O'REILLY

Prompt Engineering Deep Dive

30/04/2024

Table of Contents

Modules

Table of Contents

Modules

Table of Contents

Modules

Module #1

Introduction to Prompt Engineering

What is Prompt Engineering?

What is Prompt Engineering?

- Prompt engineering: Discipline for engineering prompts

What is Prompt Engineering?

- Prompt engineering: Discipline for engineering prompts
- The goal is to **design good prompts**

What is Prompt Engineering?

- Prompt engineering: Discipline for engineering prompts
- The goal is to **design good prompts**
- Process for developing prompts that yield high performance in a task.

Why Prompt Engineering?

Why Prompt Engineering?

- LLM's outputs are uncertain

Why Prompt Engineering?

- LLM's outputs are uncertain
- We need a systematic approach for searching solutions to problems

What is a Prompt?

What is a Prompt?

- Prompt is text that conveys the user's intention to the LLM.

What is a Prompt?

- Prompt is text that conveys the user's intention to the LLM.
- It can be a question, an instruction, request.

Prompt Basics

Components of the prompt

You are a text annotation engine.
Classify this sentence into positive or negative:
Text: *I like eating pancakes*
Output:

Prompt Basics

Components of the prompt: instruction

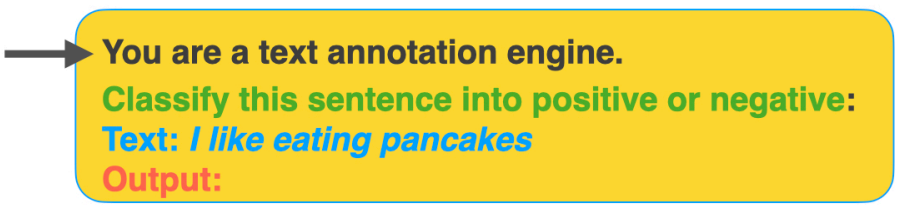
- **Instruction**: where you describe what you want

You are a text annotation engine.
Classify this sentence into positive or negative:
Text: *I like eating pancakes*
Output:

Prompt Basics

Components of the prompt: instruction, context

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.



Prompt Basics

Components of the prompt: instruction, context input data

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.
- **Input data:** example of data you wish the model to process

Prompt Basics

Components of the prompt: instruction, context input data , output indicator

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.
- **Input data:** example of data you wish the model to process
- **Output indicator:** Priming model to output what you want (e.g structure, length, etc..)

You are a text annotation engine.
Classify this sentence into positive or negative:
Text: ***I like eating pancakes***
Output:

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1: Write clear instructions**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1: Write clear instructions**
- Bad: Who’s president?

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1: Write clear instructions**
- Bad: Who’s president?
- Better: Who was the president of Mexico in 2021?

[3] OpenAI's Prompt Engineering Guide

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 2: Provide reference text**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 2: Provide reference text**

  SYSTEM
  Use the provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer."
  USER
  <insert articles, each delimited by triple quotes>
  Question: <insert question here>

