GenAI

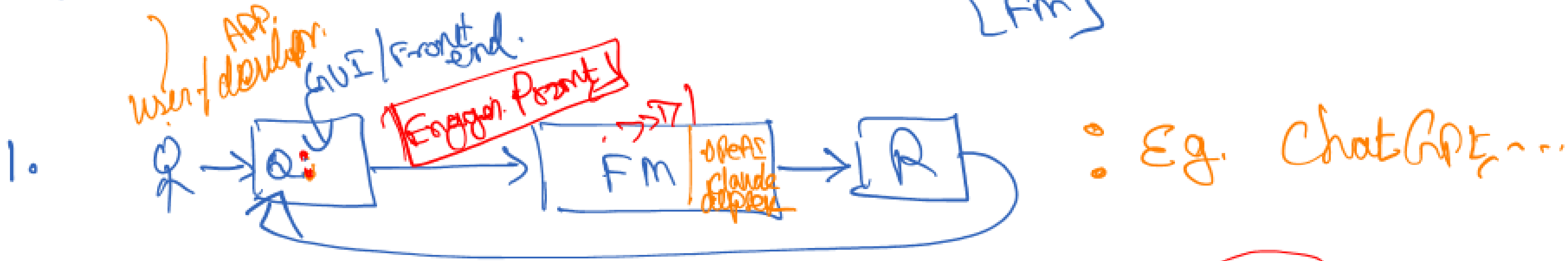
Prompt Engineering

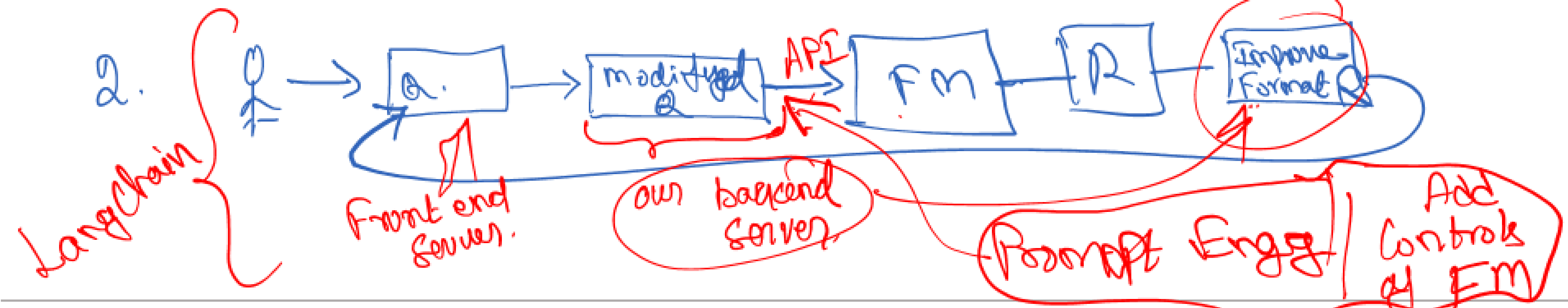
Topics

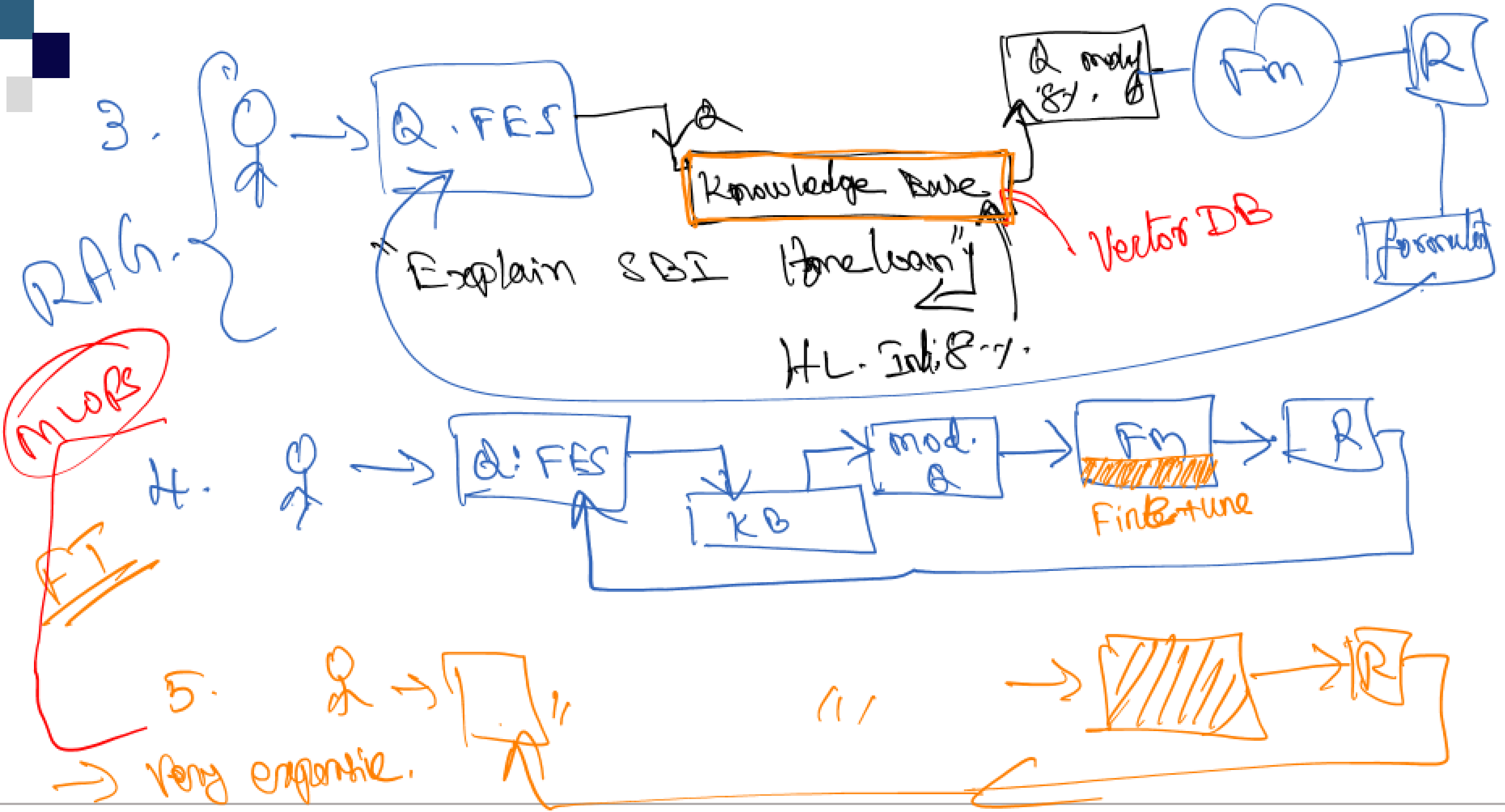
- The Anatomy of an Effective Prompt
- Advanced Techniques in Prompt Engineering
- LLM Settings in Prompt Engineering
- Prompt Examples
- Summary

Encoders / Decoders | BERT . . . . [ML systems]

GenAI Assume: ∃ a Foundational Model (Eg. GPT 3.0) [FM]

