Lecture 14

LLMs, RLHF, Instruct GPT

Core Challenge with Language Models:

- **GPT models predict the next token based on historical data, lacking an innate ability to follow instructions.**

- GPT (Generative Pre-trained Transformer) models, at their core, predict the next word or token in a sequence based on the probabilities derived from pre-training on extensive text corpora.
- They don't inherently "understand" instructions or follow commands but generate what's statistically likely to come next, given their training.

The Challenge with Large Language Models (LMs)

- **Capability:** LMs can be prompted to perform a range of NLP tasks.
- **Issue:** They often exhibit unintended behaviors:
  - Making up facts
  - Generating biased or toxic text
  - Not adhering to user instructions
- **Reason:** The language modeling objective (predicting the next token) is misaligned with the goal of "following user's instructions helpfully and safely."

Alignment in Language Models

- **Objective**: Train LMs to act in accordance with the user's intention.
- **Explicit Intentions**: Following instructions.
- **Implicit Intentions**: Staying truthful, avoiding bias, toxicity, or harm.

Core Principles for Aligned LMs:

- **Helpful**: Assist users effectively.
- **Honest**: Offer accurate, non-misleading information.
- **Harmless**: Prioritize user safety and avoid harm.

The Goal of Alignment:

- **The aim is to align GPT's responses with specific user instructions and ethical standards, beyond just generating probable text.**

- The primary objective is to bridge the gap between these statistical predictions and meaningful adherence to instructions provided by users.
- This involves ensuring that the AI's responses are not just contextually appropriate or conversationally relevant but also aligned with the specific intentions, ethical expectations, and task-oriented goals of the user.

Why It Matters:

- **This alignment is crucial for enhancing GPT's reliability, ensuring it respects user intent and ethical norms.**

- Without this alignment, while a model like GPT might produce grammatically correct and contextually relevant content, it might diverge from user instructions or produce content that's inappropriate or misaligned with the user's ethical, cultural, or personal expectations.
- The goal is to enhance the model's reliability in following directives accurately, respecting ethical boundaries, and fulfilling the user's actual needs, thereby making the technology more trustworthy and effective in real-world applications.

Modifying Language Model Behavior

- Multiple strategies have been developed to refine the generation behavior of language models.

Filtering Pretraining Datasets

- **What**: Removing or excluding certain documents or content from the dataset used for initial training.
- **Why**: Prevents the model from learning harmful or undesirable patterns present in those documents.
- **Example**: Excluding documents that contain hate speech or misinformation to reduce the chances of the model generating such content.

Fine-Tuning on Value-Targeted Datasets

- **What**: Adjusting the model's parameters using a dataset that emphasizes specific values or behaviors.
- **Why**: Helps the model better adhere to desired values in its outputs.
- **Example**: If we want a model to generate environmentally-conscious responses, we might fine-tune it on a dataset full of eco-friendly content.

Key Areas of Focus in AI Alignment

1.Learning from Human Feedback:

- Tailoring AI through human interaction.

2. Training to Follow Instructions:

- Teaching AI specific task adherence.

3. Evaluating AI Harms:

- Identifying risks in AI outputs.

4. Modifying Behavior to Mitigate Harms:

- Adjusting AI to prevent negative impacts.

Key Areas of Focus in AI Alignment

1.Learning from Human Feedback:

- Tailoring AI through human interaction.

2. Training to Follow Instructions:

- Teaching AI specific task adherence.

3. Evaluating AI Harms:

- Identifying risks in AI outputs.

4. Modifying Behavior to Mitigate Harms:

- Adjusting AI to prevent negative impacts.

Addressing GPT Challenges - Training Strategy Overview

- To address inherent challenges with GPT models, ChatGPT's training strategy mirrors the "Instruct GPT" approach, itself an amalgamation of strategies from preceding works.
- **The Three-Phased Training Approach:**

1.Supervised Fine-Tuning (SFT):

- "Refines a pre-trained GPT-3 model's responses for specific tasks or guidelines, enhancing its understanding and output relevance.“

2. Training a Reward Model (RM):

- "Develops a system that assesses the quality of text generated by the model, guiding it towards human-preferred responses."

3. Reinforcement Learning from Human Feedback (RLHF)

The Three-Phased Training Approach

1.Supervised Fine-Tuning (SFT):

- Refines a pre-trained GPT-3 model's responses for specific tasks or guidelines, enhancing its understanding and output relevance.

