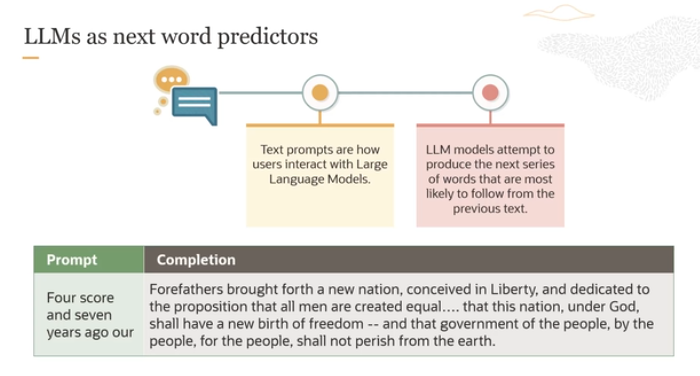


Prompt is the input or initial text provided to the large language model. Prompt engineering is the process of iteratively refining a prompt for the purpose of  eliciting/extract a particular style of response.



**Key Points:**

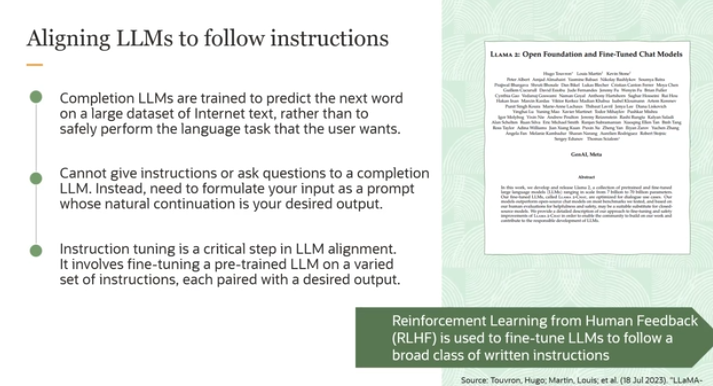
1. **LLMs as Next Word Predictors**: The diagram labels LLMs as tools that predict the next word or sequence of words based on the input they receive.
2. **User Interaction**: Users interact with these models by providing text prompts.
3. **Model's Task**: The model attempts to predict and produce a continuation of the text that logically follows from the given prompt.

**Explanation of the Process:**

* **Text Prompts**: Users start by typing a prompt, which is a piece of text. In your example, the prompt is the beginning of Abraham Lincoln's Gettysburg Address: "Four score and seven years ago our".
* **Model's Prediction**: Based on this input, the model predicts the next words. It does this by analyzing the input text and using its training on vast amounts of data to generate the most probable subsequent words.
* **Text Completion**: The model continues the text in a way that aligns with how the text usually proceeds based on its training. For the given example, since the model has likely been trained on historical speeches including the Gettysburg Address, it can accurately complete the excerpt from Lincoln's speech.

**How it Works:**

The LLM has been trained on a large dataset that includes books, articles, websites, and other text sources. This training involves statistically analyzing sequences of words and learning patterns of language usage. When given a prompt, the LLM uses this learned information to predict and generate text that follows naturally from the prompt. The model uses algorithms to calculate the probability of each possible next word and selects the most likely ones to form coherent and contextually appropriate text.



The image and the detailed explanation you provided discuss the concept of aligning Large Language Models (LLMs) to better follow user instructions, a process known as instruction tuning, and the use of Reinforcement Learning from Human Feedback (RLHF).

**Overview of LLMs:**

* **Completion LLMs**: Traditionally, LLMs are trained to predict the next word based on vast amounts of Internet text. Their primary function is to complete the text rather than perform specific tasks safely or as intended by the user.

**Challenges:**

* **Instruction Limitation**: Due to their completion-oriented training, these models can't inherently understand and execute user instructions directly. Users must craft prompts that naturally lead the model to generate the desired output, which isn't always intuitive or effective.

**Solution: Instruction Tuning**

* **Purpose**: Instruction tuning is designed to realign LLMs from merely completing text to performing tasks as specified by user instructions. This is done by training the model on a varied set of instructions, each paired with the desired output.
* **Process**: This involves fine-tuning a pre-trained LLM on specific tasks that reflect real-world use cases, making the model more responsive and accurate in following instructions.

