Table Of Contents

How-to guides

Prompting

Link to Notebook showing examples of the techniques discussed in this section.

Prompt engineering is a technique used in natural language processing (NLP) to improve the performance of the language model by providing them with more context and information about the task in hand. It involves creating prompts, which are short pieces of text that provide additional information or guidance to the model, such as the topic or genre of the text it will generate. By using prompts, the model can better understand what kind of output is expected and produce more accurate and relevant results. In Llama 2 the size of the context, in terms of number of tokens, has doubled from 2048 to 4096.

Crafting Effective Prompts

Crafting effective prompts is an important part of prompt engineering. Here are some tips for creating prompts that will help improve the performance of your language model:

1. Be clear and concise: Your prompt should be easy to understand and provide enough information for the model to generate relevant output. Avoid using jargon or technical terms that may confuse the model.
2. Use specific examples: Providing specific examples in your prompt can help the model better understand what kind of output is expected. For example, if you want the model to generate a story about a particular topic, include a few sentences about the setting, characters, and plot.
3. Vary the prompts: Using different prompts can help the model learn more about the task at hand and produce more diverse and creative output. Try using different styles, tones, and formats to see how the model responds.

4. Test and refine: Once you have created a set of prompts, test them out on the model to see how it performs. If the results are not as expected, try refining the prompts by adding more detail or adjusting the tone and style.
5. Use feedback: Finally, use feedback from users or other sources to continually improve your prompts. This can help you identify areas where the model needs more guidance and make adjustments accordingly.

Explicit Instructions

Detailed, explicit instructions produce better results than open-ended prompts: You can think about giving explicit instructions as using rules and restrictions to how Llama 2 responds to your prompt.

Stylization

```
Explain this to me like a topic on a children's educational network show
teaching elementary students.

I'm a software engineer using large language models for summarization.
Summarize the following text in under 250 words:

Give your answer like an old timey private investigator hunting down a
case step by step.
```

Formatting

```
Use bullet points.

Return as a JSON object.

Use less technical terms and help me apply it in my work in
communications.
```

Restrictions

```
Only use academic papers.

Never give sources older than 2020.

If you don't know the answer, say that you don't know.
```

Here's an example of giving explicit instructions to give more specific results by limiting the responses to recently created sources:

```
Explain the latest advances in large language models to me.
#  More likely to cite sources from 2017

Explain the latest advances in large language models to me. Always cite
your sources.
Never cite sources older than 2020.
# Gives more specific advances and only cites sources from 2020
```

Prompting using Zero- and Few-Shot Learning

A shot is an example or demonstration of what type of prompt and response you expect from a large language model. This term originates from training computer vision models on photographs, where one shot was one example or instance that the model used to classify an image.

Zero-Shot Prompting

Large language models like Meta Llama are capable of following instructions and producing responses without having previously seen an example of a task. Prompting without examples is called "zero-shot prompting".

```
Text: This was the best movie I've ever seen!
The sentiment of the text is:
```

```
Text: The director was trying too hard.
The sentiment of the text is:
```

Few-Shot Prompting

Adding specific examples of your desired output generally results in a more accurate, consistent output. This technique is called "few-shot prompting". In this example, the generated response follows our desired format that offers a more nuanced sentiment classifier that gives a positive, neutral, and negative response confidence percentage.

```
You are a sentiment classifier. For each message, give the percentage of
positive/netural/negative. Here are some samples:
Text: I liked it
Sentiment: 70% positive 30% neutral 0% negative
Text: It could be better
Sentiment: 0% positive 50% neutral 50% negative
Text: It's fine

Sentiment: 25% positive 50% neutral 25% negative

Text: I thought it was okay

Text: I loved it!

Text: Terrible service 0/10
```

Role Based Prompts

Creating prompts based on the role or perspective of the person or entity being addressed. This technique can be useful for generating more relevant and engaging responses from language models.

Pros:

1. Improves relevance: Role-based prompting helps the language model understand the role or perspective of the person or entity being addressed, which can lead to more relevant and engaging responses.
2. Increases accuracy: Providing additional context about the role or perspective of the person or entity being addressed can help the language model avoid making mistakes or misunderstandings.

Cons:

1. Requires effort: Requires more effort to gather and provide the necessary information about the role or perspective of the person or entity being addressed.

Example:

```
You are a virtual tour guide currently walking the tourists Eiffel Tower
on a night tour. Describe Eiffel Tower to your audience that covers its
history, number of people visiting each year, amount of time it takes to
do a full tour and why do so many people visit this place each year.
```

Chain of Thought Technique

Involves providing the language model with a series of prompts or questions to help guide its thinking and generate a more coherent and relevant response. This technique can be useful for generating more thoughtful and well-reasoned responses from language models.

Pros:

1. Improves coherence: Helps the language model think through a problem or question in a logical and structured way, which can lead to more coherent and relevant responses.
2. Increases depth: Providing a series of prompts or questions can help the language model explore a topic more deeply and thoroughly, potentially leading to more insightful and informative responses.

Cons:

1. Requires effort: The chain of thought technique requires more effort to create and provide the necessary prompts or questions.

