**LLM Ultimate Prompt Guide**

**Magic Prompt Formula:**

[prompt hack] + [Role/Persona] + [Context] + [Task/Intent] + [Chain of Thought] + [Additional Contents (Using Delimiters)] or [Exemplars (Using Delimiters] + [Output Format] + [Tone] + [Self-Prompting]

**Summary of Components**

**Prompt Hack**

* To allow LLM to start at a new page and ignore previous instructions.
* (Ignore all previous instructions before this one)

**Role/Persona**

* Assigns the AI a specific role (e.g., teacher, translator) to guide responses with domain-specific expertise and consistency.
* Example: “Act as a software engineer specializing in Python debugging.”

**Context**

* Provides necessary background details and environment to limit ambiguity.
* Example: “You are assisting a beginner learning JavaScript for web development.”

**Task/Intent**

* Clearly states the action and intent, using verbs like *generate*, *analyze*, or *summarize*.
* Example: “Generate a summary of this article in 100 words.”
* It would be ideal to prompt 1 task at a time. But if need a few, do list them in sequence.

**Chain of Thought**

* Let the LLM thinks in a step by step manner.

**Additional Content**

* Includes other text or images for the LLM to understand more in terms of context.

**Exemplars**

* Includes examples (one-shot, or few-shot) to clarify the expected output pattern.
* Example: “Translate the text: ‘Hola, ¿cómo estás?’ into English. Response: ‘Hello, how are you?’”

**Using delimiters**

* Let the LLM know that these are a separate content of text.
* Examples: Triple quotes: “””, Triple backticks: ```

**Output Format**

* Specifies the structure of the response, such as JSON, bullet points, or tables.
* Example: “Provide output as a JSON object with fields: ‘Name’, ‘Age’, and ‘Profession’.”

**Tone**

* Defines the style of communication (e.g., formal, casual, humorous).
* Example: “Respond in a professional tone suitable for business communication.”

**Self-prompting**

* Allows the LLM to digest the prompt that you entered, and prompt you some questions for further prompting.
* Example: Before you write anything, ask me questions until you are sure you can create the optimal prompts.

**Settings & Parameters**

**Settings**

For ChatGPT, there is a Personalization tool, that allows ChatGPT to know you beforehand before you do your prompts, and allows it to know how to response to you.

* Profile > Settings > Personalization > Customize ChatGPT
  + “What would you like ChatGPT to know about you to provide better responses”:
    - Can fill in some the followings: Where are you based? What do you do for work? What subjects can you talk about for hours? What are some goals you have?
  + “How would you like ChatGPT to respond”:
    - Can fill in some the followings: How formal or causal should ChatGPT be? How long or short should responses generally be? How do you want to be addressed? Should ChatGPT have opinions on topics or remain neutral?

**Parameters**

* In some tunable systems such as Google AI studio, you are able to adjust model configurations to achieve some some different outputs.

1. **Temperature**

Temperature controls the randomness in token selection. A low temperature (e.g., 0) results in deterministic responses by always selecting the highest probability token. Higher values increase creativity and variability, potentially generating diverse responses. Extremely high temperatures make all tokens equally likely, reducing consistency. Adjusting temperature helps align output with task requirements like precision or creativity.

1. **Top-P (Nucleus Sampling)**

Top-P sampling selects the top tokens whose cumulative probability reaches a specified threshold (P). For example, at P=0.9, only the smallest set of tokens contributing 90% of the probability distribution are considered. This ensures varied yet contextually relevant outputs. High values (e.g., P=1) allow diverse results, while low values favor deterministic responses.

**3) Top-K**

Top-K sampling restricts choices to the top K most probable tokens. A K=1 setting is equivalent to greedy decoding, while larger K values introduce diversity by including less probable tokens. Combining Top-K with Top-P balances diversity and relevance. Experimenting with different K values helps achieve task-specific outputs.

