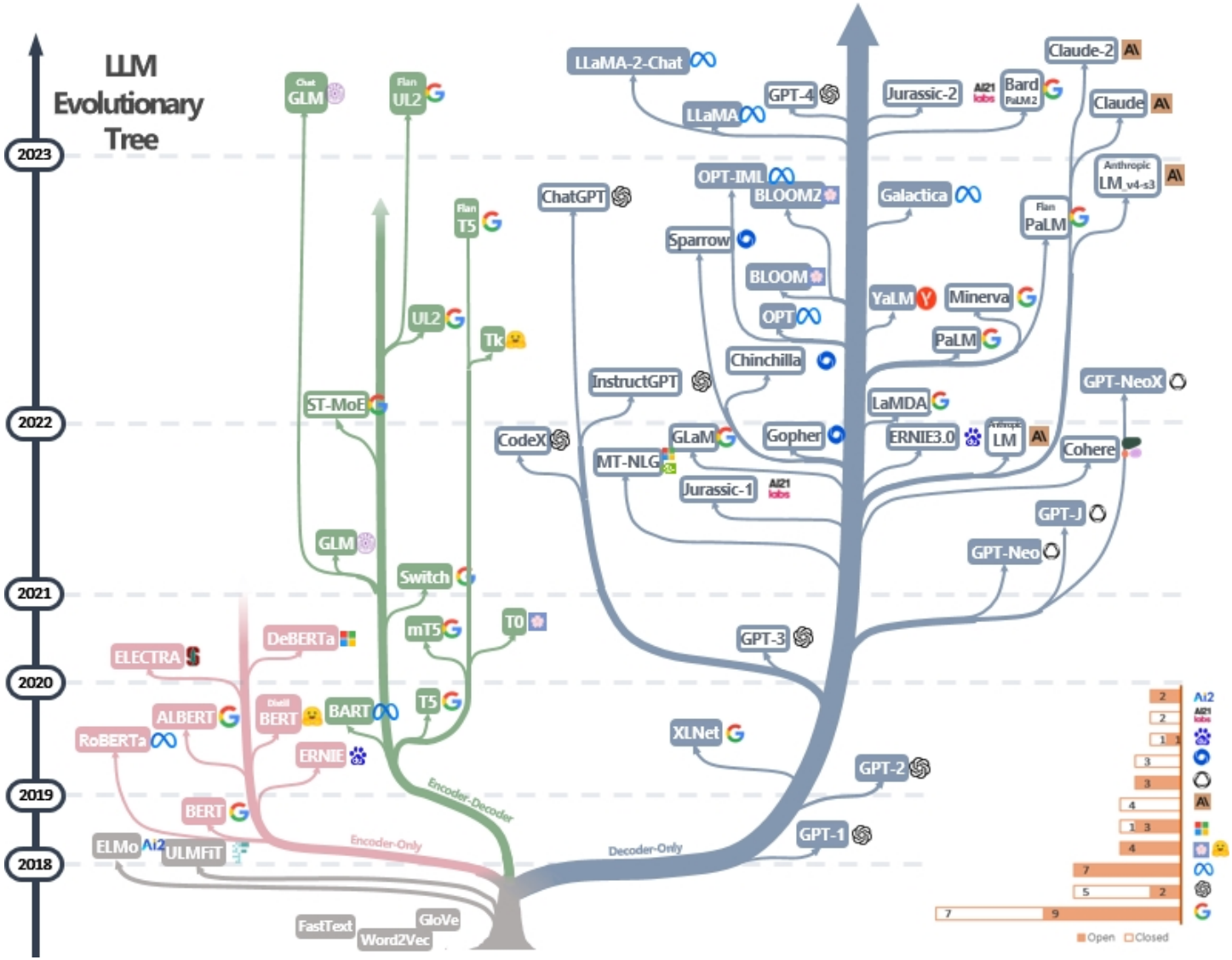
Large Language Models

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

Outline

History of Transformer-based LLMs



(Yang et al. 2023) *Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond*
https://arxiv.org/abs/2304.13712

History of Transformer-based LLMs

Year	*Name*	*Architecture*	*Size*	*Training data*
2018	**BERT**	Encoder-only	340M	13GB (Wikipedia+BookCorpus)
	Pre-training goal: Masked Language Modeling (MLM) Contribution: Introduced bidirectional context and MLM for pre-training			
2019	**T5**	Encoder-decoder	11B	750GB (C4)
	Pre-training goal: Text-to-Text Transfer Transformer Contribution: Unified natural language understanding and generation tasks under a text-to-text framework			
	BART	Encoder-decoder	400M	160GB (Wikipedia+BookCorpus)
	Pre-training goal: Denoising Autoencoder (DAE) Contribution: Introduced a flexible DAE objective that can handle several types of noise and improve text generation quality			
	GPT-2	Decoder-only	1.5B	40GB (WebText)
	Pre-training goal: Autoregressive Language Modeling (ALM) Contribution: Improved the quality and diversity of text generation using a larger model and dataset			
	XLNet	Encoder-only	340M	40GB (WebText)
	Pre-training goal: Permutation Language Modeling (PLM) Contribution: Introduced a novel PLM objective that preserves bidirectional context and avoids the pretrain-finetune discrepancy			

History of Transformer-based LLMs

Year	*Name*	*Architecture*	*Size*	*Training data*
2020	**GPT-3**	Decoder-only	175B	570GB(*)
	Pre-training goal: Autoregressive Language Modeling (ALM) Contribution: Scaled up ALM to a very large model which showed zero-shot and few-shot capabilities			
2022	**GPT-3.5**	Decoder-only	175B	570GB(?)
	Pre-training goal: Autoregressive Language Modeling (ALM) Contribution: Reinforcement Learning from Human Feedback (RLHF) and Supervised FineTuning (SFT) to align with human values.			
	BLOOM	Decoder-only	175B	1.6TB (ROOTS)
	Pre-training goal: Autoregressive Language Modeling (ALM) Contribution: Open source and trained on a large multilingual corpus covering 46 languages and 13 programming languages			
2023	**GPT4**	Decoder-only(?)	+175B(?)	(?)
	Pre-training goal: Autoregressive Language Modeling (ALM) Contribution: Multimodality, processes both images and text input and produces both images and text output			

