Advanced Methods in Natural Language Processing

Session 9: LLMs Basics

Arnault Gombert

May 2025

Barcelona School of Economics

Introduction

Introduction to Today's Lecture on LLMs and ChatGPT

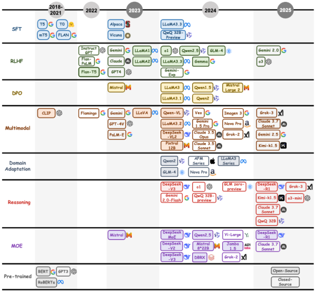
Today, we delve into the world of LLMs such as ChatGPT, exploring their advancements, applications, and the intricacies of prompt engineering, fine-tuning, and retrieval-augmented generation (RAG).

Session Overview:

- **Overview of LLMs:** Introducing text-only and multimodal models, and their evolution since 2022.
- **Main Use Cases:** Exploring the diverse applications of LLMs in various domains.
- **Prompt Engineering:** Understanding the art of effectively communicating with LLMs to achieve desired outcomes.
- **Retrieval-Augmented Generation (RAG):** Leveraging external knowledge bases to enhance LLMs' responses.
- **Fine-Tuning Techniques:** Techniques to customize LLMs for specific tasks or datasets.

Training of ChatGPT: Process and Innovations

- **Foundation Model**: ChatGPT builds upon a large transformer-based language model, similar to GPT-3, trained on a diverse range of internet text.

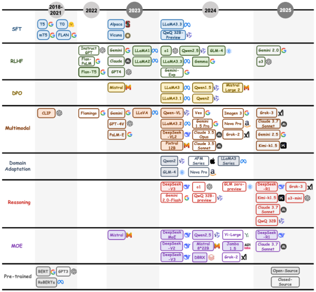


Evolution of Texts LLM

Tie et al. (2025)

Training of ChatGPT: Process and Innovations

- **Foundation Model**: ChatGPT builds upon a large transformer-based language model, similar to GPT-3, trained on a diverse range of internet text.
- **Reinforcement Learning from Human Feedback (RLHF)**:

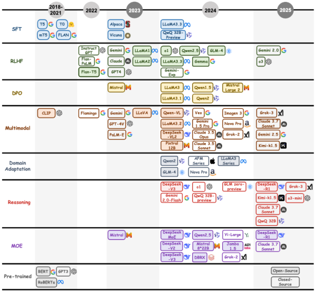


Evolution of Texts LLM

Tie et al. (2025)

Training of ChatGPT: Process and Innovations

- **Foundation Model**: ChatGPT builds upon a large transformer-based language model, similar to GPT-3, trained on a diverse range of internet text.
- **Reinforcement Learning from Human Feedback (RLHF)**:
  - *Supervised Fine-Tuning (SFT)*: Initial fine-tuning on a dataset of conversational prompts and responses.

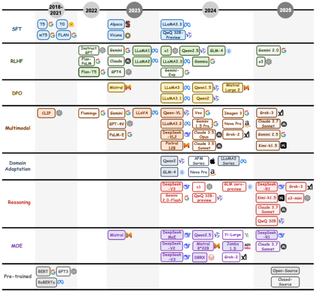


Evolution of Texts LLM

Tie et al. (2025)

Training of ChatGPT: Process and Innovations

- **Foundation Model**: ChatGPT builds upon a large transformer-based language model, similar to GPT-3, trained on a diverse range of internet text.
- **Reinforcement Learning from Human Feedback (RLHF)**:
  - *Supervised Fine-Tuning (SFT)*: Initial fine-tuning on a dataset of conversational prompts and responses.
  - *Reward Modeling (RM)*: Trained a reward model to predict scores given by human trainers for model-generated responses.

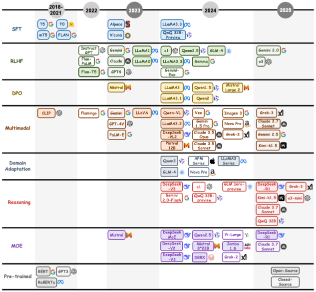


Evolution of Texts LLM

Tie et al. (2025)

Training of ChatGPT: Process and Innovations

- **Foundation Model**: ChatGPT builds upon a large transformer-based language model, similar to GPT-3, trained on a diverse range of internet text.
- **Reinforcement Learning from Human Feedback (RLHF)**:
  - *Supervised Fine-Tuning (SFT)*: Initial fine-tuning on a dataset of conversational prompts and responses.
  - *Reward Modeling (RM)*: Trained a reward model to predict scores given by human trainers for model-generated responses.
  - *Proximal Policy Optimization (PPO)*: Final fine-tuning phase using the reward model to guide training toward human preferences.

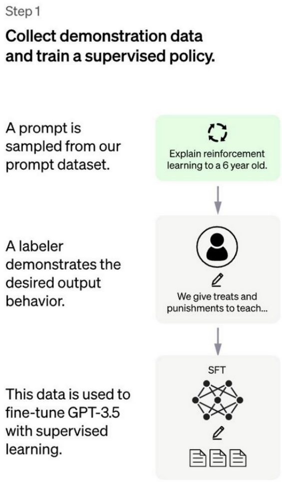


Evolution of Texts LLM

Tie et al. (2025)

Supervised Fine-Tuning (SFT) in ChatGPT

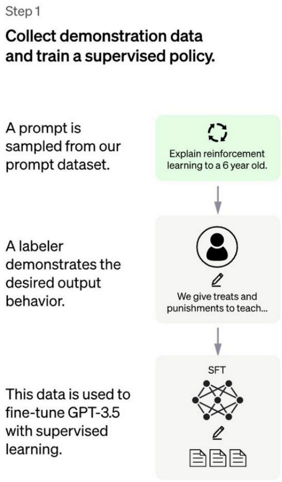
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

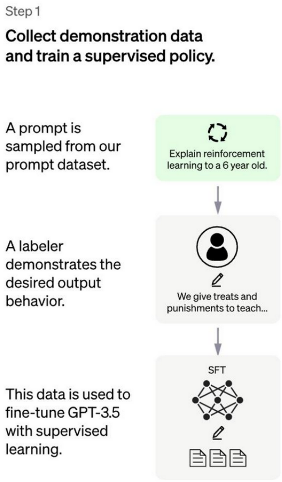
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

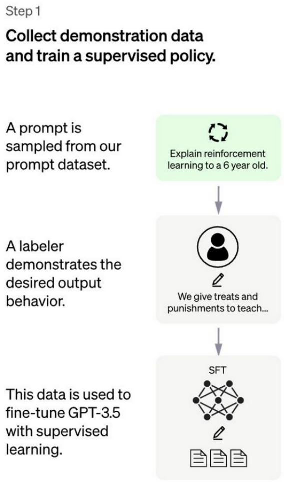
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:
  - Utilizes a curated dataset comprising diverse conversational prompts and corresponding human-written responses.



