Intro to LLMs

Victoria Zhang

Feb 10, 2025

Outline

- LM Background
- LLM Modeling and Pre-training
- Adapt LLMs to Downstream Tasks
- More about LLMs

LM Background

From LMs to LLMs

Language Models (LMs)

- ***Corpus*** = raw text data
- ***Token*** = word & punctuation x_l
- ***Vocabulary*** = a set of tokens, $x_1, \ldots, x_L$

- **LM** = probability distribution over ***a sequence of tokens*** $p(x_1, \ldots, x_L)$

$$p(\text{the, mouse, ate, the, cheese}) = 0.02,$$

$$p(\text{the, cheese, ate, the, mouse}) = 0.01,$$

$$p(\text{mouse, the, the, cheese, ate}) = 0.0001.$$

- **Generation** = sample from the distribution $x_{1:L} \sim p$

Autoregressive LMs

- Joint distribution → chain rule (efficient, e.g., via feedforward NN)

$$p(x_{1:L}) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_L \mid x_{1:L-1}) = \prod_{i=1}^{L} p(x_i \mid x_{1:i-1})$$

$$\begin{aligned} p(\text{the, mouse, ate, the, cheese}) = \; & p(\text{the}) \\ & p(\text{mouse} \mid \text{the}) \\ & p(\text{ate} \mid \text{the, mouse}) \\ & p(\text{the} \mid \text{the, mouse, ate}) \\ & p(\text{cheese} \mid \text{the, mouse, ate, the}). \end{aligned}$$

Autoregressive LMs generation

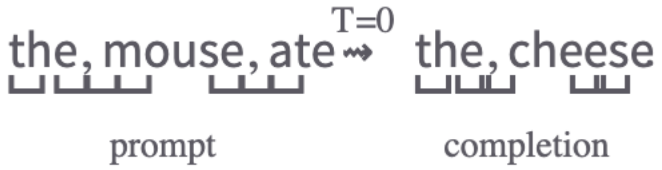
- Generation:

$$\text{for } i = 1, \dots, L:$$
$$x_i \sim p(\, x_i \mid x_{1:i-1} \,)^{1/T}$$

where $T \geq 0$ is a <u>temperature</u> parameter that controls how much randomness:

  - $T = 0$: deterministically choose the most probable token x_i at each position i
  - $T = 1$: sample “normally” from the pure language model
  - $T = \infty$: sample from a uniform distribution over the entire vocabulary

- Prompt $x_{1:i}$ = prefix
- Completion $x_{i+1:L}$= conditional generation



Towards Large LMs (LLMs)

- N-gram models

$p(\text{cheese} \mid \text{the, mouse, ate, the}) = p(\text{cheese} \mid \text{ate, the}).$ $p(x_i \mid x_{1:i-1}) = p(x_i \mid \underline{x_{i-(n-1):i-1}}).$

$n = 3$

if n is too small, or too big?

- Neural LMs

$p(\text{cheese} \mid \text{ate, the}) = \text{some-neural-network}(\text{ate, the, cheese}).$ $p(x_i \mid x_{1:i-1}) = p(x_i \mid \underline{x_{i-(n-1):i-1}}).$

