

Prompt Engineering

The power of LLMs, large language models as a developer tool, that is using API calls to LLMs to quickly build software applications.

Two Types of large language models:

1. Base LLM: Predict the next word, based on the text training data.
 - a. This is often trained on a large amount of data on the internet.

Once upon a time, there was a unicorn
that lived in a magical forest with
all her unicorn friends

What is the capital of France?
What is France's largest city?
What is France's population?
What is the currency of France?

2. Instruction Tuned LLM: A lot of momentum and research is going on.
 - a. This LLM tries to follow instructions.
 - b. Fine tune on instructions and good attempts at following instructions.
 - c. LLM that's been trained on a huge amount of text data and further train it, further fine tune it with inputs and outputs that are instructions.
 - d. Further refined by using a technique called RLHF: Reinforcement Learning from Human Feedback to make the system better.

We have to be clear and specific which is an important principle of prompting LLMs.

Principle 1: Write clear and specific instructions:

- a. We should express what we want the model to do.
- b. This will guide the model to correct output and prevent incorrect responses.
- c. It provides more clarity and context for the model.

Tactic 1: Use delimiters.

1. Triple quotes: `"""`
2. Triple backticks: `````
3. Triple dashes: `- - -`
4. Angle brackets: `<>`
5. XML tags: `<tag> </tag>`

Tactic 2: Ask for structured output.

HTML, JSON

Tactic 3: Check whether conditions are satisfied.

1. Check assumptions required to do the task.

Tactic 4: Few Shot Prompting.

1. Give successful examples of completing tasks, the as the model to perform the task.

Principle 2: Give the model time to think:

If the model is rushing to an incorrect conclusion, we should try reframing the query to request a chain or series of relevant reasoning before the model provides its final answer.

You can ask the model to spend more time on the model which is more computational effort.

Tactic 1: Specify the steps to complete the task.

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

Model Limitations:

1. Hallucination
2. Makes statements that sound plausible but are not true.

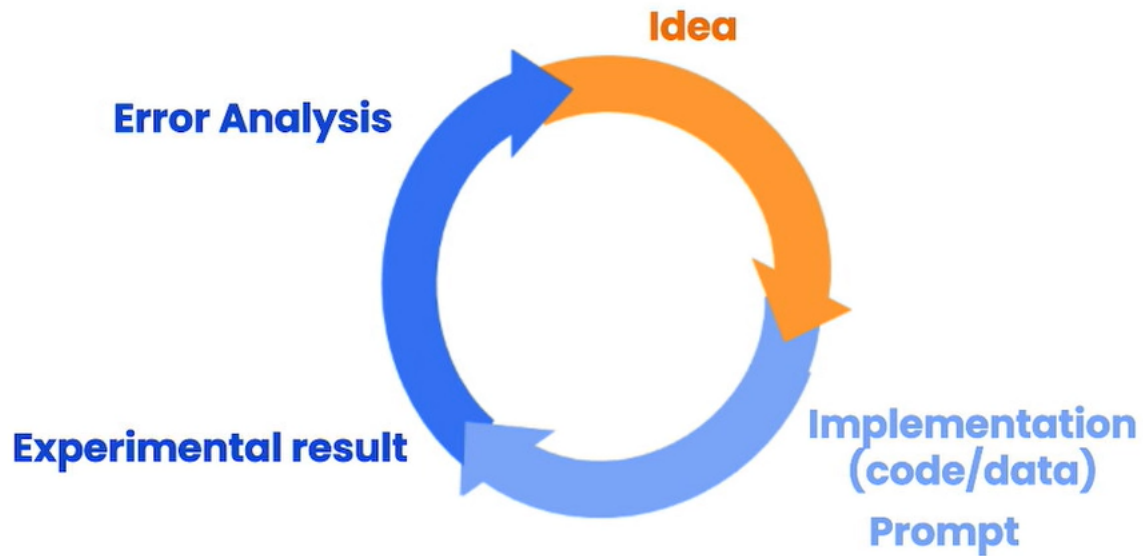
Reducing Hallucinations:

1. First find relevant information, then answer the question based on the relevant information.

Iterative Prompt Development:

In this lesson, we'll iteratively **analyze** and **refine** our prompts to generate **marketing copy** from a product fact sheet.

