# 1227

## Goal

### 1.1 Content

Master the practical skills of AI Agent/ agent

- Understand the principles and trends of AI Agent technology

- Learn the basic use of Alipay treasure box

- Practice advanced: Agent function design, Prompt engineering, workflow and iterative thinking

Make your own work and participate in the learning competition

- Learn to apply, complete their own creation

- Review summary, let the works and ideas go further

### 1.2 Result

The final learning results of the entire study period, combined with their own passion and interest, develop and complete the work, submit your work, submit the work should include:

Application scheme (required) : project background, application value, technical scheme, project effect, team introduction, etc. PDF or PPT format, no more than 50MB.

Application Works (Required) : Application experience entry and experience method

Talk is cheap, Show me the demo, works focus on completion rather than perfection

## 2.Grading rules

### 2.1 New and unique

- Whether the idea of the innovative work

### 2.2 Application value

- Whether the Agent can effectively solve practical problems

- whether it has high commercial potential/social value

### 2.3 Technical implementation

- Based on Alipay treasure box design Agent implementation scheme

- Evaluation of its technical difficulty and complexity

### 2.4 User experience

- Whether the user interface is intuitive and easy to use

- whether the interaction logic is clear and reasonable, and the user experience is comfortable

## 3.Learning

### 3.1 How I Won Singapore’s GPT-4 Prompt Engineering Competition[1]

A deep dive into the strategies I learned for harnessing the power of Large Language Models (LLMs)

Singapore’s first ever GPT-4 Prompt Engineering competition

organised by the Government Technology Agency of Singapore (GovTech)

Prompt engineering is a discipline that blends both art and science — it is as much technical understanding as it is of creativity and strategic thinking.

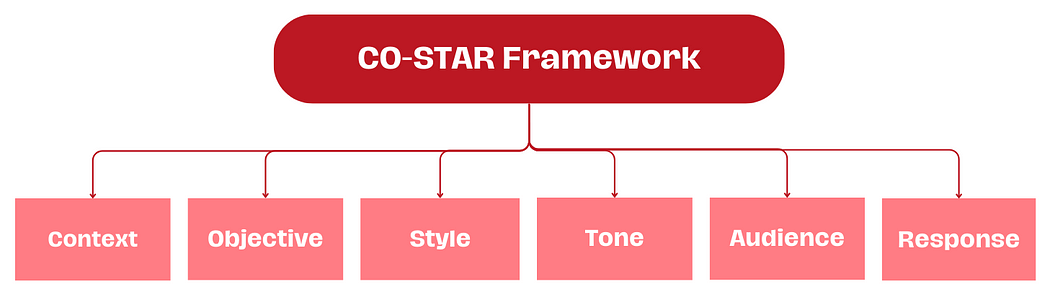
This is a compilation of the prompt engineering strategies I learned along the way, that push any LLM to do exactly what you need and more!

#### 3.1.1. Structuring prompts using the CO-STAR framework

Effective prompt structuring is crucial for eliciting optimal responses from an LLM.

The CO-STAR framework, a brainchild of GovTech Singapore’s Data Science & AI team, is a handy template for structuring prompts.

It considers all the key aspects that influence the effectiveness and relevance of an LLM’s response, leading to more optimal responses.



CO-STAR framework — Image by author

##### 3.1.1.1 Here’s how it works:

(C) Context: Provide background information on the task

This helps the LLM understand the specific scenario being discussed, ensuring its response is relevant.

(O) Objective: Define what the task is that you want the LLM to perform

Being clear about your objective helps the LLM to focus its response on meeting that specific goal.

(S) Style: Specify the writing style you want the LLM to use

This could be a particular famous person’s style of writing, or a particular expert in a profession, like a business analyst expert or CEO. This guides the LLM to respond with the manner and choice of words aligned with your needs.

(T) Tone: Set the attitude of the response

This ensures the LLM’s response resonates with the intended sentiment or emotional context required. Examples are formal, humorous, empathetic, among others.

