

Foundations of Large Language Models

An In-Depth Summary of Core Concepts

Gemini & AM Alvarez

Universidad Nacional de Colombia - sede Manizales

from T. Xiao & J. Zhu work (<https://arxiv.org/abs/2501.09223>)

July 7, 2025

Outline

- 1 Chapter 1: Pre-training
- 2 Chapter 2: Generative Models & Scaling
- 3 Chapter 3: Prompting
- 4 Chapter 4: Alignment
- 5 Conclusion



From Task-Specific to Foundation Models

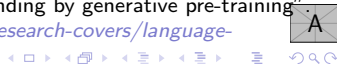
The development of LLMs is rooted in a fundamental shift in NLP methodology.

- **Old Paradigm:** Train specialized models from scratch for each task (e.g., translation, sentiment analysis) using large, task-specific labeled datasets.
- **New Paradigm (Pre-train, then adapt):**
 - 1 **Pre-train** a single, very large "foundation model" on vast amounts of unlabeled text using a self-supervised objective.
 - 2 **Adapt** this general-purpose model to specific downstream tasks using much less data via methods like fine-tuning or prompting.

This approach, pioneered by models like BERT¹ and GPT², leverages the structure inherent in language to build powerful, general-purpose representations.

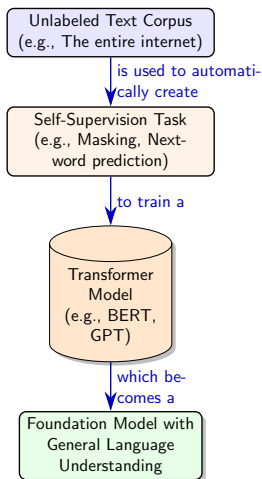
¹Jacob Devlin et al. (2019). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

²Alec Radford et al. (2018). "Improving language understanding by generative pre-training". In: URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.



Learning Without Human Labels

Self-supervised learning is the engine of pre-training. The model creates its own supervisory signals from the unlabeled input data.



The Three Transformer Families

Pre-trained models are typically based on one of three variations of the Transformer architecture.

Encoder-Only

(e.g., BERT)

- Sees the full input at once (bidirectional context).
- Excellent for understanding tasks (classification, NER).
- Pre-trained with Masked LM.

Decoder-Only

(e.g., GPT)

- Sees only past tokens (left-to-right context).
- Excellent for text generation.
- Pre-trained with Causal LM.

Encoder-Decoder

(e.g., T5, BART)

- Maps an input sequence to an output sequence.
- Excellent for sequence-to-sequence tasks (translation, summarization).



The Autoregressive Approach (Decoder-Only)

The model's task is to predict the next token given all preceding tokens. This is the core objective for generative models like GPT.

$$\mathcal{L}_{CLM} = - \sum_{i=1}^T \log P(x_i | x_{<i}; \theta)$$



Figure: At each step, the model uses a **causal attention mask** to prevent it from "seeing" future tokens.

The Denoising Approach (Encoder-Only)

Introduced by BERT, this objective corrupts the input by replacing tokens with a special '[MASK]' token, and trains the model to reconstruct the original tokens.

$$\mathcal{L}_{MLM} = - \sum_{i \in \mathcal{M}} \log P(x_i | \mathbf{x}_{\setminus \mathcal{M}}; \theta)$$

where \mathcal{M} is the set of masked indices.

- This allows the model to learn from both left and right context simultaneously (**bidirectional**).
- It is highly effective for building deep contextual representations.
- **Drawback:** Creates a pre-train/fine-tune mismatch, as the '[MASK]' token does not appear in downstream tasks.



A Hybrid Approach (e.g., XLNet)

PLM combines the strengths of autoregressive (CLM) and autoencoding (MLM) methods.

- 1 For a sequence, it randomly permutes the factorization order of the likelihood.
- 2 It then predicts tokens autoregressively based on this new order.
- 3 This effectively allows a token to be conditioned on a random subset of other tokens from the sequence, capturing bidirectional context without a '[MASK]' token.

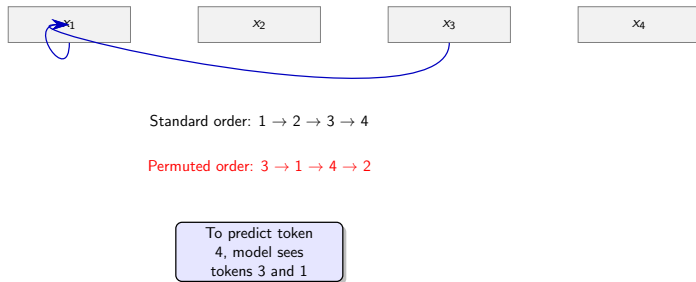


Figure: For prediction order $3 \rightarrow 1 \rightarrow 4 \rightarrow 2$, predicting x_4 depends on x_3 and x_1 , while predicting x_2 depends on all others.