Iterative Prompt Development Process:



Recall Guidelines:

Prompt guidelines

- Be clear and specific
- Analyze why result does not give desired output.
- Refine the idea and the prompt
- Repeat

Application example: Generate a marketing product description from a product fact sheet

```
fact_sheet_chair = ""
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
```

```

- Soft or hard-floor caster options.
- Two choices of seat foam densities:
  medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.
SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

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.

Technical specifications: ```{fact_sheet_chair}```
"""
response = get_completion(prompt)
print(response)

```

Introducing our stunning mid-century inspired office chair, the perfect addition to any home or business setting. Part of a beautiful family, this chair is designed to provide you with the ultimate in comfort and style. The chair is constructed with a 5-wheel plastic coated aluminum base and features a pneumatic chair adjust for easy raise/lower action. It is designed to be both functional and aesthetically pleasing. Measuring at a width of 53 cm, depth of 51 cm, and height of 80 cm, with a seat height of 44 cm and seat depth of 41 cm, this chair is designed to fit perfectly into any space. The materials used in the construction of this chair are of the highest quality. The shell base glider is made of cast aluminum with modified nylon PA6/PA66 coating. This chair is made in Italy and is the perfect combination of style and functionality. Upgrade your workspace with our mid-century inspired office chair.

Issue 1: The text is too long.

1. Limit the number of words, sentences or characters.

Issue 2: Text focuses on the wrong details.

1. Ask it to focus on the aspects that are relevant to the intended audience.

Issue 3: Description needs a table of dimensions.

1. Ask it to extract information and organize it in a table.

Summarization techniques:

- Summarize with a word/sentence/character limit
- Summarize with a focus on shipping and delivery
- Summarize with a focus on price and value
- Try "extract" instead of "summarize"
- Summarize multiple product reviews

Inferring:

In this lesson, you will infer `sentiment` and `topics` from product reviews and news articles.

`Inferring` : set of tasks where the model takes a text as input and performs some kinds of analysis.

- Ex: Extracting labels, names, understand the sentiment of text, etc

Inferring LLM vs classical ML Model

- classical ML Model :
 - complex and slow process
 - need of a different model for each task (extraction, sentiment analysis..)
- LLM models : much faster and all you need is a `prompt`
 - only need one API (model) to perform many different tasks

Applications:

```
lamp_review = """
Needed a nice lamp for my bedroom, and this one had \
additional storage and not too high of a price point. \
Got it fast. The string to our lamp broke during the \
transit and the company happily sent over a new one. \
Came within a few days as well. It was easy to put \
together. I had a missing part, so I contacted their \
support and they very quickly got me the missing piece! \
Lumina seems to me to be a great company that cares \
about their customers and products!!
"""
```

Sentiment analysis (positive/negative):

```
prompt = f"""
What is the sentiment of the following product review,
which is delimited with triple backticks?

Review text: '{lamp_review}'
"""
response = get_completion(prompt)
print(response)
```

Completion:

```
The sentiment of the product review is positive.
```

Transforming:

In this notebook, we will explore how to use Large Language Models for text transformation tasks such as **language translation, spelling and grammar checking, tone adjustment, and format conversion**.

E.g: Inputting HTML and outputting JSON.

Transformation Tasks:

Translation

- ChatGPT is trained with sources in many languages. This gives the model the ability to do translation. Here are some examples of how to use this capability.

Universal Translator:

- Imagine you are in charge of IT at a large multinational e-commerce company. Users are messaging you with IT issues in all their native languages. Your staff is from all over the world and speaks only their native languages. You need a universal translator!

Tone Transformation:

- Writing can vary based on the intended audience. ChatGPT can produce different tones.

Format Conversion:

- ChatGPT can translate between formats. The prompt should describe the input and output formats.

Spellcheck/Grammar Check

- Here are some examples of common grammar and spelling problems and the LLM's response.
- To signal to the LLM that you want it to proofread your text, you instruct the model to 'proofread' or 'proofread and correct'.

Expanding:

In this lesson, you will generate customer service emails that are tailored to each customer's review.

Warnings and good usage:

- Expanding can be used to generate a large amount of spam.
- It's strongly advised to use AI & LLMs in a responsible way or in a way that helps people.