(A) Audience: Identify who the response is intended for

Tailoring the LLM’s response to an audience, such as experts in a field, beginners, children, and so on, ensures that it is appropriate and understandable in your required context.

(R) Response: Provide the response format

This ensures that the LLM outputs in the exact format that you require for downstream tasks. Examples include a list, a JSON, a professional report, and so on. For most LLM applications which work on the LLM responses programmatically for downstream manipulations, a JSON output format would be ideal.

##### 3.1.1.2 A practical application of CO-STAR

Here’s a real-world example of why the CO-STAR framework is useful.

Let’s say you work as a social media manager, and you need help drafting a Facebook post to advertise your company’s new product.

A quick prompt without CO-STAR might look something like this:

Write a facebook post to advertise my company’s new product. My company’s name is Alpha and the product is called Beta, a new ultra-fast hairdryer.

Here’s the output from GPT-4:

Introducing the Revolutionary Beta Hairdryer by Alpha!

- Say goodbye to long, tedious hair drying sessions! Alpha is proud to unveil the Beta, our latest innovation in haircare technology.

- Ultra-Fast Drying: The Beta is designed to cut your drying time in half, thanks to its advanced, powerful motor. Get ready faster and enjoy more time doing what you love!

Shop Now: [Link to product page]

While adequate, this output is painfully generic, lacking the specific nuances and targeted appeal necessary to resonate with your company’s audience.

Here’s an example of using the CO-STAR template instead, which reminds you to incorporate other aspects of your task into your prompt, specifically the STYLE, TONE, and AUDIENCE aspects that are missing from the previous quick prompt:

# CONTEXT #

I want to advertise my company’s new product. My company’s name is Alpha and the product is called Beta, which is a new ultra-fast hairdryer.

# OBJECTIVE #

Create a Facebook post for me, which aims to get people to click on the product link to purchase it.

# STYLE #

Follow the writing style of successful companies that advertise similar products, such as Dyson.

# TONE #

Persuasive

# AUDIENCE #

My company’s audience profile on Facebook is typically the older generation. Tailor your post to target what this audience typically looks out for in hair products.

# RESPONSE #

The Facebook post, kept concise yet impactful.

By applying CO-STAR, GPT-4’s response becomes significantly more targeted and effective:

Embrace Effortless Elegance with Alpha Beta — The Ultra-Fast Hairdryer for the Wise Generation.

Rediscover the joy of simple, effective hair care with Alpha Beta. Our latest innovation is more than just a hairdryer; it’s a promise of swift, gentle, and reliable hair styling for those who appreciate the finer things in life.

- Easy and Intuitive Use: Say goodbye to complicated gadgets. Alpha Beta is crafted for comfort and simplicity, perfect for those who value straightforward, hassle-free technology.

- Time-Saving Technology: We understand your time is precious. That’s why Alpha Beta cuts down drying time significantly, giving you more moments to enjoy life’s pleasures.

Make every day a good hair day with Alpha Beta. Experience the blend of sophistication and simplicity today.

[Your Product Link Here]

The CO-STAR framework guides you to provide all of the crucial pieces of information about your task to the LLM in a structured manner, ensuring a tailored and optimized response to exactly what you need.

#### 3.1.2.Sectioning prompts using delimiters

Delimiters are special tokens that help the LLM distinguish which parts of your prompt it should consider as a single unit of meaning.

This is important because your entire prompt arrives to the LLM as a single long sequence of tokens.

Delimiters provide structure to this sequence of tokens by fencing specific parts of your prompt to be treated differently.

It is noteworthy that delimiters may not make a difference to the quality of an LLM’s response for straightforward tasks.

However, the more complex the task, the more impact the usage of delimiters for sectioning has on the LLM’s response.

Delimiters as Special Characters

A delimiter could be any sequence of special characters that usually wouldn’t appear together, for example:

###

===

>>>

The number and type of special characters chosen is inconsequential, as long as they are unique enough for the LLM to understand them as content separators instead of normal punctuation.

Here’s an example of how you might use such delimiters in a prompt:

Classify the sentiment of each conversation in <<<CONVERSATIONS>>> as

‘Positive’ or ‘Negative’. Give the sentiment classifications without any other preamble text.