(*) Common Crawl + WebText2 + Books1/2/3 + Wikipedia + CC-News + OpenWebText + Stories + RealNews

(?) Guessed information, actually undisclosed by model owners

Outline

Datasets for LLM

- To train LLMs and ChatGPT-like chatbots, we need large and varied datasets
- Large datasets help the models achieve higher accuracy, fluency, and diversity in their outputs
- Varied datasets help the models cover different domains, languages, and modalities, such as web pages, books, code, images, and audio
- Two types of datasets are used:
  - **Pre-training datasets**: Used to train the Language Model and create a *foundational* LLM.
  - **Instruction datasets**: Used to fine-tune the foundational LLM intro an *instructed* chatbot able to follow instructions.

Datasets for LLM: Examples

Some LM Pre-training datasets

- The Pile (by EleutherAI, curated, multiple sources)
- Common Crawl (massive open data, raw internet text)
- MassiveText (highly curated, not openly acessible)
- Wikipedia
- GitHub (code in many programming languages)

The Pile Dataset (I)

For example, *The Pile* Dataset (GPT3-NeoX) includes:

- Pile-CC: A collection of website crawls from 2008 onwards, with diverse domains but varying quality data.
- PubMed Central: A subset of the PubMed repository for biomedical articles providing open, full-text access to ∼5M medical domain publications.
- Books3: Book dataset, almost an order of magnitude larger than the second largest book dataset. Valuable for long-range context modeling and coherent storytelling.
- OpenWebText2: A generalized multilingual web scrape dataset including recent content from Reddit submissions up until 2020.
- ArXiv: Preprint server for research papers in math, computer science, and physics. Written in LaTeX.
- GitHub: Large corpus of open-source code repositories.
- FreeLaw: Provides access to legal opinions from federal and state courts.

The Pile Dataset (II)

- Stack Exchange: Publicly available repositories of question-answer pairs in diverse domains.
- USPTO Backgrounds: Dataset of technical writing on applied subjects.
- Wikipedia (English): Source of high-quality text written in expository prose spanning many domains.
- PubMed Abstracts: Biomedical article abstracts from PubMed and MEDLINE.
- Project Gutenberg: Classic Western literature, PG-19 dataset includes books before 1919, distinct from modern books.
- OpenSubtitles: English language dataset of subtitles from movies and TV shows, natural dialog source, useful for creative writing tasks.
- DeepMind Mathematics: Mathematical problems from various topics, formatted as natural language prompts, improve mathematical ability of language models.
- BookCorpus2: Expanded version of BookCorpus, a widely used language modeling corpus, unlikely to overlap with other datasets.

The Pile Dataset (III)

- Ubuntu IRC: Publicly available chatlogs from Ubuntu-related channels on Freenode IRC chat server, model real-time human interactions.
- EuroParl: Multilingual parallel corpus of European Parliament proceedings in 21 languages from 1996 to 2012.
- YouTube Subtitles: Parallel corpus of text from human-generated closed captions on YouTube, provides multilingual data, educational content, popular culture, and natural dialog.
- PhilPapers: Open-access philosophy publications, spanning a wide body of abstract/conceptual discourse in high-quality academic writing.
- NIH Grant Abstracts: Awarded grant applications from 1985 to present, high-quality scientific writing.
- Hacker News: Link aggregator operated by Y-Combinator, focus on computer science and entrepreneurship, comment trees provide high-quality dialogue and debate on niche topics.
- Enron Emails: Valuable corpus for research on email usage patterns, aids in understanding the modality of email communications.

Datasets for LLM: Examples

Chatbot Instruct datasets

- **Alpaca**
  Stanford's instruction-following dataset, based on OpenAI's GPT responses, 50,000 question-answers
- **FLAN**
  Fine-tuned LAnguage Net, by Google, contains diverse NLP tasks
- **Awesome Instruction Dataset**
  Curated open-source dataset including several sources, multimodal
- **OASST**
  Open-Assistant Conversations, community-driven instruction data
- **Dolly 2.0 Dataset**
  by Databricks, for instruction tuning
- **LAION-5B**
  Large-scale Artificial Intelligence Open Network, multimodal, 5 bilion text-image pairs

Outline

Evaluation of LLM

Evaluating LLMs and ChatGPT-like models is challenging for several reasons:

- There is no clear definition of what constitutes a good or bad output for these models, as different tasks and domains may have different criteria and expectations.
- There is a lack of standardized benchmarks and metrics to measure the quality and reliability of these models, especially for complex and open-ended tasks.
- There is a risk of generating erroneous, misleading, or harmful outputs that may not be easily detected or corrected by human users or reviewers.
- There is a need for ethical and responsible use of these models, as they may have social, legal, and moral implications for various stakeholders.

Evaluation of LLMs at the BigScience Workshop

https://bigscience.huggingface.co/

Workshop goals included developing standard LLM evaluation measures. Some considered tasks were:

- Extrinsic evaluation: downstream applications (e.g., sentiment analysis, natural language inference, MT, text summarization)
- Intrinsic evaluation: internal properties and capabilities (e.g., syntactic parsing, semantic role labeling, named entity recognition, fact verification)
- Few-shot generalization: minimal or no supervision tasks (e.g., text classification, text generation, question answering, summarization)
- Bias and social impact: potential harms and benefits (e.g., gender/race bias, toxicity, hate speech, misinformation, privacy)
- Multilingualism: performance and limitations across languages and scripts (e.g., English, French, Spanish, German, Chinese, Arabic, Hindi, Bengali, ...)