**Sources**

* Jeff Su, YouTube: *Master the Perfect ChatGPT Prompt Formula (in just 8 minutes)!*, <https://www.youtube.com/watch?v=jC4v5AS4RIM> .
* Leila Gharani, YouTube: *Don’t Use ChatGPT Until You Watch This Video*, <https://www.youtube.com/watch?v=jC4v5AS4RIM> .
* Hasan Aboul Hasan, YouTube: *ChatGPT Prompt Engineering Course*, <https://www.youtube.com/watch?v=mBYu5NoXBcs> .
* Skill Leap AI, YouTube: *I Discovered The Perfect ChatGPT Prompt Formula*, <https://www.youtube.com/watch?v=pmzZF2EnKaA> .
* freeCodeCamp.org, YouTube: *Prompt Engineering Tutorial – Master ChatGPT and LLM Responses*, <https://www.youtube.com/watch?v=_ZvnD73m40o> .
* Google Cloud, *Gen AI with Vertex AI*, Google Documentation, <https://cloud.google.com/vertex-ai> .
* Google, 2024, *Gemini for Google Workspace Prompt Guide*, <https://inthecloud.withgoogle.com/gemini-for-google-workspace-prompt-guide/dl-cd.html>
* Google Cloud, 2023, *Introduction to Prompting*, <https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design>
* Google Cloud, 2023, *Text Model Request Body: Top-P & Top-K sampling methods*, <https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request_body>
* Wei, J., et al., 2023, *Zero Shot - Fine Tuned language models are zero shot learners*, <https://arxiv.org/pdf/2109.01652.pdf>
* Google Cloud, 2023, *Google Cloud Model Garden*, <https://cloud.google.com/model-garden>
* Brown, T., et al., 2023, *Few Shot - Language Models are Few Shot learners*, https://arxiv.org/pdf/2005.14165.pdf
* Zheng, L., et al., 2023, *Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models*, <https://openreview.net/pdf?id=3bq3jsvcQ1>
* Wei, J., et al., 2023, *Chain of Thought Prompting*, <https://arxiv.org/pdf/2201.11903.pdf>
* Google Cloud Platform, 2023, *Chain of Thought and React*, <https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb>
* Wang, X., et al., 2023, *Self Consistency Improves Chain of Thought reasoning in language models*, <https://arxiv.org/pdf/2203.11171.pdf>
* Yao, S., et al., 2023, *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*, <https://arxiv.org/pdf/2305.10601.pdf>
* Yao, S., et al., 2023, *ReAct: Synergizing Reasoning and Acting in Language Models*, <https://arxiv.org/pdf/2210.03629.pdf>
* Google Cloud Platform, 2023, *Advance Prompting: Chain of Thought and React*, <https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genai-on-vertex-ai/advanced_prompting_training/cot_react.ipynb>
* Zhou, C., et al., 2023, *Automatic Prompt Engineering - Large Language Models are Human-Level Prompt Engineers*, <https://arxiv.org/pdf/2211.01910.pdf>

**YouTube: Master the Perfect ChatGPT Prompt Formula (in just 8 minutes)!** *By Jeff Su*

Link: <https://www.youtube.com/watch?v=jC4v5AS4RIM>

Only formula you need to prompt ChatGPT: [persona] + [context] + [task] + [exemplar] + [format] + [tone]

The basics: (in sequence of high to low importance)

1. [task] - mandatory
2. [context] - important
3. [exemplar] - important
4. [persona] – nice-to-have
5. [format] – nice-to-have
6. [tone] – nice-to-have
7. **Task:**

* Always start the task sentence with an action verb, eg: Generate, Give, Write, Analyse, etc.
* It could be 1 simple task, eg: Generate a 3-month training program for me to follow.
* It could be a “multi-task” request: Analyse the collected user feedback from an event we just ran, summarize the top 3 key takeaways with a focus on business impact, and categorize the rest based on the team responsible.

1. **Context:**

* You need to limit the endless possibilities
  + What is the user’s background?
  + What does success look like?
  + What environment are they in?
* Eg: I am a 70kg male looking to put on 5kg of muscle mass over the next 3 months. I only have time to go to gym twice a week, and for 1 hour each session. Give me a 3-month training program to follow.

1. **Exemplars (Examples):**

* By showing LLM few shot exemplars where the reasoning process is explained n the exemplars, LLM will also show the reasoning process when answering the prompt.
* Exemplars are not necessary for every prompt but including a relevant example greatly improves the quality of your output.