###

EXAMPLE CONVERSATIONS

[Agent]: Good morning, how can I assist you today?

[Customer]: This product is terrible, nothing like what was advertised!

[Customer]: I’m extremely disappointed and expect a full refund.

[Agent]: Good morning, how can I help you today?

[Customer]: Hi, I just wanted to say that I’m really impressed with your

product. It exceeded my expectations!

###

EXAMPLE OUTPUTS

Negative

Positive

###

<<<

[Agent]: Hello! Welcome to our support. How can I help you today?

[Customer]: Hi there! I just wanted to let you know I received my order, and

it’s fantastic!

[Agent]: That’s great to hear! We’re thrilled you’re happy with your purchase.

Is there anything else I can assist you with?

[Customer]: No, that’s it. Just wanted to give some positive feedback. Thanks

for your excellent service!

[Agent]: Hello, thank you for reaching out. How can I assist you today?

[Customer]: I’m very disappointed with my recent purchase. It’s not what I expected at all.

[Agent]: I’m sorry to hear that. Could you please provide more details so I can help?

[Customer]: The product is of poor quality and it arrived late. I’m really

unhappy with this experience.

>>>

Above, the examples are sectioned using the delimiter ###, with the section headings EXAMPLE CONVERSATIONS and EXAMPLE OUTPUTS in capital letters to differentiate them.

The preamble states that the conversations to be classified are sectioned inside <<<CONVERSATIONS>>>, and these conversations are subsequently given to the LLM at the bottom of the prompt without any explanatory text, but the LLM understands that these are the conversations it should classify due to the presence of the delimiters <<< and >>>.

Here is the output from GPT-4, with the sentiment classifications given without any other preamble text outputted, like what we asked for:

Positive

Negative

Delimiters as XML Tags

Another approach to using delimiters is having them as XML tags. XML tags are tags enclosed in angle brackets, with opening and closing tags. An example is <tag> and </tag>. This is effective as LLMs have been trained on a lot of web content in XML, and have learned to understand its formatting.

Here’s the same prompt above, but structured using XML tags as delimiters instead:

Classify the sentiment of the following conversations into one of two classes, using the examples given. Give the sentiment classifications without any other

preamble text.

<classes>

Positive

Negative

</classes>

<example-conversations>

[Agent]: Good morning, how can I assist you today?

[Customer]: This product is terrible, nothing like what was advertised!

[Customer]: I’m extremely disappointed and expect a full refund.

[Agent]: Good morning, how can I help you today?

[Customer]: Hi, I just wanted to say that I’m really impressed with your

product. It exceeded my expectations!

</example-conversations>

<example-classes>

Negative

Positive

</example-classes>

<conversations>

[Agent]: Hello! Welcome to our support. How can I help you today?

[Customer]: Hi there! I just wanted to let you know I received my order, and

it’s fantastic!

[Agent]: That’s great to hear! We’re thrilled you’re happy with your purchase.

Is there anything else I can assist you with?

[Customer]: No, that’s it. Just wanted to give some positive feedback. Thanks

for your excellent service!

[Agent]: Hello, thank you for reaching out. How can I assist you today?

[Customer]: I’m very disappointed with my recent purchase. It’s not what I

expected at all.

[Agent]: I’m sorry to hear that. Could you please provide more details so I

can help?

[Customer]: The product is of poor quality and it arrived late. I’m really

unhappy with this experience.

</conversations>

It is beneficial to use the same noun for the XML tag as the words you have used to describe them in the instructions. The instructions we gave in the prompt above were:

Classify the sentiment of the following conversations into one of two classes, using the examples given. Give the sentiment classifications without any other

preamble text.

Where we used the nouns conversations, classes, and examples. As such, the XML tags we use as delimiters are <conversations>, <classes>, <example-conversations>, and <example-classes>. This ensures that the LLM understands how your instructions relate to the XML tags used as delimiters.