The ELECTRA Approach

A more sample-efficient pre-training task. Instead of masking, a small "generator" model replaces some tokens, and a larger "discriminator" model is trained to identify which tokens were replaced.

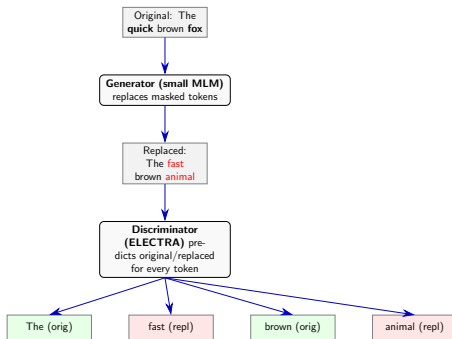


Figure: The discriminator learns from all tokens, not just the masked ones, making it much more computationally efficient. Clark et al. (2020)



Dissecting BERT

BERT (Bidirectional Encoder Representations from Transformers) is a multi-layer Transformer encoder.

Key Inputs

The final embedding for each token is the sum of three parts:

- **Token Embeddings:** The identity of the word/subword.
- **Segment Embeddings:** Indicates if the token belongs to sentence A or B (for sentence-pair tasks).
- **Position Embeddings:** Indicates the token's position in the sequence.

Special Tokens

- **[CLS]:** A special token prepended to every input. Its final hidden state is used as the aggregate sequence representation for classification tasks.
- **[SEP]:** A separator token used between sentences in sentence-pair tasks.



Adapting the Foundation Model

Once pre-trained, BERT can be adapted to various tasks by adding a small, task-specific layer on top of the base model and fine-tuning the entire network on labeled data.

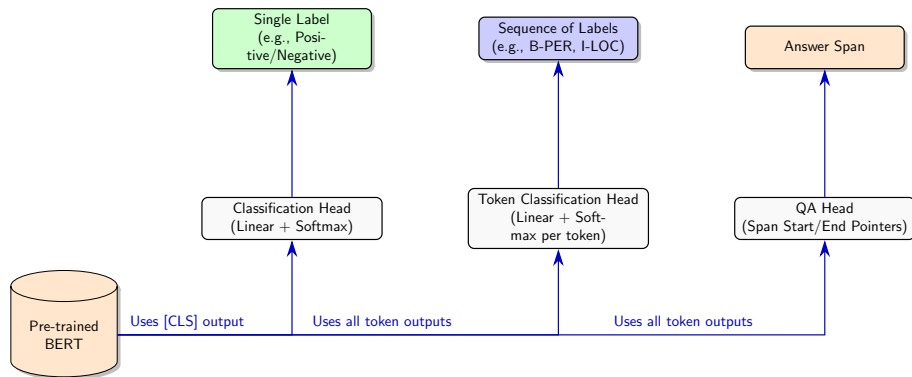


Figure: The same core BERT model can be used for diverse tasks by swapping the final layer.

Chapter 1 Summary

- Pre-training leverages massive unlabeled corpora to build foundation models.
- Self-supervision (MLM, CLM, etc.) is the key technique for learning from this data.
- Model architecture (Encoder, Decoder, Encoder-Decoder) is chosen based on the intended use case.
- Pre-trained models like BERT learn rich, contextual language representations.
- These models are then adapted to specific tasks via fine-tuning, which is far more efficient than training from scratch.



From Representation to Generation

While BERT-like models generate representations, GPT-like models generate text.

- **Architecture:** Decoder-only Transformer.
- **Objective:** Causal Language Modeling (predicting the next word).
- **Process:** Autoregressive generation. The model's output at step t is fed back as part of the input for step $t + 1$.

Key Insight

By simply training a model to predict the next word on a massive and diverse dataset, it learns not only grammar and facts but also reasoning, style, and world knowledge.^a

^aTom B Brown et al. (2020). “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33, pp. 1877–1901.



Architectural Deep Dive

The core of a generative LLM is a stack of identical decoder blocks. Each block has two main sub-layers.