**Example - Llama 2 by Meta:**

* **Foundation and Fine-Tuning**: The Llama 2 model was initially trained on a dataset comprising approximately two trillion tokens. For tasks specific to conversations (Llama 2 chat), it underwent additional fine-tuning on 28,000 prompt-response pairs tailored to improve interaction quality.
* **RLHF**: Reinforcement Learning from Human Feedback is a technique used in further fine-tuning stages where human annotators evaluate the model's responses to prompts. Feedback from these evaluations helps train a reward model, which then guides the LLM to better align with human preferences and instructions.

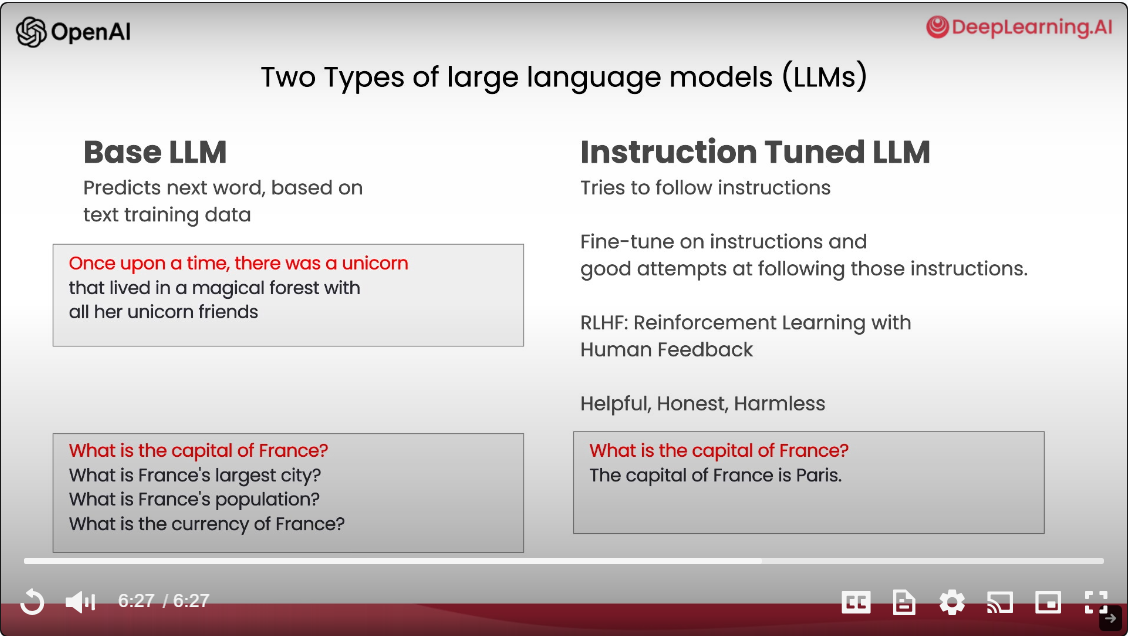
**How RLHF Works:**

1. **Human Annotators**: They interact with the model by providing prompts and reviewing the outputs.
2. **Feedback Utilization**: Their feedback is used to develop a reward model that understands and replicates the preferences shown by humans.
3. **Model Alignment**: This reward model is then used to adjust the LLM’s responses, ensuring that they meet the quality and relevance standards as perceived by human evaluators.

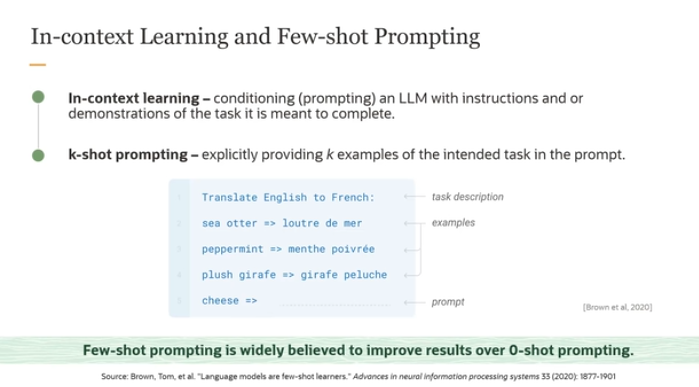
**Conclusion:**

Instruction tuning, particularly through methods like RLHF, is crucial for adapting completion-based LLMs to serve practical purposes effectively. It helps transform these models from simple text generators into versatile tools capable of executing complex, instruction-based tasks as demanded by users in real-world applications.