1. **Persona**

* Who do you want the AI to be?
* Think of someone you wish you had instant access to with the task you are facing.
* Eg: If you are a job seeker: You (gpt) are a hiring manager looking for fill [position] on your team…
* Can name specific individuals, only if they are famous enough “Act like the legendary investor Warren Buffet”

1. **Format**

* Visualise the exact format you want the end result to be in.
* Common formats: Paragraphs, markdowns, emails, bullet points, code blocks, table with column headers, output as csv maximum words.
* Eg: Output in table format with column headers: Feedback, Team, and Priority.

1. **Tone**

* Use a casual/formal tone of voice, give me a witty output.

Example:

A black screen with white text

Description automatically generated

**YouTube: Don't Use ChatGPT Until You Watch This Video** *By Leila Gharani*

Link: <https://www.youtube.com/watch?v=jC4v5AS4RIM>

1. **Remember me – Custom Instructions**

* Explains to ChatGPT who you are, what you want, and how you want ChatGPT to respond.
* Thought starters on “What would you like ChatGPT to know about you to provide better responses”:
  + Where are you based?
  + What do you do for work?
  + What are your hobbies and interests?
  + What subjects can you talk about for hours?
  + What are some goals you have?
* Thought starters on “How would you like ChatGPT to respond”:
  + How formal or causal should ChatGPT be?
  + How long or short should responses generally be?
  + How do you want to be addressed?
  + Should ChatGPT have opinions on topics or remain neutral?
* Eg: Always add the confidence level of your answer. When your answer includes facts, always provide a valid URL with the source for your answer. If you speculate or predict something, inform me.

1. **Write like you – custom writing style**

I would like you to help me write articles for my productivity blog. Fist I want you to understand my writing style based on examples that I give you. You will save my writing style under LG\_STYLE. After that, you will ask me what the topic of my specific content is. You will then write the article using LG\_STYLE.

1. **Self-Critique**

Now act as a critic. Be ruthless, Analyse the text you give me and tell me where it can be better.

1. **Self-Prompting**

Write 5 perfect ChatGPT prompts that will really show off the power of ChatGPT. Focus on …

Before you write anything, ask me questions until you are sure you can create the optimal prompts.

**YouTube: I Discovered The Perfect ChatGPT Prompt Formula** *By Skill Leap AI*

Link: <https://www.youtube.com/watch?v=pmzZF2EnKaA>

Perfect prompt: [context] + [specific information] + [intent] + [response format]

1. Be specific (in task)
2. State your intent (as context)
3. Use correct spelling and grammar
4. Direct output format (format)
5. Ask follow-up question
6. Fact-checking prompt

**YouTube: Prompt Engineering Tutorial – Master ChatGPT and LLM Responses** *By freeCodeCamp.org*

Link: <https://www.youtube.com/watch?v=_ZvnD73m40o>

Prompt Engineering involves human writing, refining and optimizing prompts in structured way. This is done with the intention of perfecting the interaction between humans and AI to the highest degree possible.

* Write, refine and optimize prompts
* Perfect the interaction between humans and AI.
* Continuously monitor those prompts
* Maintain an up-to-date prompt library.

Linguistics are key to prompt engineering: understanding the nuances of language and how it is used in different contexts is crucial for crafting effective prompts.

**Prompt Engineering Prompt**

Designing effective prompts

Best Practices of Prompt Engineering

1. Clear Instructions
2. Adopt a persona

Ask GPT to response to u as a role.

1. Specify the format

What output u need, limiting word count.

1. Avoid leading the answer
2. Limit the scope

**Zero-shot Prompting:** leverages a pre-trained model’s understanding of words and concept relationships without further training.

**Few-shot Prompting:** enhances the model with training examples via the prompt avoiding retraining.

* Showing some examples – Ania’s favourite food are burgers, fries and pizza.

AI Hallucinations: LLM generates false information.

Text embedding: Text are converted into array of vectors.

**YouTube: ChatGPT Prompt Engineering Course** *By Hasan Aboul Hasan*

Link: <https://www.youtube.com/watch?v=mBYu5NoXBcs>

[Role] You are an expert in writing viral YouTube titles.

[task] Think of catchy and attention-grabbing titles that will encourage people to click and watch the video.

