Advanced Methods in Text Analytics

Large Language Models

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Large Language Models (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **LLMs:** Large Language Models (vast majority are **causal LMs**, i.e. CLMs)
  - Active/trendy research topic, **many open questions/challenges**
  - **Above line:** open source; **orange cards:** instruction tuned

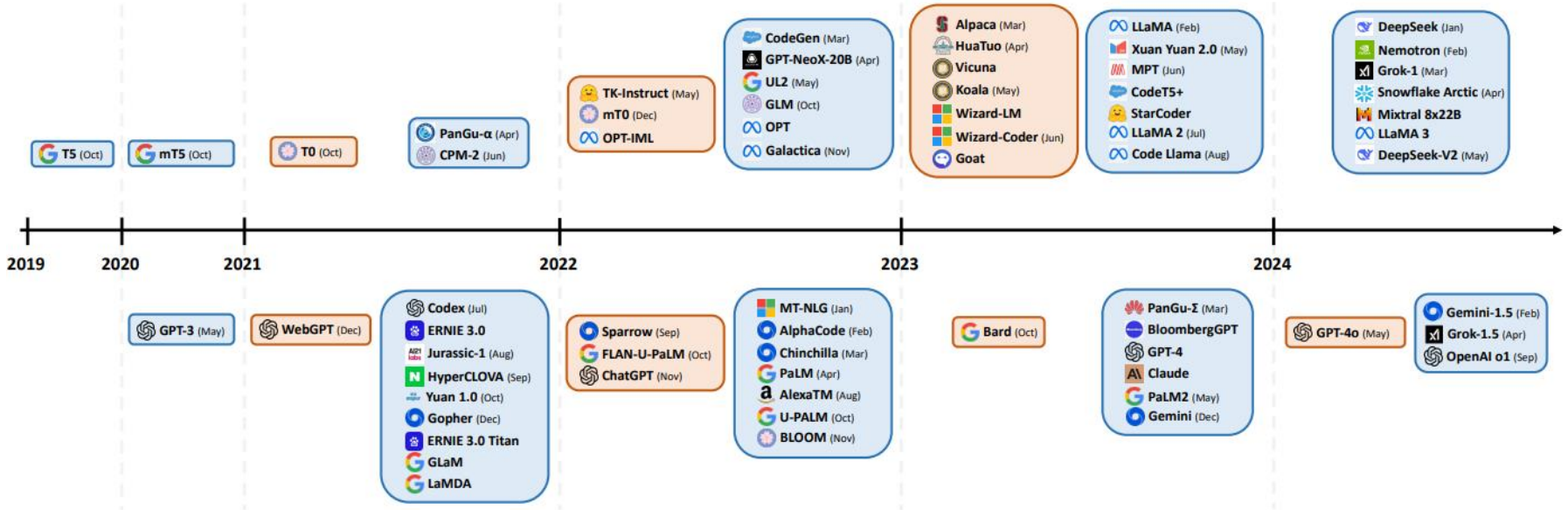


Image source

Large Language Models (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **LLMs distinguished by:**
  - (i) size --> (ii) what they can do --> (iii) how they are used
- **The way we use LMs** has changed over time, **rough history**:
  - **Statistical LMs:** count-based, e.g. *n*-gram models
  - **Neural LMs:** FNNs/RNNs, task-specific downstream data and architectures
  - **Pre-trained LMs:** transfer learning, task-specific data, general architecture
  - **Large LMs:** as PLMs but *sometimes* no task-specific data required

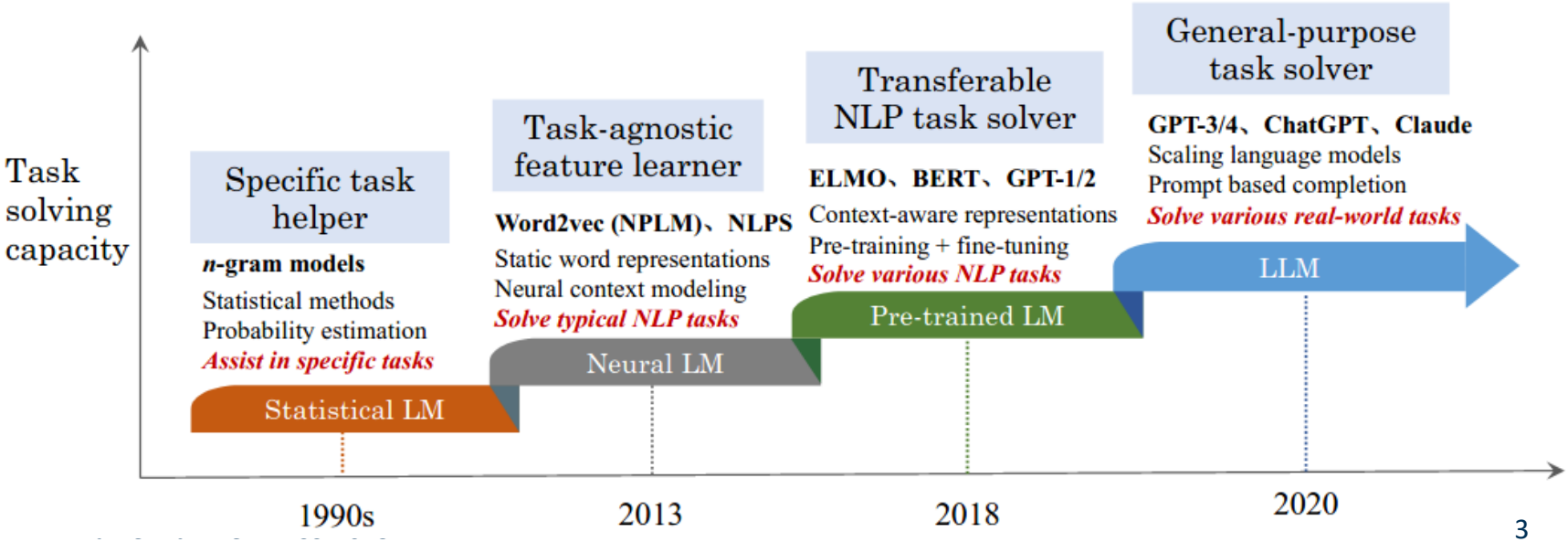


Image source

Large Language Models (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- What can LLMs do? (1) **In-context learning abilities**
  - Zero shot, or few shot tasks via in-context learning
  - Other “emergent abilities”, mostly related to text generation
- What can LLMs do? (2) **Solve *new tasks* by following instructions**
  - ***New*** = not trained or tuned for it, ***task*** = "summarize", "write a poem", etc.
  - E.g. *"summarize the following article into a single sentence"*
  - **New tuning methods** developed for this
- **Still, essentially pre-trained LMs** (can do transfer learning, PEFT, etc.)
- LLMs also come with **new challenges:**
  - Expensive to train, bias and fairness, data privacy, hallucinations
- In this lecture, **we focus on key methods/models**
  - Not a laundry list! (New models regularly, many won't stand test of time)
  - For comprehensive overview of methods, see this survey and this survey
  - **Industry:** get LLMs to perform better, apply them to real-world, sell them
  - **Academia:** study LLMs, understand LLMs

Outline

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

1. Large Language Models
    - In-Context Learning
    - Model Overview

2. LLM Tuning
    - Instruction Tuning
    - Model Alignment

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Large Language Models

In-Context Learning

Emergent Abilities

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- "An ability is emergent if it is not present in smaller models but is present in larger models." Wei et al. 2022

1. **In-context learning (ICL)** (this section)
    - Provide model with **instruction/examples**, then **model performs task**
    - Brown et al. 2020 found that larger models much better at this
2. **Instruction following** (next section)
    - Model is **tuned on multiple tasks** via natural language instructions
    - Then model **performs** well on **new unseen tasks** described similarly
    - Thoppilan et al. (2022) observed this for their model with 68B parameters, but not for the one with 8B parameters

- Such "abilities" are **task dependent**
    - They work well on some tasks, not so much on others
    - **Prompting generally brittle**, performance dependent on chosen prompts
- Note: term "ability" is vague, we care about what these mean *concretely*
    - I.e. we focus on solving NLP (i.e. text-based) tasks, not human-like abilities

Few-Shot Tasks (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Background:** concept of few-shot, one-shot, zero-shot common in ML
  - E.g. in computer vision, robotics, recommender systems, etc.
- ***n*-shot**: *n* number of training examples
  - Usually, *n* is a very small number; **few** = 5, 10, 20, etc.; **one** = 1; **zero** = none
  - Deep learning models notoriously data-hungry, *n*-shot can be challenging
  - But humans known to do *n*-shot quite well
- **Zero-shot** task: "Identify the word"
  - rnmtrfraeso =>
- **One-shot** task: "Identify the word"
  - eanlingr => learning
  - rnmtrfraeso =>
- **Few-shot** task: "Identify the word"
  - eanlingr => learning
  - rcnpisgose => processing
  - rnmtrfraeso =>

Few-Shot Tasks (2)

- **Good few-shot performance** can potentially be **very useful**!
  - Getting lots of data for training/tuning is often costly, may be impossible
  - Learning from few examples may be key for human-level intelligence
- In NLP, good few-shot performance **achieved with transfer learning**
  - Typically fine-tuning on few shots
- E.g. given pre-trained LM PLM_θ parameterized by θ, N task-specific examples (x_i, y_i) and loss function L:
  - Update θ given $L(PLM_\theta(x_1), y_1)$
  - Update θ given $L(PLM_\theta(x_2), y_2)$
  - …
  - Update θ given $L(PLM_\theta(x_N), y_N)$
- Main point: **we update parameters**
  - Catastrophic forgetting may occur

In-Context Learning (1)

- **In-context learning:** given **(i) instruction** of task *and/or* **(ii) examples** of task (often called demonstrations), language model performs the task
  - Important: **no weight updates!**
  - Not learning in the traditional sense
- Note ICL also **based on transfer learning**
  - The model in question is a *pre-trained* language model (typically a CLM)
- Example, **few-shot** setting (most common ICL setting):

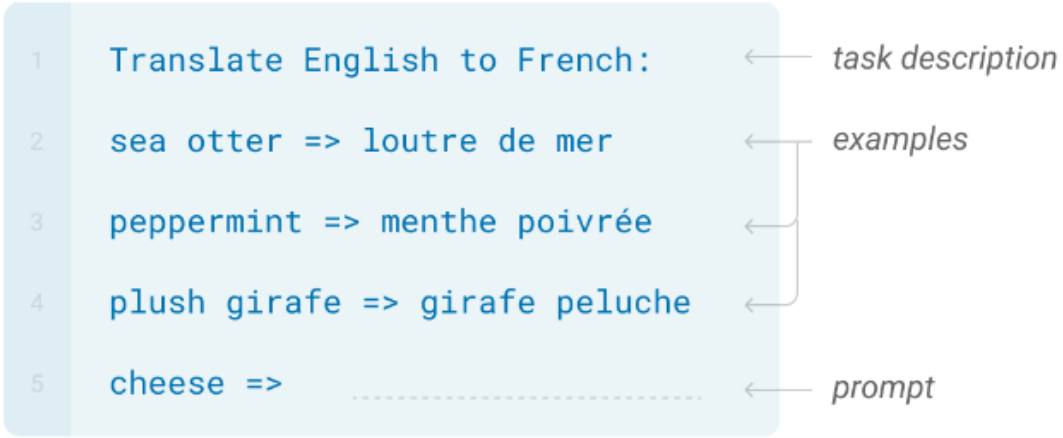


Image source

In-Context Learning (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Can also be used in **one-shot** setting:

Image source

- And **zero-shot** setting:

Image source

In-Context Learning (3)

- Can also be done **without task instructions**, just examples:
  - CLM naturally completes given prompt

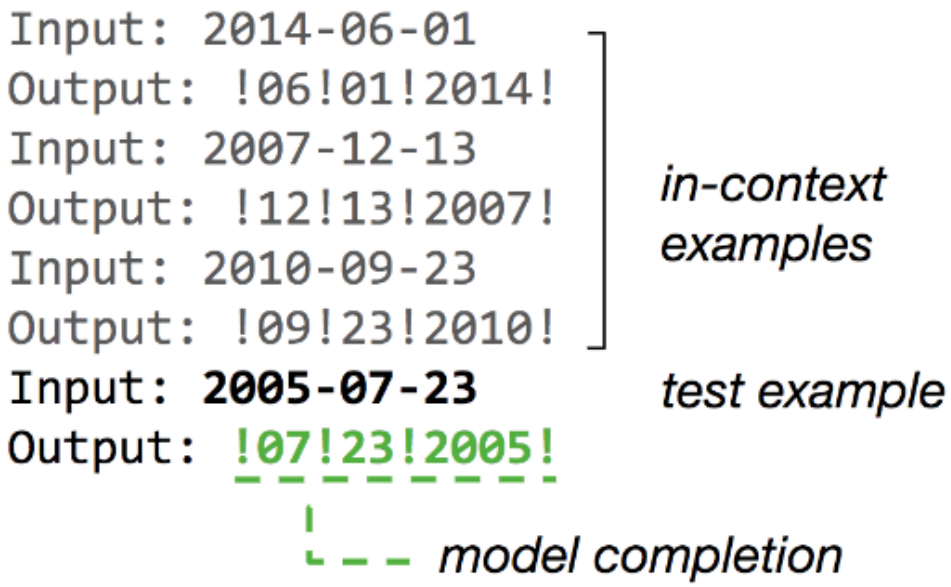
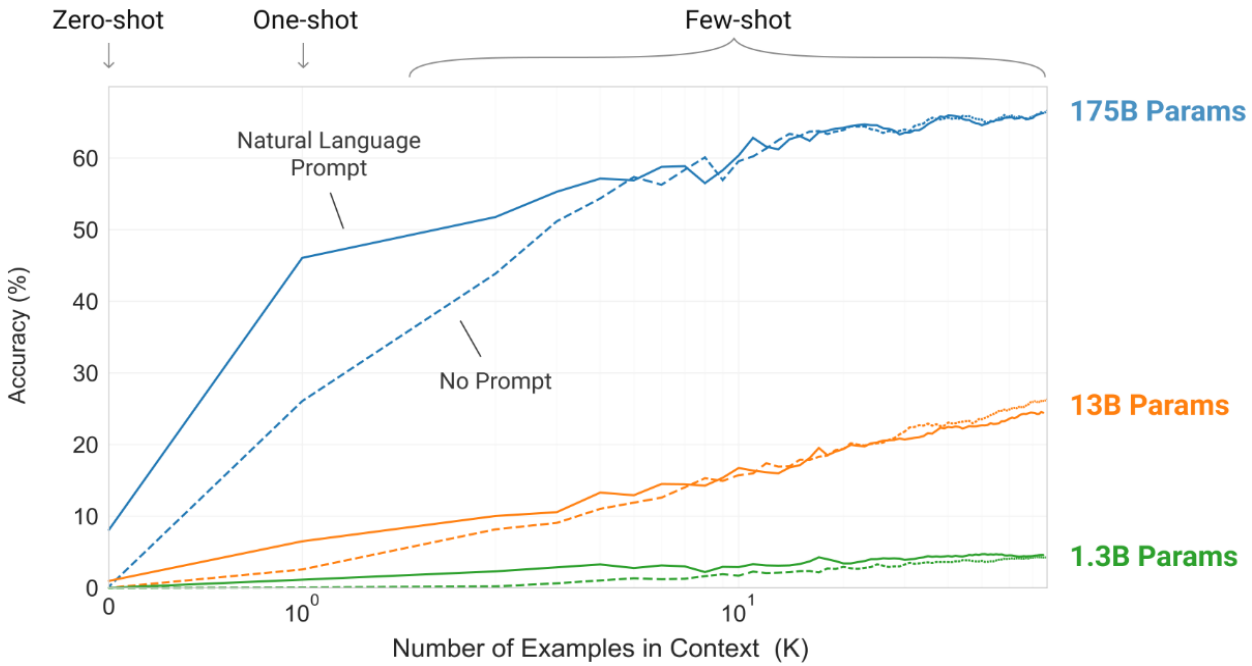