Again, the sectioning of your instructions in a clear and structured manner through the use of delimiters ensures that GPT-4 responds exactly how you want it to:

Positive

Negative

#### 3.1.3. Creating system prompts with LLM guardrails

Before diving in, it is important to note that this section is relevant only to LLMs that possess a System Prompt feature, unlike the other sections in this article which are relevant for any LLM. The most notable LLM with this feature is, of course, ChatGPT, and therefore we will use ChatGPT as the illustrating example for this section.

Terminology surrounding System Prompts

First, let’s iron out terminology: With regards to ChatGPT, there exists a plethora of resources using these 3 terms almost interchangeably: “System Prompts”, “System Messages”, and “Custom Instructions”. This has proved confusing to many (including me!), so much so that OpenAI released an [article](https://help.openai.com/en/articles/8234522-chat-completions-api-system-message-vs-custom-instructions-in-ui" \t "https://towardsdatascience.com/_blank) explaining these terminologies. Here’s a quick summary of it:

“System Prompts” and “System Messages” are terms used when interacting with ChatGPT programmatically over its Chat Completions API.

On the other hand, “Custom Instructions” is the term used when interacting with ChatGPT over its user interface at [https://chat.openai.com/](https://chat.openai.com/" \t "https://towardsdatascience.com/_blank).

Overall, though, the 3 terms refer to the same thing, so don’t let the terminology confuse you! Moving forward, this section will use the term “System Prompts”. Now let’s dive in!

##### What are System Prompts?

System Prompts are an additional prompt where you provide instructions on how the LLM should behave. It is considered additional as it is outside of your “normal” prompts (better known as User Prompts) to the LLM.

Within a chat, every time you provide a new prompt, System Prompts act like a filter that the LLM automatically applies before giving its response to your new prompt. This means that the System Prompts are taken into account every time the LLM responds within the chat.

When should System Prompts be used?

The first question on your mind might be: Why should I provide instructions inside the System Prompt when I can also provide them in my first prompt to a new chat, before further conversations with the LLM?

The answer is because LLMs have a limit to their conversational memory. In the latter case, as the conversation carries on, the LLM is likely to “forget” this first prompt you provided to the chat, making these instructions obsolete.

On the other hand, when instructions are provided in the System Prompt, these System Prompt instructions are automatically taken into account together with each new prompt provided to the chat. This ensures that the LLM continues to receive these instructions even as the conversation carries on, no matter how long the chat becomes.

In conclusion:

Use System Prompts to provide instructions that you want the LLM to remember when responding throughout the entire chat.

##### What should System Prompts include?

Instructions in the System Prompt typically includes the following categories:

Task definition, so the LLM will always remember what it has to do throughout the chat.

Output format, so the LLM will always remember how it should respond.

Guardrails, so the LLM will always remember how it should \*not\* respond. Guardrails are emerging field in LLM governance, referring to configured boundaries that an LLM is allowed to operate in.

For example, a System Prompt might look like this:

ou will answer questions using this text: [insert text].

You will respond with a JSON object in this format: {“Question”: “Answer”}.

If the text does not contain sufficient information to answer the question, do not make up information and give the answer as “NA”.

You are only allowed to answer questions related to [insert scope]. Never answer any questions related to demographic information such as age, gender, and religion.

Where each portion relates to the categories as follows:

Breaking down a System Prompt — Image by author

But then what goes into the “normal” prompts to the chat?

Now you might be thinking: That sounds like a lot of information already being given in the System Prompt. What do I put in my “normal” prompts (better known as User Prompts) to the chat then?

The System Prompt outlines the general task at hand. In the above System Prompt example, the task has been defined to only use a specific piece of text for question-answering, and the LLM is instructed to respond in the format {"Question": "Answer"}.

You will answer questions using this text: [insert text].

You will respond with a JSON object in this format: {“Question”: “Answer”}.

In this case, each User Prompt to the chat would simply be the question that you want answered using the text. For example, a User Prompt might be "What is the text about?". And the LLM would respond with {"What is the text about?": "The text is about..."}.