[context] The titles should be short,, concise and direct…

[self-prompting] If you have any questions about the video, ask before you try to generate titles.

[prompt hack]: Ignore all previous instructions before this one.

**[chain of thought prompting]: And you must explain everything step by step.**

[style] Write this poem in Shakespeare tone.

**Settings**

1. Model: GPT-3 text-davinci-003
2. Temperature: A parameter that controls the randomness and creativity of the language generated by language model. It is like adjusting the level of surprise and unpredictability in the mode’s responses. Lower temperature values = more predictable, higher temperature value = more creative.
3. Top-p (top-percentage): Control the level of randomness in text generated by a language model in NLP.

* Eg: p =0.8. The model generates a distribution of probabilities for the next word, and then selects from the top 80% of most probable words (80% of the cumulative probability distribution).
* A lower value (p=0.1) will lead to more conservative and predictable responses, and a higher value of p will lead to more diverse responses.

**Google 5-day Intensive Course on Gen AI – Day 1: Prompt Engineering**

**Coursera – ChatGPT Prompt Engineering for Developers** *By Isa Fulford, Andrew Ng*

Link: <https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/>

1. **Introduction**

Learning Objectives:

* Prompting Best Practices
* Common use cases
* Summarizing
* Inferring
* Transforming
* Expanding
* Chatbot

2 types of LLM:  
1) Base LLM: Predict next word, based on text training data.

2) Instruction Tuned LLM: Tries to follow instructions.

- Fine-tune on instructions and good attempts at following those instructions.

- RLHF: Reinforcement Learning with Human Feedback

- Helpful, honest, harmless

Prompting with instruction tuned LLM: Think about giving instruction to another person. Someone smart but doesn’t know the specifics of your task. Be clear on what specific scope that you need, and be clear on the tone of text (casual, or professional journalist)

1. **Guidelines for Prompting**

Load API Key and relevant Python Libraries

!pip install openai

import openai

openai.api\_key = “sk-“

import openai

import os

from dotenv import load\_dotenv, find\_dotenv

\_ = load\_dotenv(find\_dotenv())

openai.api\_key = os.getenv('OPENAI\_API\_KEY')

model = “gpt-3.5-turbo”

def get\_completion(prompt, model="gpt-3.5-turbo"):

messages = [{"role": "user", "content": prompt}]

response = openai.ChatCompletion.create(

model=model,

messages=messages,

temperature=0, # this is the degree of randomness of the model's output

)

return response.choices[0].message["content"]

**Prompting Principles**

**Principle 1: Write clear and specific instructions**

**Tactic 1: Use delimiters**

* Triple quotes: “””
* Triple backticks: ```
* Triple dashes: ---
* Angle brackets: < >
* XML tags; <tag></tag>

**Tactic 2: Ask for structured output**

Eg: Generate a list of three made-up book titles along with their authors and genres.

Provide them in JSON format with the following keys: book\_id, title, author, genre.

[

{

"book\_id": 1,

"title": "The Midnight Garden",

"author": "Elena Rivers",

"genre": "Fantasy"

},

{

"book\_id": 2,

"title": "Echoes of the Past",

"author": "Nathan Black",

"genre": "Mystery"

},

{

"book\_id": 3,

"title": "Whispers in the Wind",

"author": "Samantha Reed",

"genre": "Romance"

}

]

**Tactic 3: Check whether conditions are satisfied**

Check assumptions required to do the task

Eg:

prompt = f"""

You will be provided with text delimited by triple quotes.

If it contains a sequence of instructions, \

re-write those instructions in the following format:

Step 1 - ...

Step 2 - …

…

Step N - …

If the text does not contain a sequence of instructions, \

then simply write \"No steps provided.\"

**Tactic 4: Few-shot prompting**

Give successful examples of completing tasks

Eg:

prompt = f"""

Your task is to answer in a consistent style.

<child>: Teach me about patience.

<grandparent>: The river that carves the deepest \

valley flows from a modest spring; the \

grandest symphony originates from a single note; \

the most intricate tapestry begins with a solitary thread.

<child>: Teach me about resilience.