context

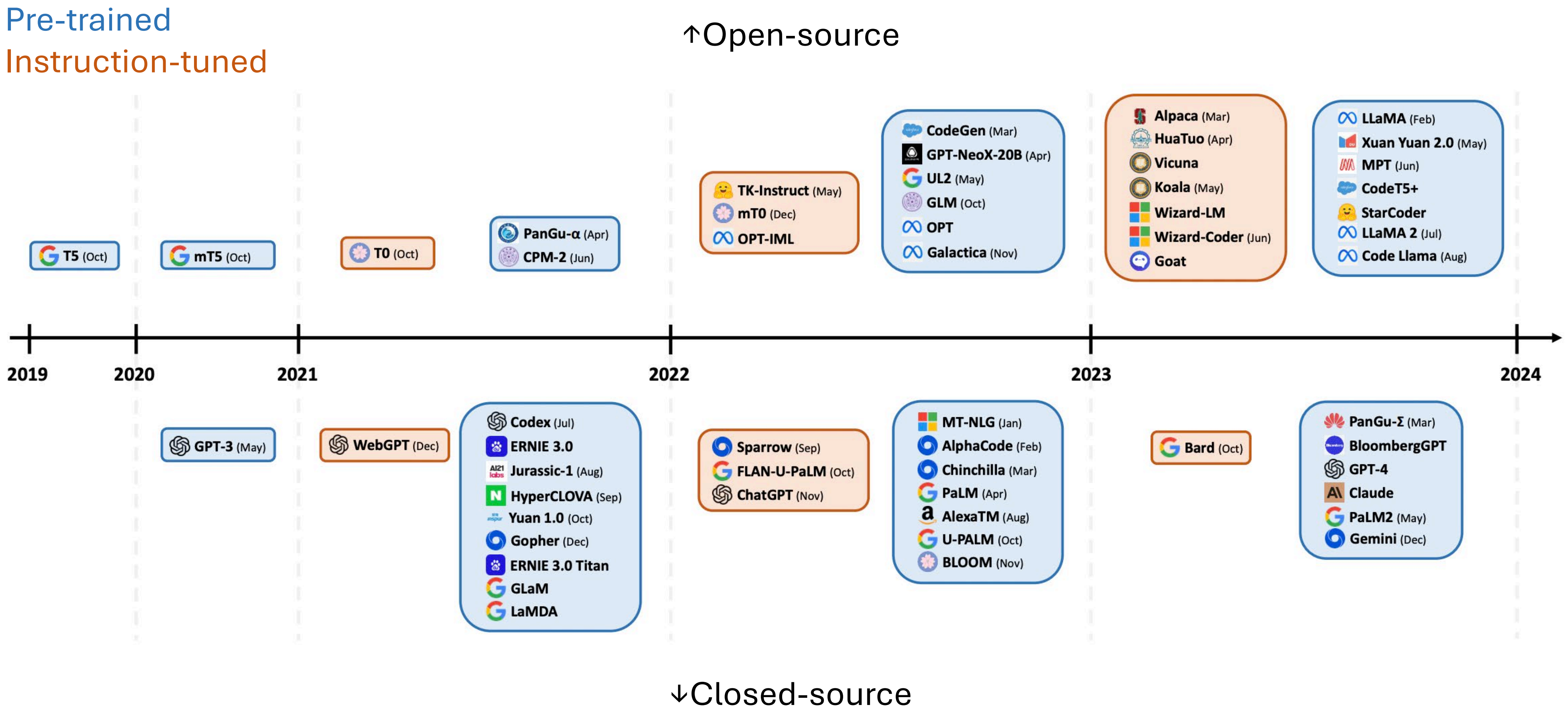
  - Recurrent neural networks (RNNs, LSTMs) effectively $n = \infty$
  - Transformers (TFs)

Towards Large LMs (LLMs)

	N-gram	**RNN / LSTM**	**Transformer**
Context	n	∞	n
Training - statistically	Infeasible	Feasible	Feasible
Training - computationally	Efficient	Inefficient	Still inefficient, but better due to GPUs
Application	Limited to speech recognition & machine translation	Speech recognition, machine translation, text completion, …	Modern language tasks: translation, Q&A, arithmetic, …