[3] OpenAI's Prompt Engineering Guide

Prompt Engineering Guide

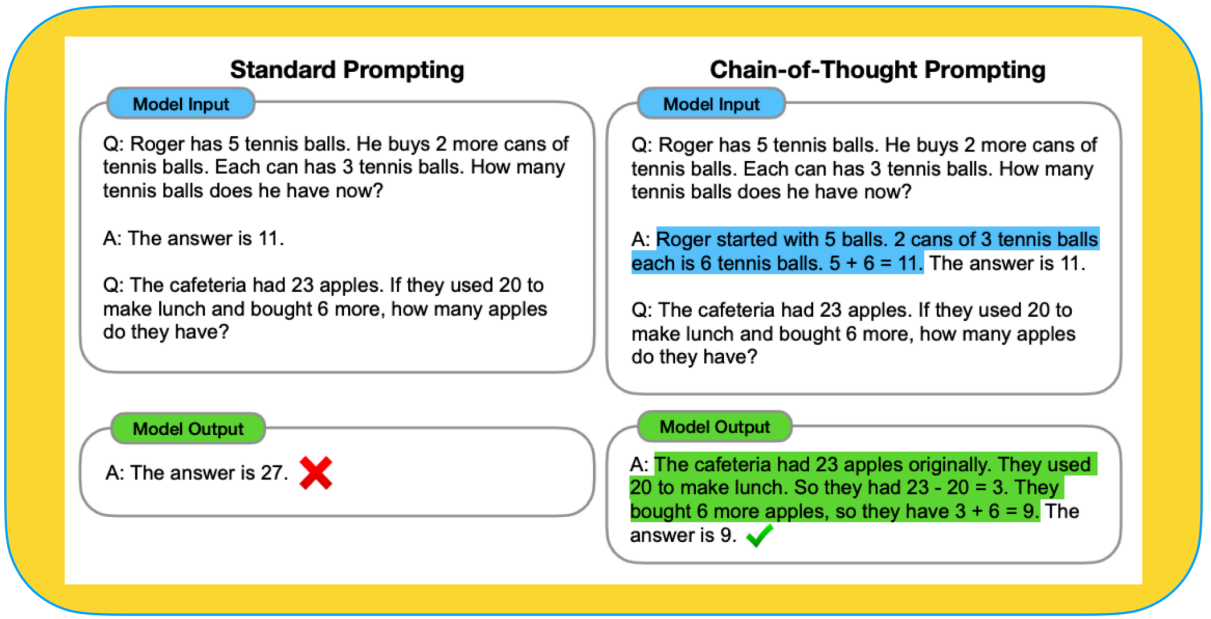
OpenAI's Guide for Building Good Prompts

- **Strategy 3: Break tasks into subtasks**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 3: Break tasks into subtasks**



[3] OpenAI's Prompt Engineering Guide

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 4: Give the model time to think**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 4: Give the model time to think**

SYSTEM
Follow these steps to answer the user queries.

Step 1 - First work out your own solution to the problem. Don't rely on the student's solution since it may be incorrect. Enclose all your work for this step within triple quotes (""").

Step 2 - Compare your solution to the student's solution and evaluate if the student's solution is correct or not. Enclose all your work for this step within triple quotes (""").

Step 3 - If the student made a mistake, determine what hint you could give the student without giving away the answer. Enclose all your work for this step within triple quotes (""").

Step 4 - If the student made a mistake, provide the hint from the previous step to the student (outside of triple quotes). Instead of writing "Step 4 - ..." write "Hint:".
USER
Problem Statement: <insert problem statement>

Student Solution: <insert student solution>

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 5: Use external tools**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 5: Use external tools**

SYSTEM
You can write and execute Python code by enclosing it in triple backticks, e.g. ```code goes here```. Use this to perform calculations.
USER
Find all real-valued roots of the following polynomial: 3*x**5 - 5*x**4 - 3*x**3 - 7*x - 10.

[3] OpenAI's Prompt Engineering Guide

Prompt Engineering Guide

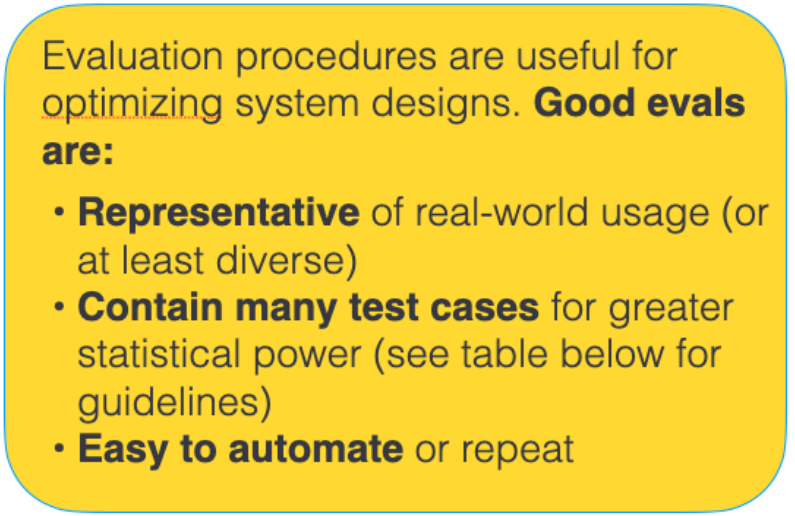
OpenAI's Guide for Building Good Prompts

- **Strategy 6: Test changes systematically**

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 6: Test changes systematically**