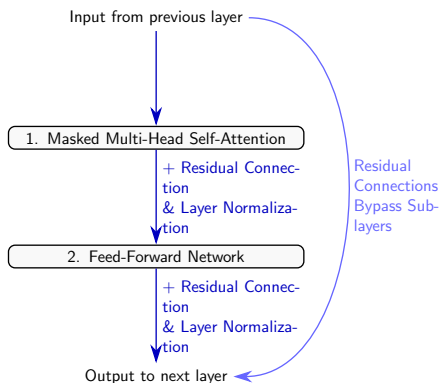


Figure: The Masked Self-Attention prevents positions from attending to subsequent positions, ensuring the autoregressive property is maintained.



Training LLMs is a Monumental Task

Developing modern LLMs pushes the boundaries of engineering.

Data Challenges

- **Volume:** Requires trillions of tokens.
- **Quality:** Unfiltered web data can be noisy, toxic, or factually incorrect.
- **Diversity:** Must cover many domains (prose, code, conversation) to be general.
- **Bias:** Data reflects societal biases, which the model will learn.
- **Privacy:** Data may contain personally identifiable information.

Compute Challenges

- **Cost:** Requires thousands of GPUs running for weeks or months.
- **Stability:** Training can diverge due to numerical precision issues (exploding/vanishing gradients).
- **Parallelism:** Must efficiently split the model and data across thousands of devices.



Making Large-Scale Training Feasible

Training a model with billions of parameters requires sophisticated parallelism.

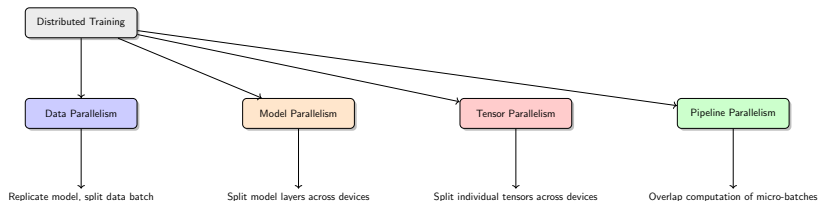


Figure: Modern frameworks like Megatron-LM combine these strategies to train massive models efficiently.

Predictable Improvements in Performance

A key finding in LLM research is that model performance (measured by test loss) improves as a smooth, predictable power-law function of model-dataset size, and compute.

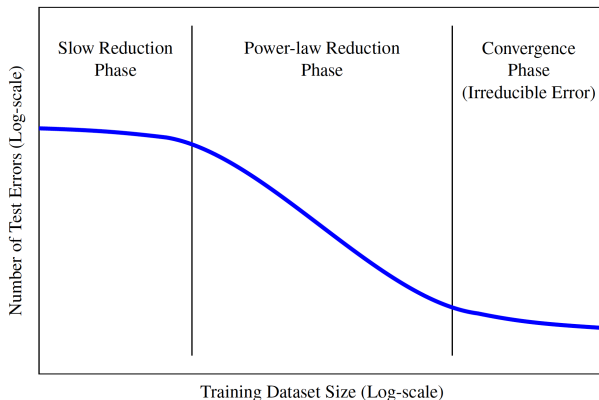


Figure: Power Law Curve



When Scale Unlocks New Skills

Emergent abilities are skills that are not present in smaller models but appear in larger models, often suddenly, once a certain scale threshold is crossed.

- Performance on these tasks is near-random for small models but becomes significantly better for large models.
- Examples include multi-step arithmetic, answering questions in a new language, and Chain-of-Thought reasoning.
- This phenomenon is a primary motivation for the continued scaling of LLMs³.

³Jason Wei, Yi Tay, et al. (2022). “Emergent abilities of large language models”. In: *Transactions on Machine Learning Research*.



Extending the Context Window

Standard Transformers have a computational cost that grows **quadratically** with sequence length ($O(n^2)$), making it very expensive to process long documents.

Challenges

- **Training Cost:** Prohibitively expensive for very long sequences.
- **Inference Memory (KV Cache):** The memory required to store intermediate key-value states grows linearly with sequence length, becoming a bottleneck.

Solutions

- **Efficient Architectures:** Sparse attention, linear attention, etc.
- **Positional Embedding Generalization:** Modifying positional encodings (like RoPE) to extrapolate beyond the trained context length, often through **interpolation**.