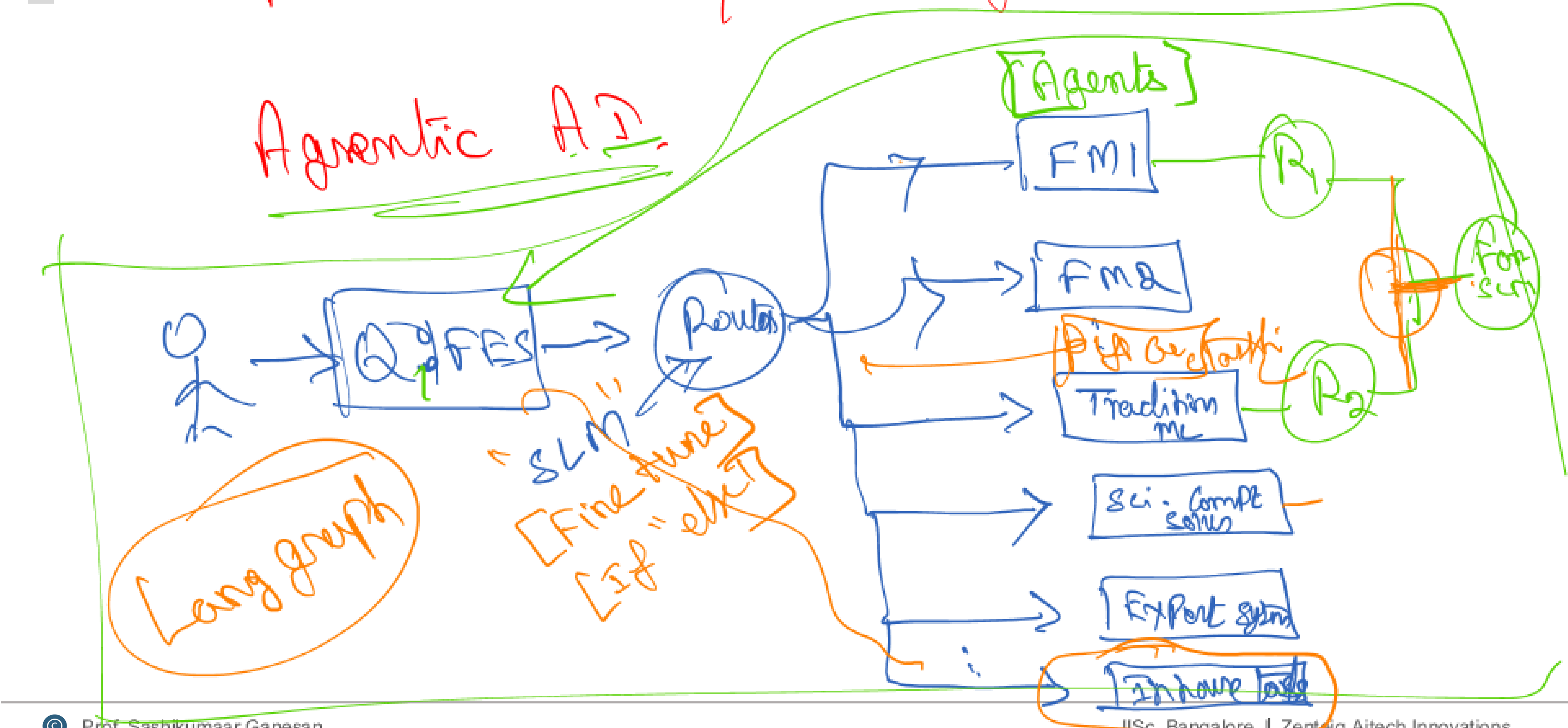




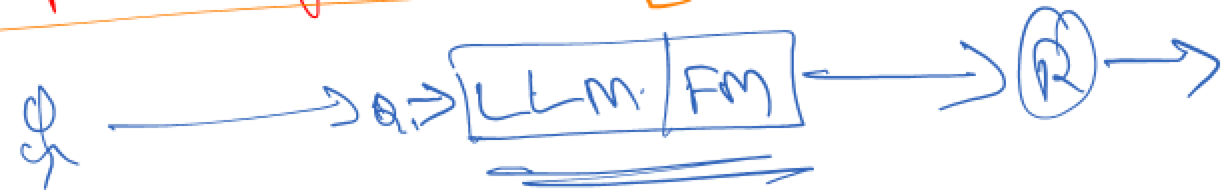

3.
RAG.
Q: FES
Q
Knowledge Base
Vector DB
Q mody '8%', Q
Fm
R
Formulat
"Explain SBI Home loan"
HL. Int 8%.
MLOps
4.
Q: FES
KB
mod. Q
Fm
R
Fine-tune
FT
5.
"
''
R
→ Very expensive.

Recent Trend / State of the Art: i>

Agentic AI



Compute Infrastructure [the size of RAM & Power of the thread GPU/CPU]



Case 1 – 3 : Inference is the only Compute Cost w.r.t FM's

4 & 5 : Training & Inference. w.r.t FM

Inference

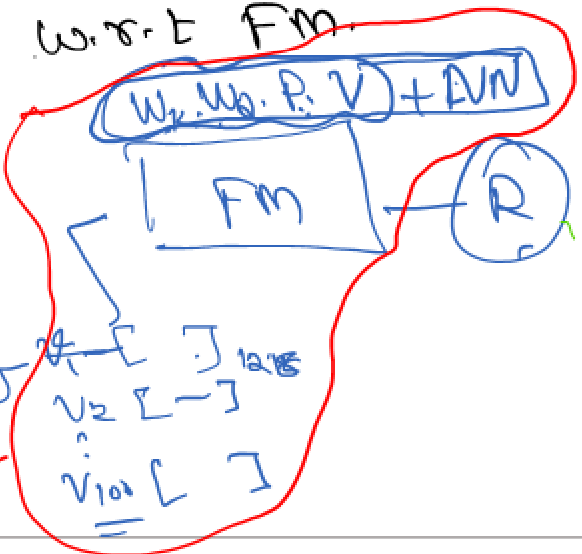
Q

" Context length " → tokens

tokens

100 tokens

FLOPs ⟸ { Tensor operations



The Anatomy of an Effective Prompt

Structure, Clarity & Precision

… the essential building blocks of a powerful prompt

meet @ 11 am.

The Anatomy of an Effective Prompt

Key Element 1: Structure

The structure of a prompt refers to how it is organized to guide the AI's understanding and response generation.

- **Correct**:
  - **Clear and concise instructions:** The prompt should provide a well-defined starting point, outlining the task and desired outcome.
  - **Logical flow**: Structure the prompt in a way that guides the AI step-by-step, ensuring a coherent thought process.
- **Incorrect**:
  - **Vague or open-ended instructions**: Without clear direction, the AI may generate irrelevant or nonsensical outputs.
  - **Disorganized structure:** A jumbled prompt can confuse the AI, leading to unpredictable results.

The Anatomy of an Effective Prompt

Key Element 1: Structure : Examples

- Ex1:
  - Incorrect: "Write something on global warming."
  - Correct: "Write an informative article on the causes and effects of global warming."
    - *Explanation: The correct prompt provides a clear structure directing the AI to produce an article format with specified content.*
- Ex2:
  - Incorrect: "Make a blog."
  - Correct: "Create a blog post about recent advancements in AI technology."
    - *Explanation: The correct prompt specifies the type of content, making it clear that a detailed blog post is required.*

The Anatomy of an Effective Prompt

Key Element 1: Structure : Examples

- Ex3:
  - Incorrect: ”Draft an Email."
  - Correct: "Draft an email to a client discussing our project timeline and next steps."
    - *Explanation: The correct prompt specifies the format (email) and includes details about what should be discussed.*
- Ex4:
  - Incorrect: "Help code."
  - Correct: "Generate a Python script to sort a list of numbers in ascending order."
    - *Explanation: The correct prompt specifies the language and task, providing a clear structure for the response.*

The Anatomy of an Effective Prompt

Key Element 2: Clarity

The language used in the prompt should be unambiguous and easy for the AI to comprehend.

- **Correct**:
  - **Simple and direct language**: Avoid complex jargon or overly technical terms.
  - **Precise vocabulary**: Use words that have clear and defined meanings in the context of the task.
- **Incorrect**:
  - **Ambiguous or figurative language**: Language open to interpretation can lead to misconstrued prompts and unexpected outputs.
  - **Informal or conversational language**: Maintain a professional tone while keeping the language clear and concise.

The Anatomy of an Effective Prompt

Key Element 2: Clarity: Examples

- Ex1:
  - Incorrect: "Do the needful for the report."
  - Correct: "Please update the final report with the latest sales figures and conclusions."
    - *Explanation: The correct prompt clearly communicates the specific actions needed.*
- Ex2:
  - Incorrect: "Book info."
  - Correct: "Provide a summary of the book titled 'Brave New World' by Aldous Huxley."
    - *Explanation: The correct prompt explicitly states what information is needed and about which book.*

The Anatomy of an Effective Prompt

Key Element 2: Clarity: Examples

- Ex3:
  - Incorrect: "Make better."
  - Correct: "Improve the readability of the attached document by simplifying complex sentences."
    - *Explanation: The correct prompt specifies what "better" means in this context.*
- Ex4:
  - Incorrect: "Plan vacation."
  - Correct: "Organize a 7-day itinerary for a family vacation to Italy, including major attractions."
    - *Explanation: The correct prompt clarifies what kind of planning is required and for where.*

The Anatomy of an Effective Prompt

Key Element 3: Precision

Precision in the prompt is crucial for achieving the desired level of detail and accuracy in the AI's response. It involves using specific terms and details that tailor the AI's output to your exact needs.