2.Training a Reward Model (RM):

- Develops a system that assesses the quality of text generated by the model, guiding it towards human-preferred responses.

3. Reinforcement Learning from Human Feedback (RLHF)

- Refining AI behavior through direct human feedback.

1. Supervised Fine-Tuning (SFT)

Data Collection Approach

- **Source**: OpenAI API with InstructGPT.
- **Human-in-the-loop**: Integral part of the process to ensure quality and diversity.
- **Goal**: Capture a broad spectrum of language patterns and ensure diversity in prompts.

Data Collection Approach

- **Writing Prompts & Responses**:

- Labelers generate diverse prompts and craft human-written responses.

Introduction to Labeler-Written Prompts

- **Objective**: Train the initial InstructGPT model with instruction-like prompts.
- **Source**: Prompts crafted by contractors (labelers).
- **Types of Prompts**: Plain, Few-shot, User-based.

Types of Labeler-Written Prompts

1. **Plain Prompts**:
    1. Goal: Generate arbitrary tasks ensuring diversity.
    2. Example: "Describe the process of evaporation."
2. **Few-shot Prompts**:
    1. Goal: Provide an instruction with multiple query/response pairs.
    2. Example:
        1. **Instruction**: "Give the sentiment for a tweet."
        2. **Queries**: Sample tweets.
        3. **Responses**: "Positive" or "Negative."

Types of Labeler-Written Prompts

3.User-based Prompts:

- Derived from real-world applications to the OpenAI API.
- Reflect specific needs or interests of potential users.
- Example:
  - Original Application: "I want to use the API for a bird identification app."
  - User-based Prompt: "Describe characteristics of a common bird."

User-based vs Plain Prompts

- Both prompts are crafted by labelers.
- User-based prompts are inspired by real-world applications, ensuring they align with practical needs.
- Plain prompts are more open-ended and not tied to any specific application.

Illustrative User Prompts

1. **Brainstorming**
   Example: "List five ideas for how to regain enthusiasm for my career."
2. **Classification**
   Example: "Given the following text, rate, on a scale from 1-10, how sarcastic the person is being."
3. **Extract**
   Example: "Extract all course titles from the table below:
   |Title|Lecturer|Room|"
4. **Generation**
   Example: "Write a short story where a brown bear goes to the beach, makes friends with a seal, and then returns home."

Illustrative User Prompts

5. Chat

Example: "This is a conversation with an AI assistant. The assistant is helpful, creative, clever, and very friendly. Human: 'Hello, who are you?' AI: 'I am an AI created by OpenAI. How can I help you today?’”

6. Rewrite

Example: "Translate this sentence to Spanish: <English sentence>"

7. Closed QA

Example: "Answer the following question: What shape is the earth?

A) A circle B) A sphere C) An ellipse D) A plane"

Illustrative User Prompts

8. Open QA

Example: "Who built the statue of liberty?"

9. Summarization

Example: "Summarize this for a second-grade student: {text}"

10. Other

Example: "Lookup 'cowboy' on Google and give me the results."

Supervised Fine-Tuning (SFT)

- **Objective**: Train the model to generate desired responses to given prompts.
- **Method**:
  - **Concatenate the prompt and the desired response.**
  - Use this concatenated text as input for the model.
  - Train the model to predict the next token in the sequence.

Supervised Fine-Tuning (SFT)

- **Example**:
  - **Prompt**: "What's the capital of France?"
  - **Desired Response**: "Paris."
  - **Concatenated Input**: "What's the capital of France? Paris."
- **Outcome**: The model learns to generate concise and accurate responses to a wide range of prompts.

2. Training a Reward Model (RM)

Dataset Creation for RM

1. **Generating Responses**: For a given prompt, produce multiple responses using the SFT model.
2. **Pairing & Ranking**: Create pairs from these responses and have a human rank the better response in each pair.

- **Prompt**: "What is ice?"
- **Responses**:
  - A. "It's the solid form of water."
  - B. "Frozen water that's cold to touch."
  - C. "Water that has turned solid due to low temperatures."
  - D. "A crystalline substance formed when water freezes."

Dataset Creation for RM

Create Response Pairs: Pair the responses with each other.

- Total Pairs: $C(4,2) = 6$ pairs
- Example Pairs:
  1. (A, B)
     A: It's the solid form of water.
     B: Frozen water that's cold to touch.