Chapter 2 Summary

- Generative models are typically decoder-only Transformers trained with a Causal Language Modeling objective.
- **Scaling Laws** are the empirical principle that performance predictably improves with more data, parameters, and compute.
- This scaling gives rise to **emergent abilities** not seen in smaller models.
- Training at scale is a massive engineering challenge requiring data, model, and pipeline parallelism.
- Extending models to handle **long context** is a key research area, addressing the quadratic complexity of attention and the limitations of positional embeddings.



Unlocking Knowledge without Retraining

Prompting is the art of designing the input text given to an LLM to elicit a desired output. It is the primary way we interact with and control pre-trained foundation models.

The Power of Prompts

A well-crafted prompt can guide an LLM to perform complex tasks, access specific knowledge, adopt a certain persona, and follow constraints—all without updating a single model weight. This is the essence of **prompt engineering**.



Learning from Examples in the Prompt

ICL is an emergent ability where an LLM learns to perform a task simply by seeing a few examples (demonstrations) in its input prompt.

Zero-shot Prompting

Provide only the task instruction. The model relies entirely on its pre-trained knowledge.

Translate to French:

Sea otter = ?

Few-shot Prompting

Provide the instruction plus a few input-output examples.

Translate to French:

Sea otter = ? loutre de mer

cheese = ? fromage

what is your name = ?



Guiding the LLM to "Think Step-by-Step"

CoT prompting dramatically improves performance on tasks requiring multi-step reasoning by explicitly asking the model to generate the intermediate steps.⁴

Q: The cafeteria had 23 apples. If they used 20 for lunch and bought 6 more, how many apples do they have?

A: 29. (*Incorrect*)

Q: The cafeteria had 23 apples. If they used 20 for lunch and bought 6 more, how many apples do they have?

A: The cafeteria started with 23 apples. They used 20, so they had $23 - 20 = 3$. Then they bought 6 more, so they have $3 + 6 = 9$. The answer is **9**. (*Correct*)

⁴Jason Wei, Xuezhi Wang, et al. (2022). "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in Neural Information Processing Systems 35*, pp. 24824–24837.



Robustness Through Majority Vote

Self-consistency improves on CoT by sampling multiple diverse reasoning paths and then taking the most frequent answer as the final output.⁵

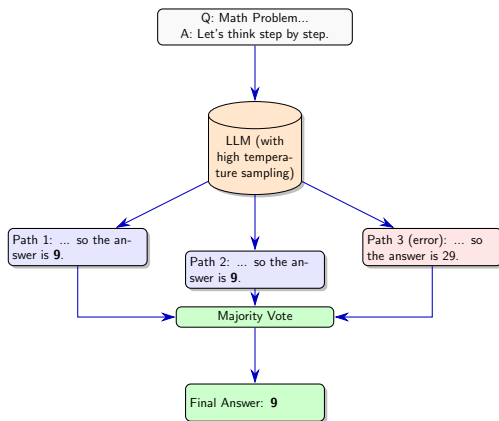


Figure: This makes the final answer more robust by marginalizing out incorrect reasoning.

⁵Xuezhi Wang et al. (2022). "Self-consistency improves chain of thought reasoning in language models". In: *arXiv preprint arXiv:2203.11171*.

Augmenting LLMs with External Knowledge

LLMs can be prompted to call external tools (calculator, search engine, or code interpreter) to overcome their limitations (e.g., knowledge cutoffs, arithmetic errors).

Retrieval-Augmented Generation (RAG)

A specific form of tool use where the LLM uses a retrieval system to find relevant documents and uses them as context to answer a question.

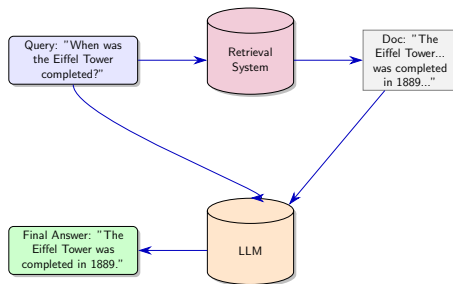


Figure: The RAG process: query → retrieve → augment prompt → generate.



Beyond Human-Written Text

Instead of manually engineering discrete text prompts ("hard prompts"), we can learn continuous vector representations ("soft prompts") that are optimized directly for a task.

- **Prompt Tuning:** A small set of trainable "soft prompt" vectors are prepended to the input sequence's embeddings.
- **Prefix Tuning:** A more powerful variant where trainable prefixes are added to the key and value vectors in every Transformer layer.
- During fine-tuning, the main LLM parameters are frozen, and only these small sets of prompt parameters are updated.
- This is a highly **parameter-efficient fine-tuning (PEFT)** method.