But let’s generalize this task example further. In practice, it would be more likely that you have multiple pieces of text that you want to ask questions on, rather than just 1. In this case, we could edit the first line of the above System Prompt from

You will answer questions using this text: [insert text].

to

You will answer questions using the provided text.

Now, each User Prompt to the chat would include both the text to conduct question-answering over, and the question to be answered, such as:

<text>

[insert text]

</text>

<question>

[insert question]

</question>

Here, we also use XML tags as delimiters in order to provide the 2 required pieces of information to the LLM in a structured manner. The nouns used in the XML tags, text and question, correspond to the nouns used in the System Prompt so that the LLM understands how the tags relate to the System Prompt instructions.

In conclusion, the System Prompt should give the overall task instructions, and each User Prompt should provide the exact specifics that you want the task to be executed using. In this case, for example, these exact specifics are the text and the question.

Extra: Making LLM guardrails dynamic

Above, guardrails are added through a few sentences in the System Prompt. These guardrails are then set in stone and do not change for the entire chat. What if you wish to have different guardrails in place at different points of the conversation?

Unfortunately for users of the ChatGPT user interface, there is no straightforward way to do this right now. However, if you’re interacting with ChatGPT programmatically, you’re in luck! The increasing focus on building effective LLM guardrails has seen the development of open-source packages that allow you to set up far more detailed and dynamic guardrails programmatically.

A noteworthy one is NeMo Guardrails developed by the NVIDIA team, which allows you to configure the expected conversation flow between users and the LLM, and thus set up different guardrails at different points of the chat, allowing for dynamic guardrails that evolve as the chat progresses. I definitely recommend checking it out!

#### 3.1.4.Analyzing datasets using only LLMs, without plugins or code

You might have heard of OpenAI’s Advanced Data Analysis plugin within ChatGPT’s GPT-4 that is available to premium (paid) accounts. It allows users to upload datasets to ChatGPT and run code directly on the dataset, allowing for accurate data analysis.

But did you know that you don’t always need such plugins to analyze datasets well with LLMs? Let’s first understand the strengths and limitations of purely using LLMs to analyze datasets.

Types of dataset analysis that LLMs are \*not\* great at

As you probably already know, LLMs are limited in their ability to perform accurate mathematical calculations, making them unsuitable for tasks requiring precise quantitative analysis on datasets, such as:

Descriptive Statistics: Summarizing numerical columns quantitatively, through measures like the mean or variance.

Correlation Analysis: Obtaining the precise correlation coefficient between columns.

Statistical Analysis: Such as hypothesis testing to determine if there are statistically significant differences between groups of data points.

Machine Learning: Performing predictive modelling on a dataset such as using linear regressions, gradient boosted trees, or neural networks.

Performing such quantitative tasks on datasets is why OpenAI’s Advanced Data Analysis plugin exists, so that programming languages step in to run code for such tasks on a dataset.

So, why would anyone want to analyze datasets using only LLMs and without such plugins?

Types of dataset analysis that LLMs are great at

LLMs are excellent at identifying patterns and trends. This capability stems from their extensive training on diverse and voluminous data, enabling them to discern intricate patterns that may not be immediately apparent.

This makes them well-suited for tasks based on pattern-finding within datasets, such as:

Anomaly detection: Identifying unusual data points that deviate from the norm, based on one or more column values.

Clustering: Grouping data points with similar characteristics across columns.

Cross-Column Relationships: Identifying combined trends across columns.

Textual Analysis (For text-based columns): Categorization based on topic or sentiment.

Trend Analysis (For datasets with time aspects): Identifying patterns, seasonal variations, or trends within columns across time.

For such pattern-based tasks, using LLMs alone may in fact produce better results within a shorter timeframe than using code! Let’s illustrate this fully with an example.

Analyzing a Kaggle dataset using only LLMs

We’ll use a popular real-world Kaggle dataset curated for Customer Personality Analysis, wherein a company seeks to segment its customer base in order to understand its customers better.