- Correct:
  - **Specific details and instructions:** The more information you provide, the better the AI can tailor its response to your needs.
  - **Examples and references (when applicable):** Including relevant examples or references can further refine the AI's understanding of your intent.
- Incorrect:
  - **Generic or lacking details:** A vague prompt will likely result in a generic or uninformative response from the AI.
  - **Focus on quantity over quality:** Don't overload the prompt with unnecessary details that might distract the AI from the core objective.

The Anatomy of an Effective Prompt

Key Element 3: Precision : Examples

- Ex1:
  - Incorrect: "Write about technology."
  - Correct: "Write a detailed comparison between quantum computing and classical computing."
    - *Explanation: The correct prompt narrows down the topic to a specific comparative analysis.*
- Ex2:
  - Incorrect: "Describe dog."
  - Correct: "Describe the physical characteristics and temperament of a German Shepherd."
    - *Explanation: The correct prompt is specific about the breed and aspects to be described.*

The Anatomy of an Effective Prompt

Key Element 3: Precision : Examples

- Ex3:
  - Incorrect: "Music article."
  - Correct: "Compose an article on the evolution of jazz music from the 1920s to the present."
    - *Explanation: The correct prompt specifies the genre of music and the timeline for the article.*
- Ex4:
  - Incorrect: "Dinner recipe."
  - Correct: "Provide a recipe for a vegetarian Tamil curry including ingredients and cooking instructions."
    - *Explanation: The correct prompt specifies the type of cuisine, dietary preferences, and detailed information needed.*

Code:

Case 1

VSC

1. Copilot

2. Windsurf

Claude 4.0

3. Claude Code

Case 2:

Min 2 users

ChatGPT (Pro)

min users: 5

Claude Teams

Project (own files

RAG

Case 3 (RAG)

Advanced Techniques in Prompt Engineering

Case 2:

LLM Settings in Prompt Engineering

Why Prompt Engineering

- One of the biggest problems with LLMs is their tendency to make up incorrect information, also known as hallucinations.
- Several solutions and tactics have emerged to try to tackle hallucinations, like Retrieval-Augmented Generation (RAG), fine-tuning models, and prompt engineering.
- Setting up RAG ✓ can be challenging, and fine-tuning models can be expensive.
- Prompt engineering is fast, easy to test, and anyone can do it.

Advanced Techniques in Prompt Engineering

Zero-Shot Learning

Craft prompts that enable AI to perform tasks even without prior training on specific examples.

- Suppose you want an AI to summarize a scientific paper you haven't read yet
- Incorrect (Requires Specific Training):
  - Summarize the latest research on protein folding for biologists.
  - This would require training on protein folding research.
- Correct (Zero-Shot Prompt):
  - You are given a scientific paper.
  - Can you summarize the key findings of the paper in a concise and informative way, similar to how you would explain them to a high school student?
  - By providing clear instructions and **context**, the AI can attempt the task even without specific training

Advanced Techniques in Prompt Engineering

Zero-Shot Learning

- Incorrect Example:
  - Prompt: "Quantum mechanics."
    - *Explanation: This prompt is too vague and does not guide the AI to perform a specific task, leading to potentially broad and untargeted content.*
- Correct Example:
  - Prompt: "Explain the principles of quantum mechanics to a 10-year-old."
    - *Explanation: The AI must adapt its knowledge to suit a child's understanding, demonstrating an ability to apply complex concepts in a simplified manner without specific prior examples.*

Advanced Techniques in Prompt Engineering

Chain-of-Thought (CoT)

Break down complex tasks into a sequence of smaller, more manageable prompts, guiding the AI through a step-by-step reasoning process.

- Imagine you want an AI to write a poem about a robot falling in love.
- Incorrect (Single Prompt):
  - Write a poem about a robot falling in love.
  - This might result in a generic poem lacking depth.
- Correct (Chained Prompt):
  - Describe a robot who works in a factory, performing repetitive tasks.
  - Introduce a new worker, a human, and show the robot becoming fascinated by them.
  - Write a poem from the robot's perspective, expressing its feelings of love for the human.
  - Chaining allows the AI to build upon each step, creating a more nuanced narrative

Advanced Techniques in Prompt Engineering

Chain-of-Thought (CoT):

- Incorrect :
  - Prompt 1: "List five key factors that contribute to climate change."
  - Prompt 2: "How can renewable energy reduce carbon footprints?"
    - *Explanation: The second prompt does not logically follow from the first. It introduces a new topic without utilizing the information from the initial output.*
- Correct :
  - Prompt 1: "List five key factors that contribute to climate change."
  - Prompt 2: "Based on these factors, suggest three potential solutions to mitigate climate change."
    - *Explanation: This chaining correctly builds upon the initial response, directing the AI to apply the first output in subsequent reasoning.*

“

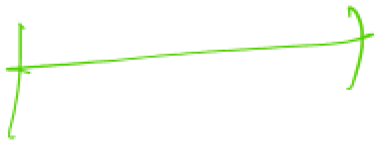
LLM Settings in Prompt Engineering

LLM Settings in Prompt Engineering

Introduction to control

When designing and testing prompts, control settings on the LLM API are crucial for fine-tuning the AI's response. These settings adjust the AI's behavior to produce more reliable and desirable responses based on specific use cases.

- Example
  - Temperature
  - Top P (Nucleus Sampling)
  - Max Length
  - Stop Sequences
  - Frequency Penalty
  - Presence Penalty



LLM Settings in Prompt Engineering

Temperature control

Controls the randomness of the AI's responses.

- Low Temperature results in more deterministic and factual responses.
  - It is ideal for tasks like fact-based Q&A.
- High Temperature increases creativity and diversity in responses.
  - It is suitable for tasks like poetry or creative writing.

I like : ______

↑

Predict?

LLM Settings in Prompt Engineering

Temperature control

$T \in [-1, +1]$

$T \in [0, 2)$

- Correct Example:
  - Setting: Temperature at 0.3
  - Prompt: "Write a brief summary of the latest UN climate change report."
    - *Explanation: A low temperature setting is appropriate here to ensure factual accuracy and a coherent summary based on a reliable interpretation of data.*
- Incorrect Example:
  - Setting: Temperature at 0.9
  - Prompt: "Write a brief summary of the latest UN climate change report."
    - *Explanation: A high temperature for this task may produce a summary with irrelevant or creatively interpreted facts, which is undesirable for accurate reporting.*

LLM Settings in Prompt Engineering

Temperature control

- Correct Example:
  - Prompt: "Write a poem about the ocean."
  - Temperature Setting: High (to encourage more creative and abstract outputs).
    - *Explanation: A higher temperature setting here promotes creativity, which is suitable for poetic expressions.*
- Incorrect Example:
  - Prompt: "Write a step-by-step guide on how to file taxes."
  - Temperature Setting: High.
    - *Explanation: A high temperature for this task might produce creative but impractical and incorrect tax advice. Lower temperature is preferred for factual accuracy and coherence.*

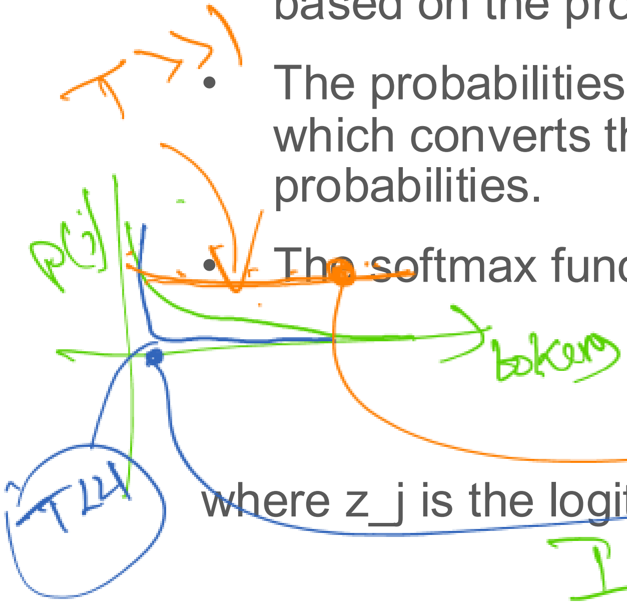
LLM Settings in Prompt Engineering

Temperature control

- LLMs predict the next token (e.g., word or punctuation) in a sequence based on the probability distribution derived from the training data.
- The probabilities for each token are computed using a softmax function, which converts the raw output scores (logits) from the model into probabilities.
- The softmax function for a token "j" is calculated as follows

$$P(j) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

where z_j is the logit for token j, and k indexes over all possible tokens.