  2. (A, C)
  3. (A, D)
  4. (B, C)
  5. (B, D)
  6. (C, D)

Human Ranking: Ask a human to rank responses within each pair.

- For pair (A, B), the human might prefer response A over B.

"It's the solid form of water."
"Frozen water that's cold to touch."

"It's the solid form of water."

"Frozen water that's cold to touch."

"Water that has turned solid due to low temperatures."
"A crystalline substance formed when water freezes."

"Water that has turned solid due to low temperatures."

"A crystalline substance formed when water freezes."

- **Total Prompts Used**: 33,000
- **Responses per Prompt**: Between 4 to 9
- **Sample Calculation**:
- For 4 responses per prompt: $33{,}000 \times C(4,2) = 198{,}000$ pairs
- For 9 responses per prompt: $33{,}000 \times C(9,2) = 1{,}188{,}000$ pairs

Reward Model (RM)

- **Objective**: Align model generation with human preference.
- **Method**: Rate model responses based on human preference.
- **Outcome**: Train a model (Reward Model) to simulate these ratings.

Reward Model Architecture

- **Starting Point**: Use the SFT model from Phase 1.

Modifications:

  - Remove the last linear layer (unembedding layer).
  - Add a randomly initialized linear layer.
  - The model now outputs a scalar value, effectively becoming a regressor.

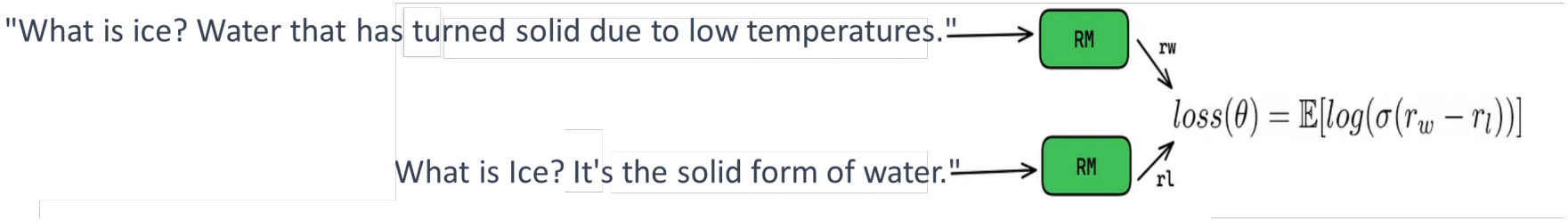
Training Regime for Reward Model

- **Pair Selection**:

- **Losing Pair**: "What is Ice? It's the solid form of water."
- **Winning Pair**: "What is ice? Water that has turned solid due to low temperatures."
-

Reward Calculation

1. **Winning Reward**: Pass the winning prompt-response to the reward model.
2. **Losing Reward**: Pass the losing prompt-response to the same model.
3. **Difference**: Calculate the difference between the two rewards.



Loss Function

$$\text{loss}(\theta) = \mathbb{E}\left[\log\left(\sigma(r_w - r_l)\right)\right]$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- The loss represents the difference in log-odds of a high reward (winning) and low reward (losing) response.
- Intuitively, the model aims to maximize the difference between the winning and losing responses.

Inference with Reward Model

- **Post-Training**: For any prompt-response combination, get a scalar reward.
- **Purpose**: Simulate human preference for a given prompt-response.

3. Reinforcement Learning from Human Feedback (RLHF)

Introduction to Reinforcement Learning Model (RL)

- **Objective**: Train a model that generates text aligning with human preferences.
- **Components**:
  - SFT Model: Generates responses but may not align with human preferences.
  - Reward Model: Provides a scalar reward but doesn't generate text.
- **Goal of RL**: Combine the capabilities of the above models to produce text that maximizes the reward.

Aligning Generation with Human Preference

- The RL model's aim is to maximize the reward.
- This can be formulated mathematically as an objective maximization problem.
- Objective:

$$J(w) = \mathbb{E}_{x \sim p_w}[r(x)]$$

where:

- $J(w)$ is the objective function.
- x is the generated response.
- $r(x)$ is the reward for response x.

Training the RL Model

- Use gradient ascent to maximize the objective function.
- Gradient Ascent Update:

$$w \leftarrow w + \alpha \nabla_w J(w)$$

where:

- α is the learning rate.
- $\nabla_w J(w)$ is the gradient of the objective function.