Image source

- Does this really work?
  - Yes! First work to document this well was GPT-3 (Brown et al. 2020)

In-Context Learning (4)



- Brown et al. (2020) observed (using between 10 and 100 examples):
  - Larger model -> better at ICL, largest models competitive with fine-tuning
  - Largest model made good use of task instruction in zero/one shot setting
- Later studies found ICL to be comparable to fine-tuning
- Important question: **why does this work?**

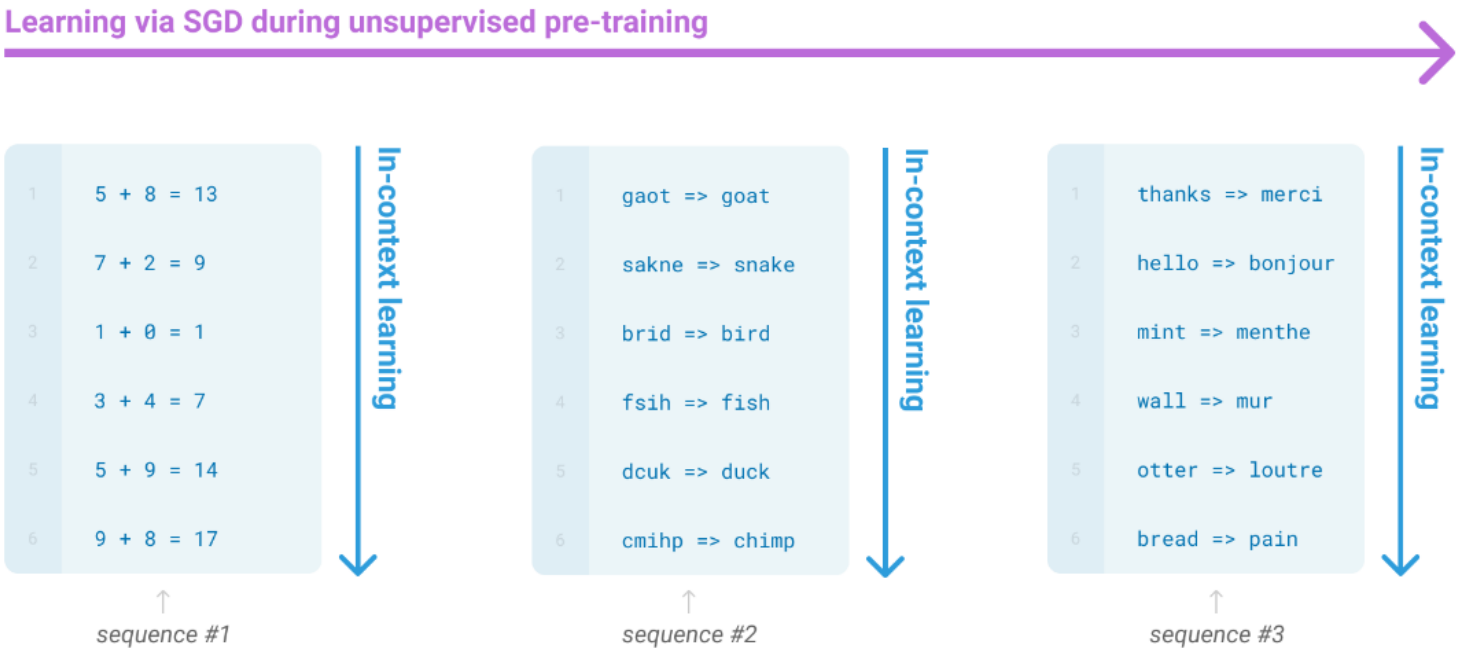
Why does ICL Work? (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- ICL *ability* reported by GPT-3 authors has since been observed multiple times on multiple models
  - Generally observed only after models reach certain size
  - But other factors do matter, e.g. data quality, model architecture
  - Through the years, smaller models get better at this
  - E.g. Llama3-8B (2024) comparable to/better than Llama2-70B (2023)
- **Why does ICL work?**
  - There are no weight updates!
  - Models not (directly) trained on these tasks, e.g. sentiment analysis
- **Open research question**, two main directions
  1. There is indeed some form of learning happening in the forward pass
     - E.g. ICL seen from a Bayesian inference perspective
     - Or ICL from a gradient-descent perspective
  2. Models don't learn the new tasks, exploit patterns seen during pre-training
     - E.g. correct labels not needed, just label distribution/space (Min et al. 2022)

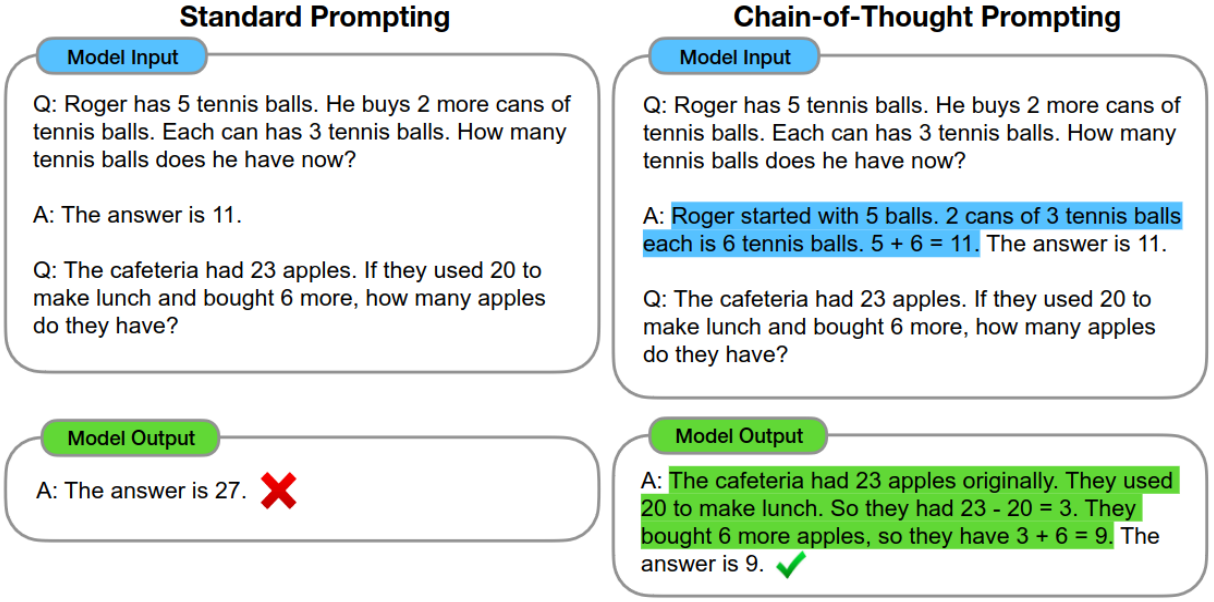
Why does ICL Work? (2)

- Authors of GPT-3 suggested models learn "skills" during pre-training
  - Then use these "pattern recognition abilities" to perform new tasks
  - They highlight that training examples may contain many such "subtasks"



Prompting (1)

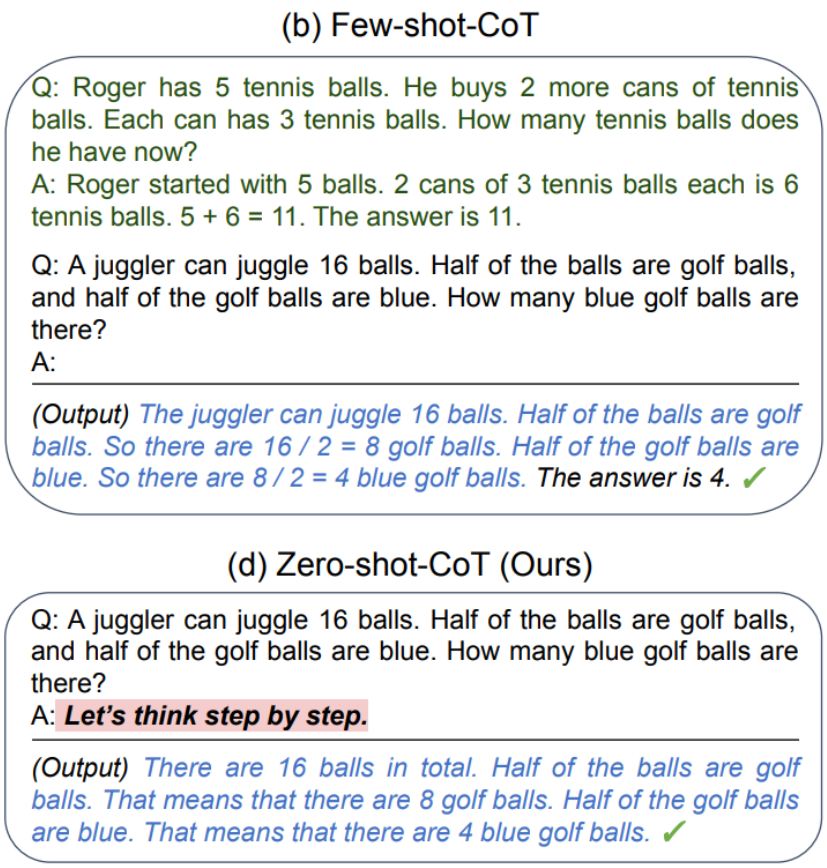
- Also observed multiple times: **ICL performance highly sensitive to prompt given to model** (e.g. see Chen et al. 2023, Lu et al. 2023)
  - E.g. order of examples is relevant, distribution of labels is relevant, etc.
  - Thus, **prompting:** finding right prompt for model to perform well on task
- Important example of impact of prompting: **chain-of-thought**
  - Examples of thinking in steps helps e.g. math problems (Wei et al. 2022)



Prompting (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Kojima et al. 2022 extended the idea of chain-of-thought by simply prompting models to **"think step by step"**
  - They tested their approach with GPT-3 and in a zero-shot setting, i.e. no demonstrations as in the top picture
  - This small addition to the prompt **significantly improved zero-shot performance**
- **Interpretation:**
  - It seems by having models also predict text that explains reasoning increases probability of tokens in correct answer
- Next: overview of some LLMs



UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

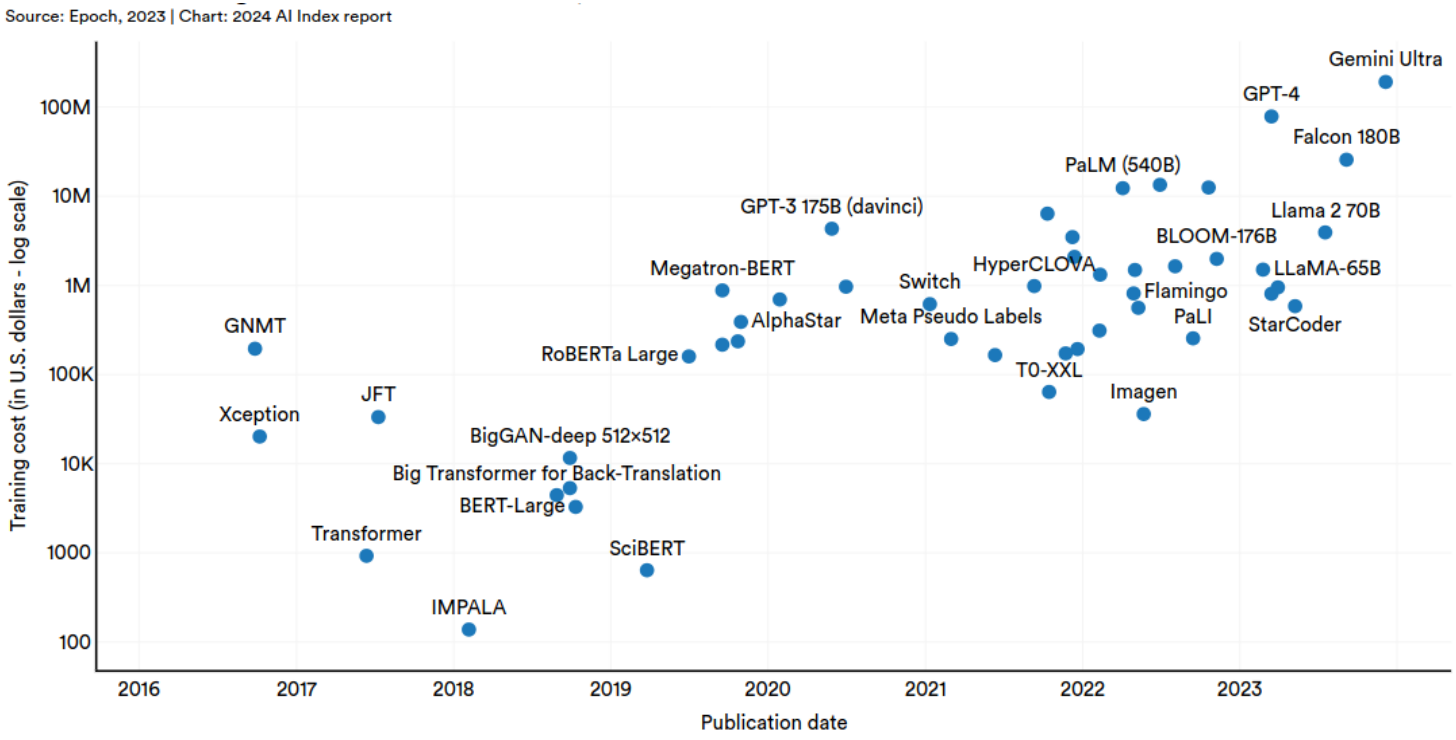
Large Language Models

Model Overview

LLMs are Expensive (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Training LLMs** has become **prohibitively expensive**
  - Example: LLaMa-2 70B (2023) trained on 2 trillion tokens (~10TB of open data), 6K GPUs, 12 days, 2M USD (source: Intro to LLMs by A. Karpathy)
- Some estimates go beyond 100M USD (source: Stanford's AI Index Report 2024)



LLMs are Expensive (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Once pre-trained, **fine-tuning is cheaper** (computationally)
  - Computationally much more feasible, e.g. with PEFT, quantization
- But **getting data can be challenging**
  - **Data quality matters** more than quantity at this stage
  - Data **more reflective of relevant** downstream **tasks**, e.g. chatbots
- **Also, fine-tuning** is **iterative** process
  - Observe problems, create data for adjusting, fine-tune, repeat
  - I.e. costs can be continuous
  - More in next section
- Let's have a look at some models
  - But first, a quick overview of architectures and training objectives

Architectures

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- All architectures are transformer-based
- **Causal LM, decoder-only (most common by far)**
  - **Decoder** autoregressively predicts new tokens conditioned only on output of decoder
  - E.g. GPT series (closed source, OpenAI), LlaMA series (open source, MetaAI)
- **Encoder/decoder (less adopted in recent years)**
  - **Encoder** produces sequence of contextualized tokens
  - **Decoder** autoregressively predicts new tokens conditioned on output of decoder *and* encoder
  - **Cross-attention:** attends to output tokens from encoder
  - Example model: T5, UL2
- Other less common architectures (for LLMs) exist
  - Encoder-only
  - Non-causal decoder only (bidirectional attention across *past* tokens)

Training Objectives

- **(Full) Autoregressive/Causal** language modeling (CLM)
  - Most common approach, highly adopted since GPT-2 (2019)
  - Causal masking on (self-)attention scores (***KQ***) matrix
- **Masked** language modeling: fill-in-the-blank task (less adopted)
- **Prefix** language modeling (less adopted)
  - **Prefix:** fixed subsequence of first *n* tokens
  - **Task:** predict tokens after prefix
  - Difference with CLMs: attention *bidirectional* over prefix token

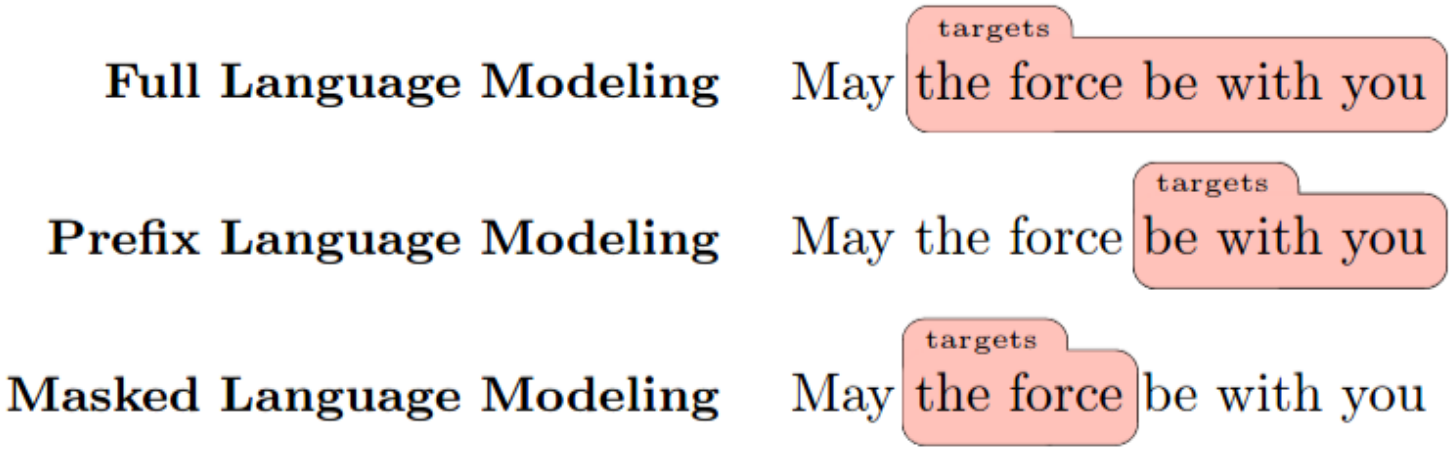


Image source

The GPT Series (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Proprietary** models! **Closed source**, not all information is available
- We've already discussed the **GPT architecture** in a previous lecture
  - Causal language modeling, decoder-only
- In terms of **number of parameters**
  - **GPT-1:** 117M
  - **GPT-2:** 1.5B
  - **GPT-3:** 175B
- In terms of decoder **layers**
  - **GPT-1:** 12 layers of size 768
  - **GPT-2:** 48 layers of size 1600
  - **GPT-3:** 96 layers of size 12228
- **GPT-2 vs GPT-1**
  - Small changes in architecture (discussed in Transfer Learning lecture)
  - Performed well on zero-shot tasks, but performance below fine-tuning

The GPT Series (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **GPT-3 vs GPT-2**
  - Leap in few-shot performance via **in-context learning (ICL)**
  - Inspired to continue scaling up models

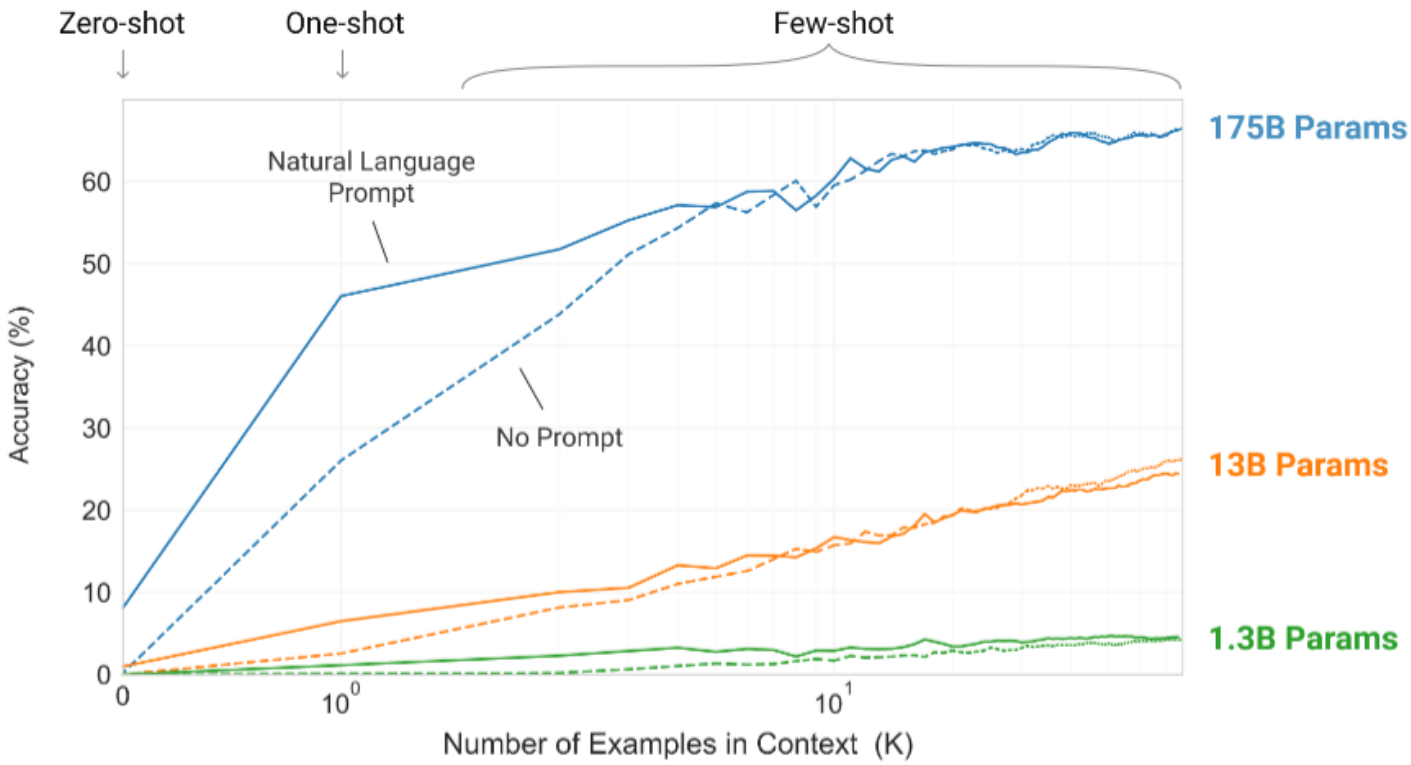


Image source

The GPT Series (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Codex**
  - GPT-3 fine-tuned on large corpus of GitHub code
  - Able to solve coding problems, improved performance on math problems
- **GPT-3.5 models**
  - GPT-3-based models with stronger abilities
  - Abilities -> not benchmarked tasks, evaluated by humans
  - How did they get better? **Model alignment** (more in next section)
- **ChatGPT:** fine-tuned for dialogue (ability: communicate with humans)
- **GPT-4**
  - Superior performance on human-generated tests compared to ChatGPT
- Still **many challenges**
  - Mainly, **hallucinations**: false information presented as fact
  - Perhaps a reminder: these are still *language models*
    - LLMs can predict facts *sometimes*, i.e. may be used as *unreliable* databases
    - But no internal "stop-and-think" mechanism (on-going research, e.g. here, here)

The LLaMA Series (1)

- **Highly valuable contribution!** "LlaMA is the new BERT"
  - Being open source, it sparked large number of research/applications

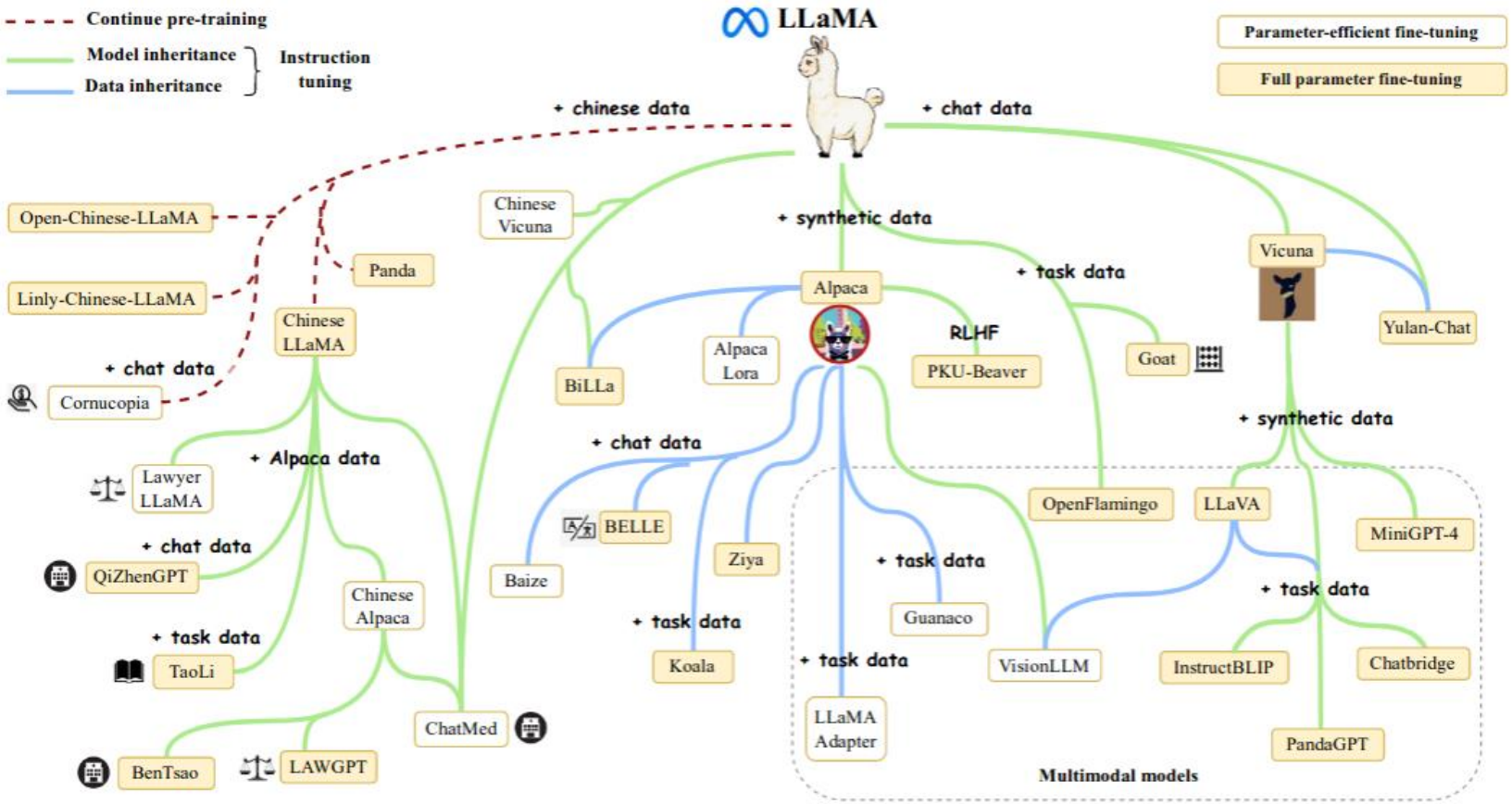


Image source

The LLaMA Series (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Important **open source** LLM developed by META AI
- LLaMA-1 (2023)
  - Decoder-only causal LM
  - Several sizes: 7B to 65B parameters
  - Largest model has 80 layers of size 8192
  - Trained only on publicly available data
  - Competitive with best LLMs
  - All weights (checkpoints), code made available
- LLaMA-2 (2023)
  - Focused on fine-tuning for dialogue applications
  - Same architecture as LLaMA-1, but larger model at 70B parameters
  - Trained on 40% more data than LLaMA-1
  - Released all pre-trained and fine-tuned models