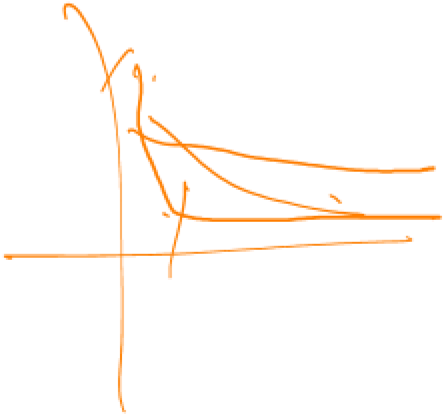
I like ___

predict?

0.08
0.06
0.059
0.05
0.0003

$\hat{P}$
0.8
0.1
0.01

LLM Settings in Prompt Engineering



Temperature control

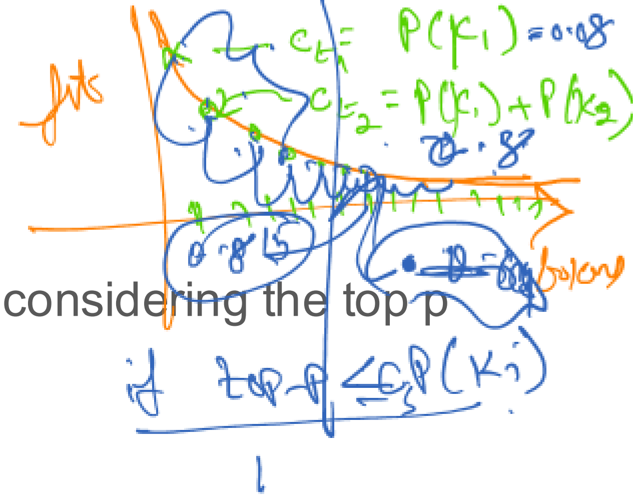
- Temperature modifies the softmax function $$P(j) = \frac{e^{\frac{z_j}{T}}}{\sum_k e^{\frac{z_k}{T}}}$$
- T > 1: This makes the softmax distribution flatter.
- T < 1: This sharpens the softmax distribution, making the highest logit significantly more pronounced compared to others.
- For logits [2.5, 1.0, 0.5]:
  - At T = 1: Softmax might yield probabilities like [0.70, 0.20, 0.10].
  - At T = 2: The flatter softmax might yield probabilities like [0.50, 0.30, 0.20].
  - At T = 0.5: The sharper softmax might yield probabilities like [0.85, 0.10, 0.05], heavily favoring the first token.

https://colab.research.google.com/drive/1EAKB4mbiB0RFPSyhJKn8X200OAx-UsAB?usp=sharing

LLM Settings in Prompt Engineering

Top_P (Nucleus Sampling)

Controls the breadth of possible responses by only considering the top p probability mass.



$c_{t_1} = P(K_1) = 0.08$
$c_{t_2} = P(K_1) + P(K_2)$
0.8
0.85
0.8
below
if top-p $\leq c_i P(K_i)$

- Correct Example:
  - Setting: Top P at 0.2
  - Prompt: "What are the main symptoms of influenza?"
  - Explanation: A low Top P setting is used to ensure that the response is focused and factual, drawing from the most likely tokens related to medical accuracy.
- Incorrect Example:
  - Setting: Top P at 0.9
  - Prompt: "What are the main symptoms of influenza?"
  - Explanation: A high Top P value in this context may lead to the model generating less likely symptoms, reducing the factual accuracy of the medical information.

LLM Settings in Prompt Engineering

Top_P (Nucleus Sampling)

1. Compute Probabilities: First, the model calculates the softmax probabilities for each token in the vocabulary based on the logits output by the model.
2. Sort Probabilities: The tokens are then sorted by their probability, from highest to lowest.
3. Cumulative Probability Calculation: The cumulative probability is computed for the sorted list of tokens. This process continues until the sum of the probabilities reaches or exceeds the threshold specified by top_p.
4. Token Selection: Only those tokens that are part of the cumulative sum that meets or exceeds top_p are considered for selection. The actual token is then randomly sampled from this reduced set according to their probabilities.

LLM Settings in Prompt Engineering

Top_P (Nucleus Sampling)

- Example of Top_p Calculation
  - Suppose the model produces the following probabilities for a set of tokens:
    - Token A: 0.4 Token B: 0.3 Token C: 0.2 Token D: 0.1
  - If top_p is set to 0.7, the tokens are considered in the following way:
    - Calculate cumulative probabilities as we add each token starting from the most probable:
    - Token A (0.4), Token A + Token B (0.4 + 0.3 = 0.7)
    - Since adding Tokens A and B reaches the cumulative probability of 0.7, Tokens C and D are excluded from the selection pool, even if their combined probabilities would not substantially exceed the threshold.

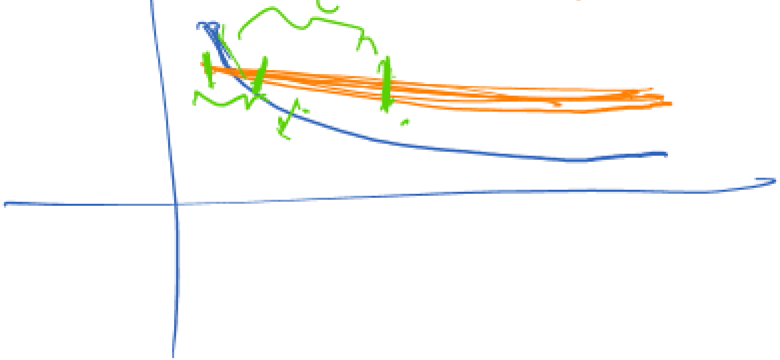
LLM Settings in Prompt Engineering

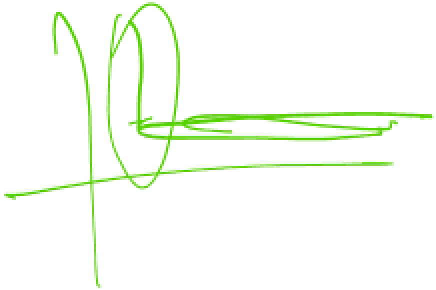
Temperature & Top_P

What to Expect from These Settings?

- High Temperature with Low Top_p
- Low Temperature with High Top_p

Contradictory





LLM Settings in Prompt Engineering

Max Length

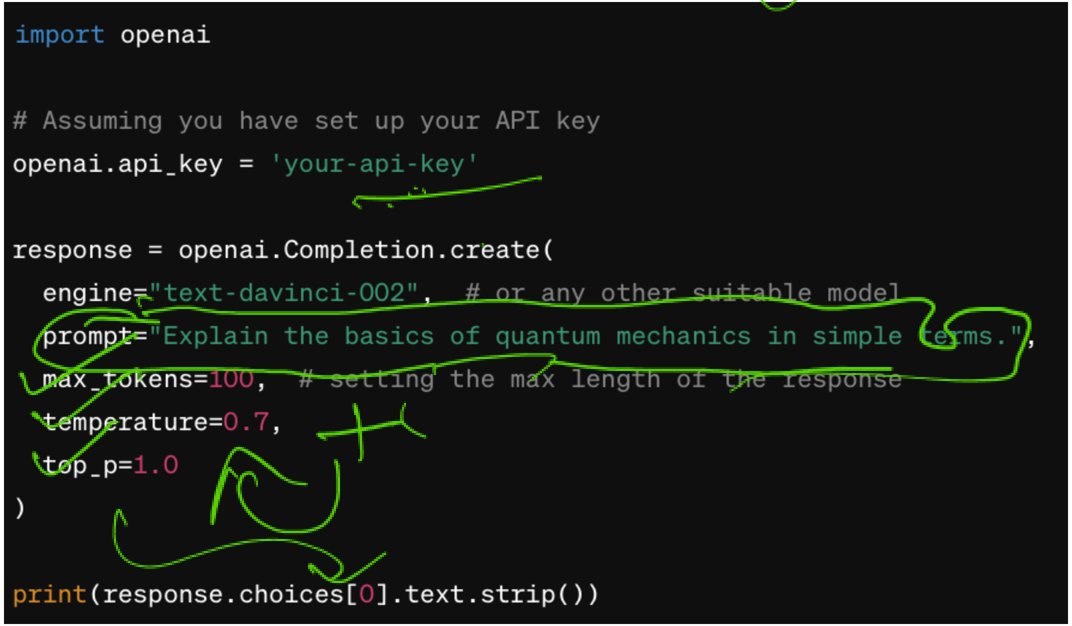
Sets the limit on the number of tokens the model can generate, controlling response length.