Chapter 3 Summary

- Prompting is the primary method for interacting with pre-trained LLMs without changing their weights.
- **In-Context Learning** allows LLMs to learn tasks from a few examples provided in the prompt.
- **Chain-of-Thought (CoT)** and **Self-Consistency** are advanced techniques that elicit step-by-step reasoning, significantly improving performance on complex tasks.
- LLMs can be augmented with **external tools** (like search engines in RAG) to overcome their inherent limitations.
- **Soft prompts** (e.g., Prompt/Prefix Tuning) offer a parameter-efficient alternative to full fine-tuning by learning continuous prompt vectors instead of discrete text.



Helpful, Honest, and Harmless

A pre-trained base LLM is a "next-word predictor," not a helpful assistant. It may generate outputs that are factually incorrect, unhelpful, toxic, biased, or that reveal private information.

What is Alignment?

Alignment is the process of fine-tuning a base model to align its behavior with human goals, preferences, and ethical principles. It's about teaching the model *how* to behave, not just what it knows.



A Multi-Stage Process

Modern alignment typically follows a three-step process after pre-training.

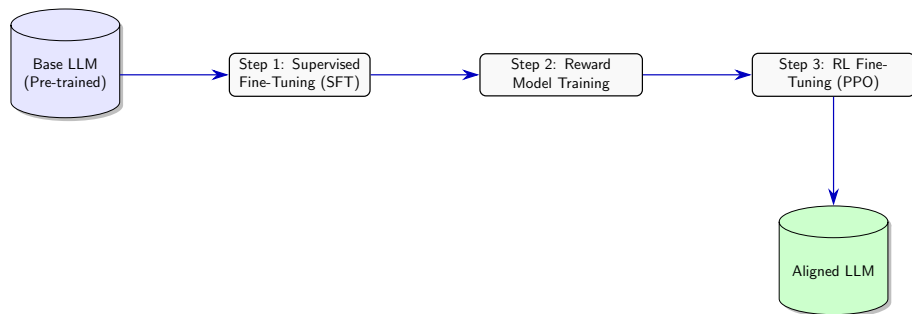


Figure: From a general-purpose predictor to a specialized, aligned assistant.



Teaching the Model to Follow Instructions

SFT adapts the base LLM to act as a helpful assistant by training it on a curated, high-quality dataset of instruction-response pairs.

- **Data:** Consists of prompts (instructions) and desired outputs, often created by human labelers.
- **Objective:** Standard language modeling loss, but computed only on the tokens of the desired response, not the input prompt.
- **Outcome:** The model learns the format of a conversation and how to follow explicit instructions. It's the first step towards helpfulness.

This step alone can significantly improve a model's utility, but it doesn't capture nuanced human preferences.



Learning Human Preferences

It's easier for humans to compare two outputs than to write a perfect one from scratch. The RM is trained to capture this preference signal.

- 1 Take a prompt and generate several responses from the SFT model.
- 2 Have human labelers rank the responses from best to worst.
- 3 This creates a preference dataset of comparisons (e.g., for a given prompt, $y_{chosen} \succ y_{rejected}$).
- 4 Train a **reward model** (RM) that takes a prompt and a response, and outputs a scalar score (a reward). The RM is trained to give a higher score to the chosen response than the rejected one.

This RM now serves as an automated, scalable proxy for human feedback.⁶

⁶Long Ouyang et al. (2022). "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems* 35, pp. 27730–27744.

From Human Ranking to a Scalar Score

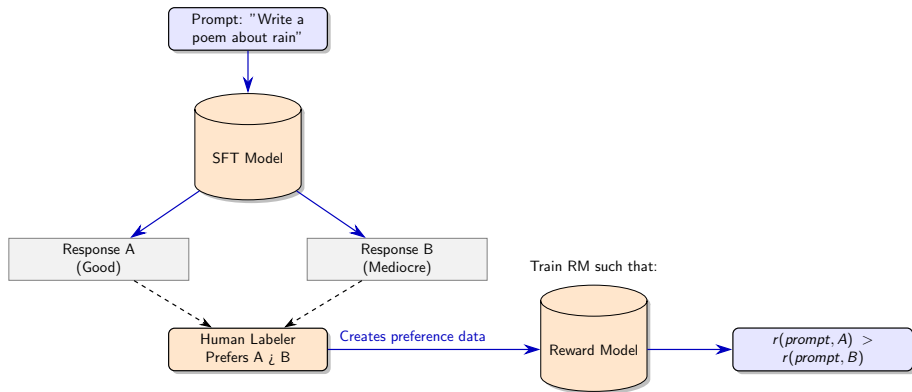


Figure: The reward model learns a scalar function that aligns with human judgments.