For easier validation of the LLM’s analysis later, we’ll subset this dataset to 50 rows and retain only the most relevant columns. After which, the dataset for analysis looks like this, where each row represents a customer, and the columns depict customer information:

First 3 rows of dataset — Image by author

Say you work on the company’s marketing team. You are tasked to utilize this dataset of customer information to guide marketing efforts. This is a 2-step task: First, use the dataset to generate meaningful customer segments. Next, generate ideas on how to best market towards each segment. Now this is a practical business problem where the pattern-finding (for step 1) capability of LLMs can truly excel.

Let’s craft a prompt for this task as follows, using 4 prompt engineering techniques (more on these later!):

1. Breaking down a complex task into simple steps

2. Referencing intermediate outputs from each step

3. Formatting the LLM’s response

4. Separating the instructions from the dataset

System Prompt:

I want you to act as a data scientist to analyze datasets. Do not make up information that is not in the dataset. For each analysis I ask for, provide me with the exact and definitive answer and do not provide me with code or instructions to do the analysis on other platforms.

Prompt:

# CONTEXT #

I sell wine. I have a dataset of information on my customers: [year of birth, marital status, income, number of children, days since last purchase, amount spent].

#############

# OBJECTIVE #

I want you use the dataset to cluster my customers into groups and then give me ideas on how to target my marketing efforts towards each group. Use this step-by-step process and do not use code:

1. CLUSTERS: Use the columns of the dataset to cluster the rows of the dataset, such that customers within the same cluster have similar column values while customers in different clusters have distinctly different column values. Ensure that each row only belongs to 1 cluster.

For each cluster found,

2. CLUSTER\_INFORMATION: Describe the cluster in terms of the dataset columns.

3. CLUSTER\_NAME: Interpret [CLUSTER\_INFORMATION] to obtain a short name for the customer group in this cluster.

4. MARKETING\_IDEAS: Generate ideas to market my product to this customer group.

5. RATIONALE: Explain why [MARKETING\_IDEAS] is relevant and effective for this customer group.

#############

# STYLE #

Business analytics report

#############

# TONE #

Professional, technical

#############

# AUDIENCE #

My business partners. Convince them that your marketing strategy is well thought-out and fully backed by data.

#############

# RESPONSE: MARKDOWN REPORT #

<For each cluster in [CLUSTERS]>

— Customer Group: [CLUSTER\_NAME]

— Profile: [CLUSTER\_INFORMATION]

— Marketing Ideas: [MARKETING\_IDEAS]

— Rationale: [RATIONALE]

<Annex>

Give a table of the list of row numbers belonging to each cluster, in order to back up your analysis. Use these table headers: [[CLUSTER\_NAME], List of Rows].

#############

# START ANALYSIS #

If you understand, ask me for my dataset.

Below is GPT-4’s reply, and we proceed to pass the dataset to it in a CSV string.

GPT-4's response — Image by author

Following which, GPT-4 replies with its analysis in the markdown report format we asked for:

GPT-4's response — Image by author

GPT-4's response — Image by author

GPT-4's response — Image by author

Validating the LLM’s analysis

For the sake of brevity, we’ll pick 2 customer groups generated by the LLM for validation — say, Young Families and Discerning Enthusiasts.

Young Families

- Profile synthesized by LLM: Born after 1980, Married or Together, Moderate to low income, Have children, Frequent small purchases.

- Rows clustered into this group by LLM: 3, 4, 7, 10, 16, 20

- Digging into the dataset, the full data for these rows are:

Full data for Young Families — Image by author

Which exactly correspond to the profile identified by the LLM. It was even able to cluster the row with a null value without us preprocessing it beforehand!

Discerning Enthusiasts

- Profile synthesized by LLM: Wide age range, Any marital status, High income, Varied children status, High spend on purchases.

- Rows clustered into this group by LLM: 2, 5, 18, 29, 34, 36

- Digging into the dataset, the full data for these rows are:

Full data for Discerning Enthusiasts — Image by author

Which again align very well with the profile identified by the LLM!

This example showcases LLMs’ abilities in pattern-finding, interpreting and distilling multi-dimensional datasets into meaningful insights, while ensuring that its analysis is deeply rooted in the factual truth of the dataset.