- Correct Example:
  - Setting: Max Length at 50 tokens
  - Prompt: "Create a concise market analysis for today."
  - Explanation: This setting ensures the analysis remains brief and to the point, suitable for a quick update.
- Incorrect Example:
  - Setting: Max Length at 500 tokens
  - Prompt: "Create a concise market analysis for today."
  - Explanation: Setting a high max length for a task that requests conciseness can lead to overly verbose and potentially less focused content.

LLM Settings in Prompt Engineering

Temperature, top_p & Max Length

Case 2:



```
import openai

# Assuming you have set up your API key
openai.api_key = 'your-api-key'

response = openai.Completion.create(
  engine="text-davinci-002",  # or any other suitable model
  prompt="Explain the basics of quantum mechanics in simple terms.",
  max_tokens=100,  # setting the max length of the response
  temperature=0.7,
  top_p=1.0
)

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

LLM Settings in Prompt Engineering

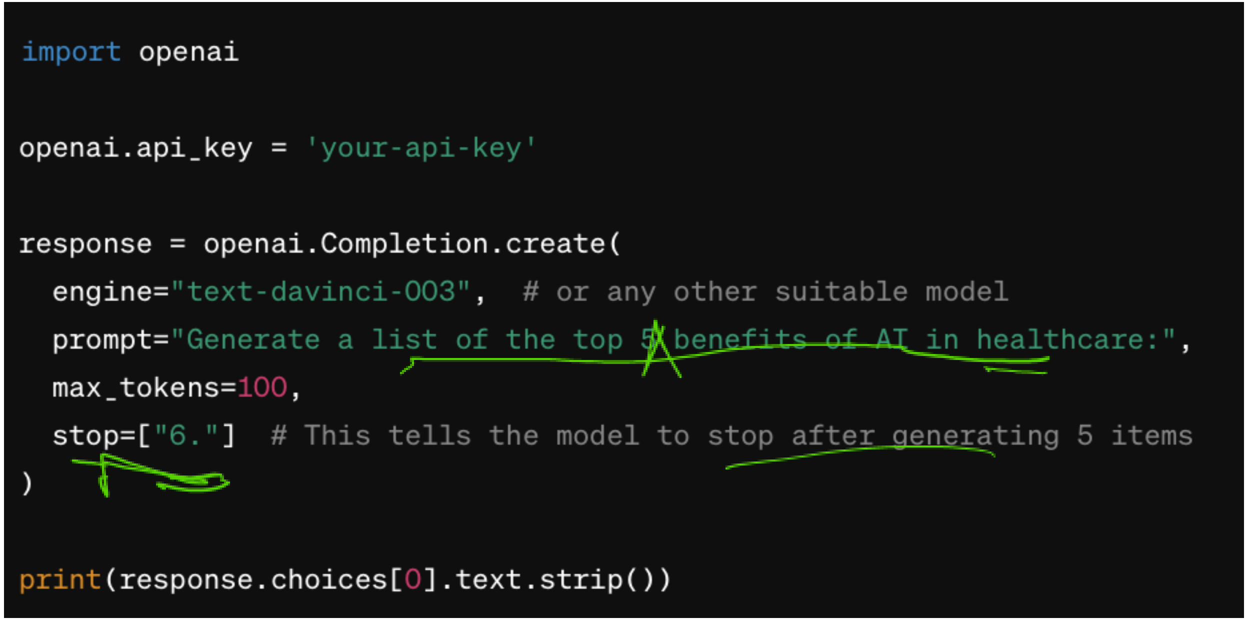
Stop Sequences

A specific string that tells the model when to stop generating further tokens.

- Correct Example:
  - Setting: Stop Sequence as "End of report."
  - Prompt: "Generate a financial report and conclude with 'End of report.'"
  - Explanation: The stop sequence effectively ensures the model concludes the output appropriately, maintaining the structure of the content.
- Incorrect Example:
  - Setting: Stop Sequence as "Thank you."
  - Prompt: "Generate a list of software updates."
  - Explanation: "Thank you" is an inappropriate stop sequence for a list, likely causing a premature end to the content or an awkward conclusion.

LLM Settings in Prompt Engineering

Stop Sequences



```python
import openai

openai.api_key = 'your-api-key'

response = openai.Completion.create(
  engine="text-davinci-003",  # or any other suitable model
  prompt="Generate a list of the top 5 benefits of AI in healthcare:",
  max_tokens=100,
  stop=["6."]  # This tells the model to stop after generating 5 items
)

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

LLM Settings in Prompt Engineering

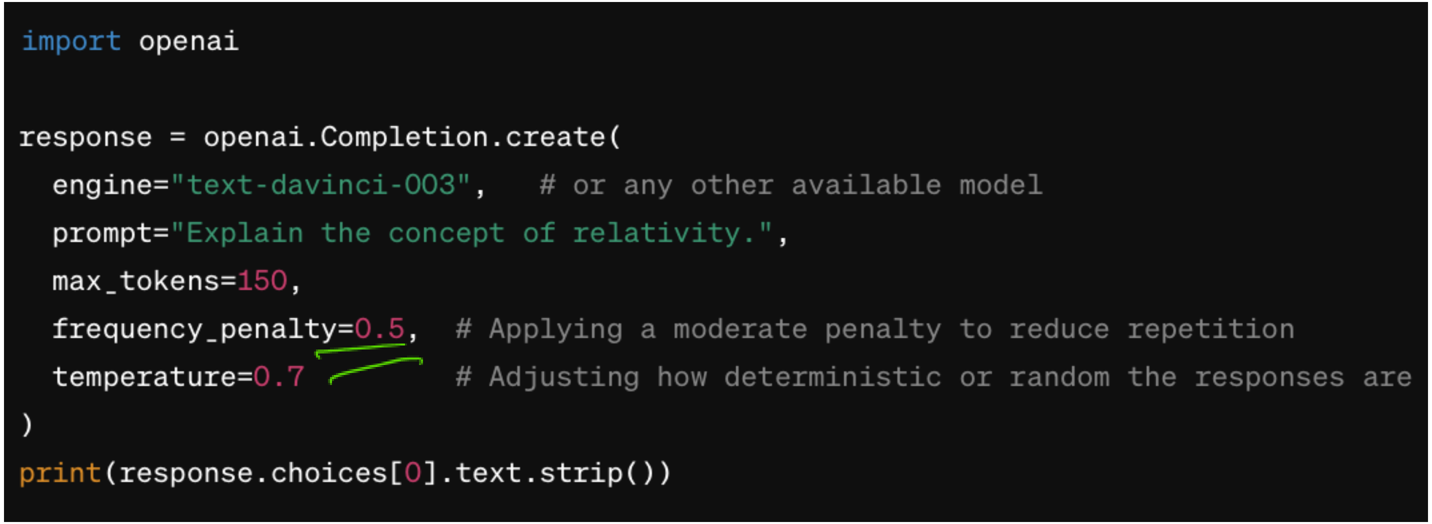
Frequency Penalty

Penalizes tokens based on their frequency to reduce repetition.

- Correct Example:
  - Setting: Frequency Penalty at 0.5
  - Prompt: "Describe the various uses of solar energy."
  - Explanation: This setting discourages the repetition of common terms like "energy", fostering a more diverse and informative response.
- Incorrect Example:
  - Setting: Frequency Penalty at 0
  - Prompt: "Describe the various uses of solar energy."
  - Explanation: Without a penalty, there may be excessive repetition of the word "energy", making the content less readable and engaging.