[3] OpenAI's Prompt Engineering Guide

Prompt Engineering Template

- Define Your Task Clearly
- Define the Eval Metric
- Generate Candidate Prompts
- Experiment
- Learning When to Stop Experimenting

Define Your Task Clearly

Define Your Task Clearly

- Clear and concise

Define Your Task Clearly

- Clear and concise
- Progress should be measurable

Define Your Task Clearly

- Clear and concise
- Progress should be measurable
- Settle on scope and constraints

Define Your Task Clearly

- Clear and concise
- Progress should be measurable
- Settle on scope and constraints
- Settle on your audience

Define Your Task Clearly

- Clear and concise
- Progress should be measurable
- Settle on scope and constraints
- Settle on your audience

Examples

- **Bad**: Summarize this paper

Define Your Task Clearly

- Clear and concise
- Progress should be measurable
- Settle on scope and constraints
- Settle on your audience

Examples

- **Bad**: Summarize this paper
- **Good**: Summarize the main contributions of this paper in bullet points for a non-technical audience.

Define the Eval Metric

Define the Eval Metric

- Establish criteria for evaluating the effectiveness of your prompts.

Define the Eval Metric

- Establish criteria for evaluating the effectiveness of your prompts.
- *What should I take into account when evaluating a prompt?*

Define the Eval Metric

- Establish criteria for evaluating the effectiveness of your prompts.
- *What should I take into account when evaluating a prompt?*
- This could include accuracy, creativity, relevance, or other metrics pertinent to your task.

Define the Eval Metric

- Establish criteria for evaluating the effectiveness of your prompts.
- *What should I take into account when evaluating a prompt?*
- This could include accuracy, creativity, relevance, or other metrics pertinent to your task.
- Problem: classifying emails into urgent and not urgent

Define the Eval Metric

- Establish criteria for evaluating the effectiveness of your prompts.
- *What should I take into account when evaluating a prompt?*
- This could include accuracy, creativity, relevance, or other metrics pertinent to your task.
- Problem: classifying emails into urgent and not urgent
- Metric: Accuracy

Generate Candidate Prompts

Generate Candidate Prompts

- Diverse set of prompts

Generate Candidate Prompts

- Diverse set of prompts
- Aiming for variety & creativity

Generate Candidate Prompts

- Diverse set of prompts
- Aiming for variety & creativity
- Keep them as short as possible without losing information

Experiment

Experiment

- **Test**: Run your candidate prompts through the intended system.

Experiment

- **Test**: Run your candidate prompts through the intended system.
- **Evaluate**: Assess the outputs based on your defined evaluation metrics.

Experiment

- **Test**: Run your candidate prompts through the intended system.
- **Evaluate**: Assess the outputs based on your defined evaluation metrics.
- **Compare**: Analyze the performance of each prompt against others.

Experiment

- **Test**: Run your candidate prompts through the intended system.
- **Evaluate**: Assess the outputs based on your defined evaluation metrics.
- **Compare**: Analyze the performance of each prompt against others.
- **Best Candidate**: Determine which prompt(s) best fulfill your task requirements.

Experiment

- **Test**: Run your candidate prompts through the intended system.
- **Evaluate**: Assess the outputs based on your defined evaluation metrics.
- **Compare**: Analyze the performance of each prompt against others.
- **Best Candidate**: Determine which prompt(s) best fulfill your task requirements.
- Tip: Utilize tools like Google Sheets for organized experimentation and tracking.

Learning When to Stop Experimenting

Learning When to Stop Experimenting

- Recognize the balance between thorough experimentation and practical limitations.

Learning When to Stop Experimenting

- Recognize the balance between thorough experimentation and practical limitations.
- Consider factors like:

Learning When to Stop Experimenting

- Recognize the balance between thorough experimentation and practical limitations.
- Consider factors like:
    - **Token/Cost**: The expense associated with processing each prompt.

Learning When to Stop Experimenting

- Recognize the balance between thorough experimentation and practical limitations.
- Consider factors like:
  - **Token/Cost**: The expense associated with processing each prompt.
  - **Time/Urgency/Priority**: Available time and the urgency of the task.