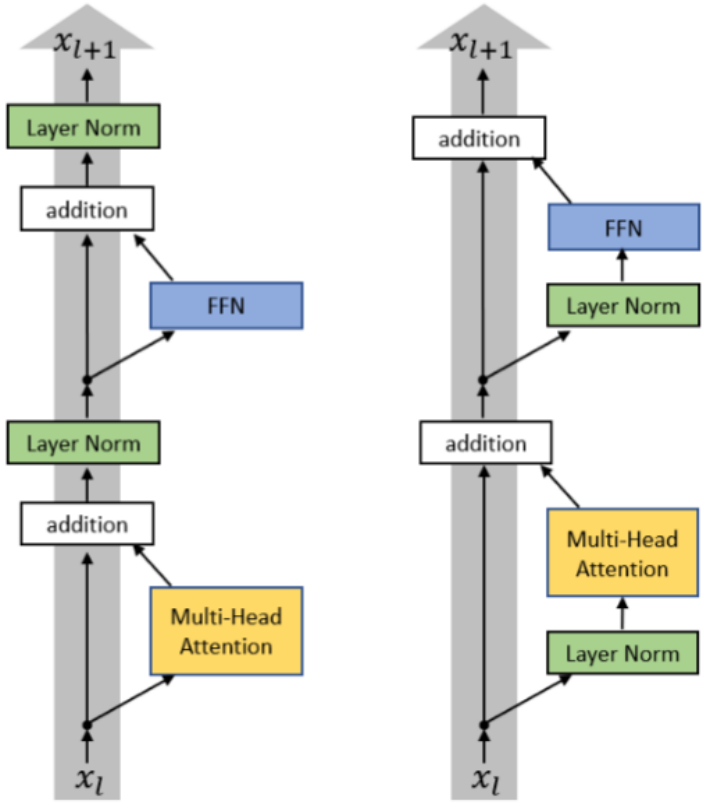
The LLaMA Series (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- LlaMA-3 (2024)
  - Improved quality and size of training data (15T vs 1.8T for Llama-2)
  - Sizes from 8B to 405B parameters (the largest model is dense, more soon)
  - Hidden representations size ranges from 4096 to 8192
  - Number of attention heads ranges from 32 to 128
  - Vocabulary size: 128.000
- Llama-4 (2025)
  - No technical report released yet.
  - Sizes from 109B to 2T parameters
  - But all models now sparse, i.e. relying on mixture-of-experts (more soon)
- Interesting "mid-season" release: Llama-3.2 (2024)
  - Small 1B and 3B models achieved via model distillation
- In the following, we *briefly* discuss some common LLM components:
  - PreNorm vs PostNorm models, RMSNorm, SwiGLU
  - Model distillation, Mixture-of-Experts (MoE)

PreNorm vs PostNorm

- Following prior work, Xiong et al. (2020) argued that using **LayerNorm** differently addresses gradient-based issues during training
  - They proposed to move the LayerNorm component from after each self-attention and MLP layer (image on left) to inside (and before) each self-attention and MLP layer (image on right)
  - This variant is commonly used by some LLMs, e.g the Llama family
- But Llama models do not use LayerNorm
  - They use a variant: **RMSNorm**



RMSNorm

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Recall **LayerNorm**: İt's a form of centering data
  - **Goal:** address internal distribution shifts to make training more stable
  - **How:** given output vector **a** for some hidden layer, compute its mean μ and standard deviation σ
  - Then **center a'** = (**a** - μ) / σ
  - Finally, **scale** the data: *LayerNorm* = γ***a'*** + β with γ**,** β being learned
- Zhang et al. (2019) argued that only the scaling component is necessary
  - They proposed simpler variant: not using mean centering
  - Result was scaling and normalization based on root-mean-square (RMS)

$$\bar{a}_i = \frac{a_i}{\mathrm{RMS}(\mathbf{a})} g_i, \quad \text{where } \mathrm{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}$$

- They showed experimentally:
  - Using RMSNorm gave similar or more effective performance

SwiGLU Activations

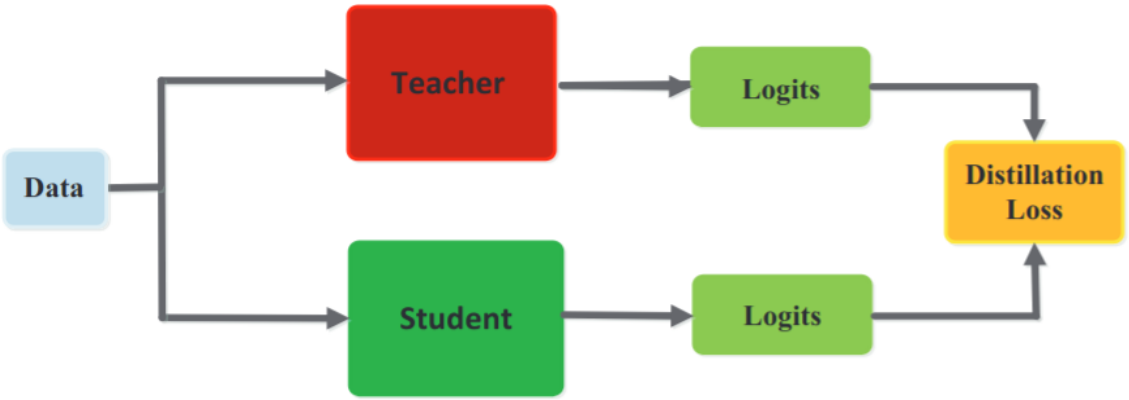
UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Combination of two forms of activations: Swish and GLU
- Swish activations:
  - $Swish(\boldsymbol{x}, \beta) = \boldsymbol{x} \cdot \sigma(\beta x)$ where $\boldsymbol{x}$ is input vector, σ is sigmoid, β is learned
  - Roughly: smooth, non-monotonic version of ReLU
- GLU activations:
  - $GLU(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{c}) = \sigma(\boldsymbol{xW} + \boldsymbol{b}) \otimes (\boldsymbol{xV} + \boldsymbol{c})$ where $\boldsymbol{W}, \boldsymbol{V}$ are linear projections, $\boldsymbol{b}$ and $\boldsymbol{c}$ their biases, $\otimes$ is element-wise product
  - Note this means **some LLMs have three projections in MLP layers**
  - Here, $\boldsymbol{W}$ is typical up projection, $\boldsymbol{V}$ is 2nd *new* up projection
- SwiGLU activations:
  - $SwiGLU(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{c}, \beta) = Swish_{\beta}(\boldsymbol{xW} + \boldsymbol{b}) \otimes (\boldsymbol{xV} + \boldsymbol{c})$
  - Found to significantly increase performance compared to ReLU and Swish activations

Model Distillation (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Proposed by Hinton et al. (2015)
  - Often referred to as *knowledge distillation*
- Motivation: models are functions
  - Have function by **smaller model (student)** behave as function by **larger model (teacher)**
- How?
  - Hinton proposed using **teacher's (token) distribution as target for student**
  - Model distribution is richer than final predictions, especially with higher temperature, which they suggested should be increased during training



Model Distillation (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Common choices for distillation loss:
  - KL-Divergence (distance between distributions)
  - Mean squared error (MSE)
  - Combination of the two, or with others
- Distillation loss often linearly combined with CE loss for LM
  - DistilBERT also used cosine embedding loss, aligns teacher/student vectors
  - Can also include targets from multiple teachers (multi-signal distillation)
- Distillation is quite popular today
  - As mentioned, the Llama-3.2 family has two small models (1B and 3B) distilled from larger models Llama-3.1 (8B and 70B)
  - Companies like DeepSeek provided smaller models in their family, students were models like Llama or Qwen, teacher was their flagship model DeepSeek-R1

Sparse vs Dense Models

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Dense model:** uses all its weights for any given input at inference time
  - I.e. the type of architecture used by all models discussed so far
- **Sparse model:** only uses subset of its weights for any given input at inference time
- Popular sparse architecture in LLMs: **mixture-of-experts (MoE)**
  - Scaling model parameters less expensive if model is sparse
  - Some MoE LLMs: the Mixtral family, DeepSeek-V3 (671B parameters)
- In the following slides, we go over the basics of MoE

Mixture of Experts (1)

- Popular architecture for deep learning models
  - Above arrow: open source, below arrow: closed source

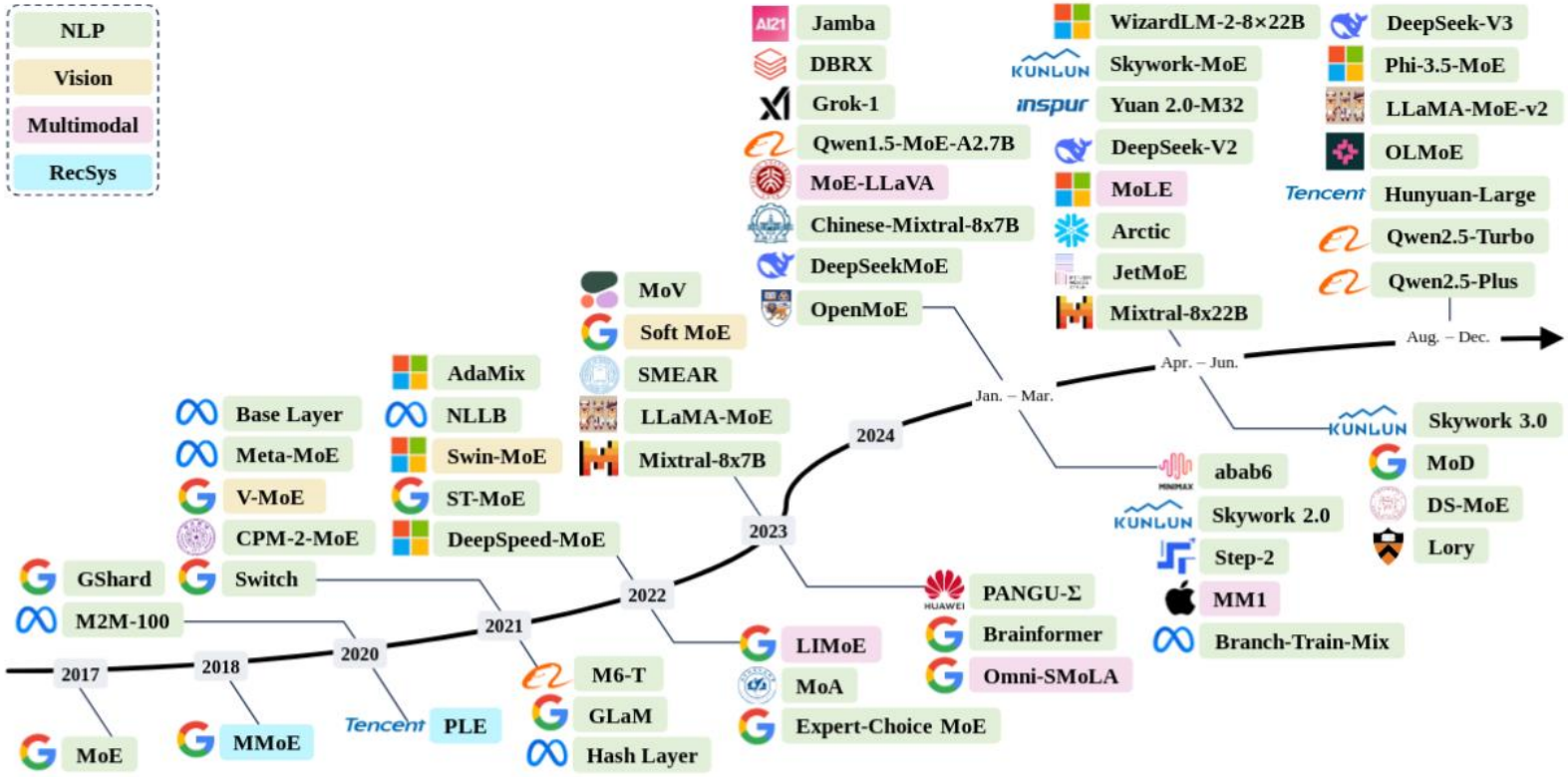


Image source

Mixture of Experts (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Part of the ML toolbox for many decades already
- **Basic intuition** of MoE:
  - Use different "parts" of the model for different inputs given to the model
  - A "part" is typically a component or subnetwork within the model
  - Thus, **expert:** subnetwork specialized to handle some type of input
- Concretely, a **MoE layer** consists of the following **components**:
  - **Set of expert networks** E = $\{E_1, E_2, \ldots, E_n\}$
  - **Gating/routing function** $G: T \dashrightarrow E^k$ designed to choose set of $k < n$ experts for each portion of training data T
  - **Routing strategy** that determines how data is distributed among experts
- **Output of MoE layer:** linear combination of experts' output weighted by G
- How could we implement function G?