Modern language tasks: prompting in disguise

LLM Timeline



LLM Modeling and Pre-training

Transformers and LLMs

Transformers

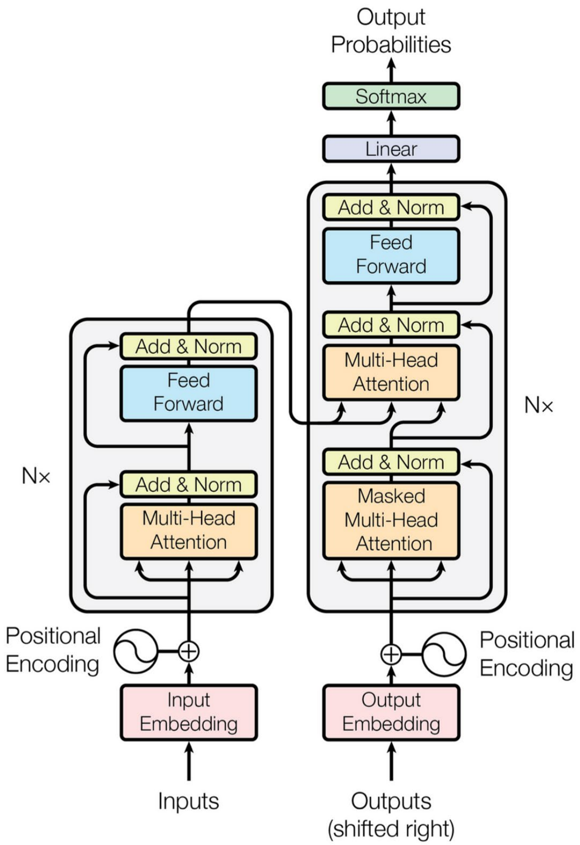


Figure 1: The Transformer - model architecture.

- Key components:
  - Self-attention
  - Positional encoding
  - Masked training
- LLM architectures:
  - Encoder-only
  - Decoder-only
  - Encoder-decoder

Encoder-Only

- Tokens → contextual embeddings

$$x_{1:L} \Rightarrow \varphi(x_{1:L}). \qquad \mathcal{V}^L \rightarrow \mathbb{R}^{d \times L}$$

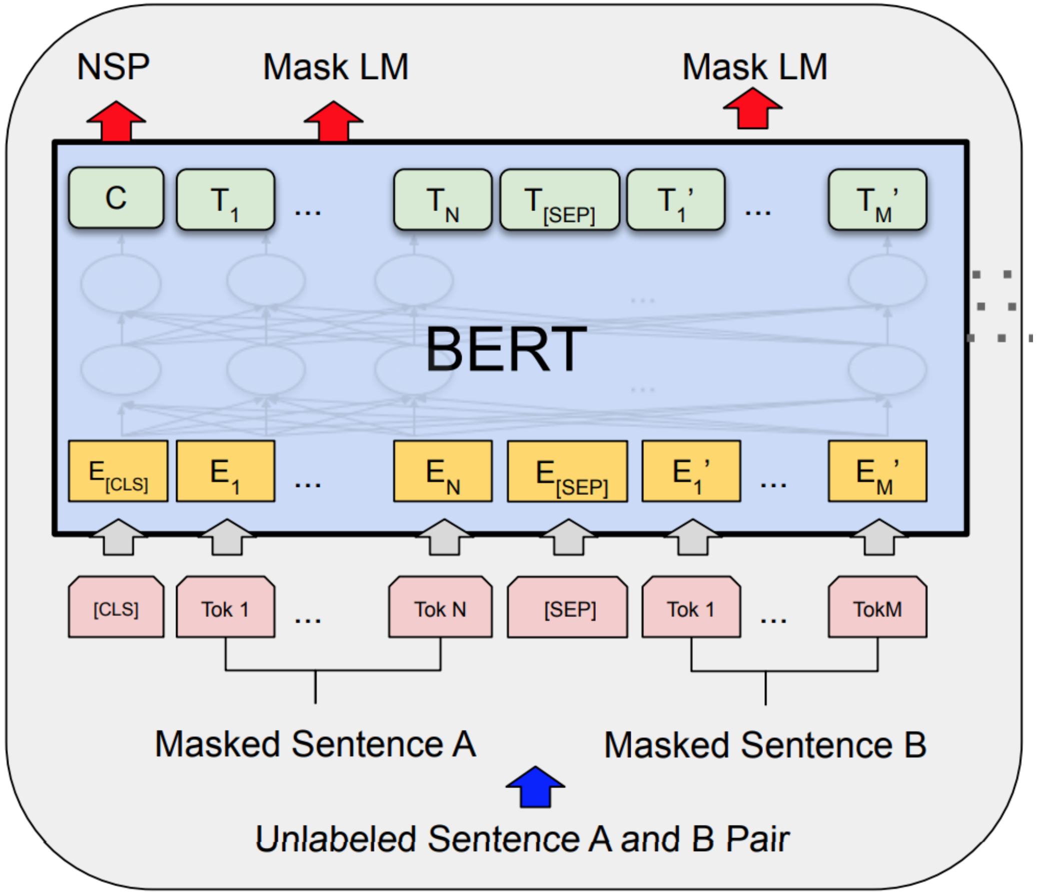
- Text classification, e.g. positive / negative reviews