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

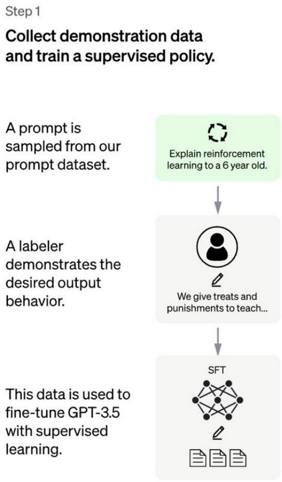
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:
  - Utilizes a curated dataset comprising diverse conversational prompts and corresponding human-written responses.
  - The model is fine-tuned to predict these responses accurately, aligning its outputs more closely with human-like conversational patterns.



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

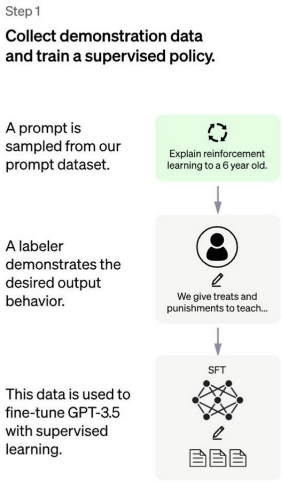
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:
  - Utilizes a curated dataset comprising diverse conversational prompts and corresponding human-written responses.
  - The model is fine-tuned to predict these responses accurately, aligning its outputs more closely with human-like conversational patterns.
- **Example**:



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

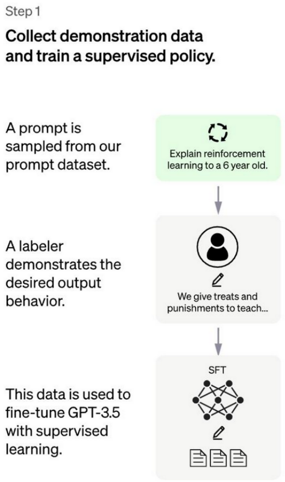
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:
  - Utilizes a curated dataset comprising diverse conversational prompts and corresponding human-written responses.
  - The model is fine-tuned to predict these responses accurately, aligning its outputs more closely with human-like conversational patterns.
- **Example**:
  - Prompt: "What's your favorite book and why?"



Supervised Fine-Tuning Phase

Supervised Fine-Tuning (SFT) in ChatGPT

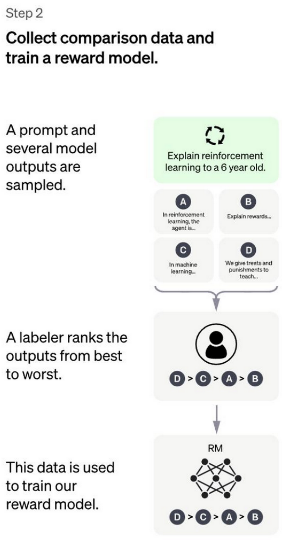
- **Objective**: Refine the foundational language model towards conversational understanding and response generation.
- **Process**:
  - Utilizes a curated dataset comprising diverse conversational prompts and corresponding human-written responses.
  - The model is fine-tuned to predict these responses accurately, aligning its outputs more closely with human-like conversational patterns.
- **Example**:
  - Prompt: "What's your favorite book and why?"
  - Model learns to generate engaging and contextually relevant responses.



Supervised Fine-Tuning Phase

Reward Modeling (RM) in ChatGPT

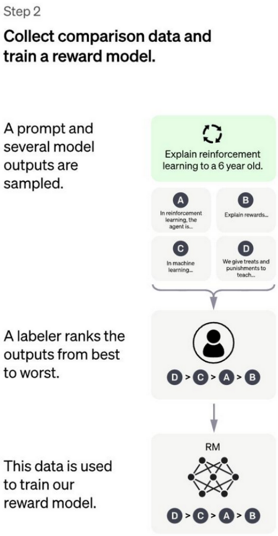
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

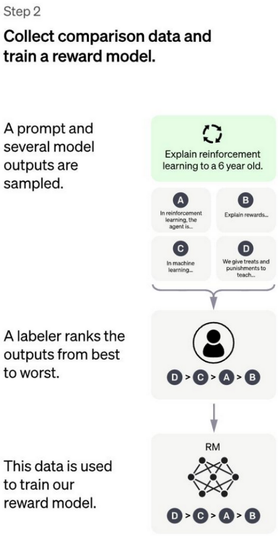
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

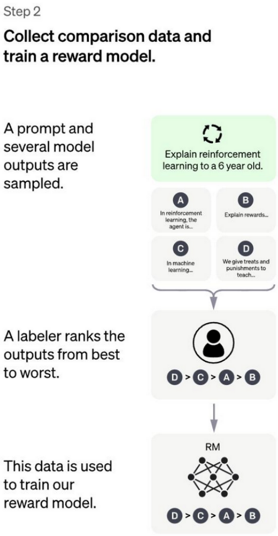
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:
  - Human trainers rate the quality of responses generated by the model, considering factors like relevance, coherence, and safety.



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

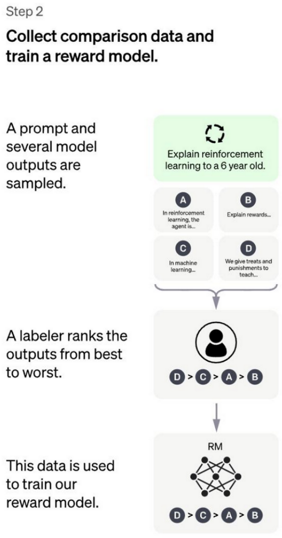
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:
  - Human trainers rate the quality of responses generated by the model, considering factors like relevance, coherence, and safety.
  - Use the ratings to train a separate reward model that learns to predict scores.