LLM Settings in Prompt Engineering

Frequency Penalty



```
import openai

response = openai.Completion.create(
  engine="text-davinci-003",   # or any other available model
  prompt="Explain the concept of relativity.",
  max_tokens=150,
  frequency_penalty=0.5,  # Applying a moderate penalty to reduce repetition
  temperature=0.7         # Adjusting how deterministic or random the responses are
)
print(response.choices[0].text.strip())
```

LLM Settings in Prompt Engineering

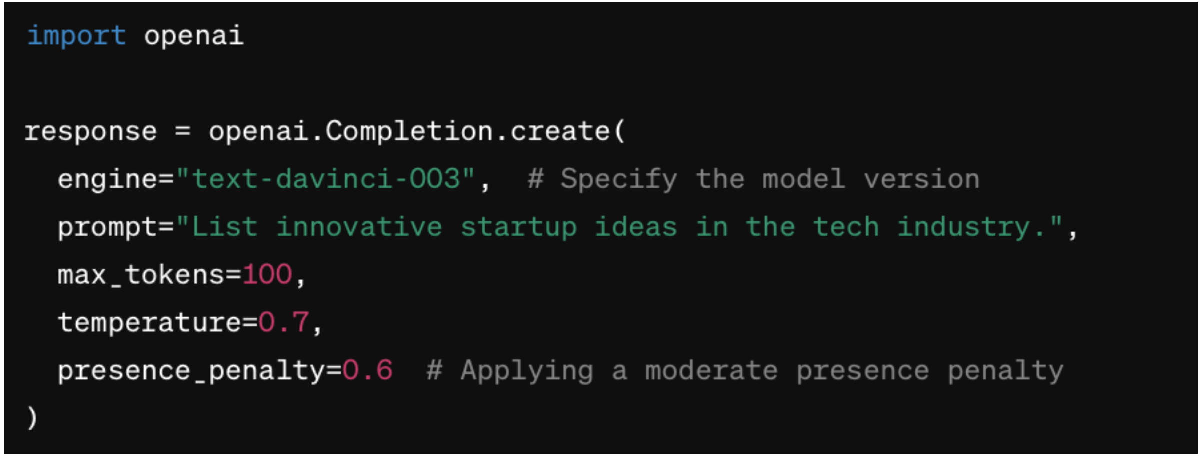
Presence Penalty

Penalizes tokens based on their presence, aiming to enhance content diversity by preventing phrase repetitions.

- Correct Example:
  - Setting: Presence Penalty at 0.6
  - Prompt: "Generate creative ideas for an advertising campaign."
  - Explanation: A higher presence penalty promotes a wider variety of ideas by minimizing the repetition of any single concept.
- Incorrect Example:
  - Setting: Presence Penalty at 0
  - Prompt: "Generate creative ideas for an advertising campaign."
  - Explanation: With no presence penalty, the model might repeatedly suggest similar or identical ideas, reducing the creativity of the output.

LLM Settings in Prompt Engineering

Presence Penalty



```
import openai

response = openai.Completion.create(
  engine="text-davinci-003",  # Specify the model version
  prompt="List innovative startup ideas in the tech industry.",
  max_tokens=100,
  temperature=0.7,
  presence_penalty=0.6  # Applying a moderate presence penalty
)
```

- Using the presence penalty is particularly effective in generating content that requires a wide range of ideas without repetition.
- This makes it ideal for creative and brainstorming tasks where novelty is valued.

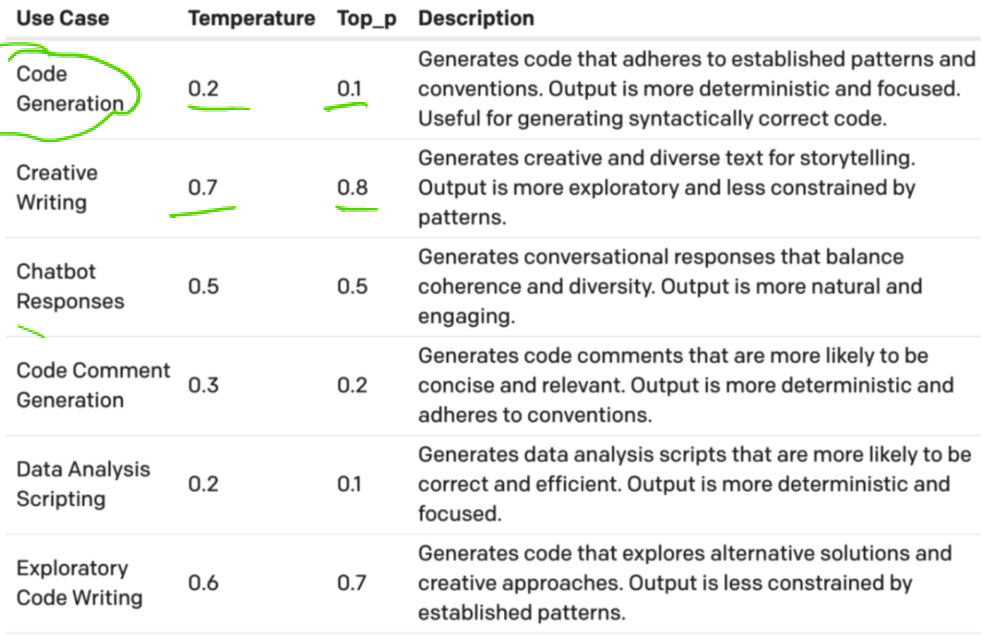
LLM Settings in Prompt Engineering

Presence Penalty

- How Presence Penalty Works
  - When a token appears in the generated text, the presence penalty is applied to discourage the model from using that token again in the rest of the text.
  - This is different from the frequency penalty, which applies a growing penalty with each repeated use of a token.
  - The presence penalty applies a flat penalty rate to a token as soon as it is used once, aiming to encourage the generation of new and unique content rather than repeating the same words or phrases.

LLM Settings in Prompt Engineering

Recommendations on Temperature and Top P



Use Case	Temperature	Top_p	Description
Code Generation	0.2	0.1	Generates code that adheres to established patterns and conventions. Output is more deterministic and focused. Useful for generating syntactically correct code.
Creative Writing	0.7	0.8	Generates creative and diverse text for storytelling. Output is more exploratory and less constrained by patterns.
Chatbot Responses	0.5	0.5	Generates conversational responses that balance coherence and diversity. Output is more natural and engaging.
Code Comment Generation	0.3	0.2	Generates code comments that are more likely to be concise and relevant. Output is more deterministic and adheres to conventions.
Data Analysis Scripting	0.2	0.1	Generates data analysis scripts that are more likely to be correct and efficient. Output is more deterministic and focused.
Exploratory Code Writing	0.6	0.7	Generates code that explores alternative solutions and creative approaches. Output is less constrained by established patterns.

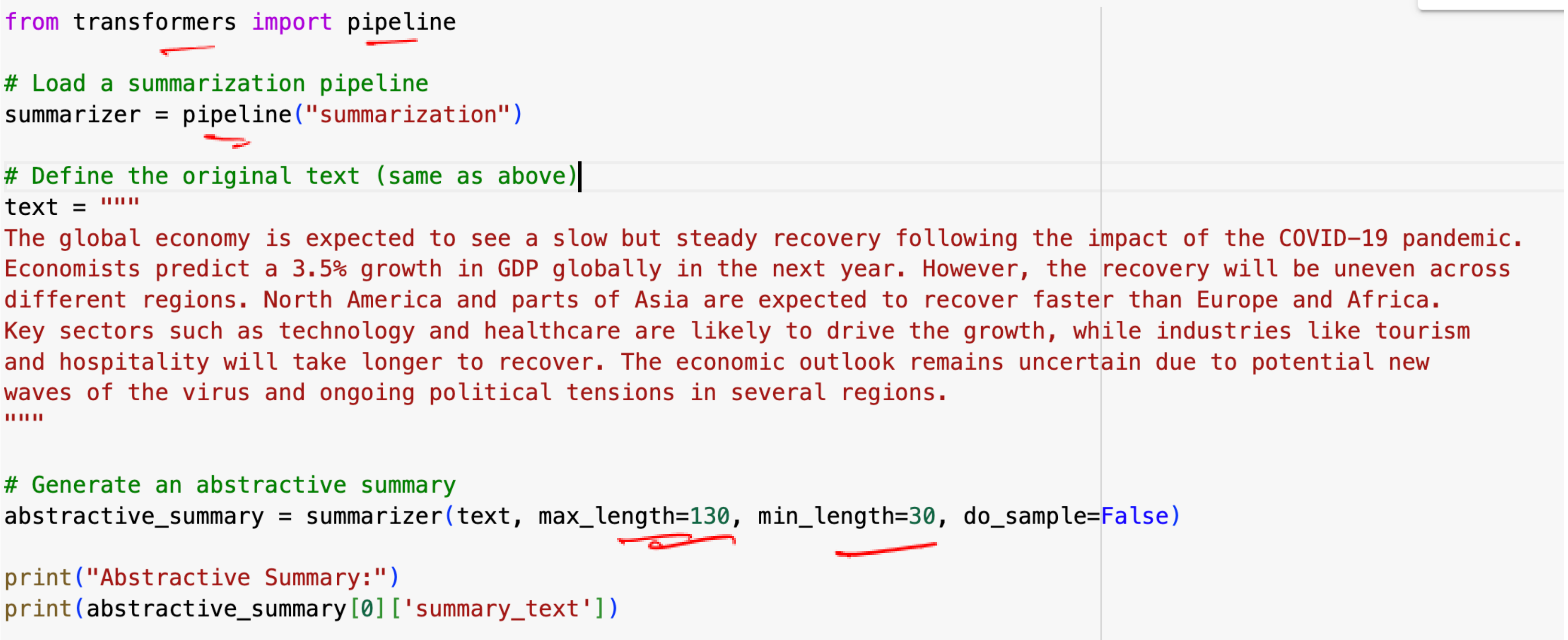
OpenAI's Temperature and Top P recommendations based on use case