$$[[\text{CLS}], \text{the}, \text{movie}, \text{was}, \text{great}] \Rightarrow \text{positive}.$$

- Examples: BERT, RoBERTa

$$\text{BERT}(x_{1:L}) = \text{TransformerBlock}^{24}(\text{EmbedTokenWithPosition}(x_{1:L}) + \text{SentenceEmbedding}(x_{1:L})) \in \mathbb{R}^{d \times L}.$$

Encoder-Only: BERT-Large



- `Transformer` block x 24
- Bidirectional
- Pre-training:
  - Masked language modeling (MLM)
  - Next sentence prediction (NSP)
- Some hyper-params
  - $d_{model} = 1024$
  - $n_{head} = 16$
  - $L = 512$
  - #params: 355M

Decoder-Only

- Tokens → contextual embeddings & next token (distribution)

$$x_{1:i} \Rightarrow \varphi(x_{1:i}), p(x_{i+1} \mid x_{1:i}).$$

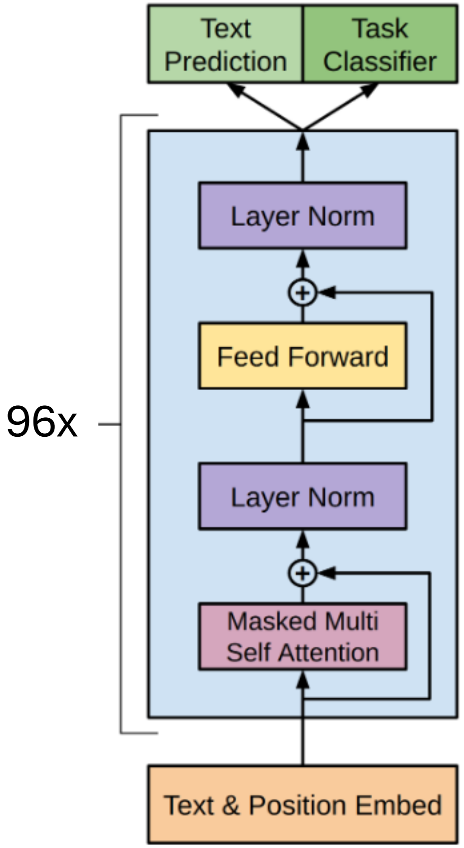
- Text completion, article generation

$$[[\text{CLS}], \text{the}, \text{movie}, \text{was}] \Rightarrow \text{great}$$

- Examples: GPT-2, GPT-3, …

$$\text{GPT-3}(x_{1:L}) = \text{TransformerBlock}^{96}(\text{EmbedTokenWithPosition}(x_{1:L}))$$

Decoder-Only: GPT-3



- `Transformer` block x 96
- Sparse Transformer
- Pre-training:
  - next token prediction
- Some hyper-params
  - $d_{model} = 12288$
  - $n_{head} = 96$
  - $L = 2048$
  - #params: 175B