Learning When to Stop Experimenting

- Recognize the balance between thorough experimentation and practical limitations.
- Consider factors like:
  - **Token/Cost**: The expense associated with processing each prompt.
  - **Time/Urgency/Priority**: Available time and the urgency of the task.
  - Aim to avoid overengineering, as it can reduce the reusability and practicality of your solutions.

Summary

- **Prompt Engineering** is the discipline for engineering prompts
- **Prompt Engineering** aims to design prompts that yield high performance across tasks.
- **OpenAI's Guide for Building Good Prompts** includes strategies like writing clear instructions, providing reference text, breaking tasks into subtasks, giving the model time to think, using external tools, and testing changes systematically.
- **Prompt Engineering Template**: task, metric, candidates, experiment

Module #2

Prompt Engineering Techniques

Prompt Engineering Techniques

- Zero Shot Prompting
- Few-shot Prompting
- Chain-of-Thought (CoT)
- Self-Consistency
- Knowledge Generation
- Tree of Thoughts (ToT)

Zero Shot Prompting

Zero Shot Prompting

- Zero-shot prompting is when you solve the task without showing any examples of what a solution might look like

Zero Shot Prompting

- Zero-shot prompting is when you solve the task without showing any examples of what a solution might look like
- One can use this as the first try at a model to see what kind of tasks LLM can already solve out of the box

Zero Shot Prompting

- Zero-shot prompting is when you solve the task without showing any examples of what a solution might look like
- One can use this as the first try at a model to see what kind of tasks LLM can already solve out of the box
- *Classify the sentiment in this sentence as negative or positive: Text: I will go to a vacation. Sentiment:*

Few-Shot

Few-Shot

- Few-shot Prompting: technique where you show a few examples of what a solution might look like.

Few-Shot

- Few-shot Prompting: technique where you show a few examples of what a solution might look like.
- Translate from English to French:

  - I am a student --> Je suis un étudiant

  - I am a teacher --> Je suis un professeur

  - I am a doctor -->

Few-Shot

- Few-shot Prompting: technique where you show a few examples of what a solution might look like.
- Translate from English to French:

  - I am a student --> Je suis un étudiant

  - I am a teacher --> Je suis un professeur

  - I am a doctor -->
- Example taken from (Brown et al. 2020)

Chain-of-Thought

Chain-of-Thought

- Chain-of-thought (CoT) enables complex reasoning capabilities through intermediate reasoning steps (Wei et al. 2022).

Chain-of-Thought

- Chain-of-thought (CoT) enables complex reasoning capabilities through intermediate reasoning steps (Wei et al. 2022).

  Q: I have one sister and one brother. I am 20 years of age. My sister is 5 years older and my brother 2 years younger than my sister. How old is my brother?

Chain-of-Thought

- Chain-of-thought (CoT) enables complex reasoning capabilities through intermediate reasoning steps (Wei et al. 2022).

  Q: I have one sister and one brother. I am 20 years of age. My sister is 5 years older and my brother 2 years younger than my sister. How old is my brother?

  A: If I am 20 years of age and my sister is 5 years older, my sister is 20+5=25 years old. If my brother is 2 years younger than my sister, my brother is 25-2=23 years old. The answer is 23 years old.

Chain-of-Thought

- Chain-of-thought (CoT) enables complex reasoning capabilities through intermediate reasoning steps (Wei et al. 2022).