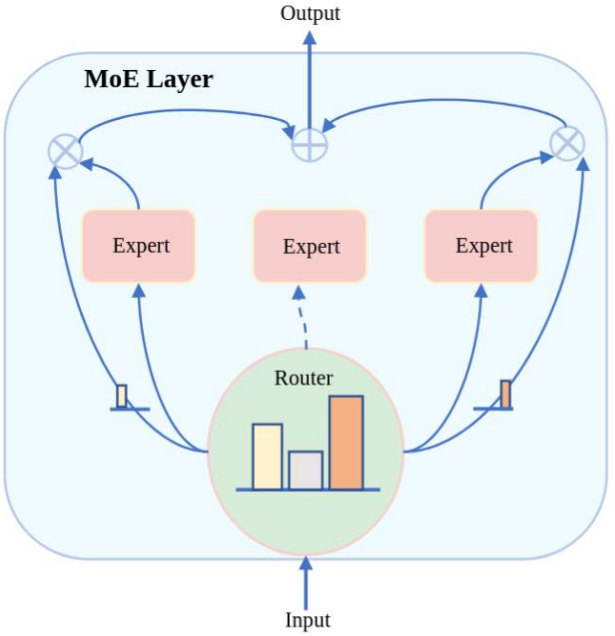


Image source

Mixture of Experts (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Common choice for routing function *G*:
  - $G(\boldsymbol{x}) = Softmax(\boldsymbol{W}_G\boldsymbol{x})$ where $\boldsymbol{x}$ is typically same input given to experts
  - Output of $G(\boldsymbol{x})$ typically referred to as routing scores
- Interpretation of $G(\boldsymbol{x})$: it acts as a "probabilistic switch"
  - Given input $\boldsymbol{x}$, $G(\boldsymbol{x})_j$ gives probability of selecting *j-th* expert
  - Thus, $\boldsymbol{W}_G \in R^{d \times n}$ where *d* is size of $\boldsymbol{x}$, *n* is number of experts
- In modern NLP, Shazeer et al. (2017) used MoE in a two-layered stacked RNN (with LSTMs) for machine translation
  - They added a MoE layer between RNN layers (input to *G*, experts: tokens)
  - **Each expert** was a **linear layer** (they used up to thousands of experts)
  - $G(\boldsymbol{x})$ was as above but they added a sparsity constraint
  - **Sparsity:** *top-k*$(\boldsymbol{x})$, i.e. only *k* experts used for each input for efficiency
- **Output of MoE layer:** $y = \sum_n G(\boldsymbol{x})_i E_i(\boldsymbol{x})$ where $G(\boldsymbol{x}) = Softmax(top\text{-}k(\boldsymbol{W}_G\boldsymbol{x}))$
- **Routing strategy:** level of specialization (token-level, task-level, etc.)
  - E.g. select *k* experts per token, use per-task experts with unlearned routing

Training MoE

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Routing examples** to different experts **can cause training issues**
- Goal of routing: select which of *n* experts to send each input to
  - **Discrete decision**, can be **problematic for gradient-based learning**
- Shazeer et al. used the "probabilistic switch"
  - Specifically, selecting **top-*k*** experts
  - Their implementation of *top-k* did not cause problems during training
  - Other strategies exist, e.g. based on hashing, reinforcement learning
- Another potential issue: **load balancing**
  - Shazeer et al., as well as prior work, noted **routing converges to favoring few experts**
  - Thus, **vicious cycle**: few experts get chosen the most, they see more data, get better at the task
  - Often **addressed with auxiliary loss function** that **encourages that each expert sees an equal amount of tokens** during training
- Same idea was applied a few years later to the transformer architecture

MoE in Transformers

- Proposed by Lepikhin et al. (2021) and Fedus et al. (2022)

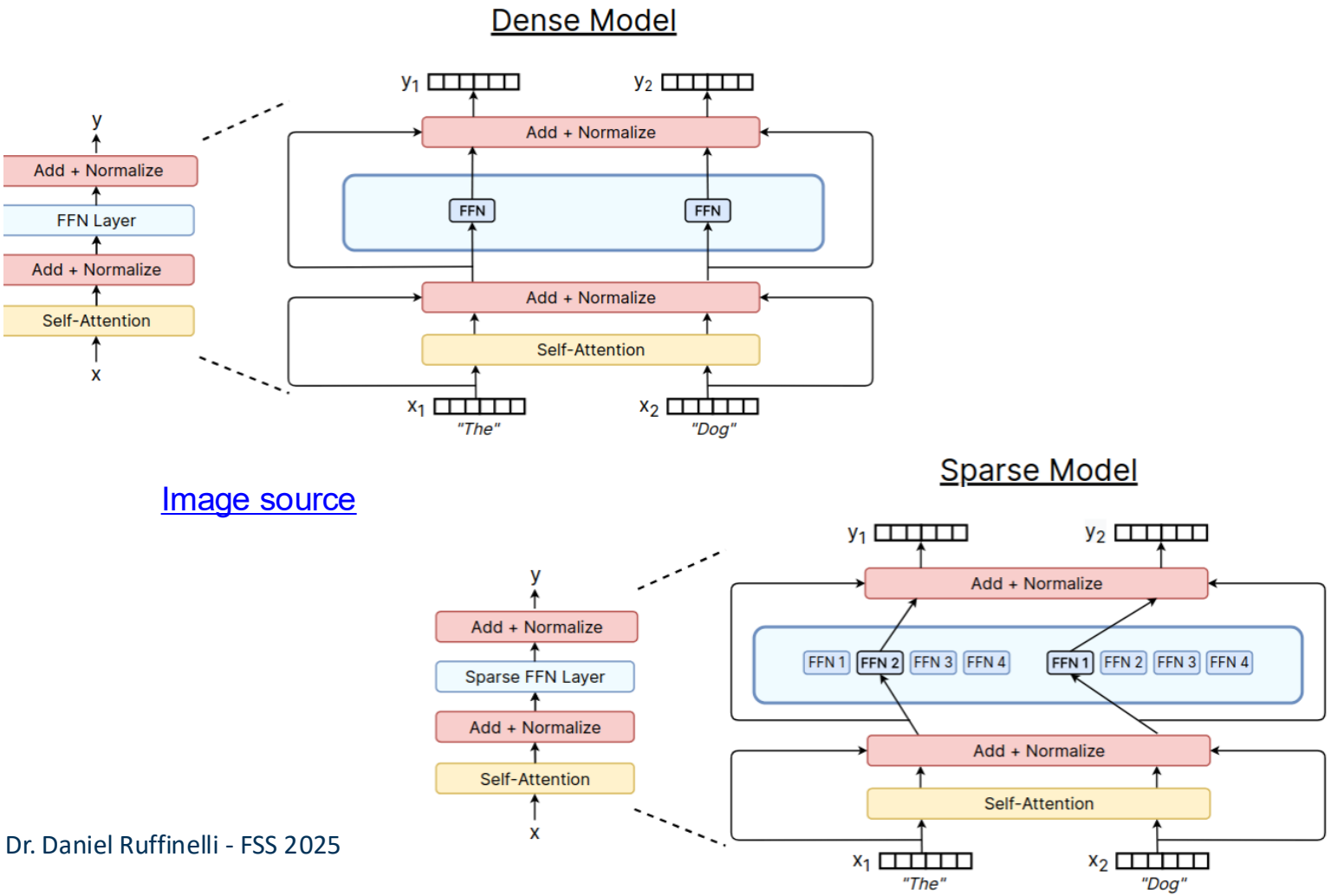


Image source

MoE in LLMs

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Mixtral** models mostly followed prior work
  - As shown in previous slide, i.e. Shazeer et al. with transformers
  - Also, softmax-based routing function, top-*k* routing strategy
  - They used SwiGLU as experts (LLM-era activations)
- **DeepSeek** models introduced **new ideas**
  - Some *shared experts* always on to capture constantly needed knowledge
  - Split internal (large) dimension in MLP layers into two separate up-projections to allow more *fine-grained experts*
- Note: **term "expert" can be misleading, depends on routing strategy**
  - Resulting experts not necessarily "experts" in math, translation, etc.
  - Routing strategy can, but often doesn't, make such decisions
  - Instead, model is allowed to organize experts as needed during training
  - It's more akin to using multiple self-attention heads
  - Result can be experts in subtasks, e.g. a "copy-paste" attention head
- For more on MoE, see here or here.

Context Window (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Something not discussed so far: **size of context window**
  - **Expensive due to quadratic cost of attention**
- Constant effort to increase size with every new model

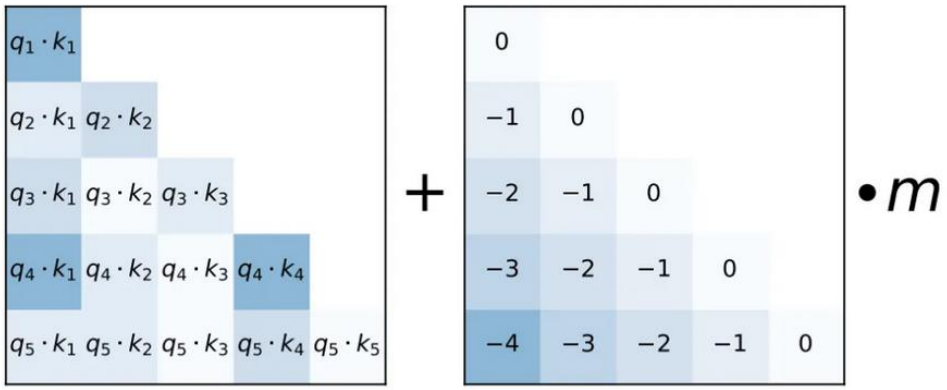
Model	Context length	Number of English pages*
GPT 3.5	4,096	6
GPT 4	8,192	12
GPT 4-32k	32,768	49
Llama 1	2,048	3
Llama 2	4,096	6

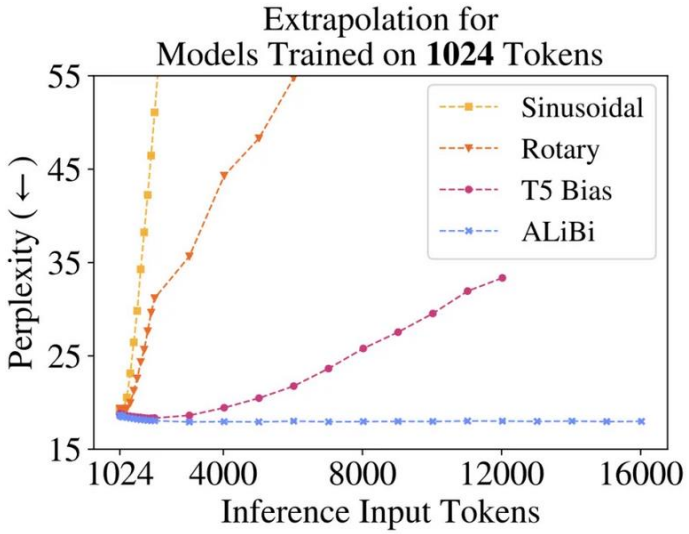
Context length comparison. (*Assuming 500 words per page.)

Image source

Context Window (2)

- **Several methods developed to increase context length**
  - Different types of methods: e.g. positional embeddings, more efficient types of attention, new model architectures, we discuss a few below
- **ALiBi** positional embeddings (Press et al. 2022)
  - **Position encoded as linear term added to attention score**
  - Allows training with shorter sequence, use longer sequence for inference
  - Requires using such positional embeddings during pre-training

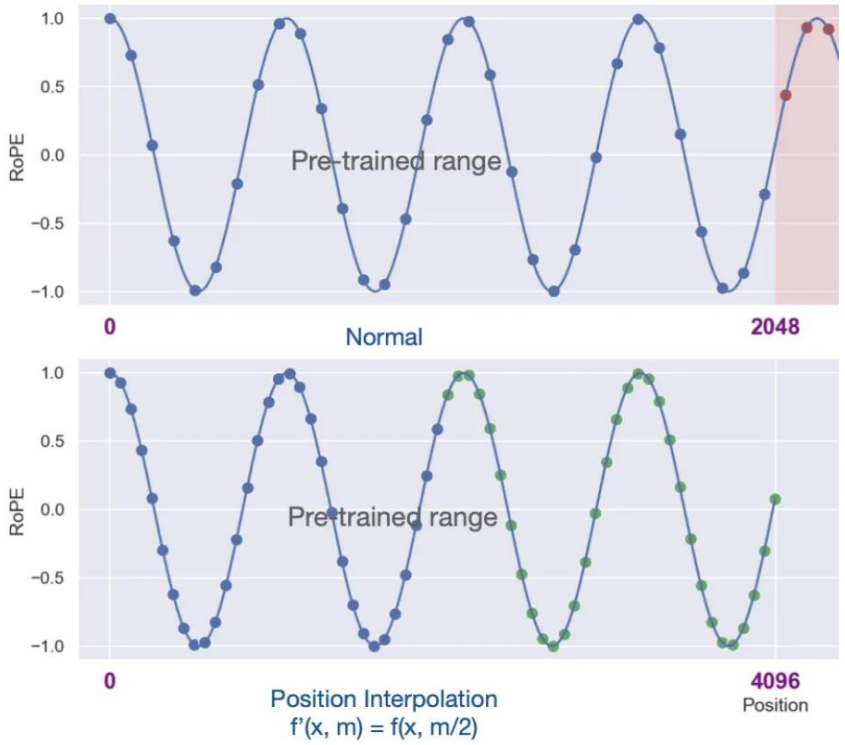




Context Window (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Position Interpolation** (Chen et al. 2023)
  - Extend position embeddings beyond size used during training
  - With a bit of fine-tuning, applicable to models that used different positional embedding during pre-training



Context Window (4)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- More **efficient forms of attention**:
  - Sparse attention: consider only subset of attention matrix, e.g. locality
  - Flash attention: GPU optimization of forward/backward pass operations
  - More in later tutorials
- More recently, **RNN-like methods**:
  - Mamba Architecture (Dao et al. 2023): state-based model allows longer input sequences similar to how RNNs did (see also Le Bronnec et al. 2024)
  - TransformerFAM (Hwang et al. 2024): introduce LSTM-like mechanism in transformers
  - Leave-No-Context-Behind (Munkhdalai et al. 2024)
- **RNN-like methods too new, impact unclear**
  - Highlights this challenge is an open-question
- Parameter count/context window show **scaling up big part of progress**
  - Why?

