Prompt Engineering Approaches and Strategies

Iterating with increased specificity and restricting output

It can often be difficult to get exactly the output you want on the first try. Don't worry, this is normal! Make note of the parts of the output that you wanted and the parts that you didn't like. Convey this information to the language model and try to give a detailed explanation of how this can be improved. It can be useful to give the model partial credit here.

Let's look at an example:

I asked ChatGPT to generate some code in python that mimics the behavior of the datetime class, with the goal of teaching students about operator overloading. However the output I got used the built in datetime methods which would undermine the purpose of the project.

Here is the prompt I used to get the desired code:

> *You can't use the built in date class since the point of this assignment is to replicate that functionally and introduce students to operator overloading. Redo that with manual calculations used to adjust the day month and year properties. Also only show me the code where you made changes*

Notice a few important things here.

First: I was very specific on the issue I was having. This helps steer our model in the direction we want it to go.

Second: I give very detailed instructions on what I want it to do. Giving broad or vague instructions will greatly increase the chance it will misinterpret you so it is often a good exercise to try and spend some time thinking about how to define your problem.

The good news here is that you can use the language model to get feedback on this!

Let’s look at some examples on how we can do this:

I was creating a comedic website and wanted a homepage, here was my first prompt:

> *For my homepage I want to have a picture on the left with a company description on the left. Then I want a picture below that on the right with a company mission statement on the left*

This wasn’t a very good prompt, it's poorly phrased. I could have one description and image pushed to the very far left of my homepage and another to the very far right leaving an ugly gap based on this description. Part of this was due to me not having a very good idea of exactly what I wanted.

The good news is once I saw the output I was able to use that as feedback for myself and better express my desires on my next prompt, getting me the desired results.

Prompt:

> *You misunderstood me. I would like effectively four quadrants. The upper right quadrant should be occupied by an image, and so should the lower left quadrant. The other two will have text.*

Quadrants is a much more descriptive term as it clearly states that each portion of my content should occupy an equal amount of space and give it an absolute position relative to my homepage which is more restrictive. This is a common theme, the more restrictive we can make our prompts the better results we can expect.

Let’s look at another example. Here I had a problem where there was a lot of dead white space in my webpage, I wanted the content to fill out more of the page to give it a better look.

Here's my first attempt, note I had already provided ChatGPT with my code so that's why there is no code being passed in my prompts.

> *I'm having an issue with my testimonials page where it's not filling the entire screen. When I zoom in to 150% I find it looks a lot better.*

Can you spot what the issues are here?

First, I don't give it any specific instructions on what I want, instead I vaguely allude to a problem, this can work but it increases the likelihood It will misconstrued my desires. Second, I explain what a better result looks like but not in a very useful way. So what if 150% zoom is better? How could I translate this realization into an actionable prompt?

Here is my next attempt with several improvements

> *I think you misunderstood me. What I mean was that currently when you load the webpage there is a lot of empty space visible below the testimonials. I would like my text and testimonials to fill up the entire scre*en.

Notice I'm giving it a specific task, I want you to do X. Also instead of a vague well this looks better I give it a set of actions it can take to get me the desired effect, "I would like my text and testimonials to fill the entire page". I also explain to it what I don't like in my current version, the dead space, this gives it something specific to focus on.

After this it gave me a better version where the spacing between elements was consistent, eliminating the large awkward chunk of dead space at the bottom on my webpage, likely because I specifically mentioned this being an issue. However, it didn't quite give me what I wanted as now the dead space was simply spread throughout the entire page.

So here was my final prompt:

> *So adding padding is better than the current implementation, however what I really want is for the actual text and images in the testimonial to be larger.*

Here I give the model partial credit, so it knows it's on the right track and specifically points out what changes need to be made to get my final output.