Outline

Outline

Adapting LLMs as Chatbots and Assistants

- Large Language Models (LLMs) can be adapted to work as chatbots/assistants to interact with humans.
- This requires several techniques such as fine-tuning (FT) and Reinforcement Learning (RL)
  - FT involves training a pre-trained LLM for generating human-like responses in conversational context, using some instruction dataset.
  - RL can be used to further improve the performance of the chatbot by rewarding it for generating appropriate responses and punishing it for inappropriate responses

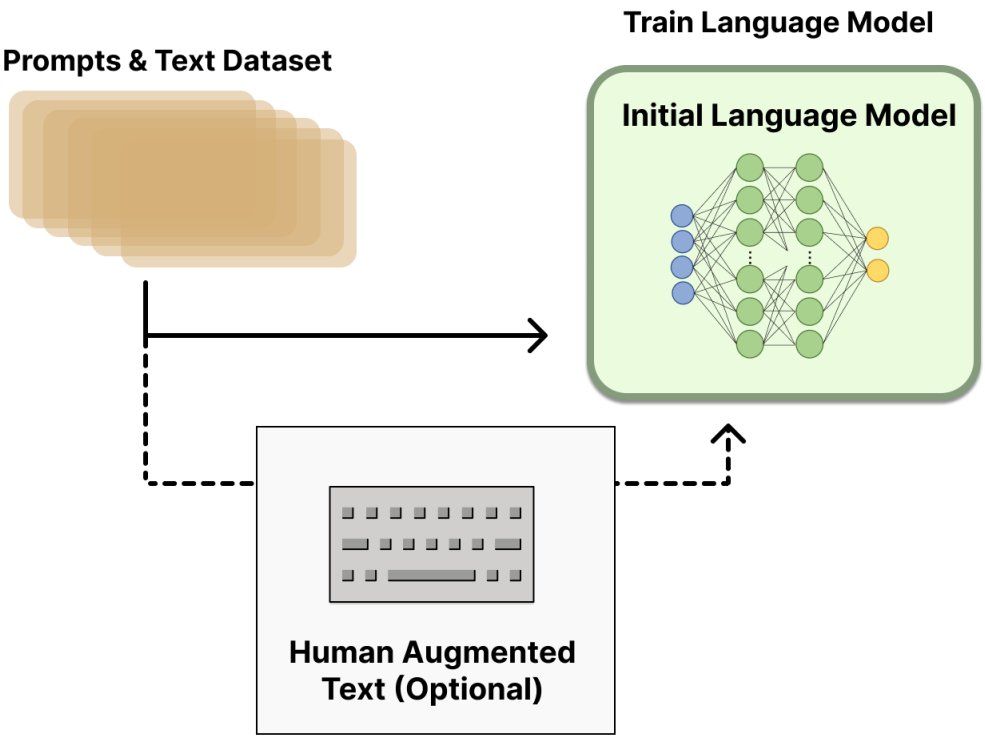
Outline

Reinforcement Learning from Human Feedback (RLHF)

- RLHF is a method to optimize a language model with human feedback
- It involves three steps:
  1. Pretraining a LM (or getting a pretrained one)
  2. Gathering data and training a reward model
  3. Fine-tuning the LM with reinforcement learning (RL) using the reward model
- It helps LMs to align with complex human values and preferences

1. Pretraining language models

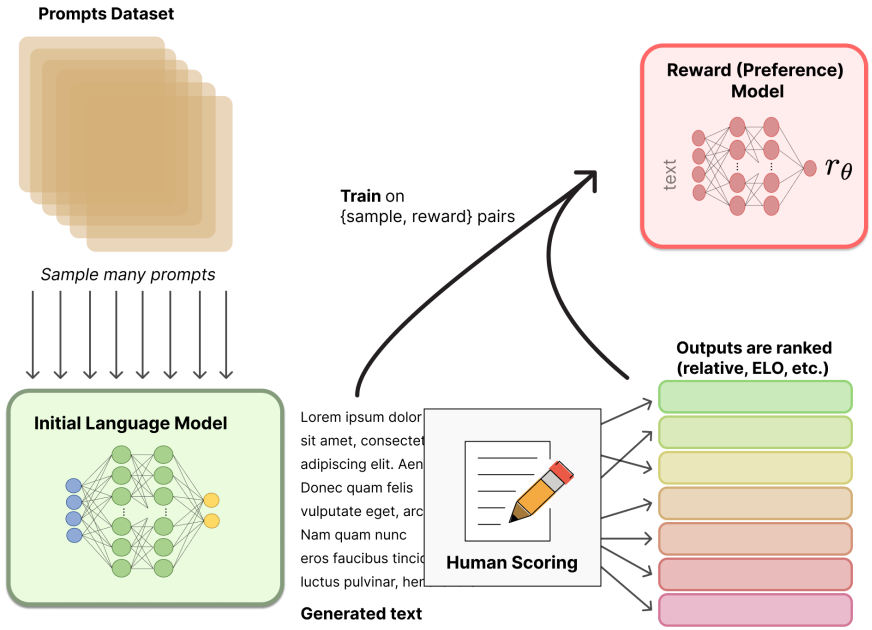
- RLHF uses a LM that has already been pretrained with a standard objective (e.g. next token prediction)
- The initial model can also be fine-tuned on additional instruction data.