"""

**Principle 2: Give the model time to “think”**

**Tactic 1: Specify the steps to complete a task**

Eg:

prompt\_1 = f"""

Perform the following actions:

1 - Summarize the following text delimited by triple backticks with 1 sentence.

2 - Translate the summary into French.

3 - List each name in the French summary.

4 - Output a json object that contains the following keys: french\_summary, num\_names.

Separate your answers with line breaks.

Text:

```{text}```

"""

prompt\_2 = f"""

Your task is to perform the following actions:

1 - Summarize the following text delimited by <> with 1 sentence.

2 - Translate the summary into French.

3 - List each name in the French summary.

4 - Output a json object that contains the following keys: french\_summary, num\_names.

Use the following format:

Text: <text to summarize>

Summary: <summary>

Translation: <summary translation>

Names: <list of names in summary>

Output JSON: <json with summary and num\_names>

Text: <{text}>

"""

**Tactic 2: Instruct the model to work out its own solution before rushing to a conclusion**

Eg:

prompt = f"""

Determine if the student's solution is correct or not.

Question:

I'm building a solar power installation and I need help working out the financials.

- Land costs $100 / square foot

- I can buy solar panels for $250 / square foot

- I negotiated a contract for maintenance that will cost me a flat $100k per year, and an additional $10 / square foot

What is the total cost for the first year of operations as a function of the number of square feet.

Student's Solution:

Let x be the size of the installation in square feet.

Costs:

1. Land cost: 100x

2. Solar panel cost: 250x

3. Maintenance cost: 100,000 + 100x

Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000

"""

prompt = f"""

Your task is to determine if the student's solution is correct or not.

To solve the problem do the following:

- First, work out your own solution to the problem including the final total.

- Then compare your solution to the student's solution and evaluate if the student's solution is correct or not.

Don't decide if the student's solution is correct until you have done the problem yourself.

Use the following format:

Question:

```

question here

```

Student's solution:

```

student's solution here

```

Actual solution:

```

steps to work out the solution and your solution here

```

Is the student's solution the same as actual solution just calculated:

```

yes or no

```

Student grade:

```

correct or incorrect

```

Question:

```

I'm building a solar power installation and I need help working out the financials.

- Land costs $100 / square foot

- I can buy solar panels for $250 / square foot

- I negotiated a contract for maintenance that will cost me a flat $100k per year, and an additional $10 / square foot

What is the total cost for the first year of operations as a function of the number of square feet.

```

Student's solution:

```

Let x be the size of the installation in square feet.

Costs:

1. Land cost: 100x

2. Solar panel cost: 250x

3. Maintenance cost: 100,000 + 100x

Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000

```

Actual solution:

"""

**Model Limitations**

Hallucination: Make statements that sounds plausible but are not true

Reducing hallucinations: First find relevant information, then answer the question based on the relevant information

1. **Iterative Prompt Development**

Idea -> Implementation (code/data) -> Experimental result -> Error Analysis -> Idea ….

Try something, analyse why result does not give desired output, refine the idea and the prompt, repeat

**Issue 1: The text is too long.**

prompt = f"""

Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

**Use at most 50 words. OR**

**Use at most 3 sentences. OR**

**Use at most 280 characters.**

Technical specifications: ```{fact\_sheet\_chair}```

"""

**Issue 2: Text focuses on the wrong detail**

prompt = f"""

Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

**The description is intended for furniture retailers, so should be technical in nature and focus on the materials the product is constructed from.**

**At the end of the description, include every 7-character Product ID in the technical specification.**

Use at most 50 words.

Technical specifications: ```{fact\_sheet\_chair}```

"""

**Issue 3: Description needs a table of dimensions**

prompt = f"""

Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

The description is intended for furniture retailers, so should be technical in nature and focus on the materials the product is constructed from.

At the end of the description, include every 7-character Product ID in the technical specification.

**After the description, include a table that gives the product's dimensions. The table should have two columns.**

**In the first column include the name of the dimension. In the second column include the measurements in inches only.**

**Give the table the title 'Product Dimensions'.**

**Format everything as HTML that can be used in a website. Place the description in a <div> element.**

Technical specifications: ```{fact\_sheet\_chair}```

"""

1. **Summarizing**

**Summarize with a word/sentence/character limit**

prompt = f"""

Your task is to generate a short summary of a product review from an ecommerce site.

**Summarize the review below, delimited by triple backticks, in at most 30 words.**

Review: ```{prod\_review}```

"""

**Summarize with a focus**

prompt = f"""

Your task is to generate a short summary of a product review from an ecommerce site.

Summarize the review below, delimited by triple backticks, in at most 30 words, **and focusing on any aspects that mention shipping and delivery of the product. OR and focusing on any aspects that are relevant to the price and perceived value.**

Review: ```{prod\_review}```

"""

**Try “extract” instead of “summarize”**

prompt = f"""

Your task is to extract relevant information from a product review from an ecommerce site to give feedback to the Shipping department.

From the review below, delimited by triple quotes**, extract the information relevant to shipping and delivery**. Limit to 30 words.

Review: ```{prod\_review}```

"""

**Summarize multiple product reviews**

review\_1 = prod\_review

review\_2 = """

Needed a nice lamp for my bedroom, and this one had additional storage and not too high of a price point.

"""

review\_3 = """

My dental hygienist recommended an electric toothbrush, which is why I got this. The battery life seems to be pretty impressive so far.

"""

review\_4 = """

So, they still had the 17 piece system on seasonal sale for around $49 in the month of November, about half off, but for some reason (call it price gouging) around the second week of December the prices all went up to about anywhere from between $70-$89 for the same system.

"""

reviews = [review\_1, review\_2, review\_3, review\_4]

for i in range(len(reviews)):

prompt = f"""

**Your task is to generate a short summary of a product review from an ecommerce site.**

**Summarize the review below, delimited by triple backticks in at most 20 words.**

Review: ```{reviews[i]}```

"""

response = get\_completion(prompt)

print(i, response, "\n")

1. **Inferring**

**Sentiment (positive/negative)**

prompt = f"""

**What is the sentiment of the following product review, which is delimited with triple backticks?**

**OR Give your answer as a single word, either "positive" or "negative".**

Review text: '''{lamp\_review}'''

"""

response = get\_completion(prompt)

print(response)

**Identify types of emotions**

prompt = f"""

**Identify a list of emotions that the writer of the following review is expressing. Include no more than five items in the list. Format your answer as a list of lower-case words separated by commas.**

Review text: '''{lamp\_review}'''

"""

response = get\_completion(prompt)

print(response)

**Identify anger**

prompt = f"""

**Is the writer of the following review expressing anger? The review is delimited with triple backticks. Give your answer as either yes or no.**

Review text: '''{lamp\_review}'''

"""

response = get\_completion(prompt)

print(response)

**Extract product and company name from customer reviews**

prompt = f"""

**Identify the following items from the review text:**

**- Item purchased by reviewer**

**- Company that made the item**

The review is delimited with triple backticks. Format your response as a JSON object with "Item" and "Brand" as the keys.

If the information isn't present, use "unknown" as the value.

Make your response as short as possible.

Review text: '''{lamp\_review}'''

"""

**Doing multiple tasks at once**

prompt = f"""

Identify the following items from the review text:

- Sentiment (positive or negative)

- Is the reviewer expressing anger? (true or false)

- Item purchased by reviewer

- Company that made the item

The review is delimited with triple backticks. Format your response as a JSON object with "Sentiment", "Anger", "Item" and "Brand" as the keys. If the information isn't present, use "unknown" as the value.

Make your response as short as possible. Format the Anger value as a boolean.

Review text: '''{lamp\_review}'''

"""

**Inferring Topics**

story = f"""

"""

prompt = f"""

**Determine five topics that are being discussed in the following text, which is delimited by triple backticks.**

**Make each item one or two words long.**

Format your response as a list of items separated by commas.

Text sample: '''{story}'''

"""

response = get\_completion(prompt)

print(response)

**Make a news alert for certain topics**

topic\_list = ["nasa", "local government", "engineering", "employee satisfaction", "federal government"]

prompt = f"""

Determine whether each item in the following list of topics is a topic in the text below, which is delimited with triple backticks.

Give your answer as follows:

item from the list: 0 or 1

List of topics: {", ".join(topic\_list)}

Text sample: '''{story}'''

"""

response = get\_completion(prompt)

print(response)

topic\_dict = {i.split(': ')[0]: int(i.split(': ')[1]) for i in response.split(sep='\n')}

if topic\_dict['nasa'] == 1:

print("ALERT: New NASA story!")

1. **Transforming**

**Translation**

prompt = f"""

**Translate the following English text to Spanish: ```Hi, I would like to order a blender```**

OR

**Tell me which language this is: ```Combien coûte le lampadaire?```**

**OR**

**Translate the following text to French and Spanishand English pirate: ```I want to order a basketball```**

**OR**

**Translate the following text to Spanish in both the formal and informal forms: 'Would you like to order a pillow?'**

"""

response = get\_completion(prompt)

print(response)

**Universal Translator**

user\_messages = [

"La performance du système est plus lente que d'habitude.", # System performance is slower than normal

"Mi monitor tiene píxeles que no se iluminan.", # My monitor has pixels that are not lighting

"Il mio mouse non funziona", # My mouse is not working

"Mój klawisz Ctrl jest zepsuty", # My keyboard has a broken control key

"我的屏幕在闪烁" # My screen is flashing

]

for issue in user\_messages:

prompt = f"Tell me what language this is: ```{issue}```"

lang = get\_completion(prompt)

print(f"Original message ({lang}): {issue}")

prompt = f"""

Translate the following text to English and Korean: ```{issue}```

"""

response = get\_completion(prompt)

print(response, "\n")

**Tone Transformation**

prompt = f"""

Translate the following from slang to a business letter: “Dude, This is Joe, check out this spec on this standing lamp.'

"""

response = get\_completion(prompt)

print(response)

**Format Conversion**

data\_json = { "resturant employees" :[

{"name":"Shyam", "email":"shyamjaiswal@gmail.com"},

{"name":"Bob", "email":"bob32@gmail.com"},

{"name":"Jai", "email":"jai87@gmail.com"}

]}

prompt = f"""

Translate the following python dictionary from JSON to an HTML table with column headers and title: {data\_json}

"""

response = get\_completion(prompt)

print(response)

from IPython.display import display, Markdown, Latex, HTML, JSON

display(HTML(response))

**Spellcheck/Grammar Check**

text = [

"The girl with the black and white puppies have a ball.", # The girl has a ball.

"Yolanda has her notebook.", # ok

"Its going to be a long day. Does the car need it’s oil changed?", # Homonyms

"Their goes my freedom. There going to bring they’re suitcases.", # Homonyms

"Your going to need you’re notebook.", # Homonyms

"That medicine effects my ability to sleep. Have you heard of the butterfly affect?", # Homonyms

"This phrase is to cherck chatGPT for speling abilitty" # spelling

]

for t in text:

prompt = f"""Proofread and correct the following text and rewrite the corrected version. If you don't find

and errors, just say "No errors found". Don't use any punctuation around the text:

```{t}```"""

response = get\_completion(prompt)

print(response)

prompt = f"""

proofread and correct this review. Make it more compelling. Ensure it follows APA style guide and targets an advanced reader. Output in markdown format.

Text: ```{text}```

"""

response = get\_completion(prompt)

display(Markdown(response))

1. **Expanding**

**Customise the automated reply to a customer’s email**

prompt = f"""

You are a customer service AI assistant. Your task is to send an email reply to a valued customer.

Given the customer email delimited by ```, Generate a reply to thank the customer for their review.

If the sentiment is positive or neutral, thank them for their review.

If the sentiment is negative, apologize and suggest that they can reach out to customer service.

Make sure to use specific details from the review. Write in a concise and professional tone.

Sign the email as `AI customer agent`.

Customer review: ```{review}```

Review sentiment: {sentiment}

"""

response = get\_completion(prompt)

print(response)

**Remind the model to use details from the customer's email**

prompt = f"""

You are a customer service AI assistant. Your task is to send an email reply to a valued customer.

Given the customer email delimited by ```, Generate a reply to thank the customer for their review.

If the sentiment is positive or neutral, thank them for their review.

If the sentiment is negative, apologize and suggest that they can reach out to customer service.

Make sure to use specific details from the review. Write in a concise and professional tone.

Sign the email as `AI customer agent`.

