Lab 2 - Prompt Engineering with LLMs

Important
Run with **JupyterLab**: We're going to use `lab-02-prompt-engineering/Lab_2_Prompt_Engineering_with_Sagemaker_Studio.ipynb` notebook for this section

Contents

Introduction to Prompt Engineering

Prompt engineering has emerged as a groundbreaking technique for optimizing the performance of large language models. This introductory lab will provide hands-on practice with key prompt engineering skills using SageMaker to interact with the Llama 2 language model.

Prompt engineering involves carefully crafting the prompts fed into language models to make them more capable, safe, and aligned with human values. As massive neural network models trained on internet data, language models excel in some areas but struggle with others. Prompt engineering allows us to unlock their full potential.

In this lab, you will learn prompt engineering fundamentals like basic prompting, few-shot learning, and chain-of-thought prompting. Through practical examples, you will gain first-hand experience using these techniques to enhance language model capabilities in areas like summarization, translation, reasoning, and classification.

For example, prompt engineering could enable a language model to accurately summarize clinical trial reports into layman's terms, which would greatly benefit medical professionals and patients.

The hands-on nature of this lab will equip you with applied skills to leverage prompt engineering in real-world contexts. You will come away with knowledge of how strategic prompts can make language models more useful, ethical, and beneficial.

As language models grow more powerful, prompt engineering is becoming an essential skill for AI practitioners, researchers, and technologists. This lab aims to provide the core conceptual and practical foundations you need to safely interact with and optimize large language models through targeted prompting.

After completing this lab, you will be ready to start leveraging prompt engineering to make language models work for you - unlocking their potential while optimizing for safety, ethics, and alignment with human values.

The techniques covered represent just the tip of the iceberg for prompt engineering's possibilities. It is an active research area with new methods rapidly emerging. You will gain invaluable introductory skills and be well-positioned to further explore advanced prompt engineering as it continues evolving alongside AI. The future will rely on prompt engineering expertise to ethically guide the development and application of language models.

We are going to continue reusing our preprovisioned Llama 2 model Endpoint, Refer to Lab 1 for instructions on how to retrieve your Endpoint Name

Prompt Engineering Basics

Basic Prompts

Lets start with some examples to get a better understanding of basic prompts. This section will introduce you to the process of giving instructions to the model for specific tasks.

We'll begin with prompts that clearly instruct the model on what text to generate or translate. These examples will show how with minimal guidance large language models can produce impressive results.

```
params = set_llama2_params(temperature=0.5)
payload, response = send_prompt(params, prompt="The sky is")
```

Although these prompts may seem basic they provide a foundation and serve as a reference point as we gradually explore more advanced techniques. Through hands on exercises we'll cover prompting fundamentals before delving into specialized methods in later sections.

By grasping the core concepts of prompting at this stage we'll be well prepared to understand and utilize advanced engineering strategies. This foundational section will lay the groundwork, for becoming proficient in prompt based interactions.

Text Summarization

In this section, we will explore using tailored prompts to distill lengthy text into concise summaries. The ability to condense complex ideas into succinct descriptions showcases the formidable text summarization capabilities of large language models. We will apply this skill through customized prompts on two examples.

First, we will provide an excerpt from the classic fable of "The Tortoise and the Hare" and craft a prompt directing the model to summarize it in one sentence. This longer narrative presents an opportunity to test summarization of intricate story details.

```
params = set_llama2_params(temperature=0.7)
prompt = """The hare was once boasting of his speed before the other animals.
"I have never yet been beaten," said he, "when I put forth my full speed. I challenge any one here to race with me."
The tortoise said quietly, "I accept your challenge." "That is a good joke," said the hare;
"I could dance round you all the way." "Keep your boasting till you've beaten me," answered the tortoise.
"Shall we race?" So a course was fixed and a start was made.
The hare darted almost out of sight at once, but soon stopped and, to show his contempt for the tortoise, lay down to have a nap.
The tortoise plodded on and plodded on, and when the hare awoke from his nap,
he saw the tortoise just near the winning-post and could not run up in time to save the race.

Explain the above in one sentence:"""
payload, response = send_prompt(params, prompt)
```

Next, we will work with a shorter sample article and prompt the model to summarize it in three bullet points. Distilling factual information into key points requires precise summarization adeptness. By testing summarization methods on varied text lengths and types, we can better discern the nuances of this AI skill.