Encoder-Decoder

- Full-transformer
- Tokens → tokens

$$x_{1:L} \Rightarrow \varphi(x_{1:L}), p(y_{1:L} \mid \varphi(x_{1:L})).$$

- Translation
- Examples: BART, T5 …

Comparing LLMs

	Encoder-only	Decoder-only	Encoder-decoder
Models	BERT, ViT	GPT-3, ChatGPT, Llama, DeepSeek	BART, T5, Google Gemini (Probably)
Pretraining	Masked Language Modelling (MLM)	Next Token Prediction	Task-dependent
Bidirectional	Yes	No	In encoder
Casual	No	Yes	Yes
Generation	No	Yes	Yes
Outputs	Bidirectional embedding	Unidirectional embedding & next token	Input embedding & output sequences
Ad-hoc Training	Yes	No	Yes
Applications	Classification	Generation	Translation

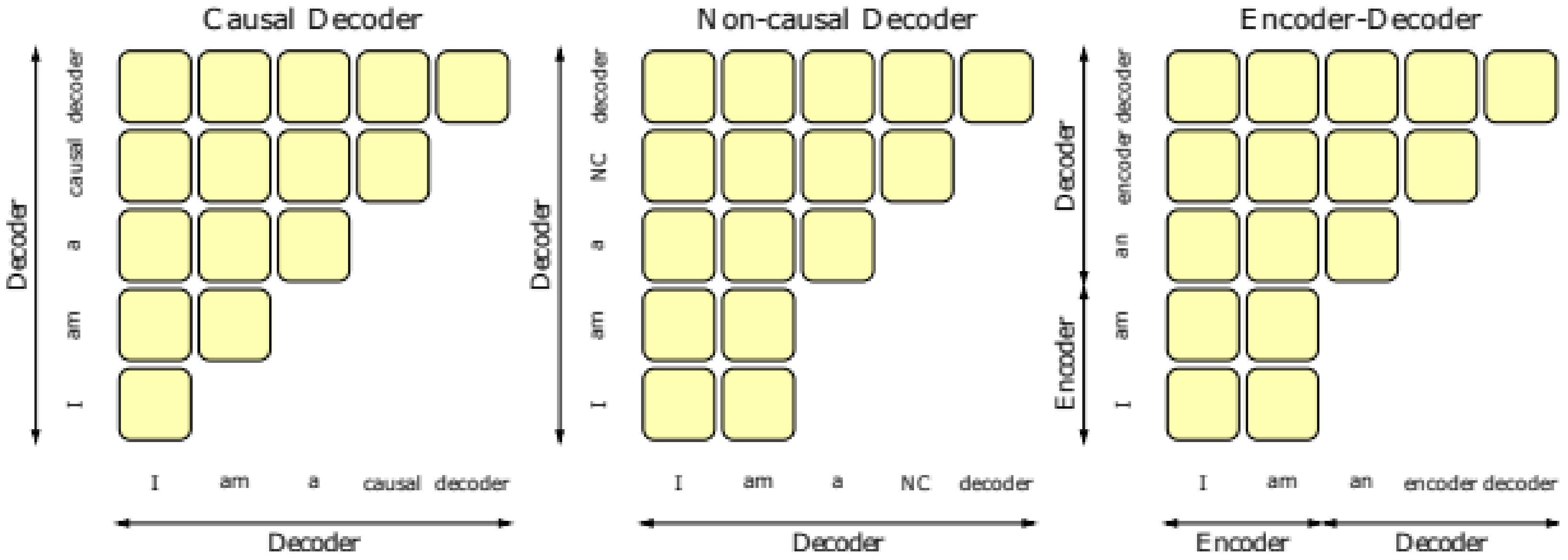
Why are more LLMs decoder-only?
Short Answer: efficient, easy to train, better adaptation

The popularity of decoder-only (1/2)

- Cost of Training
  - ED need to perform multitask finetuning (which is basically instruction finetuning) on labeled data and it could be very expensive
- In-Context Learning from Prompts
  - prompting introduced a gradient to the attention weight. decoder-only models does not need to be translated into an intermediate context first before being used for generative tasks
- Efficiency Optimization
  - in decoder-only models, the (K) and (V) matrices from previous tokens can be reused (cached) for subsequent tokens during the decoding process -> faster generation during inference