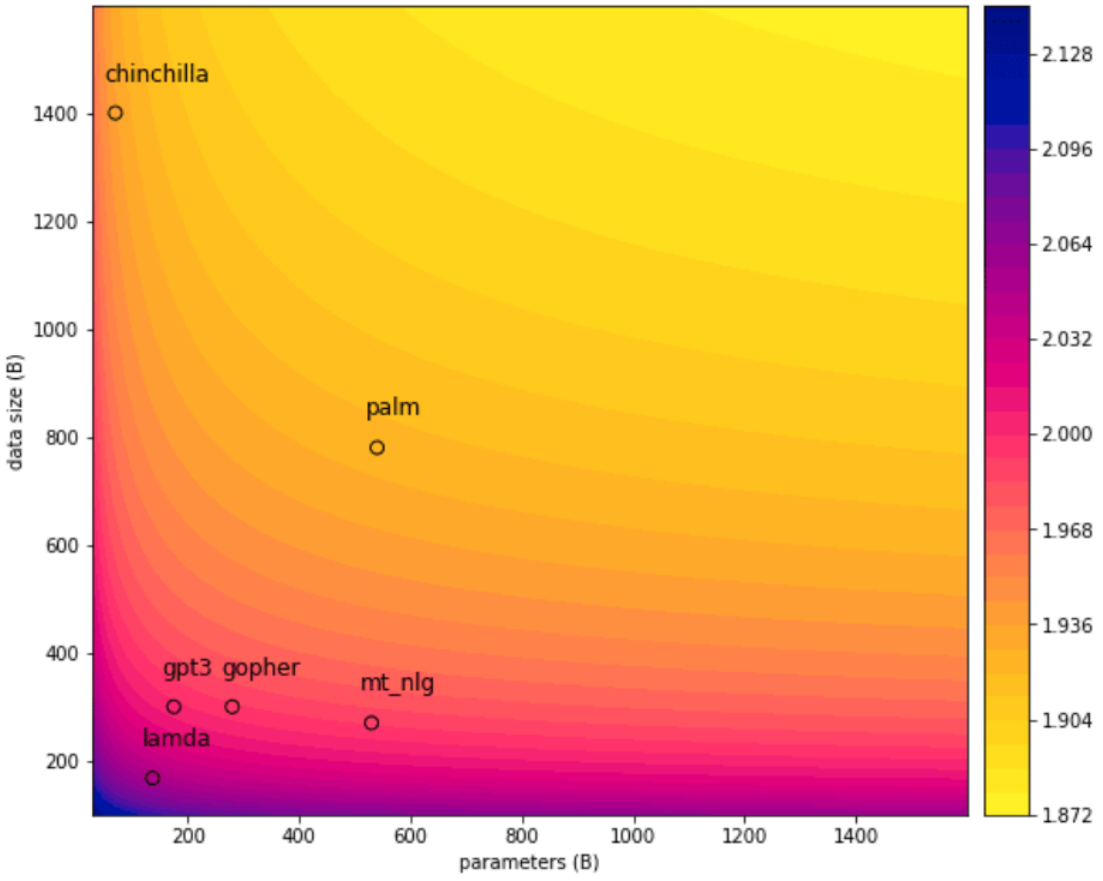
Scaling Laws (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Chinchilla scaling laws** introduced by Hoffmann et al. (2022)
- Modeled **cross-entropy loss as function of model size and data size**
  - Loss measured in *nats*, i.e. like bits but using *ln* (explained in past tutorial)
  - Model size measured in number of parameters
  - Data size measured in number of tokens in training
- Specifically: $L(N,D) = E + A/N^{\alpha} + B/D^{\beta}$
  - ***N:*** model size,
  - ***D:*** data size
  - ***E***: ideal loss from infinitely big model with infinite training data
  - A/N^{α}: represents how much model underperforms ***E*** based on its size
  - B/D^{β}: how much model underperforms ***E*** based on its training data size
- They optimized ***L*** under constraint $C = 6ND$ where ***C*** is compute power
  - They fitted **A** and **B** using data from lots of pre-training experiments
  - Different model sizes (70M to 16B), different data sizes (5B to 500B)

Scaling Laws (2)

- They derived the following **optimal sizes of model and data**
  - **N(C)** = $0.6C^{0.45}$
  - **D(C)** = $0.3C^{0.55}$
- Heatmap of loss w.r.t. model and data size
- They trained their Chinchilla model on more data, as suggested by the law
- It outperforms many models, even those with many more parameters



Scaling Laws (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- This tells us, **for given compute power C** (in flops), **how much data to use to achieve optimal loss for a given model size**
  - E.g. for 67B parameters, use 1.7 trillion tokens
  - GPT-3 trained on 300B tokens, should have had 15B parameters, not 175B
  - With that data size, growing model beyond 15B makes little difference
- Law makes **assumptions**
  - E.g. applies to CLMs using a specific form of learning rate scheduler
- Such laws **nevertheless useful**
  - E.g. test training tricks on smaller model, see how performance would scale
- When plotted, **scaling laws predict improvement with scale!**
  - **Limitation:** focused on loss reduction, may not translate to actual tasks
- For nice explanations on the topic, see here and here.

Scaling Laws (4)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Recently, these scaling laws have been under question
  - The main focus of criticism is their oversimplicity
- Diaz et al. (2024) argue that despite scaling up training sets, performance on more specific settings don't scale as described
  - They argue for considering other factors besides model and data size
  - E.g. evaluation data size, qualitative and use-case dependent evaluation metrics
- Isik et al. (2025) study scaling laws by focusing on machine translation
  - I.e. target is not training loss, but performance on translation task
  - They find that size of downstream data, as well as alignment of distribution of pre-training and downstream data are also important factors
  - Thus, scaling laws work differently in this case
- Liu et al. (2025) study how model performance can be predicted on downstream tasks
  - They find impactful factors like model architecture and dataset distribution

Pre-Training vs Fine-Tuning

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Many more models
  - And **many more details** about each model
  - E.g. size of training set, ensemble or not (i.e. sparse vs dense), etc.
- Most **proprietary LLMs** are accessed via **APIs**
  - Useful for research and applications
  - But research challenging, key information unknown
  - E.g. are benchmark test sets seen during training?
- Note a **pattern in latest LLMs**
  - **GPT-3.5:** fine-tuned GPT-3
  - **ChatGPT:** fine-tuned for dialogue
  - **LLaMA-2:** focused on fine-tuning
- **Fine-tuning methods essential** to latest LLM success
  - At least as measured by language understanding, safer use in applications
  - NLU = natural language understanding (i.e. reading comprehension)
- Let's go over the basics of fine-tuning LLMs!

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

LLM Tuning

LLMs as Useful Tools (1)

- Predicting next word is not the same as “follow user’s instructions”

PROMPT	*Explain the moon landing to a 6 year old in a few sentences.*
COMPLETION	GPT-3
	`Explain the theory of gravity to a 6 year old.`
	`Explain the theory of relativity to a 6 year old in a few sentences.`
	`Explain the big bang theory to a 6 year old.`
	`Explain evolution to a 6 year old.`

Image source

LLMs as Useful Tools (2)

- **Fine-tuning crucial** to LLMs ability to communicate, solve tasks

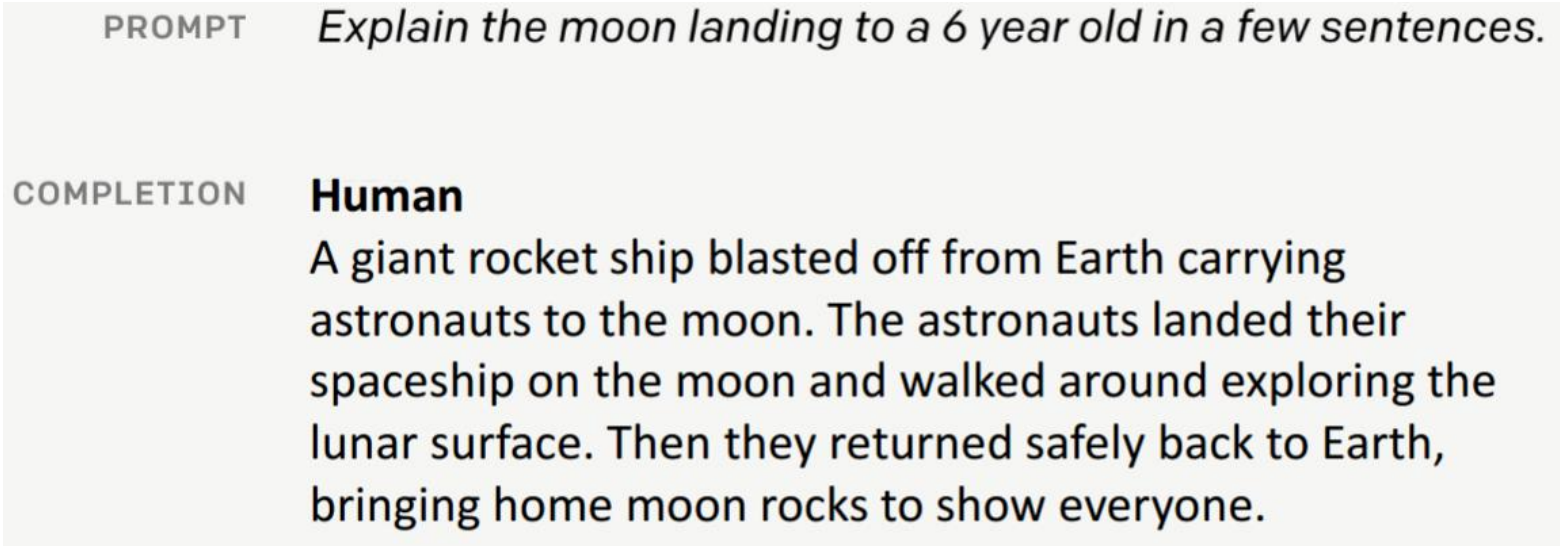


Image source

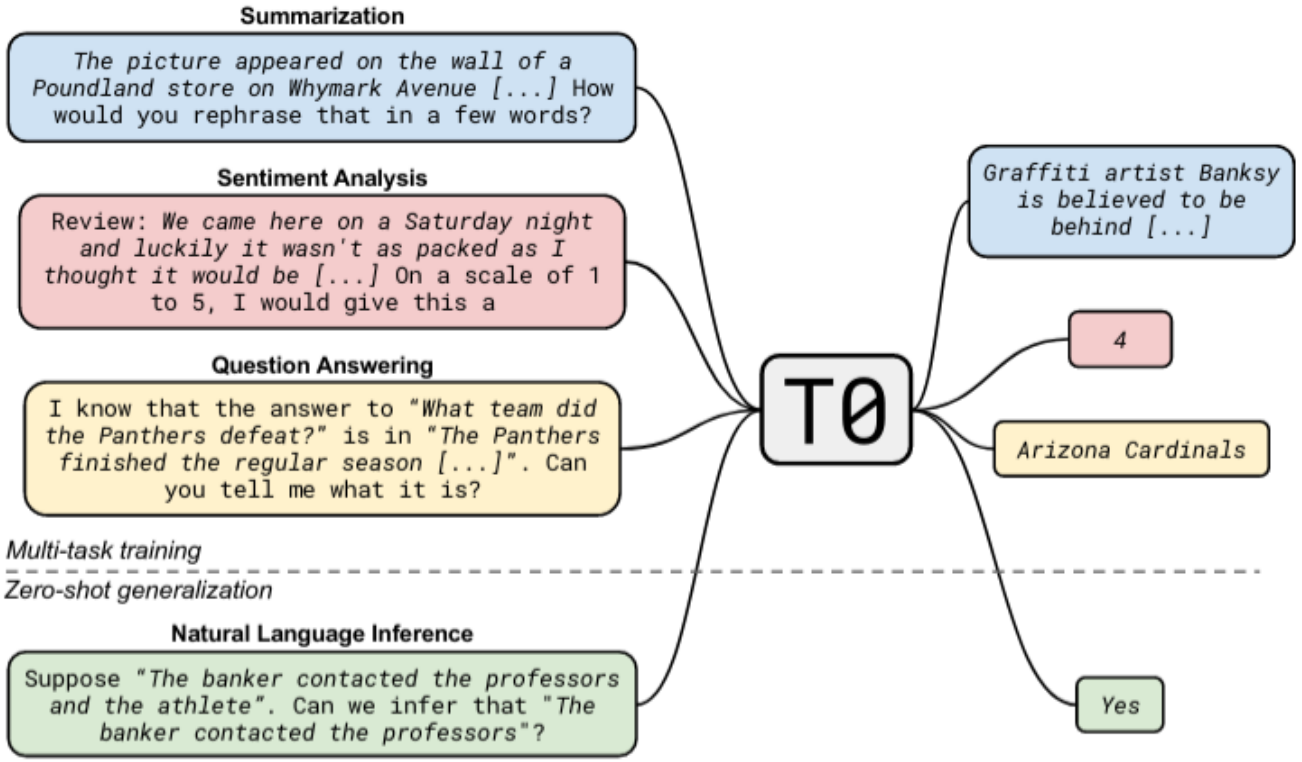
LLMs as Useful Tools (3)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **To improve usability of LLMs, tuning LLMs has focused on:**
  - 1. Performing many generative tasks, e.g. "explain moon landing"
  - 2. Providing truthful, non-toxic answers
- **How to achieve the first point?** Tune weights on **many new tasks**
  - **Before:** fine-tuning typically on chosen task, required few labels
  - **Now:** fine-tuning in many tasks, requires many labels per task
- **Why are more labels required for LLM tuning?**
  - Because goal is more general: to perform well on *many new* tasks
- In this section, common approach for LLM tuning: **instruction tuning**
- **What about the second point above?**
  - Also in this section, main approach for this goal: **RLHF**

Instruction Tuning (1)

- Question: can LLMs **perform** well on **tasks not seen in training/tuning?**
  - I.e. zero-shot generalization
- That's general goal of instruction tuning (Sanh et al., Wei et al.)