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

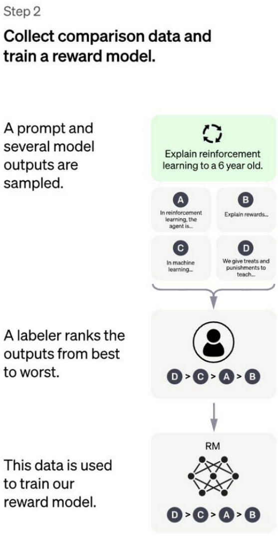
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:
  - Human trainers rate the quality of responses generated by the model, considering factors like relevance, coherence, and safety.
  - Use the ratings to train a separate reward model that learns to predict scores.
- **Example**:



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

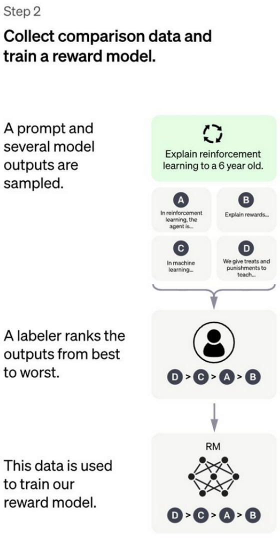
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:
  - Human trainers rate the quality of responses generated by the model, considering factors like relevance, coherence, and safety.
  - Use the ratings to train a separate reward model that learns to predict scores.
- **Example**:
  - Response: "My favorite book is 'To Kill a Mockingbird' because it tackles complex themes with compelling storytelling."



Reward Modeling Phase

Reward Modeling (RM) in ChatGPT

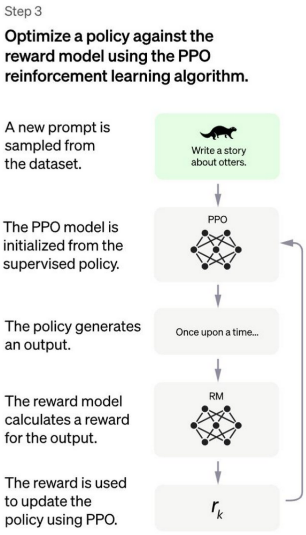
- **Objective**: Create a model to evaluate and score generated texts based on human preferences.
- **Process**:
  - Human trainers rate the quality of responses generated by the model, considering factors like relevance, coherence, and safety.
  - Use the ratings to train a separate reward model that learns to predict scores.
- **Example**:
  - Response: "My favorite book is 'To Kill a Mockingbird' because it tackles complex themes with compelling storytelling."
  - Reward model learns to score such responses for effectiveness and relevance.



Reward Modeling Phase

Proximal Policy Optimization (PPO) in ChatGPT

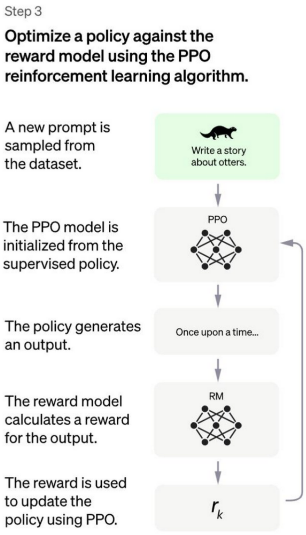
- **Objective**: Refine ChatGPT's responses to align with human preferences.



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

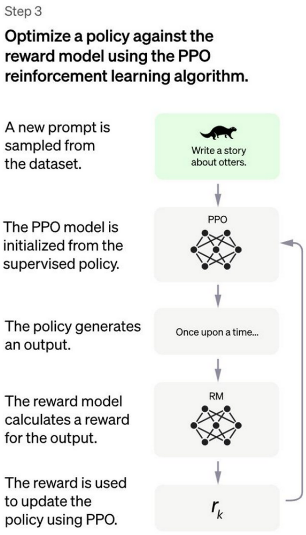
- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

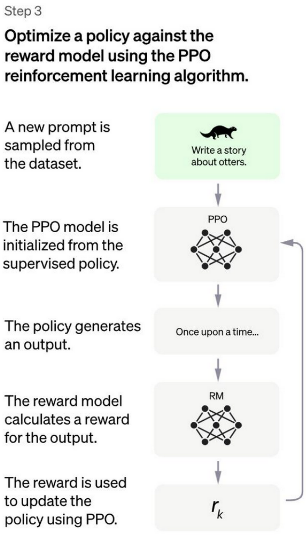
- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:
  - The model's training is guided by the reward model to generate responses that are likely to be scored highly.



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

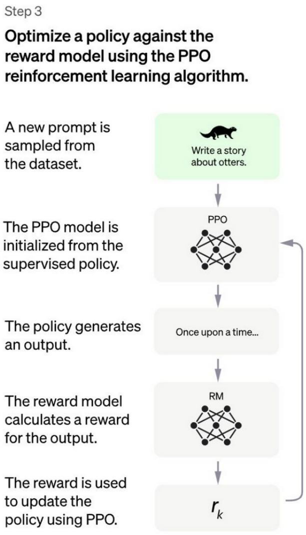
- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:
  - The model's training is guided by the reward model to generate responses that are likely to be scored highly.
  - Iteratively adjusts the model's parameters to maximize the expected reward from the reward model.