Optimizing the LLM with the Reward Model

In the final step, the SFT model is further fine-tuned using RL.

- The **LLM is the policy**, which takes a state (prompt) and produces an action (sequence of tokens).
- For a given prompt, the policy generates a response.
- The **Reward Model provides the reward**: a scalar score for that response.
- Policy's weights are updated to maximize the expected reward from the RM.

The LLM will generate outputs that humans would prefer.



Proximal Policy Optimization

PPO is the de-facto RL algorithm for LLM alignment. It prevents the policy from changing too drastically during training, which avoids catastrophic forgetting and stabilizes the process.⁷

The PPO Objective

The objective function maximizes the reward, but adds a **KL-divergence penalty** term.

$$\text{Objective} = \mathbb{E}_{(x,y) \sim \pi_{\theta}} [r(x,y)] - \beta \cdot \text{KL}(\pi_{\theta}(\cdot|x) || \pi_{\text{SFT}}(\cdot|x))$$

- This penalty keeps the updated policy (π_{θ}) from straying too far from the original SFT policy (π_{SFT}).
- This ensures the model doesn't "cheat" to get a high reward by generating gibberish that happens to score well, and it retains the general language capabilities learned during pre-training and SFT.

⁷John Schulman et al. (2017). "Proximal policy optimization algorithms". [arXiv preprint arXiv:1707.06347](#).



Bypassing the Reward Model

DPO is a more recent and simpler method that achieves alignment without explicitly training a reward model.⁸

- **Key Idea:** The optimal RLHF policy can be extracted directly from the preference data.
- DPO uses the human preference data ($y_{chosen} \succ y_{rejected}$) to directly fine-tune the LLM with a simple classification-like loss.
- The loss function encourages the model to assign a higher likelihood to the chosen responses than the rejected ones.
- This simplifies the complex, multi-stage, and often unstable RLHF pipeline into a single stage of supervised fine-tuning.

⁸Rafael Rafailov et al. (2024). "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems* 36.



Chapter 4 Summary

- **Alignment** is a crucial post-pre-training step to make LLMs helpful, honest, and harmless.
- The standard pipeline begins with **Supervised Fine-Tuning (SFT)** on instruction-response pairs to teach the model how to act as an assistant.
- **Reinforcement Learning from Human Feedback (RLHF)** further refines the model.
 - A **Reward Model** is trained on human preference data.
 - The LLM policy is then optimized using **PPO** to maximize the score from this reward model, with a KL penalty to ensure stability.
- **Direct Preference Optimization (DPO)** is a simpler and more stable alternative that achieves a similar outcome by directly fine-tuning on preference data, bypassing the need for an explicit reward model and RL.








Questions?

Summary: The Four Pillars of LLMs

- **Pre-training** builds the foundation by learning general world knowledge from unlabeled text.
- **Generative Models and Scaling Laws** define the architecture and the principle that "bigger is better," leading to emergent abilities.
- **Prompting** is the lightweight, powerful interface for unlocking and directing these abilities without retraining.
- **Alignment** is the critical final step that makes models useful, safe, and attuned to human intent.








Bibliography I

-  Schulman, John et al. (2017). “Proximal policy optimization algorithms”. In.
-  Radford, Alec et al. (2018). “Improving language understanding by generative pre-training”. In: *URL*
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.
-  Devlin, Jacob et al. (2019). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.
-  Brown, Tom B et al. (2020). “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33, pp. 1877–1901.
-  Clark, Kevin et al. (2020). “Electra: Pre-training text encoders as discriminators rather than generators”. In: *International Conference on Learning Representations*.



Bibliography II

-  Ouyang, Long et al. (2022). “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35, pp. 27730–27744.
-  Wang, Xuezhi et al. (2022). “Self-consistency improves chain of thought reasoning in language models”. In: *arXiv preprint arXiv:2203.11171*.
-  Wei, Jason, Yi Tay, et al. (2022). “Emergent abilities of large language models”. In: *Transactions on Machine Learning Research*.
-  Wei, Jason, Xuezhi Wang, et al. (2022). “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in Neural Information Processing Systems* 35, pp. 24824–24837.
-  Rafailov, Rafael et al. (2024). “Direct preference optimization: Your language model is secretly a reward model”. In: *Advances in Neural Information Processing Systems* 36.