https://huggingface.co/blog/rlhf

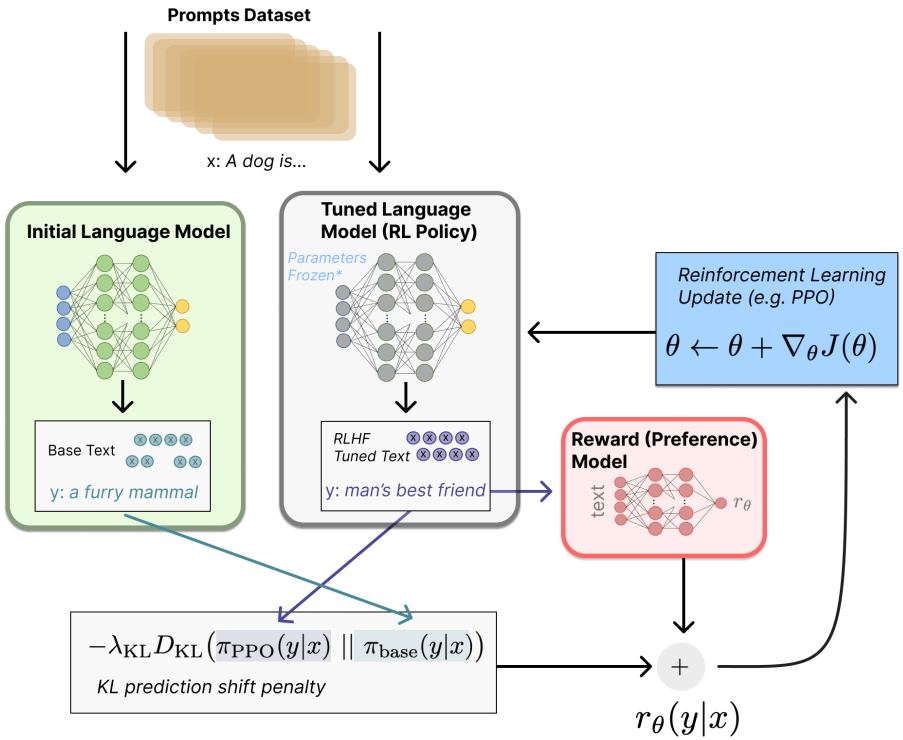
2. Gathering data and training a reward model

- RLHF collects human annotations for generated text and trains a reward model to predict them
- The reward model can be a classifier or a regressor that takes text as input and outputs a score
- The reward model can capture human preferences such as creativity, truthfulness, or executability

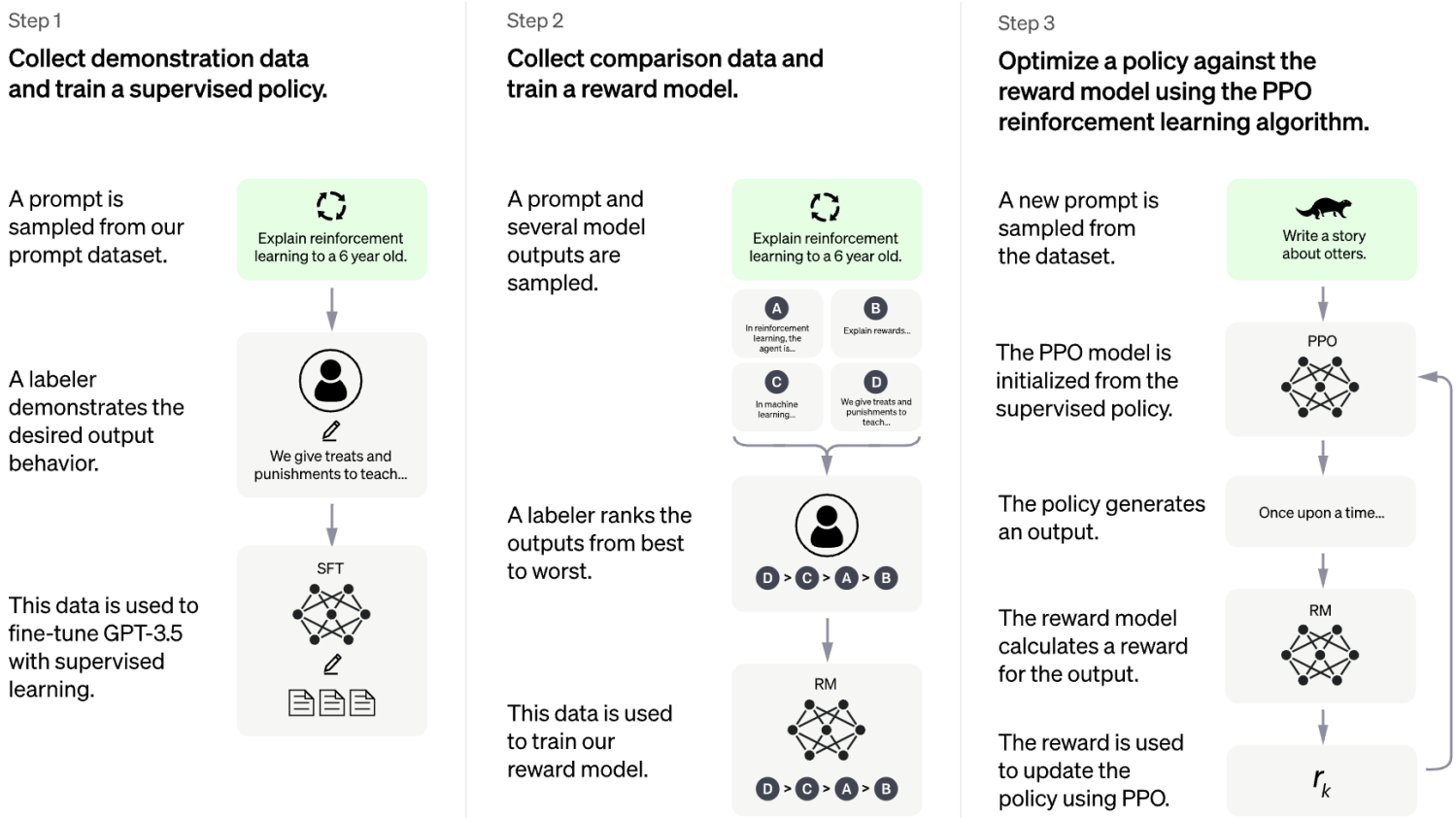


3. Fine-tuning the LM with RL, using reward model

- RLHF uses an RL algorithm (e.g. PPO) to fine-tune the LM with the reward model as the objective.
- The RL algorithm updates the LM parameters to maximize the expected reward.
- The fine-tuned LM can generate text that better fits human feedback