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

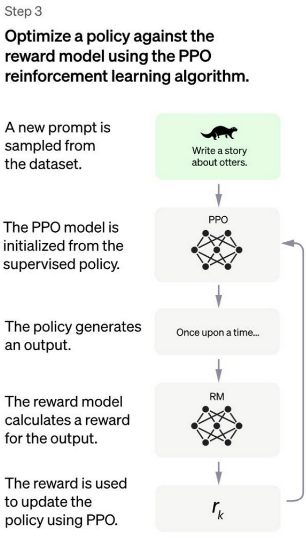
- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:
  - The model's training is guided by the reward model to generate responses that are likely to be scored highly.
  - Iteratively adjusts the model's parameters to maximize the expected reward from the reward model.
- **Example**:



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

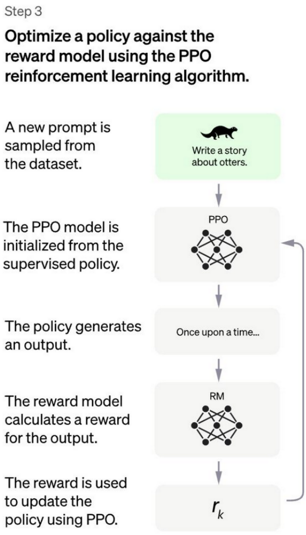
- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:
  - The model's training is guided by the reward model to generate responses that are likely to be scored highly.
  - Iteratively adjusts the model's parameters to maximize the expected reward from the reward model.
- **Example**:
  - The model generates a variety of responses to a prompt.



PPO Phase

Proximal Policy Optimization (PPO) in ChatGPT

- **Objective**: Refine ChatGPT's responses to align with human preferences.
- **Process**:
  - The model's training is guided by the reward model to generate responses that are likely to be scored highly.
  - Iteratively adjusts the model's parameters to maximize the expected reward from the reward model.
- **Example**:
  - The model generates a variety of responses to a prompt.
  - It then estimates the reward for each response and prefers choices that maximize this reward, leading to more human-aligned responses.



PPO Phase

Evolution of PPO: Newer Techniques

- **Direct Preference Optimization (DPO)**:
    - Directly optimizes for human preferences, avoiding reward model intermediaries.
    - Improves sample efficiency and stability of RL-based fine-tuning.

Evolution of PPO: Newer Techniques

- **Direct Preference Optimization (DPO)**:
  - Directly optimizes for human preferences, avoiding reward model intermediaries.
  - Improves sample efficiency and stability of RL-based fine-tuning.
- **Simple Preference Optimization (SimPO)**:
  - Focuses on stability and computational efficiency.
  - Balances expressiveness and practicality in preference learning.

Evolution of PPO: Newer Techniques

- **Direct Preference Optimization (DPO)**:
  - Directly optimizes for human preferences, avoiding reward model intermediaries.
  - Improves sample efficiency and stability of RL-based fine-tuning.
- **Simple Preference Optimization (SimPO)**:
  - Focuses on stability and computational efficiency.
  - Balances expressiveness and practicality in preference learning.
- **Others**:
  - You have KPO, ORPO, IPO, DOVE, RLAIF, SPIN...
  - You can take a look at Argilla's blog posts.

Training of ChatGPT: Process and Innovations

- **Differences from Previous Models**:

Training of ChatGPT: Process and Innovations

- **Differences from Previous Models**:
  - *Interactive Feedback*: Incorporation of dialogues and human interaction nuances.

Training of ChatGPT: Process and Innovations

- **Differences from Previous Models**:
  - *Interactive Feedback*: Incorporation of dialogues and human interaction nuances.
  - *Dynamic Learning*: Ability to learn from user interactions and adapt responses.

Training of ChatGPT: Process and Innovations

- **Differences from Previous Models**:
  - *Interactive Feedback*: Incorporation of dialogues and human interaction nuances.
  - *Dynamic Learning*: Ability to learn from user interactions and adapt responses.
  - *Ethical and Safety Considerations*: Enhanced focus on generating safe, ethical, and contextually appropriate responses.

Applications & other models

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
  - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
  - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.
  - Supports seamless code navigation and AI-enhanced coding workflows.

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
  - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.
  - Supports seamless code navigation and AI-enhanced coding workflows.
  - Provides a chat with agent or manual inputs directly within the editor.

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
  - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.
  - Supports seamless code navigation and AI-enhanced coding workflows.
  - Provides a chat with agent or manual inputs directly within the editor.
- **ROI**:

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
    - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.
    - Supports seamless code navigation and AI-enhanced coding workflows.
    - Provides a chat with agent or manual inputs directly within the editor.
- **ROI**:
    - Speeds up coding tasks with smart AI-based code generation.

Cursor: AI-Powered Code Editor

- **What it is**: A code editor with built-in AI assistance to enhance coding productivity.
- **How it works**:
  - Integrates a powerful LLM for code completions, refactoring, and natural language explanations.
  - Supports seamless code navigation and AI-enhanced coding workflows.
  - Provides a chat with agent or manual inputs directly within the editor.
- **ROI**:
  - Speeds up coding tasks with smart AI-based code generation.
  - Enhances the development process through quick code insights and debugging support.

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:
  - Combines the power of LLMs with a retrieval system to fetch relevant documents.

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:
  - Combines the power of LLMs with a retrieval system to fetch relevant documents.
  - Enhances the generation of responses by conditioning on retrieved documents.

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:
  - Combines the power of LLMs with a retrieval system to fetch relevant documents.
  - Enhances the generation of responses by conditioning on retrieved documents.
- **Usage**:

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:
  - Combines the power of LLMs with a retrieval system to fetch relevant documents.
  - Enhances the generation of responses by conditioning on retrieved documents.
- **Usage**:
  - Facilitates complex queries on internal datasets.

LlamaIndex

LlamaIndex: Leveraging RAG for Internal Data

- **What it is**: A system that utilizes Retrieval-Augmented Generation (RAG) with LLMs for internal data querying and analysis.
- **How it works**:
  - Combines the power of LLMs with a retrieval system to fetch relevant documents.
  - Enhances the generation of responses by conditioning on retrieved documents.
- **Usage**:
  - Facilitates complex queries on internal datasets.
  - Provides contextually enriched answers by combining generative power with specific data retrieval.

LlamaIndex

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.
- Provides users with quick and reliable answers to fact-check claims.

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.
- Provides users with quick and reliable answers to fact-check claims.

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.
- Provides users with quick and reliable answers to fact-check claims.

Benefits:

- **Accessibility:** A simple and direct solution to access fact-checked information.

Vera: `askvera.org`

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.
- Provides users with quick and reliable answers to fact-check claims.

Benefits:

- **Accessibility:** A simple and direct solution to access fact-checked information.
- **Public Trust:** Promotes transparency and helps build trust in information sources.