The popularity of decoder-only (2/2)

- Autoregressive vs Bidirectional Attention



Adapt LLMs to Downstream Tasks

Fine-tunning, probing, and prompting

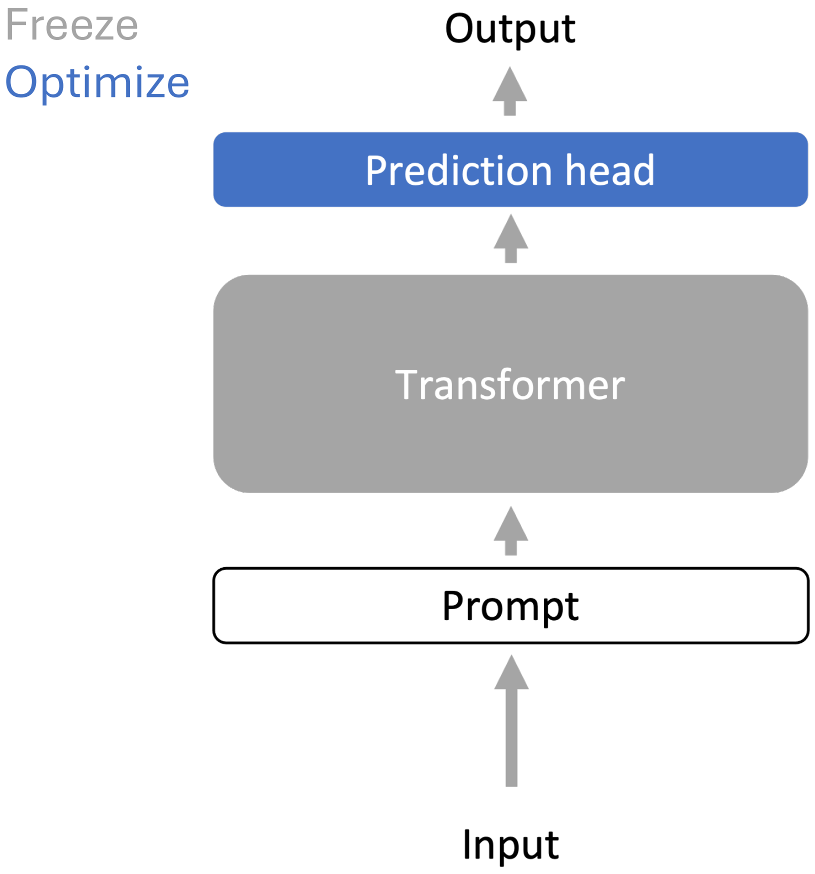
Towards Task-Specific LLMs

- Decoder-only models, e.g. GPT-3 is task-agnostic
- Not for any downstream task
  - Only pre-trained on next token prediction
  - Different formatting
  - Domain shift
  - Human preference