Example: ChatGPT fine-tuning



Outline

Using LLMs

- LLMs are pre-trained language models, able to generate plausible continuations to given input text.
- But plain models are verbose non-stop text generators, without a definite purpose.
- To perform specific tasks, they need to be instructed, prompted, or fine-tuned.

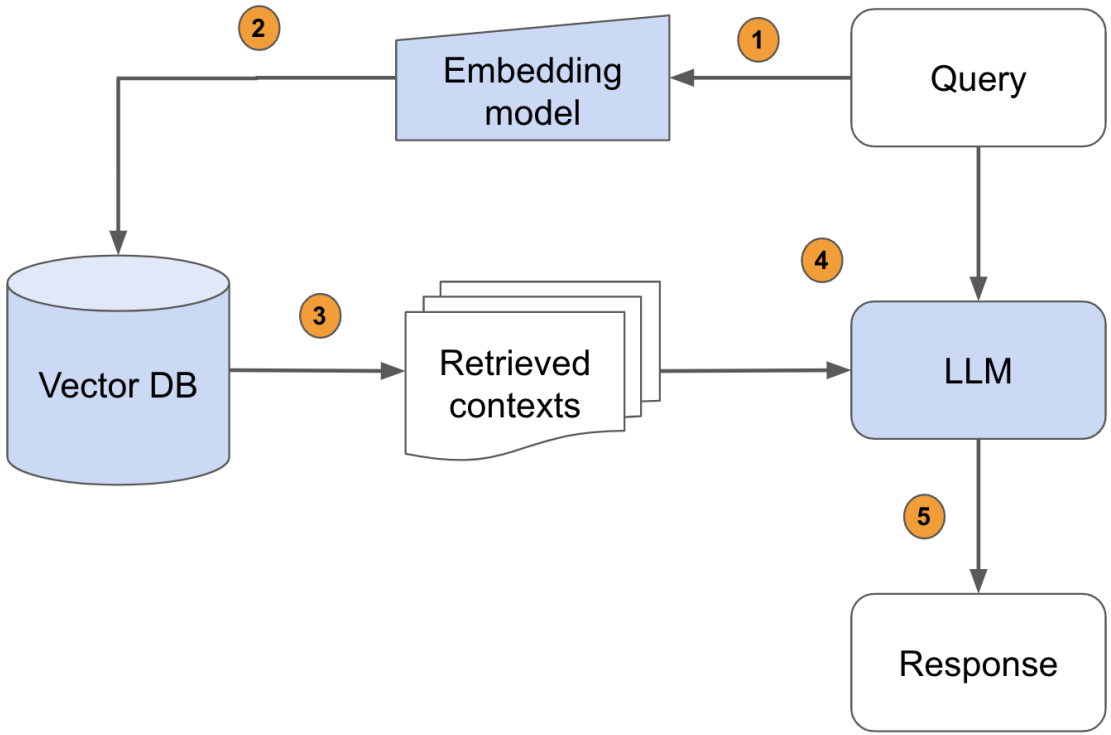
Using LLMs

Strategies to adapt LLM behaviour:

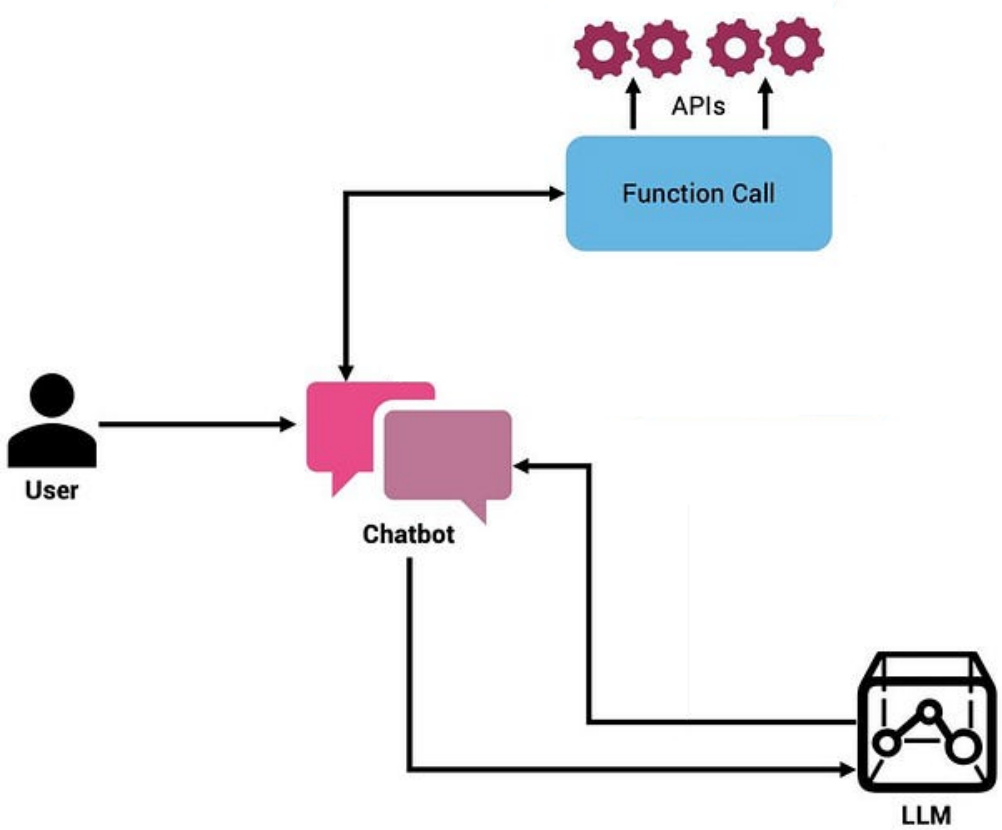
- **Instruction**: A basic model is trained with tens of thousands of examples of usual tasks (summarization, translation, question-answering, problem solving, reading comprehension, code generation, etc). RLHF.
- **Fine-tuning**: A model (usually not instructed) is trained with hundreds of examples of a specific task (e.g. semantic representations, dialog intent extraction, function calling...)
- **Prompting** (few-shot learning): An *instructed* model is shown a few examples of a desired task, and asked to repeat it on new data (e.g. NERC, information extraction, anonymization, ...)