Now basically humaray pass jo LLM hotay thy that are called **Base LLM** jokay mainly text completion ka liya create kiya gye thy and we cannot directly pass any instruction or ask question from them. But agr hum phr be desired output extract krna hai toh hum instruction ko asa write krsktay hain kay uski jo continuation ho that will be our desired output. Like: once upon a time (written by you) there was a unicorn (predict by LLM). But still this way is complex and not much efficient . So then phr humaray pass ek new version of LLM ata hai that is called **Instruction-based LLM.** Iss LLM may hum prompt ka through instruction provide krsktay hain or ek response generate krsktay hain. But your instruction should be specific for generating good response.



For more info: [https://www.ibm.com/topics/prompt-engineering](https://www.ibm.com/topics/prompt-engineering#:~:text=IBM-,What%20is%20prompt%20engineering%3F,simple%20to%20the%20highly%20technical.)



Large Language Models (LLMs): **in-context learning** and **few-shot prompting**. Here’s a breakdown of each concept:

**In-context Learning**

* **Definition**: In-context learning refers to providing an LLM with specific examples or instructions directly within the prompt to guide its response. It does not involve changing the underlying model itself (i.e., the model's parameters remain unchanged). Instead, the model uses the provided context to generate appropriate responses.
* **Mechanism**: By presenting the model with examples of the task at hand, you're effectively "conditioning" it to understand and perform the task as demonstrated in the examples. This is akin to giving the model a "mini tutorial" on what is expected.

**K-shot Prompting**

* **Definition**: K-shot prompting is a type of in-context learning where 'k' specific examples of a task are given within the prompt. These examples serve to demonstrate the task to the model. The 'k' can vary, leading to different types of prompting:
  + **0-shot prompting**: No examples are provided.
  + **1-shot prompting**: One example is provided.
  + **2-shot prompting**: Two examples are provided.
  + **K-shot prompting**: K examples are provided, where K can be any number.
  + **Few-shot prompting:** few examples such as 2,3 .

**Example from the Image**

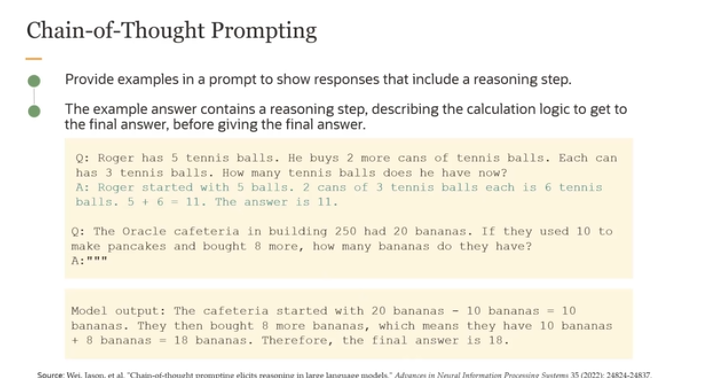
* **Task**: The task is to translate text from English to French.
* **Prompting Type**: This is a three-shot example, where three translations are provided:
  1. Sea otter => loutre de mer
  2. Peppermint => menthe poivrée
  3. Plush giraffe => girafe peluche
* **Prompt**: After these examples, a new item "cheese" is given, and the model is expected to translate it into French based on the pattern it has observed in the examples.

**Efficacy of Few-shot Prompting**

* **Advantage**: Research suggests that few-shot prompting significantly improves the model's performance compared to zero-shot prompting. By seeing a few examples, the model better understands the pattern or the task, leading to more accurate and contextually appropriate responses.

**Overall Concept**

The setup provided in the image and description showcases how giving specific examples within the prompt can effectively "teach" the model to perform tasks that require understanding of context and specifics, such as language translation.



The passage you've provided discusses **chain-of-thought prompting**, a technique used in prompt engineering for effectively interacting with large language models (LLMs) to solve complex tasks. Here’s a detailed explanation of this technique:

**Chain-of-Thought Prompting**

* **Definition**: Chain-of-thought prompting guides an LLM to break down complex problems into smaller, manageable parts, mimicking the way a human might logically think through a problem.
* **Purpose**: This technique helps the model articulate a step-by-step reasoning process before arriving at the final answer. It's particularly useful for tasks that involve calculations, logical reasoning, or any scenario where intermediate steps are beneficial to understand or verify the outcome.