Example:

```
You are a virtual tour guide from 1901. You have tourists visiting Eiffel
Tower. Describe Eiffel Tower to your audience. Begin with
1. Why it was built
2. Then by how long it took them to build
3. Where were the materials sourced to build
4. Number of people it took to build
5. End it with the number of people visiting the Eiffel tour annually in
the 1900's, the amount of time it completes a full tour and why so many
people visit this place each year.
```

```
Make your tour funny by including 1 or 2 funny jokes at the end of the
tour.
```

Self-Consistency

LLMs are probabilistic, so even with Chain-of-Thought, a single generation might produce incorrect results. Self-Consistency introduces enhanced accuracy by selecting the most frequent answer from multiple generations (at the cost of higher compute):

```
John found that the average of 15 numbers is 40.
If 10 is added to each number then the mean of the numbers is?
Report the answer surrounded by three backticks, for example: ```123```
```

Running the above several times and taking the most commonly returned value for the answer would make use of the self-consistency approach.

Retrieval-Augmented Generation

Common facts are generally available from today's large models out-of-the-box (i.e. using just the model weights). More specific data is unlikely to be available though E.g.:


```
# The capital of California is Sacramento...

What was the temperature in Menlo Park on December 12th, 2023?
# I'm just an AI, I don't have have access to real-time or historical
weather data...
```

Retrieval-Augmented Generation, or RAG, describes the practice of including information in the prompt that has been retrieved from an external database. It's an effective way to incorporate facts into your LLM application and is more affordable than fine-tuning which might also negatively impact the foundational model's capabilities.

This could be as simple as a lookup table or as sophisticated as a vector database containing all of your company's knowledge:

```
Given the following information about temperatures in Menlo Park:
2023-12-11 :  52 degrees Fahrenheit
```

```
2023-12-12 :  51 degrees Fahrenheit
2023-12-13 :  55 degrees Fahrenheit
What was the temperature in Menlo Park on 2023-12-12?

# Sure! The temperature in Menlo Park on 2023-12-12 was 51 degrees
Fahrenheit.

What was the temperature in Menlo Park on 2023-07-18 ?

# Sorry, I don't have information about the temperature in Menlo Park on
2023-07-18. The information provided only includes temperatures for
December 11th, 12th, and 13th of 2023.
```

Program-Aided Language Models

LLMs, by nature, aren't great at performing calculations. While LLMs are bad at arithmetic, they're great for code generation. Program-Aided Language leverages this fact by instructing the LLM to write code to solve calculation tasks.

```
Python code to calculate: ((-5 + 93 * 4 - 0) * (4^4 + -7 + 0 * 5))

Here is the Python code to calculate the given expression:

((-5 + 93 * 4 - 0) * (4**4 + -7 + 0 * 5))

# Calculate the expression inside the parentheses
a = -5 + 93 * 4 - 0
b = 4**4 + -7 + 0 * 5

# Multiply a and b
result = a * b

# Print the result
print(result)
```

Using the code directly provides the correct result.

Limiting Extraneous Tokens

A common challenge is generating a response without extraneous tokens (e.g. "Sure! Here's more information on...").

By combining a role, rules and restrictions, explicit instructions, and an example, the model can be prompted to generate the desired response.

```
You are a robot that only outputs JSON.
You reply in JSON format with the field 'zip_code'.
Example question: What is the zip code of the Empire State Building?
Example answer: {'zip_code': 10118}
Now here is my question: What is the zip code of Menlo Park?

# "{'zip_code': 94025}"
```

Using the code directly provides the correct result.

Reduce Hallucinations

Meta's Responsible Use Guide is a great resource to understand how best to prompt and address input/output risks of the language model. Refer to pages (14-17).

Here are some examples of how a language model might hallucinate and some strategies for fixing the issue:

Example 1:

A language model is asked to generate a response to a question about a topic it has not been trained on. The language model may hallucinate information or make up facts that are not accurate or supported by evidence.

> **Fix:** To fix this issue, you can provide the language model with more context or information about the topic to help it understand what is being asked and generate a more accurate response. You could also ask the language model to provide sources or evidence for any claims it makes to ensure that its responses are based on factual information.

Example 2:

A language model is asked to generate a response to a question that requires a specific perspective or point of view. The language model may hallucinate information or make up facts that are not consistent with the desired perspective or point of view.

Fix: To fix this issue, you can provide the language model with additional information about the desired perspective or point of view, such as the goals, values, or beliefs of the person or entity being addressed. This can help the language model understand the context and generate a response that is more consistent with the desired perspective or point of view.

Example 3:

A language model is asked to generate a response to a question that requires a specific tone or style. The language model may hallucinate information or make up facts that are not consistent with the desired tone or style.

Fix: To fix this issue, you can provide the language model with additional information about the desired tone or style, such as the audience or purpose of the communication. This can help the language model understand the context and generate a response that is more consistent with the desired tone or style.

Overall, the key to avoiding hallucination in language models is to provide them with clear and accurate information and context, and to carefully monitor their responses to ensure that they are consistent with your expectations and requirements.

Meta

Documentation

Overview

Models

Getting the Models

Running Llama

How-To Guides

Integration Guides

Community Support

Community

Resources

Trust & Safety