Adaptation

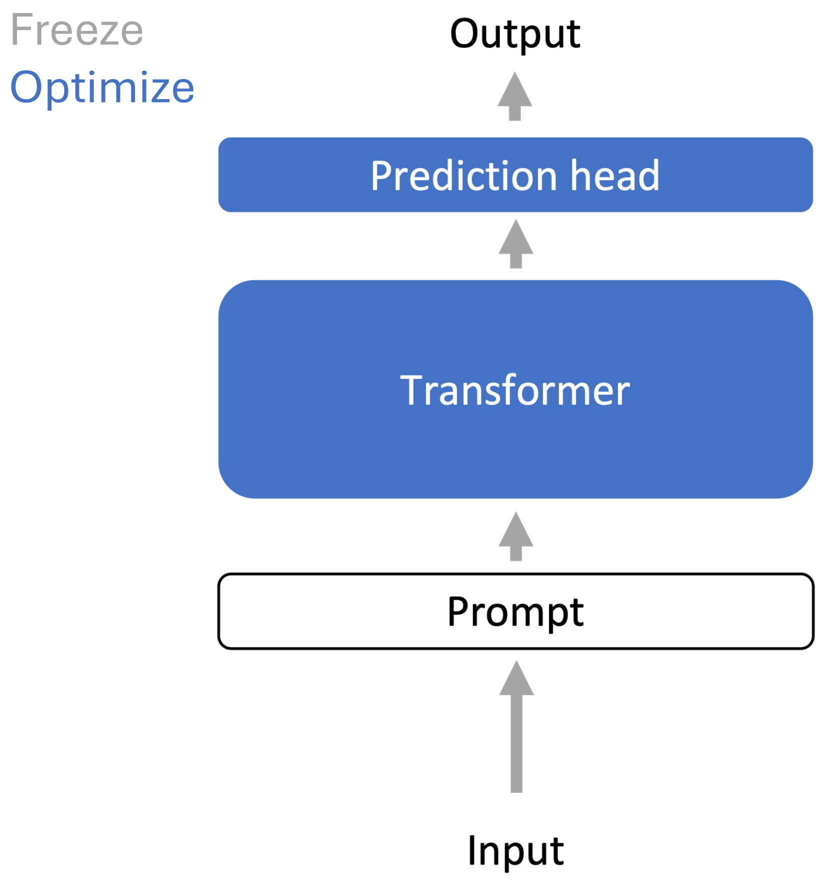
- Supervised learning
  - Fine-tuning
  - Lightweight fine-tuning
  - Probing
- In-context learning (prompting)
  - Zero-shot
  - One-shot
  - Few-shot

Probing



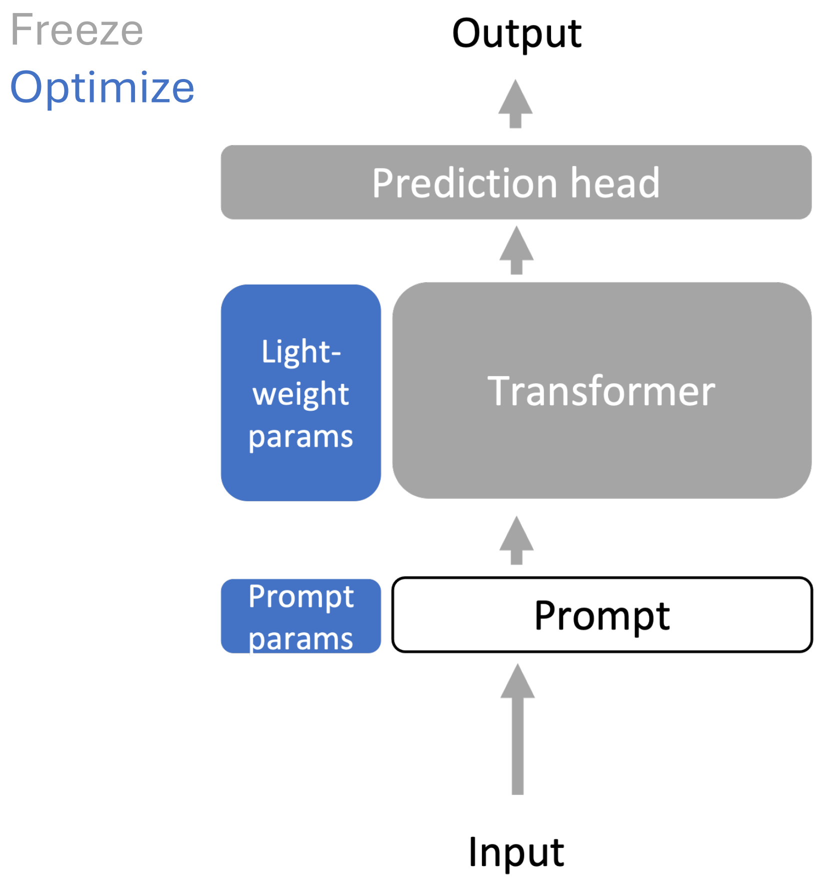
- Probe = prediction head
- Mostly on encoder-only
- Pooling:
  - CLS token
  - Average over tokens

Fine-tunning



- Smaller learning rate
- LLMs for every task - expensive

Lightweight Fine-tunning



- Optimize <1% parameters
- **Prompt** tunning
  - Append learnable embeddings
- **Prefix** tunning
  - Add learnable attention weights
- **Adapter** tunning
  - 2-layer NN between fixed transformer blocks
- Others (LoRA, BitFit, ...)