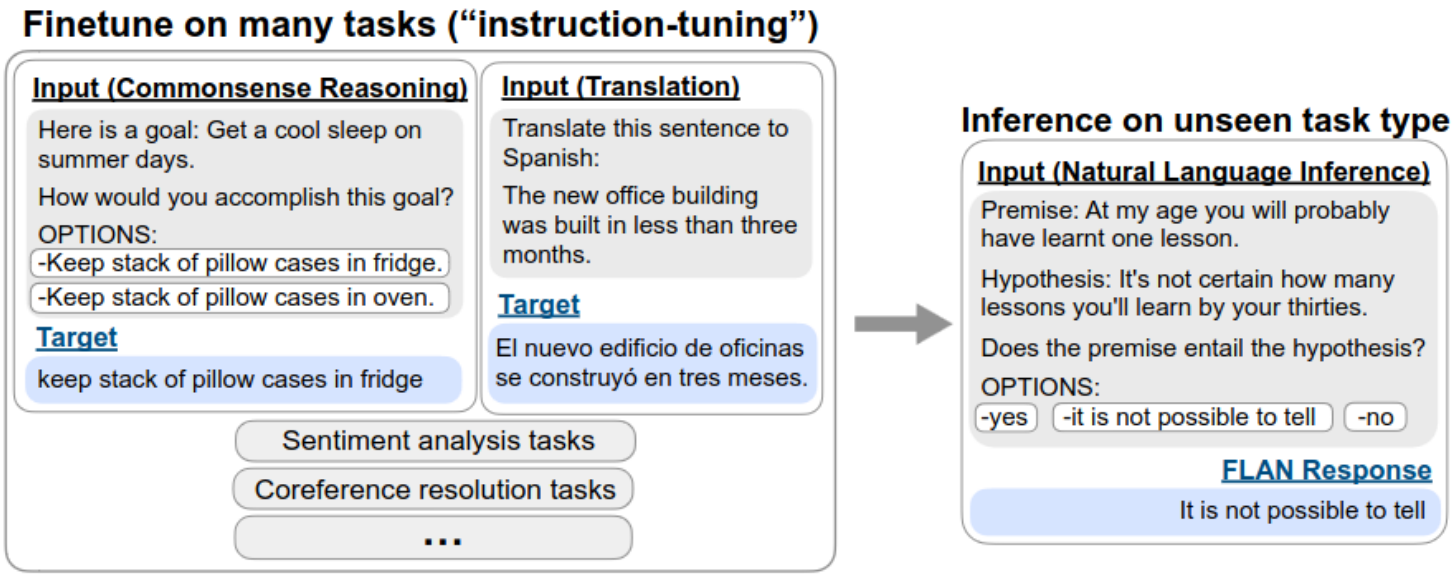
Instruction Tuning (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Instruction tuning:** train models to get better at zero-shot task solving, so long as those tasks come with text-based instructions
  - I.e. models learn to "follow instructions"
  - **Important:** measured as models performing NEW tasks not seen during pre-training or tuning, otherwise phrase "follow instructions" is vague
- **How does instruction tuning work?**
  - **Update weights** by training models **to solve many different supervised tasks** that come with instructions
  - I.e. a combination of (standard) fine-tuning with ICL
- Common approach:
  - **Cast NLP tasks as text-to-text** (see image in previous slide)
  - **Keep CLM objective** (no switch to supervised objective, e.g. classification)
  - **Train on some set of tasks, validate/test on DIFFERENT set of tasks**
- Let's visualize this.

Instruction Tuning (3)

- **How to check that models can "follow instructions"?**
  - Evaluate performance on tasks not seen in training or tuning
- Image from Wei et al.

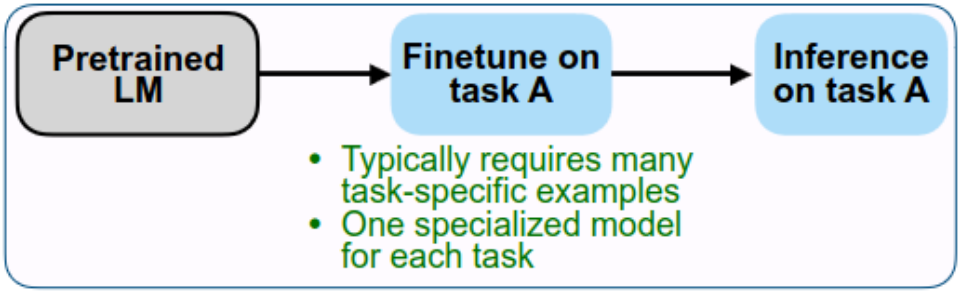


- How does this relate to standard fine-tuning and ICL?
  - Instruction tuning has components of both, but also different from both

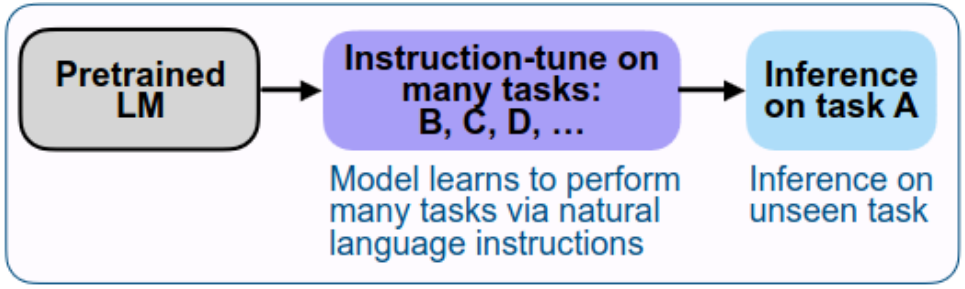
Instruction Tuning vs Fine-Tuning (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Following images from Wei et al.
- Standard **fine-tuning**:



- **Instruction-tuning:**



- In both cases, note the use of the different tasks A, B, C, D, ...

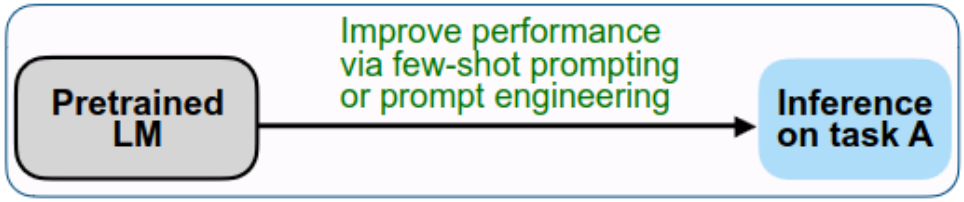
Instruction Tuning vs Fine-Tuning (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

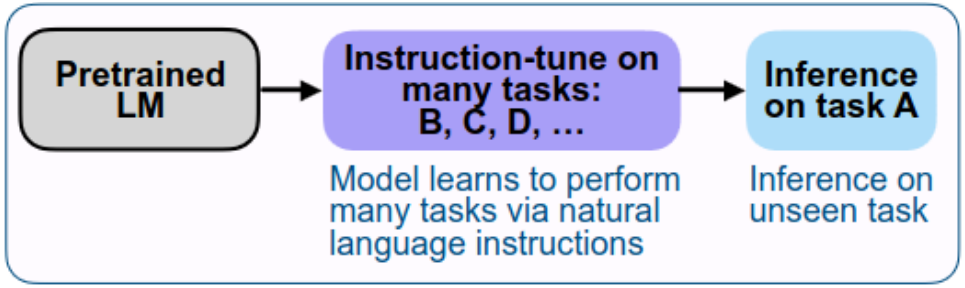
- **Fine-tuning:** update weights based on new (typically supervised) task
  - **Goal:** perform well at **specific task used in fine-tuning process**
  - Data typically as input-label pairs, e.g. *(email, label)* for spam detection.
- **Instruction-tuning:** update model weights based on several (typically supervised) tasks that are text-based and include instructions
  - **Goal:** perform well on **new tasks not seen during training or tuning**
  - Tasks must include text-based instructions
  - E.g. "Read the following email and tell me whether it's spam or not"
- In other words:
  - **Goal is different**, aim of instruction tuning is more generally useful model
  - **Data format is different**, as fine-tuning data more akin to standard ML data
  - Instruction tuning data includes text, instructions, similar to how humans follow instructions

Instruction Tuning vs ICL

- **In-context learning:** given instruction *and/or* examples of task, model performs task
  - **Similar to instruction tuning:** data format
  - **Different from instruction tuning:** no weight updates in ICL!
- **ICL:**



- **Instruction-tuning:**



Limitations of Instruction Tuning

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Achieving **generally useful models** brings **lots of challenges**
  - Some tasks don't have clear answers, e.g. write a story
  - LMs work at token level, some mistakes bigger than others
  - E.g. *"Avatar is a fantasy movie"*, replacing *fantasy* with *adventure* less problematic than replacing *movie* with *book*
- **Limitation 1:** requires **lots of data**, since **entire model often tuned**
  - PEFT methods can also be used for this, can be competitive
  - Many recent benchmarks designed for LLM fine-tuning (MMLU, Big-Bench)
- **Limitation 2:** LM objective not ***aligned*** with human intent (Ouyang et al. 2022)
  - **Explicit intentions:** models follow instructions (goal of instruction tuning)
  - **Implicit intentions:** models give answers that are truthful, non-toxic, etc.
  - Explicit intention can be achieved without implicit ones, e.g. a rude answer
- Thus, even more general goal: **tune models for human preference**
  - Often referred to as "model alignment" (Leike et al. 2018)

Tuning for Human Preference (1)

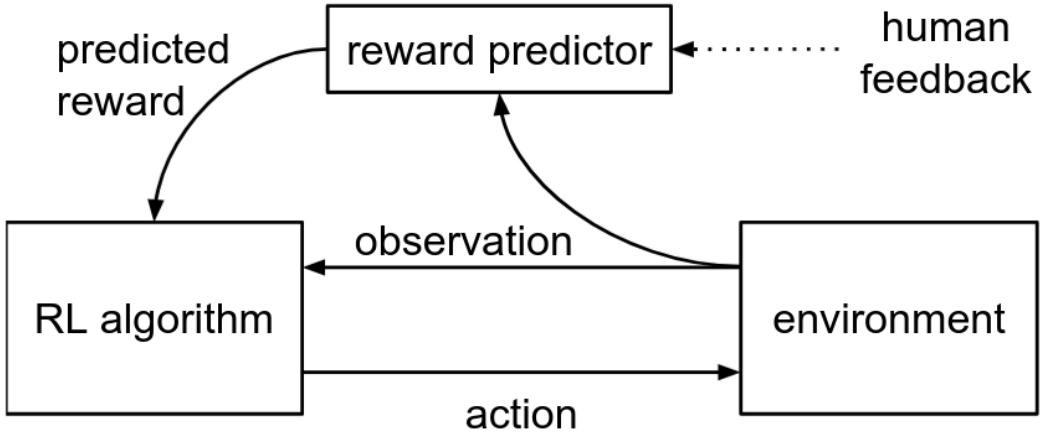
UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Main approach:** reinforcement learning for human feedback (RLHF)
  - Here: seminal work in context of deep learning by Christiano et al. (2017)
- Roughly, **basic components of reinforcement learning** (RL)**:**
  - An **agent** interacting with an **environment** over sequence of steps *T*
  - At time step $t \in T$, agent receives **observation** $o \in O$ from environment, sends **action** $a_t \in A$ to environment based on $o \dashrightarrow (o_1, a_1), (o_2, a_2), \ldots$
  - Environment provides **scalar reward** $y_t = r(a_t)$ where *r* is **reward function**
  - **Goal in RL:** learn **policy** $p: O \rightarrow A$ that predicts set of actions $a_i \in A$ for $i \in T$ that maximizes rewards over *T* (in deep RL, *r* and *p* are deep networks)
- **RL** often **successful** in tasks where **reward function is specified well**
  - E.g. playing video games (Mnih et al, 2015)
- For some tasks, **specifying the reward function can be challenging**
  - E.g. train robot to do scrambled eggs (function for good scrambled eggs?)
- **Idea:** allow humans to provide feedback on the agent's actions
  - **Use** this **human feedback to define task**

Tuning for Human Preference (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Christiano et al. 2017 proposed the following:
  - **Learn reward function** *r: O x A --> ℝ* **that provides rewards aligned with human preferences** while **simultaneously training** a successful **policy**



- Such a method should **allow solving tasks defined by desired behavior**
  - "By desired behavior" means **agents can be taught by non-experts**
  - E.g. no need to be a chef to provide feedback on scrambled eggs
- This **approach now referred to as RLHF**
  - **InstructGPT**: recent work that used RLHF to *align* LLMs