Increasing specificity with numerical instructions

Using numerical values can give you a high degree of control over your desired output. Here is an example:

Prompt:

> *I am currently making a humorous website that pokes fun of over the top claims that tech companies make about their products. An example could be a company that makes a slightly better blender might claim it's "changing the world" and use a lot of tech buzzwords that don't mean anything. I have a products page on this site where I'm advertising my "5g scooter" and the image is a poorly made drawing of the scooter. I would like a product description blurb that advertises the scooter in an over the top fashion that clearly is not justified by the image on my scooter.*

Output:

> *Introducing the revolutionary 5G Scooter: The Ultimate Quantum Mobility Experience™! Our state-of-the-art scooter is engineered with unparalleled nanotechnology and hyper-intelligent AI algorithms that redefine the very essence of personal transportation. Prepare to be awestruck by its sleek, hand-drawn design that defies logic and transcends aesthetics.*
>
> *The 5G Scooter effortlessly merges the realms of convenience, speed, and ecological sustainability. With its groundbreaking 5G connectivity, you'll glide through smart cities in perfect harmony with the Internet of Things. And the best part? It's all powered by our patented Flux Capacitor Energy System™, ensuring an infinite range of zero-emission travel.*
>
> *Experience the sublime power of our Holographic Dashboard™, as you effortlessly navigate through traffic using augmented reality and real-time data streams. The 5G Scooter's advanced gyroscopic stabilization system guarantees a ride smoother than a hoverboard on a cloud.*

And, for the first time in human history, you'll be able to unlock the full potential of the scooter's quantum entanglement technology. Yes, that's right - teleportation is finally a reality! Outmaneuver traffic jams and arrive at your destination in the blink of an eye with our cutting-edge Quantum Warp Drive™.*

Don't miss your chance to be part of this epoch-defining movement! Pre-order your very own 5G Scooter today and join the ranks of those who dared to dream beyond the limits of imagination. The future of mobility is here, and it's waiting for you to take the ride of a lifetime!

**Teleportation is not guaranteed and may or may not result in scooter and rider being lost in space-time. Use Quantum Warp Drive™ at your own risk.*

I was relatively happy with this output, however, in order to fit the dimensions of the webpage it needed to be shorter. Now I could have said something like “rewrite this again however make this prompt a bit shorter” or “make it a lot shorter”. The problem with this approach is in how we define “a lot” or “a bit”, these not very specific terms allowing for a lot of interpretation. Instead we can use numerical values which are much more specific and give us better control of the output.

New prompt:

Can you give me the product description again but make it about 40% as long

New output:

Introducing the groundbreaking 5G Scooter: Redefining Personal Mobility! Our cutting-edge scooter features unmatched nanotechnology and AI-powered performance, all wrapped up in a hand-drawn design that defies expectations.

Experience seamless 5G connectivity as you glide through smart cities, powered by our eco-friendly Flux Capacitor Energy System™. With our Holographic Dashboard™, navigate effortlessly using augmented reality and real-time data.

Unlock the future of transportation with the 5G Scooter! Pre-order now and embark on the ride of a lifetime. The future is here – and it's waiting for you to take control.

Much better! Though even here the use of the term "about" was a mistake, since my definition of about might not match that of ChatGPT.

Improving results with examples

A lot of the time you might have a good idea of the type or response you are looking for in these cases prompting your model with an example response is a very good way of improving the accuracy of your results.

Let's look at one of my earlier prompts to see how I employed this:

I am currently making a humorous website that pokes fun of over the top claims that tech companies make about their products. ***An example could be a company that makes a slightly better blender might claim it's "changing the world" and use a lot of tech buzzwords that don't mean anything****. I have a products page on this site where I'm advertising my "5g scooter" and the image is a poorly made drawing of the scooter. I would like a product description blurb that advertises the scooter in an over the top fashion that clearly is not justified by the image on my scooter.*

Even though this wasn't a particularly detailed example and more a short nudge in the right direction this prompt got me some really good results.