With some mathematical manipulations, the gradient expression becomes:

$$\nabla_w J(w) = \mathbb{E}_{x \sim p_w} \left[r(x) \nabla_w \log p_w(x) \right]$$

$$= \frac{1}{D} \sum_{D} \sum_{t=0}^{T} \nabla_w r(x) \log p_w(x_t | x_{t-1}, x_{t-2}, \ldots)$$

where:

- D refers to all the prompt-response pairs.

Generating Text with the Model

1. The model generates a token x_t based on the previous tokens $x_0, x_1, \ldots, x_{t-1}$.
2. It continues to generate tokens until the sequence is complete.
3. After generating the entire sequence, the Reward Model (RM) evaluates it and assigns a reward $r(x)$.
4. This reward informs the gradient for the policy gradient update.

Note: During sequence generation, the model is unaware of the sequence's reward. It learns the reward only post-generation, using it to refine its parameters for better future sequences.

Model, Dataset, and Training Regime

- **Model**: The model structure remains similar to the SFT model but is trained with the new objective.
- **Dataset**: Use the same dataset but now with the goal of maximizing the reward.
- **Training Regime**: Iteratively generate responses, compute rewards, and update the model using gradient ascent.

Exploring the Reinforcement Learning Phase

Model:

- Copy the SFT model trained in phase 1.
- No modifications needed.

Dataset:

- A collection of prompts.
- No responses attached.

Training Regime

1. Feed a prompt to the model.
2. The model generates a corresponding response.
3. Concatenate the prompt and response.
4. Pass this concatenated pair to the reward model to obtain a reward score.
5. Use the reward and probabilities in the gradient equation.
6. Update the model parameters using this gradient.

Challenge with Direct Model Updates

- Using the initial training method can make the model unpredictable.
- These updates can push the model to produce text that doesn't make sense or is off-topic.

Solution: KL Divergence

- Ensure updates don't deviate too much from the SFT trained in phase 1.
- Use KL divergence to measure the "distance" between the SFT and the RL model.
- KL divergence compares two distributions.

Training Process

1. For a given prompt, the RL model generates a response.
2. At each generation step, a probability distribution over the vocabulary is produced.
3. Feed the same prompt-response to the SFT model to get its probability distribution over the vocabulary.
4. Calculate the KL divergence between the distributions from the RL model and the SFT model.
5. Subtract this divergence value from the reward for the prompt-response pair.

Subtract this divergence value from the reward

$$J(p_w) = \mathop{\mathbb{E}}_{x \sim p_w} [r(x) - \beta \mathrm{KL}(p_w(x), p_{SFT}(x))]$$

- $p_w(x)$ is the probability distribution over the vocabulary produced by the RL model for the sequence x.
- $p_{SFT}(x)$ is the probability distribution over the vocabulary produced by the original SFT model for the sequence x.
- β is a hyperparamete.

Issue with the Cost Function

Problem:

- The model's performance on some datasets was inferior to the original pretrained model.

Solution:

- Introduce an additional term in the cost function to keep the RL model close to the original pretrained base.

Updated Cost Function

$$J(p_w) = \mathop{\mathbb{E}}_{x \sim p_w} \left[r(x) - \beta \mathrm{KL}\left(p_w(x), p_{SFT}(x)\right) \right] + \gamma \mathop{\mathbb{E}}_{x \sim \text{pretrain}} \left[\log\left(p_w(x)\right) \right]$$

Where:

- p_w is the RL model with parameters w.
- γ is a hyperparameter, ensuring the RL model aligns with the original pretrained base.

In Conclusion

The model is optimized to:

- Maximize reward obtained using the reward model.
- Minimize divergence from SFT model.
- Maximize likelihood of observing sequences seen during pretraining.

After updating the RL model for some steps:

- Initialize a new reward model from it.
- Ask humans to generate another dataset following the procedure described in Phase 2.
- Conduct another training session for the reward model.

LLaMA

Overview

- Released by Meta AI in February 2023.
- Available in sizes: 7B, 13B, 33B, 65B, and 70B parameters.

LLaMA-2 vs LLaMA-1

- LLaMA-2 uses 40% more training data.
- Architecture remains largely unchanged.