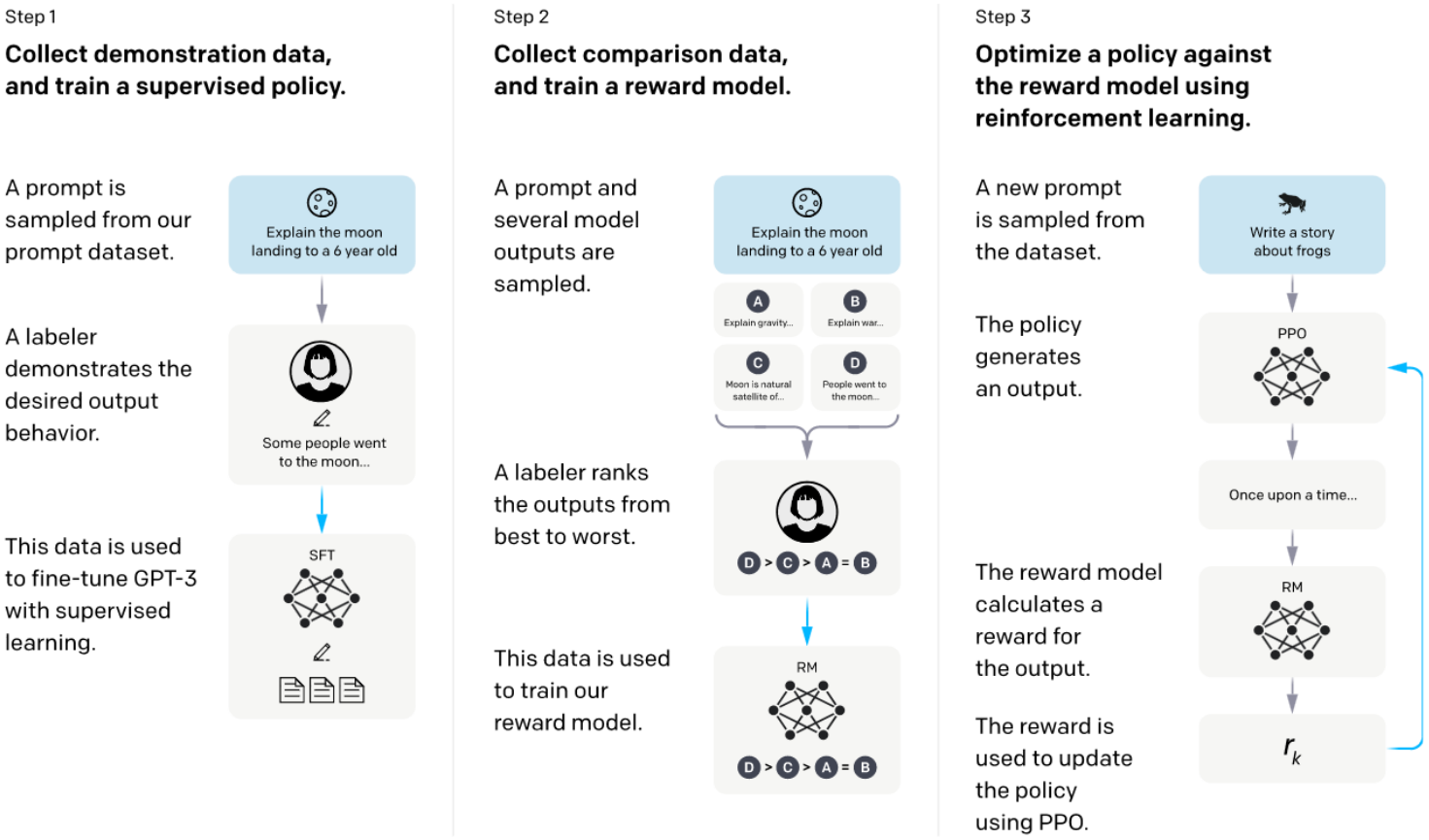
InstructGPT (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Ouyang et al. (2022) **used RLHF** to **fine-tune GPT-3** to **follow broad class of written instructions** with outputs **aligned with human preference**
- Process broken into three steps (each covered in detail in next slides):
  - **1.** Collect examples for several generative tasks, **train supervised policy**
    - **Example of task:** "List five ideas for how to regain enthusiasm for my career"
    - **Human labelers** provided **examples of desired outputs**
    - Recall **policy**: function that provides actions given observations
    - In the context of LLMs, actions are generating different text
    - So, **"train policy"** here was **fine-tuning GPT-3 with examples from labelers**
  - **2.** Collect comparison data, **train reward model to mimic humans**
    - Given tasks and model from Step 1, **generate multiple outputs for each task**
    - **Human labelers choose preferred output** from given pairs of outputs for tasks
    - Recall **reward model**: function from observations and actions to reward
    - Reward model: **copy of Step 1 model that produces scalar** (LM head replaced)
  - **3. Optimize policy (Step 1) against reward model (Step 2) using PPO**
    - **Proximal policy optimization (PPO)** (Schulman et al. 2017)**:** common in RLHF

InstructGPT (2)

- Visualization of the three steps (image from Ouyang et al. 2022):



- Let's look at each step in more detail.

InstructGPT: Step 1

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

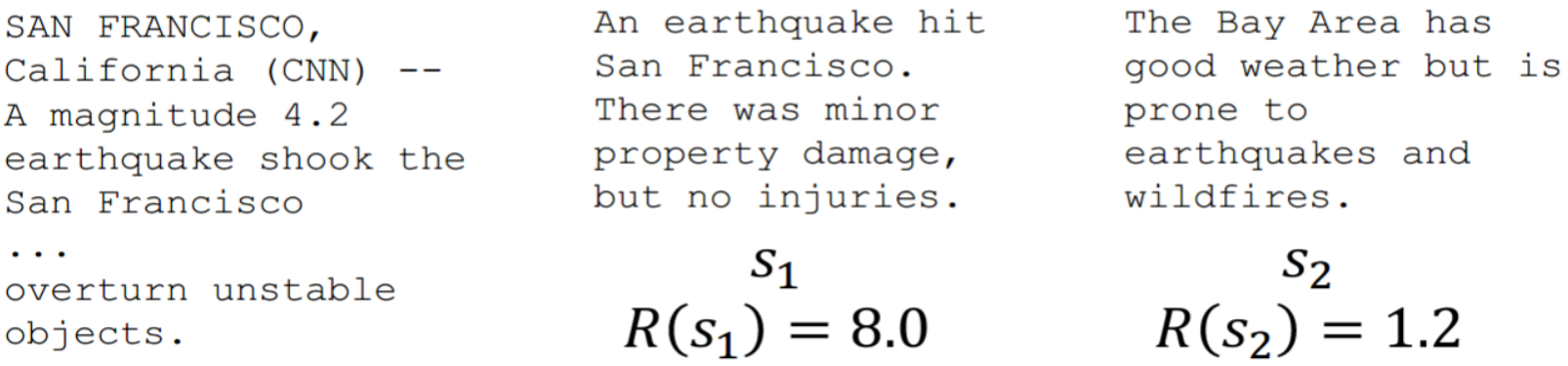
- Step 1: **collect examples for several tasks, train supervised policy**
  - **Tasks:** prompts and outputs generated by OpenAI users (volunteers)

- **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
- **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
- **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

Image source

  - **Examples of such "text generation tasks":**
    - "List five ideas for how to regain enthusiasm for my career"
    - "Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home."
  - Note that this requires **a lot of human effort**, thus very expensive!
    - This cost makes RLHF not easily accessible
  - **Training policy**: fine-tuning (all of) GPT-3 on examples from humans
    - Recall **policy**: function *p* from actions to observations, i.e. *p: O -> A*

InstructGPT: Step 2 (1)

- Step 2: **collect comparison data**, train reward model to mimic humans
- **Comparison data**: each example is a (i) task, (ii) pair of outputs (x_1, x_2) for task from model tuned in Step 1, (iii) human-chosen preference x_i
- 

  - Human-assigned rewards as ratings can be noisy, not calibrated (see image)
  - To avoid this, humans are typically asked for pairwise comparisons
  - E.g. which summary is best from the ones above, left or right?
  - Chosen summary gets label 1, other gets label 0

Image source

InstructGPT: Step 2 (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Step 2: collect comparison data, **train reward model to mimic humans**
- Recall **reward model**: function from actions and observations to reward
  - Reward is scalar number, i.e. reward model r is function *r: A x O --> $\mathbb{R}$*
- Here: **reward model is copy of model tuned in Step 1**
  - But model from Step 1 is GPT-3, i.e. a language model
  - It does not produce a scalar, but a distribution over some vocabulary
  - What can we do?
  - They replaced language modeling head with projection to scalar
- But **how do we train** this (beheaded) **reward model**?
  - (Averaged) **Cross-entropy loss** (CE), labels are 1 and 0 as needed
  - Specifically, given input *x*, model is trained to predict which output $y \in \{y_0, y_1\}$ is preferred by humans
  - For reward model r_θ parameterized by θ and preferred output *i*, we have:

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}[\log(\sigma(r_\theta(x,y_i) - r_\theta(x,y_{1-i})))]$$

InstructGPT: Step 3 (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Step 3: **optimize policy (Step 1) against reward model (Step 2)**
  - In other words, improve model from Step 1 *further* by training it to produce higher-quality outputs preferred by humans
  - How do we know which outputs are preferred by humans? Reward model!
- **Policy gradient methods**: designed for estimating RL objectives
  - InstructGPT relied on commonly used PPO method
  - **Proximal policy optimization (PPO)** (Schulman et al. 2017)
  - Details beyond our scope, e.g. value function, alternate optimization
- **Final RL objective:**
  - Given reward model r_θ parameterized by θ and *frozen* policy model π^{SFT} fine-tuned in Step 1, we tune model π^{RL} parameterized by ϕ and initialized as copy of π^{SFT}, by **maximizing** the **following objective**:

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right]$$

InstructGPT: Step 3 (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right]$$

- Note: model $\boldsymbol{\pi^{RL}}$ **regularized** so its predictions are not far from π^{SFT}
- How? Subtraction term weighted by factor β
  - Proportion between predictions produced by the model currently training, i.e. π^{RL}, and the π^{SFT} model used to train the reward model in Step 2
  - I.e. prevents policy model currently training to differ too much from model fine-tuned in Step 1 (the LLM fine-tuned on human-created labels)
- And **what is the term** weighted by factor γ?
  - Here, $D_{pretrain}$ is dataset used for pre-training original model (GPT-3)
  - This factor is designed to keep the model from differing too much from performance after pre-training and before fine-tuning in Step 1
  - I.e. we still want a LM at the end of the day

InstructGPT: All Together

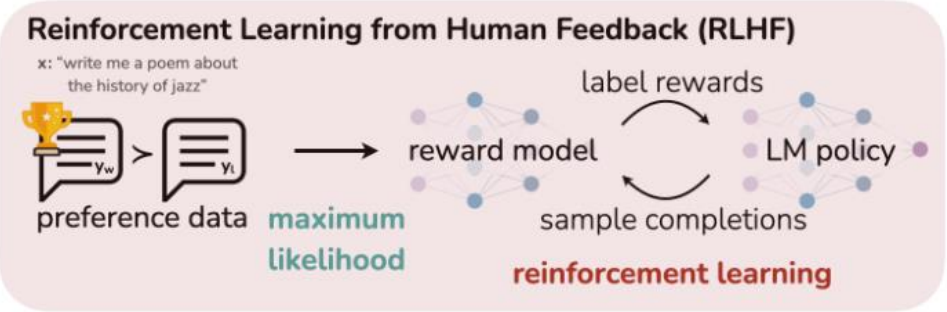
UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- There are **several models and training objectives used for InstructGPT**
  - **Policy model:** a copy of GPT-3, so a LM, fine-tuned with CLM objective
  - **Reward model:** copy of policy model (LM head replaced), trained on CE
  - **Reinforced policy model:** tuned with RL objective and signal from reward model (which learned to mimic human preference), our final LM
- Both **policy** and **reward models trained with human-labelled data**
  - Lots of data because goal was to be good at thousands of tasks
  - Thus, expensive to obtain, process required constant work with labelers
- **RLHF applied to LLMs** key step to enable construction of **ChatGPT**
  - Note that ChatGPT is a product based on some LLM (or many?)
  - Initially based on LLMs from the GPT-3 generation
  - Currently (May 2025) based on variants of GPT-4
- **High cost of RLHF means ChatGPT levels of performance not easily accessible**
  - Achieving this level of performance at a reduced cost is ongoing research

Reducing Cost of RLHF (1)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Can the cost of tuning for human preference be decreased?**
  - Focus of current research efforts
- **Direct Preference Optimization (DPO)** proposes such an approach
  - Proposed by Rafailov et al. 2023
  - Reparameterizes RLHF objective to avoid use of reward model
  - Derives closed-form solution, results in optimizing simple classification loss



- Trains on "preference pairs"
  - E.g. which of the two answers is best

Reducing Cost of RLHF (2)

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Bai et al. (2022) reduced human effort in LLM alignment to almost none
  - They build as set of principles (a "constitution") so models can use to determine whether a response is appropriate
  - For Step 1, the LLM itself judged and improved on its own answers, then was fine-tuned on its improved responses
  - For Step 2, they used the fine-tuned model to choose better aligned responses from pairs of responses
  - Step 3 was done as before.
  - They called this approach: **Reinforcement Learning from AI Feedback**
- More recently, DeepSeek-R1 skipped the entire Step 1 (expensive human labeling)
  - Used out-of-the-box software tools as reward models to train their model for reasoning tasks that typically require multiple steps
  - **Out-of-the-box reward models**: code compiler, math solvers

Open Challenges with LLMs

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- Still several challenges ahead.
- **Hallucinations:** still severely hindering potential of these tools
- **Cost:** still highly expensive to train, but costs coming down
- **Reasoning:** still limited in their ability to reason, try different solutions, identify mistakes
- **Transparency:** still huge black boxes, neither user nor models can explain their output reliably
- **Updated information:** requires updating models with newer data

Summary

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

- **Large Language Models**
  - **Size means new "abilities"**, e.g. ICL, following instructions
  - New "abilities" means new ways to use these models
- Using LMs before:
  - Fine-tune pre-trained LM, e.g. BERT, requires expert knowledge
- Using (L)LMs today:
  - Interaction via natural language, technology *potentially* democratized to end-users (ideally, if you can talk/write, you can use them)
- **In-Context Learning (ICL):** given task description and/or examples, model performs task (no weight updates on model)
- **Instruction tuning:** tune models to perform *new* tasks given instructions
- ICL and instruction tuning give rise to **prompting, prompt engineering**
- Despite progress, **models generally not *aligned* to human preference**
  - **Alignment** methods **expensive** (reinforcement learning, human feedback)

References

- References linked in corresponding slides