Here is some more detailed example prompts courtesy of Emiliano Viotti, he has a great article on prompt engineering which you can find here:

https://hackernoon.com/prompt-engineering-101-i-unveiling-principles-and-techniques-of-effective-prompt-crafting

He wants ChatGPT to generate a response to customer reviews so he prompts it by giving some example responses to get it on the right track:

Prompt:

> *Review: I am very happy with this jacket, it is worth every penny*
>
> *Response: Thank you for your review, we are very happy that you are satisfied with our products.*
>
> *---*
>
> *Review: I normally wear Columbia size L jackets, this in L is like an XL. I reordered the M… unfortunately the blue was not longer available =(*
>
> *Response: Thank you for your review! We are continuously working to bring you a better shopping experience and innovative products that will surprise you.*
>
> *---*
>
> *Review: Nobody is truly going to be wearing this unless you're in the frozen tundra…. Zero mobility, way too large in length… Absolute waste…*
>
> *Response: Thank you for your review, the feedback from our users represents an invaluable asset for us that allows us to work to improve each product and bring you a better shopping experience.*
>
> *Provide them in JSON format with the following keys: review, response. Where 'review' is the first 10 words of the customer review to easily identify it and 'response' is the generated response.*

Explaining the desired structure

Explain to your model the exact structure you want your response to follow. This may not always be possible but can significantly improve your results.

Let's go back to our customer reviews and now we want our model to convey our reviews into a json format that is useful for our data analytics. In this case we have a very specific output we want and do not want our model to deviate at all from this format.

So here is the prompt Emiliano used:

Prompt:

1 - Analyze the following product reviews delimited by

<> and summarize each review with 1 sentence.

2 - Categorize the opinion according to the following categories: super happy, happy, neutral, upset

3 - Provide an overview of the customer satisfaction, listing positive and negative aspects mentioned in the review

4 - Provide a personalized answer for each one, based on the content of the review.

5 - Output a json list object that contains the

following keys for each review: summary, category, insights, answer.

<user review>

Output:

```
[
 {
  "summary": "Reviewer thinks the jacket is an over-engineered waste of money with zero mobility.",
  "category": "upset",
  "insights": {
   "positive": [],
    "negative": [
     "over engineered",
     "zero mobility",
     "large in length"
    ]
  },
   "answer": "We're sorry to hear that you found the jacket to be over-engineered and
uncomfortable. We appreciate your feedback and will take it into consideration for
future improvements." }
]
```

An important thing to note here is that doing this will greatly reduce any type of creativity your model has and should be used in cases where you have already decided on what your output should look like. Here it works really well since we need a specific format to play nice with some software that we will be applying to our formatted output, however this would not be recommended in a case where you are still exploring your output space.

Using delimiters

Delimiters can be a useful tool in prompt engineering. They are character sequences like """, ```, or <> which are used to indicate boundaries or separations between different pieces of data within the text.

Effective usage of delimiters usually requires some trial and error, as the model's response can depend heavily on the prompt and how similar structures were represented in its training data. Here are a few tips for using them effectively:

1. **Consistency**: If you use a delimiter in a certain way once, try to use it the same way in future prompts. This can help the model understand the context better.
2. **Clarity**: Choose delimiters that clearly separate different pieces of data without confusing the model. Avoid using delimiters that are too common in normal text, as the model might not recognize them as delimiters.
3. **Context**: Delimiters that are contextually relevant to the type of text you're working with can be helpful. For example, if you're prompting the model to generate HTML, using HTML-style tags as delimiters could guide the model's output.

Here are some examples:

Block of Text: You might want to separate a large block of text for clarity:

"""

This is a long paragraph of text that needs to be set apart from the rest of the prompt. It might contain complex information that requires extra attention, or perhaps it's a quote, a song lyric, or a poem.

"""

Now you can continue with the rest of your prompt here.

Code Snippet: The ``` delimiter is used in markdown to denote code blocks:

```
def hello_world():

print("Hello, world!")
```

Creating a Dialogue: You can use delimiters to separate different speakers in a conversation:

Alice: Hello, how are you today?

Bob: I'm doing well, thanks for asking. How about you?

Alice:

Here, the names "Alice:" and "Bob:" serve as delimiters, prompting the model to generate dialogue in turn.

Question-Answer Pairs: You can use "Q:" and "A:" as delimiters to generate a series of question and answer pairs:

Q: What is the capital of France?

A: Paris

Q: What is the capital of Italy?

A: Rome

Q: What is the capital of Germany?

A:

Hopefully these tips have helped you improve your prompt engineering skills and allow you to make the most out of whatever model you end up using. Good Luck!