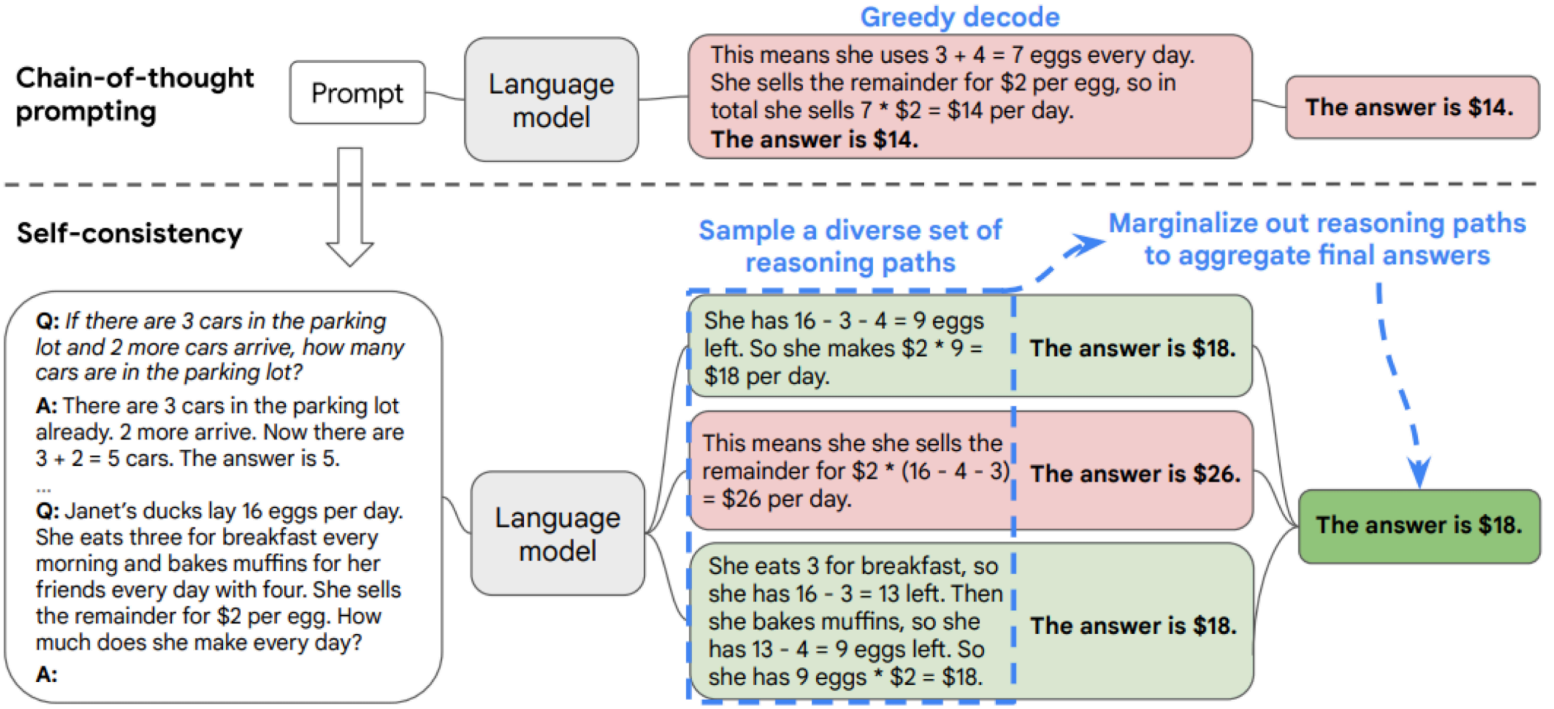
  Q: I have one sister and one brother. I am 20 years of age. My sister is 5 years older and my brother 2 years younger than my sister. How old is my brother?

  A: If I am 20 years of age and my sister is 5 years older, my sister is 20+5=25 years old. If my brother is 2 years younger than my sister, my brother is 25-2=23 years old. The answer is 23 years old.

  Q: I have 2 friends, Jack and Sally. Jack is 2 years older than Sally. Sally is 5 years younger than me. I am 17 years old. How old is Jack?