LLMs are components in bigger systems

- **Retrieval-Augmented Generation (RAG)**: Use LLM to process retrieved documents and generate an answer to the user query.
  Useful to build domain-expert assistants (medical, law, administrative, research, ...), educational tutors, customer support, etc.

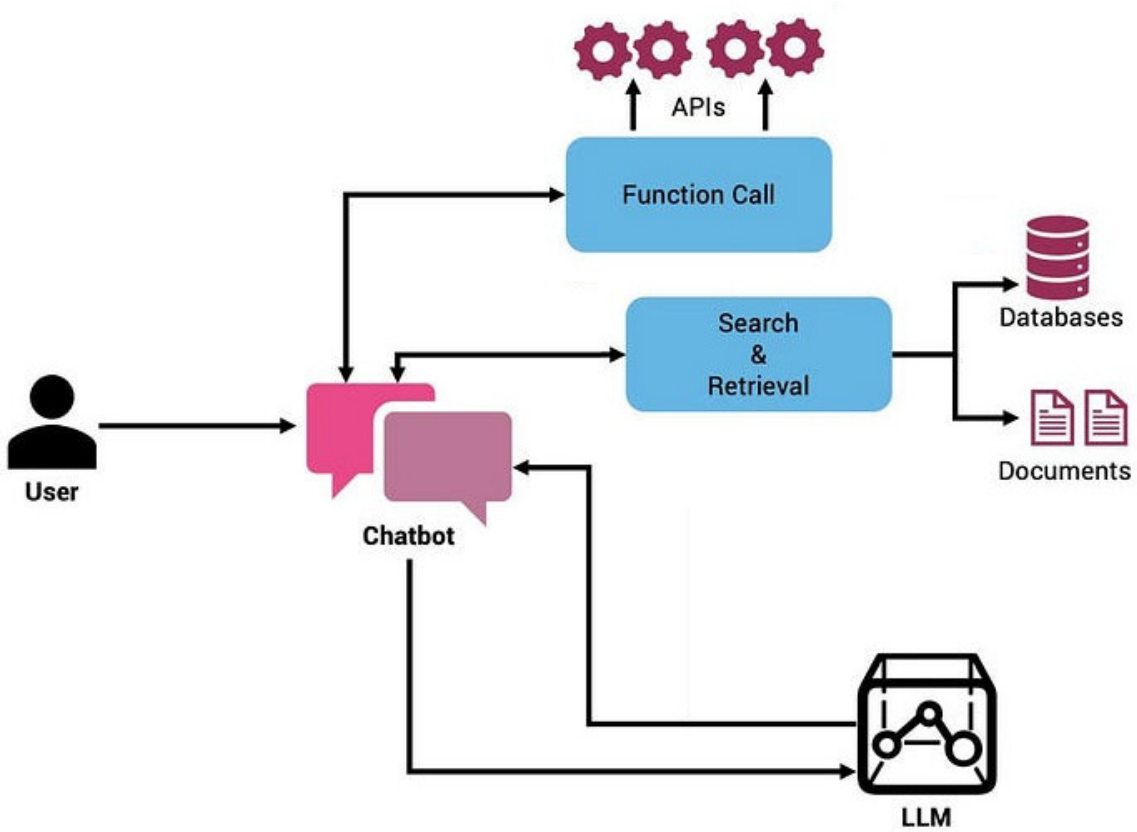


LLMs are components in bigger systems



- **Function Calling**: Use LLM to process queries and create the appropriate call to some API. Useful to build intelligent assistants, NL interfaces to databases or information systems, etc.

LLMs are components in bigger systems



- **Function Calling**: Use LLM to process queries and create the appropriate call to some API. Useful to build intelligent assistants, NL interfaces to databases orinformation systems, etc. May be combined with RAG.