Vera: askvera.org

Vera: Your Trusted Number for Fact-Checking

Functionality:

- Vera is a single, free-to-use app for verifying facts and combating misinformation.
- Provides users with quick and reliable answers to fact-check claims.

Benefits:

- **Accessibility:** A simple and direct solution to access fact-checked information.
- **Public Trust:** Promotes transparency and helps build trust in information sources.
- **Countering Misinformation:** Acts as a frontline tool in the fight against misinformation and polarization.

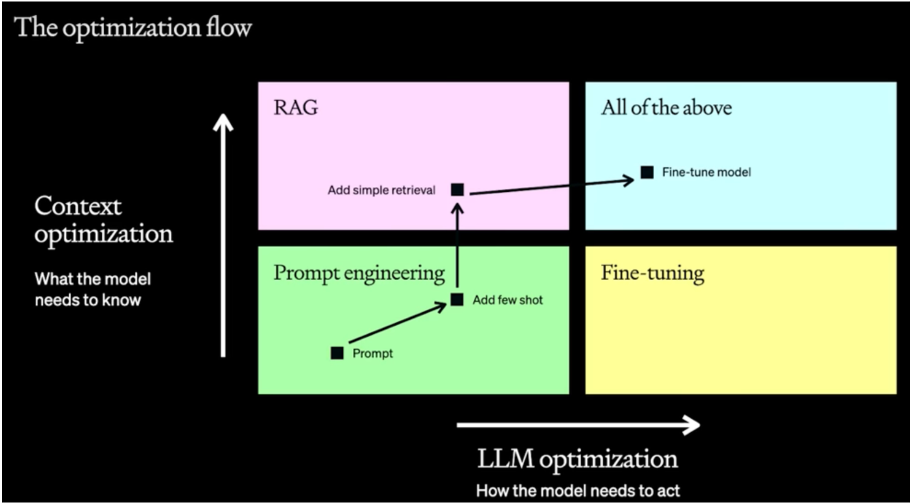
Vera: `askvera.org`

Maximizing LLM Performance

Techniques to Utilize LLMs

- **Prompt Engineering:** Crafting prompts that guide the LLM to generate the desired output. Asking the model to act in a specific way, leveraging pre-trained knowledge to fulfill complex tasks.
- **Retrieval-Augmented Generation (RAG):** Optimizing the context by providing the model with external knowledge to know before generating a response. This method enhances the model's ability to generate more contextually relevant answers.
- **Fine-Tuning:** Training the LLM on a specific dataset to optimize its performance for a particular task. It's about how the model needs to act, refining its responses based on additional training to align with the task's requirements.
- **Combining Techniques:** While each technique has its strengths, combining prompt engineering, RAG, and fine-tuning can offer a comprehensive approach to leveraging LLMs. The best direction depends on the specific use case.

Techniques to Utilize LLMs



From openAI demo day

Prompt Engineering with LLMs

Techniques for Enhanced Interaction:

- **LLM-Enhanced Prompts:** Utilizing LLMs to refine prompts for better accuracy and relevance. eg. Liu et al., 2023 - "Dynamic LLM-Agent Network"
- **Few-Shot Learning:** Incorporating examples within prompts to guide LLMs towards desired outputs. *Reference: GPT-3, OpenAI.*
- **Chain of Thoughts:** Encouraging LLMs to "think aloud," enhancing reasoning for complex queries. eg. Wei et al., 2022 - "Chain of Thought Prompting Elicits Reasoning in Large Language Models."
- **Schema-Constrained Output:** Structuring prompts to yield outputs in specific formats, like JSON for NER tasks. eg. Shin et al., 2021 - "Constrained Language Models Yield Few-Shot Semantic Parsers."

Using LLMs for Prompt Optimization

Python Example with OpenAI API:

Code Snippet

```
from openai import OpenAI
client = OpenAI(api_key='your-api-key-here')

prompt = """
I'd like to understand the two main themes of the movie description.
Please provide a list of two themes of it:

{{MOVIE}}
"""

messages = [{"role": "system",
             "content": "You're a prompt engineer that needs to
                         optimize a prompt. I'll give you a prompt
                         and you will and objective and
                         you'll improve the prompt."},
            {"role": "user", "content": prompt}]
```

Using LLMs for Prompt Optimization

Python Example with OpenAI API:

Code Snippet

```
response = client.chat.completion.create(
  engine="gpt-4",
  messages=messages,
  temperature=0.7,
  max_tokens=60,
  top_p=1.0,
  frequency_penalty=0.0,
  presence_penalty=0.0
)

print(response.choices[0].text.strip())
```

Understanding Temperature in LLMs

Temperature in Language Models:

- Controls the randomness in the prediction of the next word.
- A *lower temperature* (e.g., 0.1) results in more deterministic and confident outputs, often repeating the most likely words.
- A *higher temperature* (e.g., 1.0 or higher) increases diversity in generated text, producing more varied and sometimes more creative or unexpected results.

Understanding Temperature in LLMs

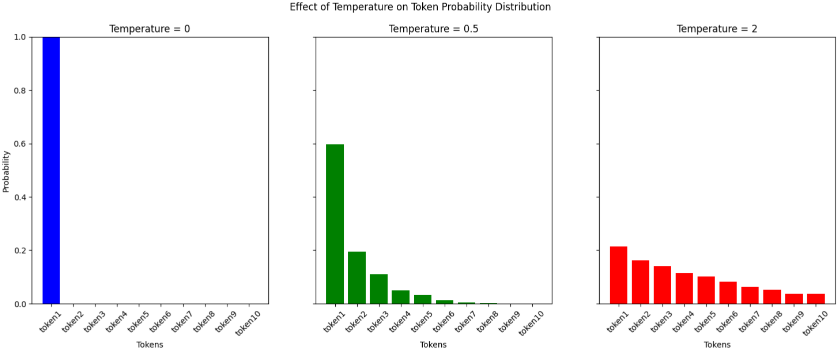
Temperature in Language Models:

- Controls the randomness in the prediction of the next word.
- A *lower temperature* (e.g., 0.1) results in more deterministic and confident outputs, often repeating the most likely words.
- A *higher temperature* (e.g., 1.0 or higher) increases diversity in generated text, producing more varied and sometimes more creative or unexpected results.