Overall, this section aims to highlight text summarization as a prime demonstration of how strategic prompts allow us to tap into profound abilities innate within language models. The hands-on examples will showcase these capabilities while underscoring the critical role of prompt engineering in eliciting top performance.

Question Answering

In this section, we will focus on crafting prompts to yield highly accurate question answering from the model. Eliciting precise responses often requires exceeding basic instructions and incorporating crucial context.

We will begin with a multi-sentence context and then supply a reasoning-based question related to the information provided. This allows us to test the model's competency in logical deductions from contextual clues. Additionally, we will demonstrate how small prompt adjustments can greatly impact the quality of answers produced.

```
params = set_llama2_params(temperature=0.7)

prompt = """Answer the following question based on the context below. Keep the answer short. Respond "Unsure about answer" if not sure about the answer.

Context: In 1849, thousands of people rushed to California in search of gold and riches.
This was known as the California Gold Rush. Prospectors came from all over the world during this time period.

Question: What year did the events take place?"""

payload, response = send_prompt(params, prompt)
```

The ability to answer questions correctly relies heavily on the information presented to the model. We will experiment with prompt structure, from basic to optimized, to showcase techniques for maximizing question-answering precision.

This section aims to highlight the possibilities enabled through deliberate prompt engineering. The hands-on examples will demonstrate how attention to detail in prompt crafting can unlock impressive reasoning abilities innate within language models.

Text Classification

In this section, we will explore using detailed prompts to enhance performance on text classification tasks. We will demonstrate the impact of providing examples and specifying potential categories within the prompt.

We will start with a basic text classification prompt, directing the model to categorize a sample text as positive or negative. Next, we will build on this prompt by clearly delineating the possible categories and supplying an example for each.

```
params = set_llama2_params(temperature=0.7)

prompt = """Classify the text into negative or positive.

Text: Apple stock is currently trading at 150 dollars per share. Given Apple's strong financial performance
lately with increased iPhone sales and new product launches planned, I predict the stock price will increase to around 160 dollars per share over the next month.

Sentiment:"""

payload, response = send_prompt(params, prompt)
```

Text classification requires accurately mapping input texts to predefined categories. Performance depends heavily on how much guidance the prompt provides through illustrations and explicit instructions.

By incrementally improving our prompts, we will showcase techniques for achieving greater precision in text classification tasks. The hands-on examples aim to highlight the critical role of prompt engineering in tapping into the categorization capabilities inherent within language models.

Role Playing

This section will illustrate role playing with an AI assistant by providing explicit personality instructions. We will craft a prompt that includes directions about the assistant's tone and role. This will shape the response when interacting with a simulated user.

```
params = set_llama2_params(temperature=0.5)

instruction = """You are an AI research assistant. Your tone is technical and scientific."""

prompt = """Human: Hello, who are you?
AI: Greeting! I am an AI research assistant. How can I help you today?
Human: Can you tell me about the creation of volcanic mountains?
AI:"""
```

```
payload, response = send_prompt(params, prompt, instruction)
```

Code Generation

This section will focus on assessing unaided code generation skills in large language models through zero-shot prompting. We will evaluate abilities in Python, JavaScript and SQL by providing high-level descriptions of functions and queries to generate code for.

The goal is to discern how well these models can produce valid code based solely on natural language instructions, without any examples. This offers insight into current zero-shot coding capabilities and opportunities to advance general-purpose code generation.

We will craft descriptions of coding problems spanning different languages and complexities. Examining the code produced in response to these zero-shot prompts will reveal strengths, limitations, and future potential.

```
params = set_llama2_params(temperature=0.5)

prompt = """Generate a Python program that prints the numbers from 1 to 10"""

payload, response = send_prompt(params, prompt)
```

Overall, this section aims to stretch the boundaries of unaided code generation as a means of better understanding innate coding abilities within large language models. The hands-on prompting experiments will uncover current skills and avenues for improvement.

Reasoning

In this section, we will explore basic reasoning abilities in large language models through introductory deductive reasoning and inference prompting examples. This provides a foundation before delving into more advanced reasoning techniques in later sections.

We will start with a simple syllogism based on the classic "Socrates is a man" example. By providing the initial premises, we can evaluate the model's competency in deducing new information through logical reasoning.

```
params = set_llama2_params(temperature=0.5)

prompt = """
Given the facts: All men are mortal. Socrates is a man.
Use logical reasoning to conclude: Is Socrates mortal? Explain your reasoning.
"""

payload, response = send_prompt(params, prompt)
```

Next, we will present a multi-step ordering problem with contextual clues. This enables testing capacities for logical inferences to determine the solution.