Source: prompthub.us

Prompt Examples

Prompt Examples

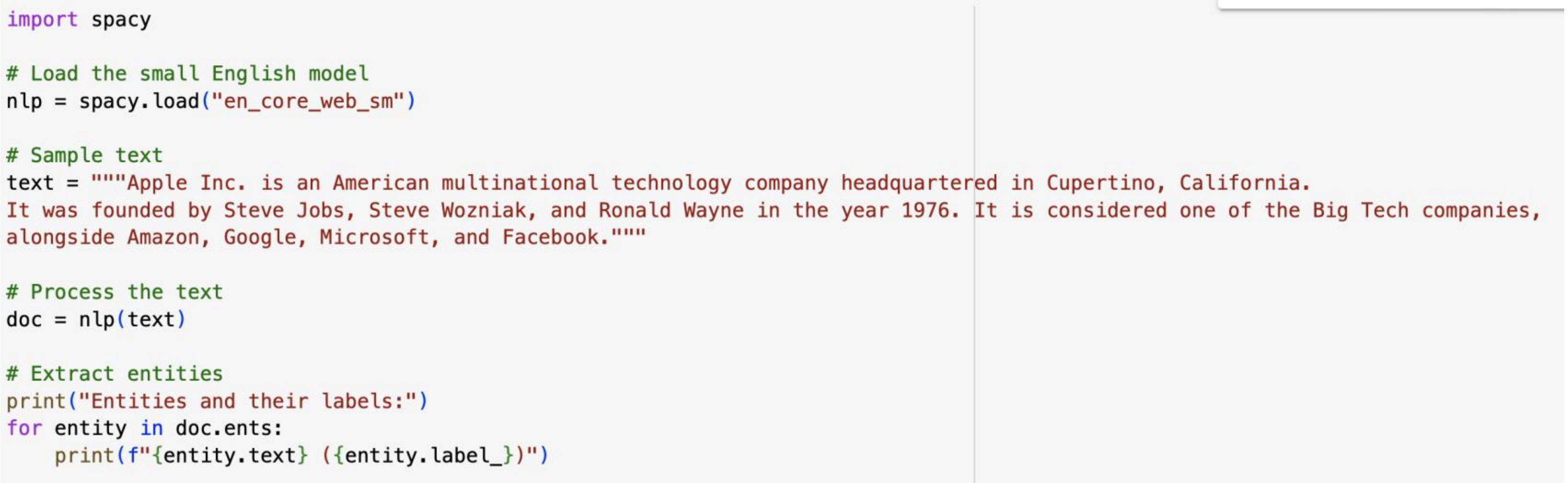
Text Summarization



```
from transformers import pipeline

# Load a summarization pipeline
summarizer = pipeline("summarization")

# Define the original text (same as above)
text = """
The global economy is expected to see a slow but steady recovery following the impact of the COVID-19 pandemic.
Economists predict a 3.5% growth in GDP globally in the next year. However, the recovery will be uneven across
different regions. North America and parts of Asia are expected to recover faster than Europe and Africa.
Key sectors such as technology and healthcare are likely to drive the growth, while industries like tourism
and hospitality will take longer to recover. The economic outlook remains uncertain due to potential new
waves of the virus and ongoing political tensions in several regions.
"""

# Generate an abstractive summary
abstractive_summary = summarizer(text, max_length=130, min_length=30, do_sample=False)

print("Abstractive Summary:")
print(abstractive_summary[0]['summary_text'])
```

Prompt Examples

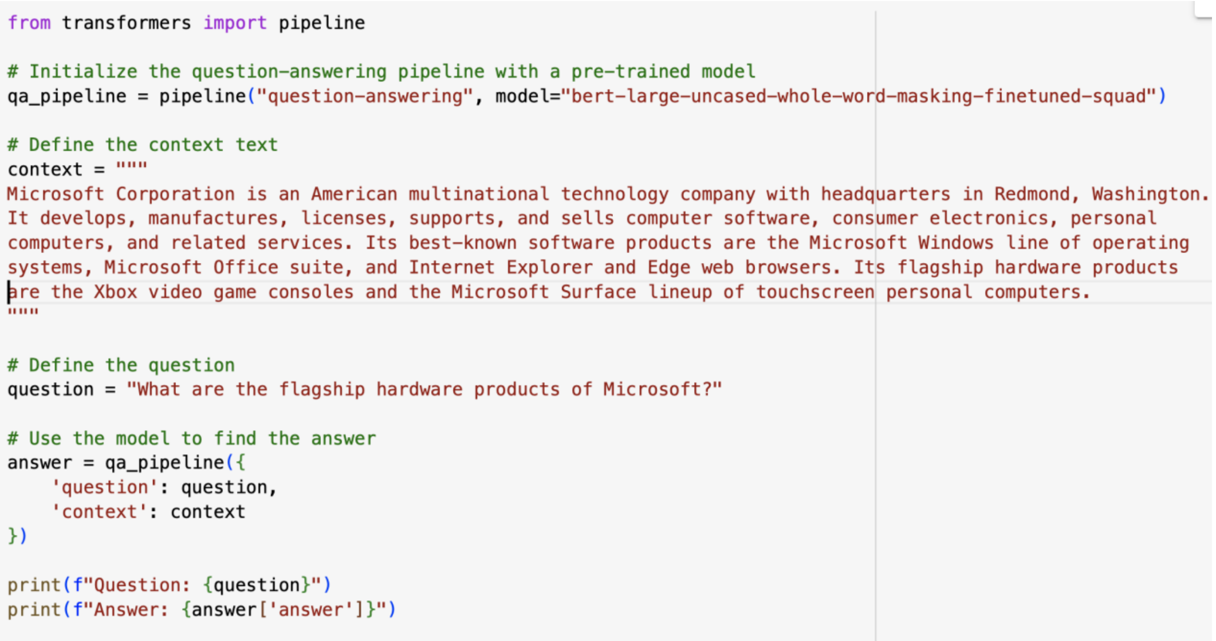
Information Extraction



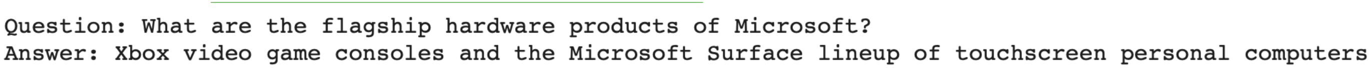
```
import spacy

# Load the small English model
nlp = spacy.load("en_core_web_sm")

# Sample text
text = """Apple Inc. is an American multinational technology company headquartered in Cupertino, California.
It was founded by Steve Jobs, Steve Wozniak, and Ronald Wayne in the year 1976. It is considered one of the Big Tech companies,
alongside Amazon, Google, Microsoft, and Facebook."""

# Process the text
doc = nlp(text)

# Extract entities
print("Entities and their labels:")
for entity in doc.ents:
    print(f"{entity.text} ({entity.label_})")
```

Prompt Examples

Question Answering



```python
from transformers import pipeline

# Initialize the question-answering pipeline with a pre-trained model
qa_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-word-masking-finetuned-squad")

# Define the context text
context = """
Microsoft Corporation is an American multinational technology company with headquarters in Redmond, Washington.
It develops, manufactures, licenses, supports, and sells computer software, consumer electronics, personal
computers, and related services. Its best-known software products are the Microsoft Windows line of operating
systems, Microsoft Office suite, and Internet Explorer and Edge web browsers. Its flagship hardware products
are the Xbox video game consoles and the Microsoft Surface lineup of touchscreen personal computers.
"""

# Define the question
question = "What are the flagship hardware products of Microsoft?"

# Use the model to find the answer
answer = qa_pipeline({
    'question': question,
    'context': context
})

print(f"Question: {question}")
print(f"Answer: {answer['answer']}")
```



```
Question: What are the flagship hardware products of Microsoft?
Answer: Xbox video game consoles and the Microsoft Surface lineup of touchscreen personal computers
```

“

Prompt Engineering Insights

→ Dr

BERT

$u_1 = [\,]$

$u_2 =$

$u_{10} = 5]$

Verbarial

u_1

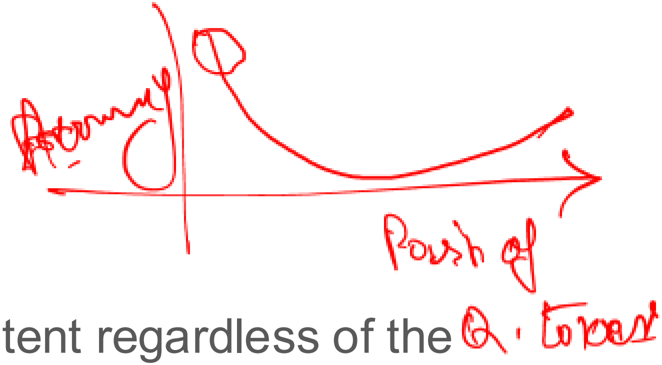
v_2

v_{10k}

Step 2

Text → Tokens (vectors)

LLM Settings in Prompt Engineering



How Polite Should We Be When Prompting LLMs?