LLaMA

Open Source & Accessibility:

- Inference code under GPL 3 license.
- Model weights access via application process.
- Trained on 20 major languages, focusing on Latin and Cyrillic alphabets.

Coding: "Code Llama"

- fine-tuned for coding tasks.
- Sizes: 7B, 13B, 34B parameters.

Stanford Alpaca

- **Dataset**:
  - Fine-tuned from LLaMA 7B on 52K instruction-following samples.
  - Comparable to OpenAI's text-davinci-003 but more cost-effective.

- **Integration with LLaMA**:
  - Used to fine-tune LLaMA v1.
  - Enhances LLaMA v1 by adding instruction-following capabilities (which it lacked by default).

https://crfm.stanford.edu/2023/03/13/alpaca.html

Open Assistant

- **Mission**: To democratize language technologies and innovations.
- **Open & Collaborative**: Provides unrestricted access to:
  - Datasets
  - Models
  - Code sources
- **Platform**: Open Assistant platform is designed for seamless interaction and collaboration.
- **Philosophy**: Emphasizes open access and community collaboration, ensuring that advancements in language technologies are accessible to all.

Open Assistant

https://github.com/LAION-AI/Open-Assistant

https://www.kdnuggets.com/2023/04/open-assistant-explore-possibilities-open-collaborative-chatbot-development.html

LLaMA v2

Variants:

- Multiple sizes available:
  - 7B (Billion parameters)
  - 13B
  - 70B

Versions:

- Chat: Features instruction-following capability.
- Non-Chat: Traditional language model without instruction-following.

Falcon

- **Model**:
  - Largest Version: 180B (Billion parameters)
  - Trained on a massive 3.5 trillion tokens.

- **Dataset**: Falcon's dataset is released for public use
- https://falconllm.tii.ae/index.html
- https://huggingface.co/datasets/tiiuae/falcon-refinedweb
- https://arxiv.org/pdf/2306.01116.pdf
- https://huggingface.co/tiiuae/falcon-180B
-

Variations of Transformer Models for Enhanced Efficiency

- The Transformer model, though powerful, is known for its computational complexity.
- Computational Complexity: $O(n^2\ d)$for self-attention, where n is the sequence length, and d is the dimensionality of the input.
- Memory Complexity: $O(n^2 + \ n\ \ d)$

Reformer

- Introduces a new method to reduce memory requirements while maintaining performance.
- Key innovation: Locality-Sensitive Hashing (LSH) to reduce the complexity of the attention mechanism.

- Computational Complexity: $O(n\ \log(n)\ d)$
- Memory Complexity: $O(n\ d)$

Lformer

- Focuses on reducing time complexity of self-attention, making it more efficient for long sequences.
- Key innovation: Linear time complexity self-attention mechanism.

- Computational Complexity: $O(n\ d)$
- Memory Complexity: $O(n\ d)$

Performer

- Replaces self-attention with a more efficient approximation.
- Key innovation: Fast Attention Via positive Orthogonal Random features (FAVOR+) allowing faster training and inference times.

- Computational Complexity: $O(n\ d)$
- Memory Complexity: $O(n\ d)$

Longformer

- Extends Transformer to handle long sequences more efficiently.
- Key innovation: Sparse attention mechanism with a mix of local and global attentions.

- Computational Complexity: $O(n\ d\ +\ n\ w)$, where w is the window size of local attention.
- Memory Complexity: $O(n\ d\ +\ n\ w)$

win Transformer

- Specifically designed for computer vision tasks.
- Key innovation: Hierarchical feature maps from smaller-sized patches and merges them for efficiency.

- Computational Complexity: Varies based on hierarchical structure, generally lower than standard Transformers.
- Memory Complexity: Also varies, generally lower.

ALBERT

- Reduces memory consumption by lowering the number of parameters.
- Key innovation: Parameter sharing across layers and separating vocabulary embedding into two smaller matrices.

- Computational Complexity: $O(n^2\ d)$, but with fewer parameters due to sharing.
- Memory Complexity: $O(n^2 + \ n\ \ d)$, but with fewer parameters.

DeBERTa

- Adds a disentangled attention mechanism to improve training efficiency.
- Key innovation: Disentangled attention mechanism allowing for more flexible and efficient attention computation.

- Computational Complexity: Similar to original Transformer but with efficiency in training.
- Memory Complexity: Similar to original Transformer.