Customize the automated reply to a customer email:

```
# given the sentiment from the lesson on "inferring",
# and the original customer message, customize the email
sentiment = "negative"

# review for a blender
review = f"""
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. And the 11 piece system went up around $10 or \
so in price also from the earlier sale price of $29. \
So it looks okay, but if you look at the base, the part \
where the blade locks into place doesn't look as good \
as in previous editions from a few years ago, but I \
plan to be very gentle with it (example, I crush \
very hard items like beans, ice, rice, etc. in the \
blender first then pulverize them in the serving size \
I want in the blender then switch to the whipping \
blade for a finer flour, and use the cross cutting blade \
first when making smoothies, then use the flat blade \
if I need them finer/less pulpy). Special tip when making \
smoothies, finely cut and freeze the fruits and \
vegetables (if using spinach-lightly stew soften the \
spinach then freeze until ready for use-and if making \
sorbet, use a small to medium sized food processor) \
that you plan to use that way you can avoid adding so \
much ice if at all-when making your smoothie. \
After about a year, the motor was making a funny noise. \
I called customer service but the warranty expired \
already, so I had to buy another one. FYI: The overall \
quality has gone down in these types of products, so \
```

```
they are kind of counting on brand recognition and \
consumer loyalty to maintain sales. Got it in about \
two days.
"""
```

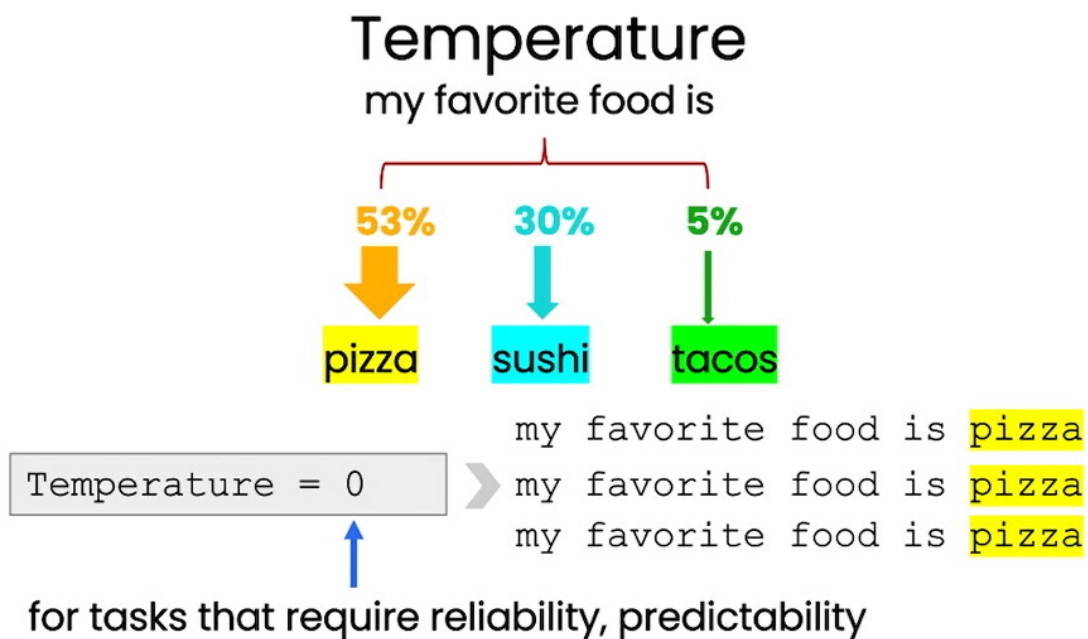
```
prompt = f"""
You are a customer service AI assistant.
Your task is to send an email reply to a valued customer.
Given the customer email delimited by ``` , \
Generate a reply to thank the customer for their review.
If the sentiment is positive or neutral, thank them for \
their review.
If the sentiment 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)
```

Model responses tuning:

- Using `Temperature` parameter allows you to vary the degree of exploration and the variety of the model's responses.