Outline

Efficiency and Optimizations

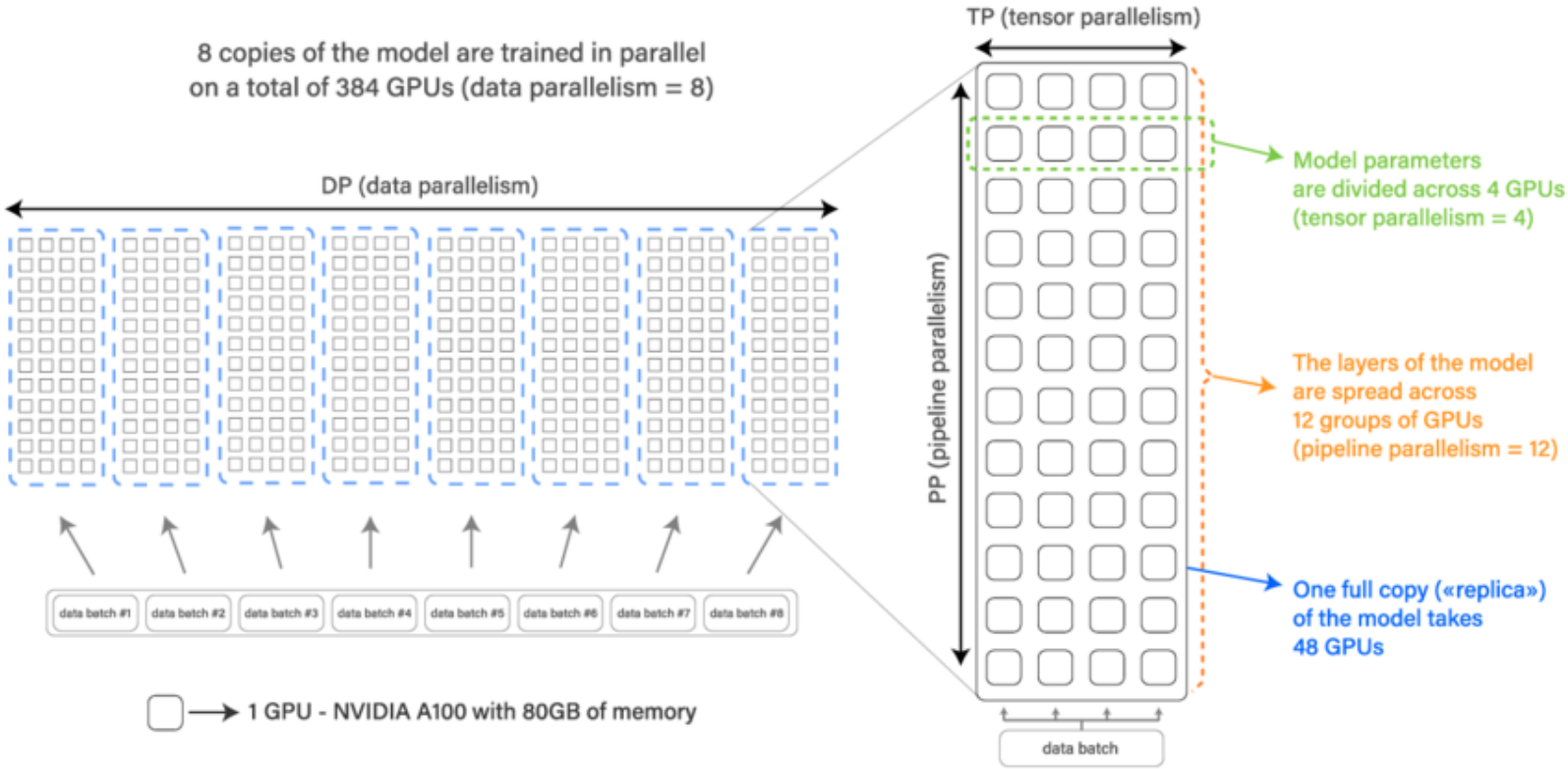
- LLMs have a huge number of parameters, and training them requires a large computational power.
- Even with the latest harwdare, sophisticated optimization techniques are required:
  - Parallelism
  - Data-type optimization
  - Low-rank adapters

Outline

3D Parallelism

- 3D parallelism is a technique for training large language models (LLM) on multiple GPUs or instances
- It combines data parallelism (DP), tensor parallelism (TP) and pipeline parallelism (PP) to address the memory efficiency and compute efficiency challenges of LLM training
  - DP replicates the model and feeds a slice of data to each replica
  - TP splits each tensor into chunks and assigns each chunk to a different device
  - PP splits the model into stages and assigns each stage to a different device

3D Parallelism



Training process of the BLOOM on 384 NVIDIA A100 80GB GPUs (48 nodes) + 32 spare gpus.
The full 175B weights model weights 329GB (2.3TB including the optimizer states).

Outline

Reduced parameter representation

LLMs use FP32 (floating point 32 bits, i.e. 4 bytes) to store their parameter values.

- 20B parameters LLM $\longrightarrow$ 20B$\times$4 bytes = 80GB of memory.
- 175B parameters LLM $\longrightarrow$ 175B$\times$4 bytes = 700GB of memory.

Powerful GPUs are required to run these models.
Even bigger to fine-tune them.

Alternative representations reduce required memory to half, storing each parameter in 16 bits

Format	Bits	Mant.	Exp.	Precision	Range
FP32 (float32)	32	23	8	high	wide
FP16 (float16)	16	10	5	middle	narrow
BF16 (bfloat16)	16	7	8	low	wide

BF16 is better for training, since it keeps the same range than FP32.
FP16 is better for inference, since it has higher precision.

Reduced parameter representation

Quantization: Even more agressive reduced representations.

Used to load trained LLMs for inference into smaller hardware

- 16-bit floating-point real numbers can be scaled to 8-bit (or even 4-bit) integers.

  (reducing model size $4\times$ or $8\times$ wrt to original FP32)

- Significant loss of precision
- However, the model keeps many of its capabilities.

Outline

Low-Rank Adapters

- Low Rank Adapters (LoRA) are a technique to fine-tune a model with low computational cost
- LoRA relies on the hypothesis that model layer weight matrices have a low "intrinsic rank", so they can be tuned modifying a reduced set of parameters instead of the whole matrix.
- LoRA allows training NN layers freezing most of their parameter values, thus reducing the computational cost.