Though reasoning remains an active challenge area for language models, these hands-on prompts offer a glimpse into fledgling reasoning skills. While there are clear limitations, targeted prompting can reveal and strengthen pockets of reasoning capacity.

This section aims to highlight the possibilities of prompting as a mechanism for developing elemental reasoning in large language models. The examples will demonstrate basic deductive and inferential reasoning abilities.

Advanced Prompting Techniques

In this section, we explore more advanced prompting techniques involving few-shot learning and chain of thought prompting. These methods can enhance model performance on complex tasks requiring reasoning across multiple steps.

We will experiment with few-shot prompting, which uses just a handful of examples within the prompt to provide the model with the context needed to make correct inferences. This leverages few-shot learning capabilities inherent in large models.

Additionally, we will demonstrate chain of thought prompting, which explicitly cues the model to showcase its logical reasoning step-by-step. This aims to strengthen rational thinking abilities.

Combining these approaches can further amplify results, as we will showcase through an illustration of few-shot chain of thought prompting.

Through hands-on examples, this section highlights advanced prompting techniques that aim to unlock greater reasoning, generalization and problem-solving skills within language models. We will explore current capabilities and future opportunities.

Zero-shot

In this section, we explore the capabilities of large language models to follow instructions without any training, known as zero-shot learning. Models like LLaMA have been optimized to excel in zero-shot scenarios by leveraging their vast datasets and parameters.

We will test zero-shot abilities by providing straightforward directives for translation and text generation without any examples. This will demonstrate the models' capacity to produce outputs based solely on prompt instructions.

```
params = set_llama2_params(temperature=0.9)
payload, response = send_prompt(params, prompt="Translate this sentence to French: I am learning to speak French.")
```

While impressive, zero-shot skills have limitations on more complex tasks. However, zero-shot prompting establishes a baseline to then showcase techniques that enhance performance, which we will cover in subsequent sections.

The hands-on examples in this section aim to highlight the innate zero-shot capabilities of large models when given clear prompting. We will also gain perspective into opportunities for improvement using more advanced methods.

Few-shot prompts

In this section, we build on zero-shot prompting by incorporating few-shot learning techniques that supply examples within the prompt. While large models can perform unaided, providing a small number of demonstrations further enhances the quality and accuracy of outputs.

We will experiment with few-shot prompting through a hands-on example focused on properly using novel words in sentences. By showing

We will experiment with few-shot prompting through a hands-on example focused on properly using novel words in sentences. By showing just a single usage example, we can evaluate the model's ability to acquire new concepts.

```
params = set_llama2_params(temperature=0.9)

prompt = """A "blicket" is a tool used for farming. An example of a sentence that uses the word blicket is:
The farmer used a blicket to dig holes and plant seeds.

"Flooping" refers to a dance move where you spin around. An example of a sentence that uses the word Flooping is:"""

payload, response = send_prompt(params, prompt)
```

Additionally, we will demonstrate scaling shots in the prompt to continue improving results. The examples provide positive conditioning that guides the model's behavior.

Overall, this section highlights the strengths of large language models in few-shot learning. Hands-on prompts showcase how combining zero-shot skills with targeted examples unlocks greater performance through in-context learning.

Chain-of-Thought (CoT) Prompting

In this section, we explore chain-of-thought (CoT) prompting, which elicits step-by-step reasoning from models to improve performance on complex, multi-step problems.

We will demonstrate zero-shot CoT prompting by adding a simple cue to think step-by-step. This encourages models to show their work without any reasoning examples.

Additionally, we will illustrate few-shot CoT prompting which combines reasoning chain demonstrations with few-shot learning. Just a couple examples provide the context needed to apply similar logic.

```
params = set_llama2_params(temperature=0.5)

prompt = """Let's think through this:
If there were 6 oranges originally and 4 were peeled, how many unpeeled oranges are left?
Step 1) Originally there were 6 oranges
Step 2) 4 oranges were peeled
Step 3) So there must be 6 - 4 = 2 unpeeled oranges left

Let's think step-by-step:

If David had 9 cakes and ate 4 of them, how many are left?"""

payload, response = send_prompt(params, prompt)
```

CoT prompting aims to strengthen rational thinking and transparency by having models explicitly detail their reasoning process. The hands-on examples will highlight techniques for unlocking greater logical capabilities.

Lab 2 Complete!
Congratulations, You've successfully completed Lab 2!

Previous Next