Temperature Effect: Prompt: "The sun sets over the"

- *Low Temperature (0.1)*: "The sun sets over the horizon."
- *Medium Temperature (0.7)*: "The sun sets over the distant mountains, casting a golden hue."
- *High Temperature (1.0)*: "The sun sets over the sea, weaving tales of ancient mariners and distant shores."

Understanding Temperature in LLMs



Higher temperature, smoother distribution

Understanding Top-p Sampling in LLMs

Top-p Sampling in Language Models:

- Selects the smallest set of words whose cumulative probability exceeds a threshold p.
- It dynamically adjusts the size of the considered vocabulary based on the p value, focusing on a more probable subset for each prediction.
- This approach helps balance between creativity and relevance by avoiding the less probable, and hence more random, words without being overly deterministic.

Understanding Top-p Sampling in LLMs

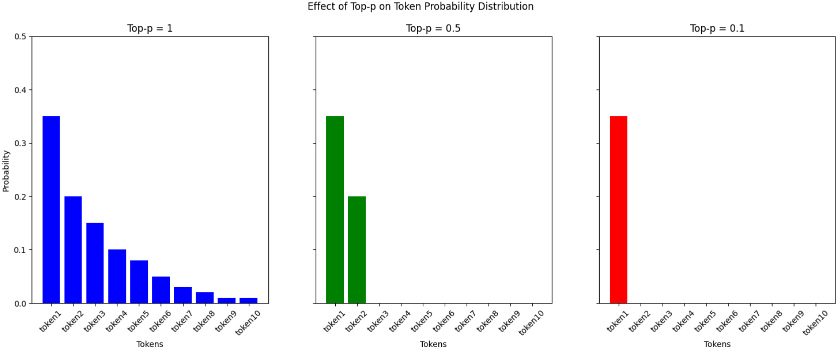
Top-p Sampling in Language Models:

- Selects the smallest set of words whose cumulative probability exceeds a threshold p.
- It dynamically adjusts the size of the considered vocabulary based on the p value, focusing on a more probable subset for each prediction.
- This approach helps balance between creativity and relevance by avoiding the less probable, and hence more random, words without being overly deterministic.

Top-p Sampling Effect: Prompt: "In the distant future, humanity"

- *Low p Value (0.2)*: "survives in a utopia."
- *Medium p Value (0.5)*: "explores new galaxies, seeking life."
- *High p Value (0.9)*: "faces challenges beyond imagination, like AI revolutions and interstellar wars."

Understanding Top-p Sampling in LLMs



Lower $\text{top}_p, smallertokenssettoconsider$

Few-Shot Learning Techniques for LLM Prompting

FSL techniques empower LLMs to adapt and generalize enhancing their ability to comprehend and respond to diverse prompts.

Main Advantages:

- **Flexibility**: LLMs can learn from a small number of examples, enabling adaptation to various tasks and contexts.
- **Efficiency**: Requires minimal labeled data, reducing the annotation burden and facilitating rapid model customization.
- **Generalization**: Promotes robustness and adaptability by extracting common patterns and concepts from limited examples.

Why Few-Shot Learning Works Better:

LLMs' extensive pre-training allows them to leverage prior knowledge and patterns from diverse domains, enabling effective transfer learning with few-shot examples.

Implementing Few-Shot Learning with OpenAI API

Python Example with OpenAI API:

Python Code Example

```
import openai
prompt = """
Translate the given text to French:
Hello world
---
Bonjour Monde !
"""

few_shot_examples = ["\My name is Harryn\n---\nje m’appelle Harry.",
                     "I love pizzas\n\n---\n j’adore les pizzas"]

messages = [{"role": "system", "content": "Your a English to French translator"},
            {"role": "user", "content": prompt},
            {"role": "user", "content": "Here are some examples:\n " +
                                        "\n\n".join(few_shot_examples)},
            {"role": "user", "content": "What is your favorite color?"}]
```

Implementing Few-Shot Learning with OpenAI API

Python Code Example

```
# Perform few-shot learning with OpenAI API
response = client.chat.completion.create(
    engine="gpt-4",
    messages=messages
    temperature=0.7,
    max_tokens=100
)

# Print the generated response
print(response.choices[0].text.strip())
```

Chain of Thoughts in LLMs

Chain of Thoughts is a technique used to guide the generation of responses in Large Language Models (LLMs) by breaking down the thought process into smaller, sequential steps.

- **Sequential Generation:** LLMs are prompted with a series of interconnected thoughts or questions, each building upon the previous one.
- **Structured Outputs:** By structuring the input as a chain of related thoughts, LLMs are encouraged to produce coherent and contextually relevant responses.
- **Enhanced Understanding:** This technique helps LLMs understand the context and intent better, leading to more accurate and meaningful outputs.
- **Improved Communication:** CoT facilitates more natural and engaging conversations, mimicking human thought processes.

Implementing Chain of Thoughts with OpenAI API

Python Example with OpenAI API:

Python Code Example

```
import openai

prompt = """
Q: What is the value of  5!?
A: 5!  = 1 x 2 x 3 x 4 x 5, so 5! = 6 x 20 = 120
A: 120

Q: What is the value of (3 x 100 ) + 5 - (43 / 7)?"

# Generate response using OpenAI API
response = client.chat.completion.create(
    engine="text-davinci-003",
    prompt=prompt,
    temperature=0.7,
    max_tokens=100)
```

Constraining LLM Outputs with JSON Schema

JSON schema provides a powerful way to define the structure of outputs from LLMs, ensuring that generated text adheres to specific formats or contains particular types of information.

- **Defining the Schema:** Specify the expected output format using JSON schema, including types of data, required fields, and descriptions.
- **Applying to LLMs:** Use the schema within the request to an LLM (such as OpenAI's GPT) to guide the generation process, ensuring outputs match the defined structure.
- **Example Use Case:** For tasks requiring structured data, like extracting specific information from text or generating content that fits a particular format.
- **Benefits:** Improves the utility of LLMs in applications where precise data structure or specific information extraction is crucial.