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'
- We can compute the difference as $\Delta W = W' - W$
  Thus, $W' = W + \Delta W$

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'
- We can compute the difference as $\Delta W = W' - W$
  Thus, $W' = W + \Delta W$
- So, fine tuning a model can be seen as adding a new matrix for each layer, which will be added to the original.

$$y' = xW' = x(W + \Delta W) = xW + x\Delta W$$

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'
- We can compute the difference as $\Delta W = W' - W$
  Thus, $W' = W + \Delta W$
- So, fine tuning a model can be seen as adding a new matrix for each layer, which will be added to the original.

$$y' = xW' = x(W + \Delta W) = xW + x\Delta W$$

- That would mean duplicating the size of the model !!

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'
- We can compute the difference as $\Delta W = W' - W$
  Thus, $W' = W + \Delta W$
- So, fine tuning a model can be seen as adding a new matrix for each layer, which will be added to the original.

$$y' = xW' = x(W + \Delta W) = xW + x\Delta W$$

- That would mean duplicating the size of the model !!
  ...unless ΔW could be reduced, e.g. using a low-rank version:

$$\Delta W = U\Sigma V^T \quad \rightarrow \quad \Delta W_k = U_k \Sigma_k V_k^T = AB$$

U: $m \times m$; Σ; $m \times n$; V_T: $n \times n$
U_k: $m \times k$; Σ_k; $k \times k$; V_T: $k \times n$
A: $m \times k$, B: $k \times n$

LoRA fine-tuning

- Any layer in a NN is a weight matrix W $(n \times m)$
- When the model is fine-tuned, those matrices are modified, obtaining a new matrix W'
- We can compute the difference as $\Delta W = W' - W$
  Thus, $W' = W + \Delta W$
- So, fine tuning a model can be seen as adding a new matrix for each layer, which will be added to the original.

$$y' = xW' = x(W + \Delta W) = xW + x\Delta W$$

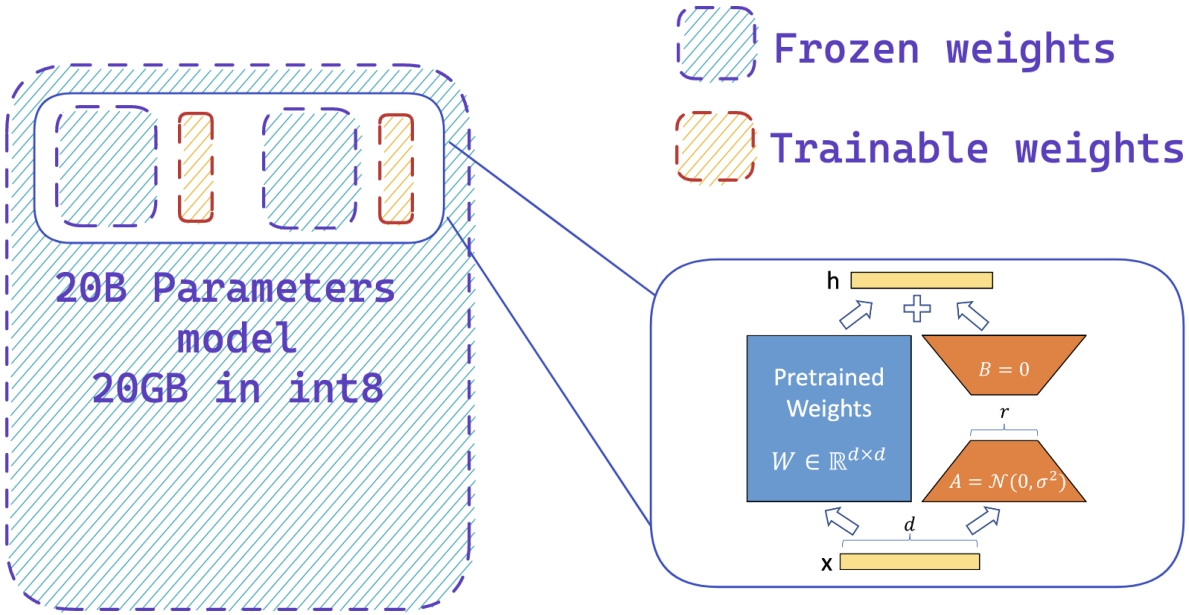
- That would mean duplicating the size of the model !!
  ...unless ΔW could be reduced, e.g. using a low-rank version:

$$\Delta W = U\Sigma V^T \quad \rightarrow \quad \Delta W_k = U_k \Sigma_k V_k^T = AB$$

U: $m \times m$; Σ; $m \times n$; V_T: $n \times n$
U_k: $m \times k$; Σ_k; $k \times k$; V_T: $k \times n$
A: $m \times k$, B: $k \times n$

- So, $\hat{y}' = xW + xAB$

LoRA fine-tuning



- Usual values for r are 4, 8, 16...
  E.g.: If ΔW is 200×200 ($40,000$ parameters), A and B are 200×8 and 8×200 ($1,600$ parameters).
- Small increase in model size (e.g. $1,600/40,000 = 4\%$)
- Much faster tuning time (e.g. only 4% of parameters to update)
- Some loss on performance of tuned model (the smaller the rank, the larger the loss)

Parameter Efficient Fine-Tuning (PEFT)

LoRA is just one among several existing PEFT methods:

- QLoRA: quantized LoRA. Same as LoRA, but using quantization
- Adapter Tuning: Small adapter layers inserted after model layers. Only adapters are tuned.
- BitFit (Bias Term Fine-Tuning): Only bias terms are tuned.

Method	**Trainable Parameters**	**Memory Usage**	**Best For**
LoRA	Low ($\sim 1\%$)	Low	General fine-tuning
QLoRA	Very Low	Very Low	Massive LLMs
Adapter Tuning	Medium	Medium	Multitask learning
BitFit	Very Low	Extremely Low	Memory-constrained fine-tuning