Prompting

- Zero-shot transferable (passable) with
  - Task description
  - Examples (input-output pairs)
- Prompt engineering

Prompting

- Arithmetic

Q: What is 556 plus 497?

A: 1053

- Translation

Mein Haus liegt auf dem Hügel. = My house is on the hill.

Keinesfalls dürfen diese für den kommerziellen Gebrauch verwendet werden. = In no case may they be used for commercial purposes.

Prompting

- Grammar correction

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you
requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I would be happy to work with you on another project.

Selective Models

Scaling laws, GPT, DeepSeek and Costs

Scaling Laws (OpenAI)

- **Model size (*N*):** 768 - 1.5 billion non-embedding parameters.
- **Dataset size (*D*):** 22M -23B tokens.
- **Model shape**: depth, width, attention heads, and feed-forward dimension.
- **Context length**: 1024 for most runs
- **Batch size**: 2^{19} for most runs.
- ***L*** be the test cross-entropy loss.
- **C** be the amount of compute used to train a model.

Performance depends strongly on model scale (**N, D, C**), weakly on model shape

GPT 3x → GPT 4o

- Unimodal → ***multi***modal (text, audio, image, and video)
- #params: 175B → ***1.8 trillion***
- context window size $L = 2048$ tokens → ***128k*** tokens.
- Reasoning: bottom 10% → ***top 10%*** passing candidates
- MoE

DeepSeek-V3

- Architecture: Innovative Load Balancing Strategy and Training Objective
  - DeepSeek-V2 + auxiliary-loss-free strategy for load balancing
  - Multi-Token Prediction (MTP) objective
- Pre-Training: Towards Ultimate Training Efficiency
  - FP8 mixed precision training framework
  - Overcame the communication bottleneck in cross-node MoE training
  - 2.664M H800 GPU hours (DeepSeek-V3 on 14.8T tokens)
- Post-Training: Knowledge Distillation from DeepSeek-R1
  - distill reasoning capabilities from the long-Chain-of-Thought (CoT) model

DeepSeek-R1

- Architecture:
  - Training Only the Important Parts (via MoE)
    - **5% of the model's parameters** were trained per token- > **95% reduction in GPU usage**
  - Faster and Cheaper AI with **KV** Compression
- Pre-Training:
  - No Fancy Chips, Just Smart (Low-level) Optimizations
- Post-Training:
  - Smarter Learning with Reinforcement Learning (RL)
    - Group Relative Policy Optimization (GRPO)

DeepSeek R1 vs GPT4o

- #params: 671B vs. 1.8 trillion
- context window size $L = 2048$ tokens vs. 2048 tokens
- # transformer: 61 vs. 120

More about LLMs

What is the future?

Data

- Effective size
- Privacy
- Fairness
- Contamination
- Ecosystem

Multi-Modality

- Input + output = multi-modal
  - T2I: Dall-E, Stable diffusion
  - I2T: GPT-4V
- Input = multi-modal
  - CLIP
- Output = multi-modal

More on LLMs…

- Parallelism
- Scaling law…(?)
- Other than transformers
- Ethical issues

Resources

- Courses
  - https://stanford-cs324.github.io/winter2022/
  - https://stanford-cs324.github.io/winter2023/
- Review papers
  - https://arxiv.org/pdf/2303.18223
  - https://arxiv.org/pdf/2307.06435
- Paper lists
  - https://github.com/Hannibal046/Awesome-LLM
- Posts
  - https://medium.com/@yumo-bai/why-are-most-llms-decoder-only-590c903e4789

Thank You

Q&A