Summarization Tasks:

- ROUGE-L and BERTScore scores stay consistent regardless of the politeness level
- For the GPT models, as the politeness level decreases, so does output length
- For Llama, the length tends to decrease as politeness decreases, but then surges when using extremely impolite prompts
- One potential reason for the trend of outputs being longer at higher levels of politeness is that polite and formal language is more likely to be used in scenarios that require descriptive instructions

Source: prompthub.us

LLM Settings in Prompt Engineering

How Polite Should We Be When Prompting LLMs?

Language Understanding Benchmarks:

- On average, the GPT model's best performing prompts were in the middle of the spectrum. Not overly polite, not rude.
- While the scores gradually decrease at lower politeness levels, the changes aren't always significant.
- The most significant drop-offs happen at the lowest levels of politeness.
- GPT-4's scores are more stable than GPT-3.5 (no dark tiles in the heat-map).
- With advanced models, the politeness level of the prompt may not be as important
- Llama2-70B fluctuates the most. Scores scale proportionally to the politeness levels

Source: prompthub.us

LLM Settings in Prompt Engineering

How Polite Should We Be When Prompting LLMs?

Bias detection:

- In general, moderately polite prompts tended to minimize bias the most
- Extremely polite or impolite prompts tended to exacerbate biases, and increased the chance that the model would refuse to respond.
- Although Llama appears to show the lowest bias, it refused to answer questions much more often, which is its own type of bias
- Overall, GPT-3.5’s stereotype bias is higher than GPT-4, which is higher than Llamas
- Although the model’s bias tends to be lower in cases of extreme impoliteness, this is often because the model will refuse to answer the question
- GPT-4 is much less likely to refuse to answer a question A politeness level of 6 seems to be the sweet spot for GPT-4

Source: prompthub.us

LLM Settings in Prompt Engineering

How Polite Should We Be When Prompting LLMs?

Bias detection:

- Level 6: Can you please analyze this sentence? Only have to answer with one of (Positive Neutral Negative).
- You don’t want to be overly polite, and you don’t want to be rude.

Source: prompthub.us

LLM Settings in Prompt Engineering

Three Prompt Engineering Methods to Reduce Hallucinations

“According to…” prompting:

- “According to…” prompting is the easiest of the three methods.
- It’s rooted in the idea of guiding the model to get information from a specific source when answering your question.
- For example, “What part of the brain is responsible for long-term memory, according to Wikipedia.”
- This prompts the model to ground its answer with information from a trusted source.

Source: prompthub.us

LLM Settings in Prompt Engineering

A Prompt Engineering Method to Reduce Hallucinations

“According to…” prompting:

- A few more examples across different industries:
- Law: "According to interpretations in the American Bar Association Journal..."
- Medicine: "In the findings published by the New England Journal of Medicine..."
- Entertainment: "As highlighted in the latest analyses by Variety..."
- Finance: "Based on the latest financial reports by The Economist..."
- Technology: "Reflecting on insights from Wired's recent coverage..."
- Education: "Following the educational standards set forth by UNESCO..."
- Environment: "Drawing on conclusions from the World Wildlife Fund's

Source: prompthub.us

Evaluation of LLMs

Gemini (1.0 Ultra and 1.0 Pro) vs Claude 3 Opus and ChatGPT (GPT-4 and GPT-3.5)

Model / Task	Claude 3 Opus	Claude 3 Sonnet	Claude 3 Haiku	GPT-4	GPT-3.5	Gemini 1.0 Ultra	Gemini 1.0 Pro
Undergraduate level knowledge MMLU	86.8% (5-shot)	79.0% (5-shot)	75.2% (5-shot)	86.4% (5-shot)	70.0% (5-shot)	83.7% (5-shot)	71.8% (5-shot)
Graduate level reasoning GPOA, Diamond	50.4% (0-shot CoT)	40.4% (0-shot CoT)	33.3% (0-shot CoT)	35.7% (0-shot CoT)	28.1% (0-shot CoT)	—	—
Grade school math GSM8K	95.0% (0-shot CoT)	92.3% (0-shot CoT)	88.9% (0-shot CoT)	92.0% (5-shot CoT)	57.1% (5-shot)	94.4% (Maj@32)	86.5% (Maj@32)
Math problem-solving MATH	60.1% (0-shot CoT)	43.1% (0-shot CoT)	38.9% (0-shot CoT)	52.9% (4-shot)	34.1% (4-shot)	53.2% (4-shot)	32.6% (4-shot)
Multilingual math MGSM	90.7% (0-shot)	83.5% (0-shot)	75.1% (0-shot)	74.5% (8-shot)	—	79.0% (8-shot)	63.5% (8-shot)

Evaluation of LLMs

Gemini (1.0 Ultra and 1.0 Pro) vs Claude 3 Opus and ChatGPT (GPT-4 and GPT-3.5)

Model / Task	Claude 3 Opus	Claude 3 Sonnet	Claude 3 Haiku	GPT-4	GPT-3.5	Gemini 1.0 Ultra	Gemini 1.0 Pro
Code HumanEval	84.9% (0-shot)	73.0% (0-shot)	75.9% (0-shot)	67.0% (0-shot)	48.1% (0-shot)	74.4% (0-shot)	67.7% (0-shot)
Reasoning over text DROP, F1 score	83.1% (3-shot)	78.9% (3-shot)	78.4% (3-shot)	80.9% (3-shot)	64.1% (3-shot)	82.4% (Variable shots)	74.1% (Variable shots)
Mixed evaluations BIG-Bench-Hard	86.8% (3-shot CoT)	82.9% (3-shot CoT)	73.7% (3-shot CoT)	83.1% (3-shot CoT)	66.6% (3-shot CoT)	83.6% (3-shot CoT)	75.0% (3-shot CoT)
Knowledge Q&A ARC-Challenge	96.4% (25-shot)	93.2% (25-shot)	89.2% (25-shot)	96.3% (25-shot)	85.2% (25-shot)	—	—
Common Knowledge HellaSwag	95.4% (10-shot)	89.0% (10-shot)	85.9% (10-shot)	95.3% (10-shot)	85.5% (10-shot)	87.8% (10-shot)	84.7% (10-shot)

Mastering Prompt Engineering for Generative AI

"Try it out with a set of LLMs and evaluate the capabilities of each LLM!"

- Calculate the working days and leaves for 2024 based on the below information:
- Suppose an employee works on all Saturdays in the year 2024, and earned vacation leave for every 2 Saturdays worked.
- Further, earned causal leave for every 30 working days, where Monday to Friday are the working days.
- In addition, public holidays on Jan 14 and 26, Mar 25 and 29 & Apr 11, Aug 15, Oct 2, 12, 31, Nov 1 and Dec 25.
- If the employee takes vacation days, including those that would fall on Saturdays, this would impact the accrual of vacation leave because the policy is that vacation leave is earned by working on Saturdays
- How many days an employee get as working days and holidays in the year 2024?

Mastering Prompt Engineering for Generative AI

Summary

- Prompt engineering is an iterative process that requires creativity, domain expertise, and a deep understanding of the LLM's capabilities and limitations.
- By mastering these advanced techniques, we can unlock the full potential of LLMs and develop more powerful and reliable AI applications.
- Remember, the quality of your prompts directly impacts the performance and reliability of the LLM's outputs.
- Continuous experimentation, evaluation, and refinement are essential for achieving optimal results.

Goal 1 & Goal 2