What if we used ChatGPT’s Advanced Data Analysis plugin?

For completeness, I attempted this same task with the same prompt, but asked ChatGPT to execute the analysis using code instead, which activated its Advanced Data Analysis plugin. The idea was for the plugin to run code using a clustering algorithm like K-Means directly on the dataset to obtain each customer group, before synthesizing the profile of each cluster to provide marketing strategies.

However, multiple attempts resulted in the following error messages with no outputs, despite the dataset being only 50 rows:

Error and no output from Attempt 1 — Image by author

Error and no output from Attempt 2 — Image by author

With the Advanced Data Analysis plugin right now, it appears that executing simpler tasks on datasets such as calculating descriptive statistics or creating graphs can be easily achieved, but more advanced tasks that require computing of algorithms may sometimes result in errors and no outputs, due to computational limits or otherwise.

So…When to analyze datasets using LLMs?

The answer is it depends on the type of analysis.

For tasks requiring precise mathematical calculations or complex, rule-based processing, conventional programming methods remain superior.

For tasks based on pattern-recognition, it can be challenging or more time-consuming to execute using conventional programming and algorithmic approaches. LLMs, however, excel at such tasks, and can even provide additional outputs such as annexes to back up its analysis, and full analysis reports in markdown formatting.

Ultimately, the decision to utilize LLMs hinges on the nature of the task at hand, balancing the strengths of LLMs in pattern-recognition against the precision and specificity offered by traditional programming techniques.

Now back to the prompt engineering!

Before this section ends, let’s go back to the prompt used to generate this dataset analysis and break down the key prompt engineering techniques used:

Prompt:

# CONTEXT #

I sell wine. I have a dataset of information on my customers: [year of birth, marital status, income, number of children, days since last purchase, amount spent].

#############

# OBJECTIVE #

I want you use the dataset to cluster my customers into groups and then give me ideas on how to target my marketing efforts towards each group. Use this step-by-step process and do not use code:

1. CLUSTERS: Use the columns of the dataset to cluster the rows of the dataset, such that customers within the same cluster have similar column values while customers in different clusters have distinctly different column values. Ensure that each row only belongs to 1 cluster.

For each cluster found,

2. CLUSTER\_INFORMATION: Describe the cluster in terms of the dataset columns.

3. CLUSTER\_NAME: Interpret [CLUSTER\_INFORMATION] to obtain a short name for the customer group in this cluster.

4. MARKETING\_IDEAS: Generate ideas to market my product to this customer group.

5. RATIONALE: Explain why [MARKETING\_IDEAS] is relevant and effective for this customer group.

#############

# STYLE #

Business analytics report

#############

# TONE #

Professional, technical

#############

# AUDIENCE #

My business partners. Convince them that your marketing strategy is well thought-out and fully backed by data.

#############

# RESPONSE: MARKDOWN REPORT #

<For each cluster in [CLUSTERS]>

— Customer Group: [CLUSTER\_NAME]

— Profile: [CLUSTER\_INFORMATION]

— Marketing Ideas: [MARKETING\_IDEAS]

— Rationale: [RATIONALE]

<Annex>

Give a table of the list of row numbers belonging to each cluster, in order to back up your analysis. Use these table headers: [[CLUSTER\_NAME], List of Rows].

#############

# START ANALYSIS #

If you understand, ask me for my dataset.

Technique 1: Breaking down a complex task into simple steps

LLMs are great at performing simple tasks, but not so great at complex ones. As such, with complex tasks like this one, it is important to break down the task into simple step-by-step instructions for the LLM to follow. The idea is to give the LLM the steps that you yourself would take to execute the task.

In this example, the steps are given as:

Use this step-by-step process and do not use code:

1. CLUSTERS: Use the columns of the dataset to cluster the rows of the dataset, such that customers within the same cluster have similar column values while customers in different clusters have distinctly different column values. Ensure that each row only belongs to 1 cluster.