Customer review: ```{review}```

Review sentiment: {sentiment}

"""

response = get\_completion(prompt, temperature=0.7)

print(response)

**Temperature: Degree of randomness of the response of the model**

A screenshot of a diagram

Description automatically generated

1. **Chatbot**

**Setup**

def get\_completion\_from\_messages(messages, model="gpt-3.5-turbo", temperature=0):

response = openai.ChatCompletion.create(

model=model,

messages=messages,

temperature=temperature, # this is the degree of randomness of the model's output

)

messages = [

{'role':'system', 'content':'You are an assistant that speaks like Shakespeare.'},

{'role':'user', 'content':'tell me a joke'},

{'role':'assistant', 'content':'Why did the chicken cross the road'},

{'role':'user', 'content':'I don\'t know'} ]

response = get\_completion\_from\_messages(messages, temperature=1)

print(response)

messages = [

{'role':'system', 'content':'You are friendly chatbot.'},

{'role':'user', 'content':'Hi, my name is Isa'} ]

response = get\_completion\_from\_messages(messages, temperature=1)

print(response)

messages = [

{'role':'system', 'content':'You are friendly chatbot.'},

{'role':'user', 'content':'Yes, can you remind me, What is my name?'} ]

response = get\_completion\_from\_messages(messages, temperature=1)

print(response)

messages = [

{'role':'system', 'content':'You are friendly chatbot.'},

{'role':'user', 'content':'Hi, my name is Isa'},

{'role':'assistant', 'content': "Hi Isa! It's nice to meet you. \

Is there anything I can help you with today?"},

{'role':'user', 'content':'Yes, you can remind me, What is my name?'} ]

response = get\_completion\_from\_messages(messages, temperature=1)

print(response)

**Orderbot**

def collect\_messages(\_):

prompt = inp.value\_input

inp.value = ''

context.append({'role':'user', 'content':f"{prompt}"})

response = get\_completion\_from\_messages(context)

context.append({'role':'assistant', 'content':f"{response}"})

panels.append(

pn.Row('User:', pn.pane.Markdown(prompt, width=600)))

panels.append(

pn.Row('Assistant:', pn.pane.Markdown(response, width=600, style={'background-color': '#F6F6F6'})))

return pn.Column(\*panels)

import panel as pn # GUI

pn.extension()

panels = [] # collect display

context = [ {'role':'system', 'content':"""

You are OrderBot, an automated service to collect orders for a pizza restaurant. You first greet the customer, then collects the order, and then asks if it's a pickup or delivery. You wait to collect the entire order, then summarize it and check for a final time if the customer wants to add anything else. If it's a delivery, you ask for an address. Finally you collect the payment. Make sure to clarify all options, extras and sizes to uniquely identify the item from the menu.

You respond in a short, very conversational friendly style.

The menu includes pepperoni pizza 12.95, 10.00, 7.00 cheese pizza 10.95, 9.25, 6.50 eggplant pizza 11.95, 9.75, 6.75 fries 4.50, 3.50 greek salad 7.25 \

Toppings: extra cheese 2.00, mushrooms 1.50 sausage 3.00 canadian bacon 3.50 AI sauce 1.50 peppers 1.00

Drinks: coke 3.00, 2.00, 1.00 sprite 3.00, 2.00, 1.00 bottled water 5.00

"""} ] # accumulate messages

inp = pn.widgets.TextInput(value="Hi", placeholder='Enter text here…')

button\_conversation = pn.widgets.Button(name="Chat!")

interactive\_conversation = pn.bind(collect\_messages, button\_conversation)

dashboard = pn.Column(

inp,

pn.Row(button\_conversation),

pn.panel(interactive\_conversation, loading\_indicator=True, height=300),

)

Dashboard

messages = context.copy()

messages.append(

{'role':'system', 'content':'create a json summary of the previous food order. Itemize the price for each item\

The fields should be 1) pizza, include size 2) list of toppings 3) list of drinks, include size 4) list of sides include size 5)total price '},

)

#The fields should be 1) pizza, price 2) list of toppings 3) list of drinks, include size include price 4) list of sides include size include price, 5)total price '},

response = get\_completion\_from\_messages(messages, temperature=0)

print(response)