Self-Consistency

Self-Consistency



Knowledge Generation

Knowledge Generation

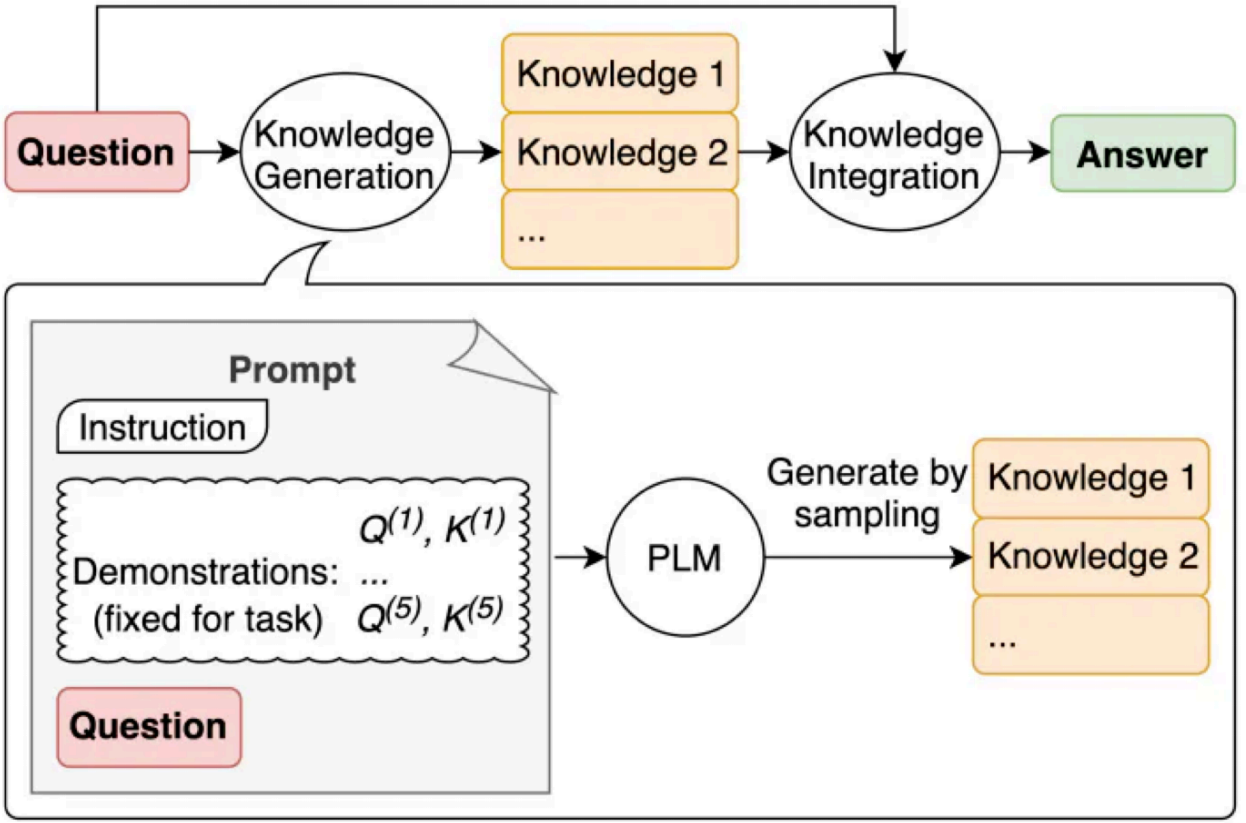
- Knowledge Generation: Generating facts related to the question.

Knowledge Generation

- Knowledge Generation: Generating facts related to the question.
- Knowledge Integration: Using generated facts to answer the question.

Knowledge Generation

- Knowledge Generation: Generating facts related to the question.
- Knowledge Integration: Using generated facts to answer the question.



Techniques

There are many more prompt engineering techniques that grow in complexity, such as:

- ToT (Yao et al. 2023)
- Retrieval Augmented Generation (Lewis et el. (2021))
- Automatic Prompt Engineer (Zhou et al., (2022))
- React Prompting (Yao et al., 2022)
- Graph Prompting (Liu et al., 2023)
- Skeleton of Thought (Ning et al. 2023)
- Step Back Prompting (Zheng et al. 2023)

Summary

- Zero-shot prompting: solve tasks without examples
- Few-shot prompting: show a few examples of what a solution might look like
- Chain-of-Thought: enables complex reasoning capabilities through intermediate reasoning steps
- Self-Consistency: ensures that the model's outputs are consistent with the input
- Knowledge Generation: generating facts related to the question
- There are many more prompt engineering techniques that grow in complexity

Module #3

Prompt Engineering Experiments

Designing Prompt Engineering Experiments

Designing Prompt Engineering Experiments

Evolving the Prompt Engineering Template

- Task, metric, candidates, experiment, iterate

Designing Prompt Engineering Experiments

Evolving the Prompt Engineering Template

- Task, metric, candidates, experiment, iterate
- **Systematic approach**
  - Task specification strategies

Designing Prompt Engineering Experiments

Evolving the Prompt Engineering Template

- Task, metric, candidates, experiment, iterate
- **Systematic approach**
  - Task specification strategies
  - Strategies for defining metrics

Designing Prompt Engineering Experiments

Evolving the Prompt Engineering Template

- Task, metric, candidates, experiment, iterate
- **Systematic approach**
  - Task specification strategies
  - Strategies for defining metrics
  - Strategies for prompt candidates generation