**How It Works**

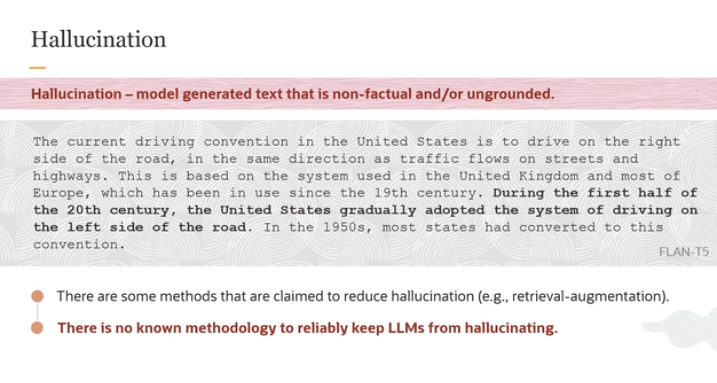
1. **Break Down the Problem**: The user structures the prompt to lead the model into dividing the problem into smaller segments or steps.
2. **Detailed Reasoning**: Each segment of the problem is addressed individually, with the model providing explicit reasoning or calculations at each step.
3. **Final Solution**: After processing through the steps, the model synthesizes the information to arrive at a final answer.

**Examples Provided**

1. **Tennis Balls Example**:
   * **Problem Statement**: Roger starts with 5 tennis balls and buys 2 more cans, each containing 3 tennis balls.
   * **Model's Reasoning**: The model calculates the total by adding the balls from the cans to the initial count:
     + Starts with 5.
     + 2 cans × 3 balls each = 6.
     + 5 + 6 = 11.
   * **Outcome**: The model outputs that Roger has a total of 11 tennis balls.
2. **Bananas Example**:
   * **Problem Statement**: The Oracle cafeteria starts with 20 bananas, uses 10 for pancakes, and buys 8 more.
   * **Model's Reasoning**: It calculates the remaining bananas after some are used and additional ones are purchased:
     + Starts with 20.
     + Uses 10, so 20 - 10 = 10 remaining.
     + Buys 8 more, so 10 + 8 = 18.
   * **Outcome**: The model concludes there are 18 bananas left.

**Importance of Chain-of-Thought Prompting**

* **Accuracy**: This prompting style helps ensure that the model's responses are not only correct but also logically sound and transparent in their derivation.
* **Debugging and Trust**: By showing intermediate steps, it becomes easier to verify or debug the model’s reasoning, which builds user trust in the model’s capabilities.



**What is Hallucination?**

* **Definition**: In the context of LLMs, hallucination refers to the generation of text that is factually incorrect, nonsensical, or not based on any data the model has been trained on.
* **Characteristics**: Hallucinated text can be:
  + **Non-factual**: Contains information that is factually wrong.
  + **Ungrounded**: Not based on any factual or historical data.
  + **Fluent**: Despite being incorrect, the text can be coherent and fluent, making it seem believable.

**Examples and Challenges**

* **Example Provided**: The example about driving on the left side of the road in the United States is factually incorrect as the U.S. follows right-hand driving. This misinformation would be considered a hallucination.
* **Detection Challenges**: While some hallucinations may be obvious and easy to spot, many can be subtle and difficult to detect, especially when the text is fluent and begins correctly.

**Implications of Hallucination**

* **Credibility and Trust**: Frequent hallucinations can undermine the credibility of the model and erode user trust.
* **Utility**: Inaccurate outputs can limit the practical uses of LLMs, particularly in applications requiring high factual accuracy, such as educational tools, content creation, and decision support systems.

**Mitigating Hallucination**

* **Retrieval-Augmented Systems**: Incorporating retrieval mechanisms, where the model accesses external databases or documents to verify information before generating text, has been shown to reduce hallucinations.
* **Research Efforts**: There is ongoing research into developing methods that can measure and ensure the groundedness of LLM outputs. This research aims to find reliable ways to prevent hallucinations and enhance the model's adherence to factual accuracy.

**Conclusion**

Understanding and addressing hallucination in LLMs is crucial for improving their reliability and expanding their applicability. While current strategies like retrieval-augmentation show promise, the complete elimination of hallucination remains a challenge, spurring active research to develop more effective solutions. This ongoing work is vital for advancing the field and maximizing the potential of language models in diverse applications.