Implementing schema's output with Dolly

Python Code Example

```
from jsonformer import Jsonformer
from transformers import AutoModelForCausalLM, AutoTokenizer
model = AutoModelForCausalLM.from_pretrained("databricks/dolly-v2-12b")
tokenizer = AutoTokenizer.from_pretrained("databricks/dolly-v2-12b")

json_schema = {
    "type": "object",
    "properties": {
        "human": {
                  "type": "object",
                  "properties": {
                                "name": {"type": "string"},
                                "occupation": {"type": "string"},
                                "is_student": {"type": "boolean"},
                              }
                }
                }
            }
```

Implementing schema's output with Dolly

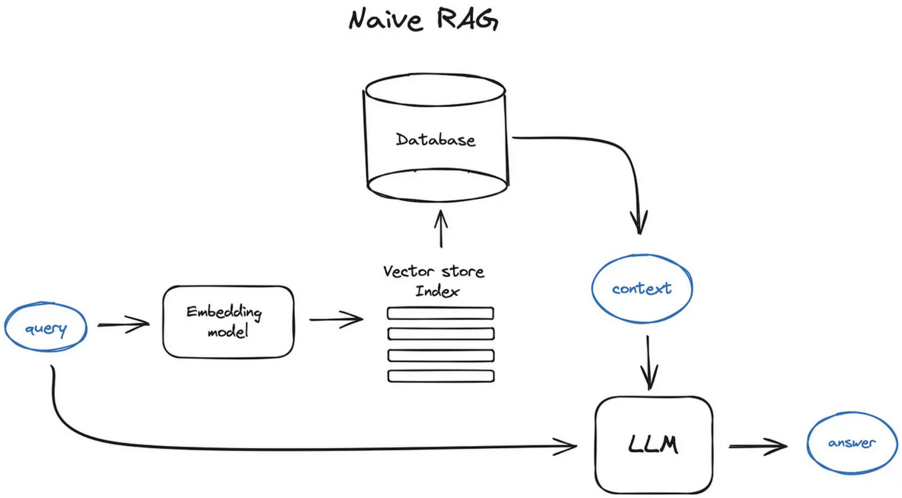
Python Code Example

```
prompt = """
Generate a person's information based on the following schema:

My name is Arnault and I work as lecturer at BSE in Barcelona.
"""
jsonformer = Jsonformer(model, tokenizer, json_schema, prompt)

{
"human": {
      "type": "object",
      "properties": {
        "name": "Arnault",
        "occupation": "lecturer",
        "is_student": False,
      }
    }
}
```

Retrieval-Augmented Generation Methods

1. **Naive RAG:**
   - Simple retrieval process to fetch relevant passages from a KB.
   - Retrieved information is concatenated with the query to augment the context before generation.
2. **Advanced RAG:**
   - Make it as complicated as you wish with hierarchical index retrieval, sentence window retrieval, similar as search engines.
3. **Hypothetical Document Embeddings:**
   - Instead of directly retrieving documents, this approach generates embeddings for hypothetical documents that could answer the query.
   - Embeddings are used to fetch the most relevant documents from the knowledge base, bridging the gap between query and available knowledge.

Naive RAG



Credit: Ivan Ilin

Naive Retrieval-Augmented Generation (RAG) Principle

How It Works:

- Uses a retriever to fetch relevant passages from a knowledge base (e.g., vector database, search engine).
- Retrieved passages are concatenated with the user's query.
- The combined context is fed into the language model.

Advantages:

- Simple and effective for queries well-covered in the knowledge base.
- Easy to implement with existing retrieval tools.

Limitations:

- Relies on exact or near-exact matches in retrieval, limiting relevance for complex queries.
- May struggle with under-specified or nuanced queries.

Hypothetical Document Embeddings:

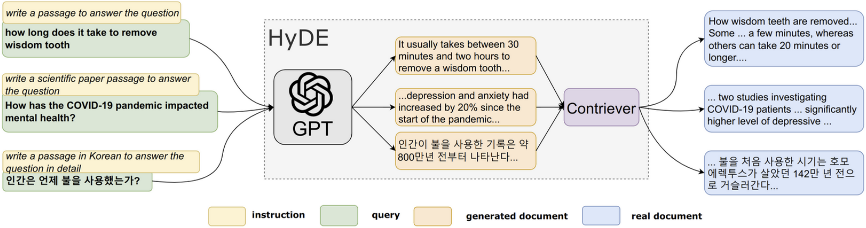


Figure 1: An illustration of the HyDE model. Documents snippets are shown. HyDE serves all types of queries without changing the underlying GPT-3 and Contriever/mContriever models.

Credit: Gao et al. (2022)

Precise Zero-Shot Dense Retrieval without Relevance Labels

Why HyDE Improves Over Naive RAG

Limitations of Naive RAG:

- Mismatch between query and retrieved texts.
- Difficulty in addressing under-specified or abstract queries.

Hypothetical Document Embeddings (HyDE) Approach:

- Generates a *hypothetical answer* to the user's query using the LLM.
- Creates an embedding of this hypothetical answer to retrieve more semantically relevant passages from the knowledge base.
- The retrieved context is closer in meaning to the user's intent.

Benefits of HyDE:

- Bridges the gap between abstract queries and available knowledge.
- Enhances the model's ability to provide nuanced, relevant answers even for vague or novel queries.

Supervised Fine-Tuning (SFT)

Overview of Supervised Fine-Tuning: leverages labeled data to enhance models' understanding & response quality in targeted domains.

Examples of Open Source Datasets:

- **GPT-4all Dataset:** A diverse QA and creative questions dataset with 400k entries, combining subsets of OIG, P3, and Stackoverflow.
- **RedPajama-Data-1T:** A massive 1.2T tokens pretraining dataset, designed following LLaMA's methodology for open pretraining.
- **OASST1:** The OpenAssistant dataset with 66,497 multilingual conversation trees, focused on enhancing LLM dialog capabilities through human-written and annotated conversations.

Significance: Supervised FT enables LLMs like ChatGPT to perform better in specialized tasks or languages, making them more versatile and effective in real-world applications.