For each cluster found,

2. CLUSTER\_INFORMATION: Describe the cluster in terms of the dataset columns.

3. CLUSTER\_NAME: Interpret [CLUSTER\_INFORMATION] to obtain a short name for the customer group in this cluster.

4. MARKETING\_IDEAS: Generate ideas to market my product to this customer group.

5. RATIONALE: Explain why [MARKETING\_IDEAS] is relevant and effective for this customer group.

As opposed to simply giving the overall task to the LLM as “Cluster the customers into groups and then give ideas on how to market to each group”.

With step-by-step instructions, LLMs are significantly more likely to deliver the correct results.

Technique 2: Referencing intermediate outputs from each step

When providing the step-by-step process to the LLM, we give the intermediate output from each step a capitalized VARIABLE\_NAME, namely CLUSTERS, CLUSTER\_INFORMATION, CLUSTER\_NAME, MARKETING\_IDEAS and RATIONALE.

Capitalization is used to differentiate these variable names from the body of instructions given. These intermediate outputs can later be referenced using square brackets as [VARIABLE\_NAME].

Technique 3: Formatting the LLM’s response

Here, we ask for a markdown report format, which beautifies the LLM’s response. Having variable names from intermediate outputs again comes in handy here to dictate the structure of the report.

# RESPONSE: MARKDOWN REPORT #

<For each cluster in [CLUSTERS]>

— Customer Group: [CLUSTER\_NAME]

— Profile: [CLUSTER\_INFORMATION]

— Marketing Ideas: [MARKETING\_IDEAS]

— Rationale: [RATIONALE]

<Annex>

Give a table of the list of row numbers belonging to each cluster, in order to back up your analysis. Use these table headers: [[CLUSTER\_NAME], List of Rows].

In fact, you could even subsequently ask ChatGPT to provide the report as a downloadable file, allowing you to work off of its response in writing your final report.

Saving GPT-4's response as a file — Image by author

Technique 4: Separating the task instructions from the dataset

You’ll notice that we never gave the dataset to the LLM in our first prompt. Instead, the prompt gives only the task instructions for the dataset analysis, with this added to the bottom:

# START ANALYSIS #

If you understand, ask me for my dataset.

ChatGPT then responded that it understands, and we passed the dataset to it as a CSV string in our next prompt:

GPT-4's response — Image by author

But why separate the instructions from the dataset?

Doing so helps the LLM maintain clarity in understanding each, with lower likelihood of missing out information, especially in more complex tasks such as this one with longer instructions. You might have experienced scenarios where the LLM “accidentally forgets” a certain instruction you gave as part of a longer prompt — for example, if you asked for a 100-word response and the LLM gives you a longer paragraph back. By receiving the instructions first, before the dataset that the instructions are for, the LLM can first digest what it should do, before executing it on the dataset provided next.

Note however that this separation of instructions and dataset can only be achieved with chat LLMs as they maintain a conversational memory, unlike completion LLMs which do not.

Closing Thoughts

Before this article ends, I wanted to share some personal reflections on this incredible journey.

First, a heartfelt thank you to GovTech Singapore for orchestrating such an amazing competition. If you’re interested in the mechanics of how GovTech organized this first-of-its-kind competition — check out this article by Nicole Lee, the lead organizer herself!

A live on-stage battle in the final round!

Second, a big shout-out to my fellow phenomenal competitors, who each brought something special, making the competition as enriching as it was challenging! I’ll never forget the final round, with us battling it out on stage and a live audience cheering us on — an experience I’ll always remember fondly.

For me, this wasn’t just a competition; it was a celebration of talent, creativity, and the spirit of learning. And I’m beyond excited to see what comes next!

I had a lot of fun writing this, and if you had fun reading, I would really appreciate if you took a second to leave some claps and a follow!

See you in the next one!

Sheila

## Refer

1. Teo, S. (n.d.). \*How I won Singapore’s GPT-4 prompt engineering competition\*. Towards Data Science. Retrieved December 29, 2023, from <https://towardsdatascience.com/how-i-won-singapores-gpt-4-prompt-engineering-competition-34c195a93d41>