Designing Prompt Engineering Experiments

Evolving the Prompt Engineering Template

- Task, metric, candidates, experiment, iterate
- **Systematic approach**
  - Task specification strategies
  - Strategies for defining metrics
  - Strategies for prompt candidates generation
  - Strategies for experimentation

Task Specification Strategies

- Writing prompts constrains continuation to task completion

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components
- Signifiers in direct specification key intended behavior without explicit instruction

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components
- Signifiers in direct specification key intended behavior without explicit instruction
- **By proxy**: Uses analogies or characters as proxies for complex or nuanced intentions

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components
- Signifiers in direct specification key intended behavior without explicit instruction
- **By proxy**: Uses analogies or characters as proxies for complex or nuanced intentions
- Specification by proxy keys behaviors from cultural consciousness rather than direct naming

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components
- Signifiers in direct specification key intended behavior without explicit instruction
- **By proxy**: Uses analogies or characters as proxies for complex or nuanced intentions
- Specification by proxy keys behaviors from cultural consciousness rather than direct naming
- **By demonstration (n-shot)**: Effective for tasks requiring specific formats or instructive examples

Task Specification Strategies

- Writing prompts constrains continuation to task completion
- Few-shot and 0-shot: key jargon in prompt programming
- 0-shot prompts split into direct and proxy specifications
- **Direct**: Tells the model to perform a known task or construct a task from known components
- Signifiers in direct specification key intended behavior without explicit instruction
- **By proxy**: Uses analogies or characters as proxies for complex or nuanced intentions
- Specification by proxy keys behaviors from cultural consciousness rather than direct naming
- **By demonstration (n-shot)**: Effective for tasks requiring specific formats or instructive examples
- Constraining behavior to guide models towards intended continuations and avoid ambiguity

Strategies for defining metrics

Strategies for defining metrics

- **Clear objective**

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable
- **Systematic evaluation**

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable
- **Systematic evaluation**
  - Review Relevant Metrics: BLEU, ROUGE, METEOR, etc.

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable
- **Systematic evaluation**
  - Review Relevant Metrics: BLEU, ROUGE, METEOR, etc.
  - Task-Specific

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable
- **Systematic evaluation**
  - Review Relevant Metrics: BLEU, ROUGE, METEOR, etc.
  - Task-Specific
  - Automated evaluation

Strategies for defining metrics

- **Clear objective**
  - Understand and define the task unambiguously
- **Specify criteria**
  - Relevance, creativity, accuracy, fluency, coherence, etc.
- **Measurable outcomes**
  - Progress should be clearly measurable
- **Systematic evaluation**
  - Review Relevant Metrics: BLEU, ROUGE, METEOR, etc.
  - Task-Specific
  - Automated evaluation
- **Iterative feedback!**

Strategies for prompt candidates generation

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**
  - Meta prompts

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**
  - Meta prompts
  - Prompt templates

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**
  - Meta prompts
  - Prompt templates
  - Tooling: guidance, langchain, lmql, dspy, marvin, outlines, etc.

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**
  - Meta prompts
  - Prompt templates
  - Tooling: guidance, langchain, lmql, dspy, marvin, outlines, etc.
  - It's not about the tool!

Strategies for prompt candidates generation

- **Field expertise + effective prompt generation strategies**
  - Meta prompts
  - Prompt templates
  - Tooling: guidance, langchain, lmql, dspy, marvin, outlines, etc.
  - It's not about the tool!
  - It's about design patterns and best practices!

Strategies for experimentation

Strategies for experimentation

- **LLM specificity, domain knowledge, and iteration**

Strategies for experimentation

- **LLM specificity, domain knowledge, and iteration**
- **Trial and error with measurable outcomes**

Strategies for experimentation

- **LLM specificity, domain knowledge, and iteration**
- **Trial and error with measurable outcomes**
- **Systematic evaluation + iterative feedback**

Demo - Text Summarization Use Case

Demo - Code Generation Use Case

Demo - Q&A - Prompting to Understand Papers

Demo - Designing Prompt Engineering Workflow

Summary

- Evolving the Prompt Engineering Template
- Task specification strategies
- Strategies for defining metrics
- Strategies for prompt candidates generation
- Strategies for experimentation