Fine Tuning dataset example

prompt (string)	response (string)
"<p>Good morning</p> <p>I have a Wpf datagrid that is displaying an observable collection of a custom…	"One possible solution is to use a fixed width for the GroupItem header and align the header and the…
"<h2>Hi, How can I generate a pdf with the screen visual data, or generate a pdf of the data being…	"To generate a PDF with the screen visual data, you can use a library such as pdf. Here's an example:…
"<pre><code>package com.kovair.omnibus.adapter.platform; import…	"The issue might be related to class loading and garbage collection. When a class loader loads a…
"<p>I'm trying to get it so that all of the items in ListView.builder can be displayed on the screen…	"To make the whole page scrollable, remove the `SingleChildScrollView` and wrap the entire…
"<p>I have used a <code>ListView</code> and the parent in the <code>xml</code> is…	"The issue seems to be with the layout parameters being set in the `getView()` method. The code is…
"<p>I am calling a stored proc [MS SQL] using EF5 from a .net application</p> <p>The call from EF</p…	"<p>This is likely due to the fact that CHAR columns are fixed-length and padded with spaces to…
"<p>This code is about viewing a published consultation schedule. Unfortunately I'm stuck wit…	"The issue with the if statement is that it is inside the for loop, but it should be outside of…

GPT-4all Dataset

Implementing Supervised Fine-Tuning with SFTTrainer

Using Hugging Face's ‘SFTTrainer‘:

The ‘SFTTrainer‘ is a tool in Hugging Face's Transformers library designed to streamline the fine-tuning of large language models on specific datasets.

Python Code Example

```
from transformers import AutoTokenizer, AutoModelForCausalLM
from transformers SFTTrainer, TrainingArguments

tokenizer = AutoTokenizer.from_pretrained("databricks/dolly-v2-12b")
model = AutoModelForCausalLM.from_pretrained("databricks/dolly-v2-12b")

train_dataset = load_dataset("path/to/dataset")
```

Implementing Supervised Fine-Tuning with SFTTrainer

Python Code Example

```
# Define training arguments
training_args = TrainingArguments(
    output_dir="./fine_tuned_model",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    logging_dir="./logs",
)

# Initialize SFTTrainer
trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    tokenizer=tokenizer,
)

trainer.train()
```

Limitations of Supervised Fine-Tuning without RHLF

Challenges: While SFT can significantly enhance LLMs' performance on specific tasks, doing so without access to RLHF introduces several limitations:

- **Bias Amplification:** Fine-tuning on biased datasets without RHLF can lead to the amplification of existing biases, affecting the fairness and neutrality of the model's outputs.
- **Overfitting:** The lack of diverse human feedback during fine-tuning may result in models that overfit to the training data, hindering their generalization to unseen contexts.
- **Missed Learning Opportunities:** RLHF can provide unique insights and corrections that are crucial for improving models. Without it, models miss out on learning from complex human interactions and corrections.

Conclusion: Incorporating RLHF is crucial for developing more adaptable, unbiased, and generalizable LLMs. Overcoming these limitations requires innovative approaches to integrate human feedback effectively.

Training Complexity of a 7B Parameter Model

Resource Requirements:

- **Model size:** $\approx$ 28 GB in 32-bit (fp32) precision.
- **Training memory:** Gradient storage + optimizer states increase the memory footprint $\approx$ 3-4x.
- **Total memory footprint:** $\approx$ 140 GB.

GPU Requirements:

- Requires multiple high-end GPUs (A100 80GB, H100 80GB, etc.).
- **Example setup:** 2$\times$ A100 80 GB or 4$\times$ A100 40 GB GPUs for data/gradient parallelism.

Time Complexity:

- Days to weeks of training depending on dataset size and compute budget.
- Infeasible on a single GPU.

LoRA: Train Large Models on a Single GPU

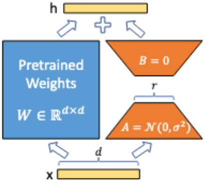
- LoRA stands for **Low-Rank Adaptation**.
- It lets you fine-tune LLM using small, trainable adapters instead of updating the whole model.

How does it work?

- Original model weights are **frozen**.
- Add two small matrices, A and B, to capture the task-specific adaptation.
- Only these small matrices are trained.

Why is it great?

- Needs much less memory and compute.
- You can fine-tune a huge model (e.g., 7B LLaMA) on a single 24GB GPU.

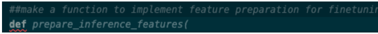


LoRA matrices.

Credit: Hu et al. (2021)

Integrating Human Feedback with UX in LLMs: Advanced UX mechanisms, like those in GitHub Copilot, demonstrate the power of integrating human feedback into LLM fine-tuning:

- **Effortless Feedback Integration:**
  - Direct workflow integrations for accepting suggestions (e.g., using TAB).
  - Navigation shortcuts for efficient suggestion review.

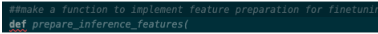


GitHub Copilot UX

RLHF through UX: Copilot Completion - Part 2

User-Friendly Interaction Optimized Feedback:

- **User-Friendly Interaction:**
  - Suggestions are passive, maintaining user workflow integrity.
  - High latency sensitivity for timely, relevant suggestions.
- **Optimized Feedback for Fine-Tuning:**
  - Capturing implicit signals from user interactions for model improvement.
  - Encourages user engagement through low requirements and high incentives.



GitHub Copilot UX: Feedback

Impact: UX designs that facilitate easy human feedback collection empower continuous LLM learning and user satisfaction.

QA and Takeaways

QA

Open Discussion

- Feel free to ask questions or share your thoughts about today's topics.
- Any insights, experiences, or perspectives you'd like to discuss are welcome.

Summary of Key Takeaways

- **Advancements in LLMs:** Explored the evolution of Large Language Models since 2022, highlighting the transition from text-only to multimodal models and the emergence of platforms like ChatGPT.
- **Applications Unveiled:** Delved into various use cases of LLMs, from GitHub Copilot to internal data querying with LlamaIndex, showcasing their wide-ranging impact across sectors.
- **Mastering Prompt Engineering:** Discussed techniques for effective prompt engineering employing few-shot examples, and applying chain of thoughts and structured outputs for enhanced interactions.
- **Innovation with RAG:** Introduced Retrieval-Augmented Generation (RAG) methods, accentuating their role in optimizing context and LLM performance for intricate querying tasks.
- **Fine-Tuning LLMs:** Covered fine-tuning methods for personalizing LLMs to specific tasks, emphasizing the importance of UX in providing human feedback for continuous model improvement.