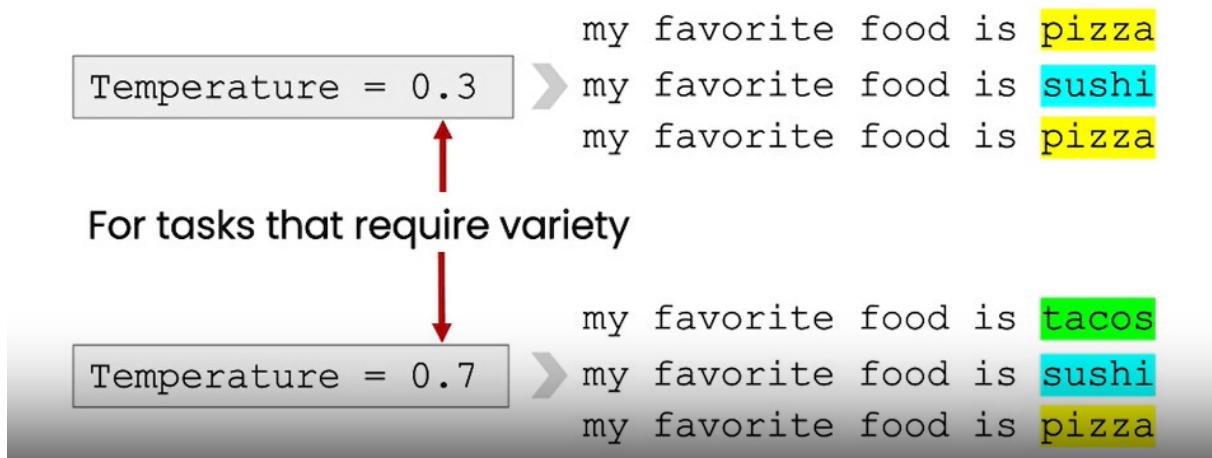
Temperature = `0`

- for tasks that require reliability, predictability



Temperature = `0.3 - 0.7`

- For tasks that require variety



- We set the temperature to 0.7, which will change the variety/randomness of the model's responses.

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)
```

Expected Completion:

```
Dear valued customer,

Thank you for taking the time to leave a review about our product. We are sorry to hear that you experienced a price increase and issue

Please know that we take all feedback seriously and we would like to assist you further. If you have any other concerns, please do not

Thank you again for your honest review and we hope to have the opportunity to serve you better in the future.

Best regards,

AI customer agent
```

Chatbot:

One of the interesting things about LLMs is we can build a custom chatbot with only a modest amount of effort.

ChatGPT, the web interface is a way for you to have a conversational interface, a conversation via a large language model.

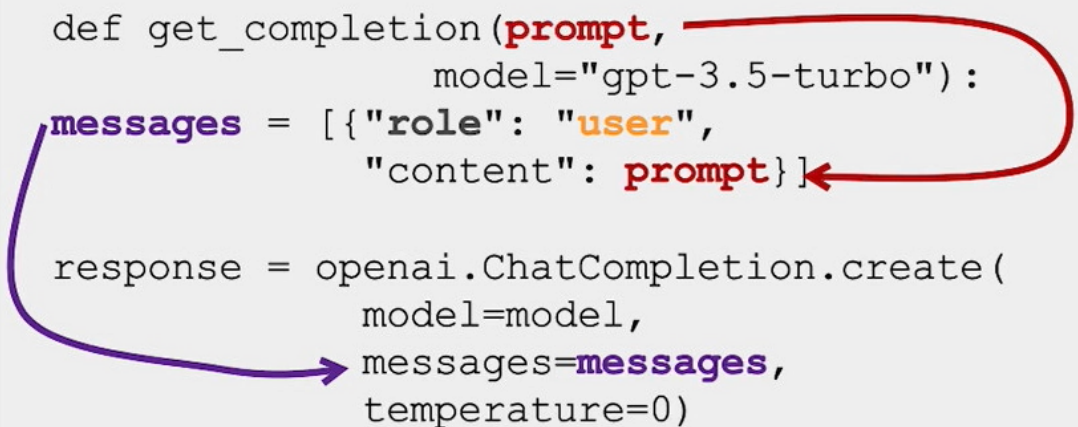
In this notebook, you will explore how you can utilize the chat format to have extended conversations with chatbots personalized or specialized for specific tasks or behaviors.

OpenAI API call Architecture:

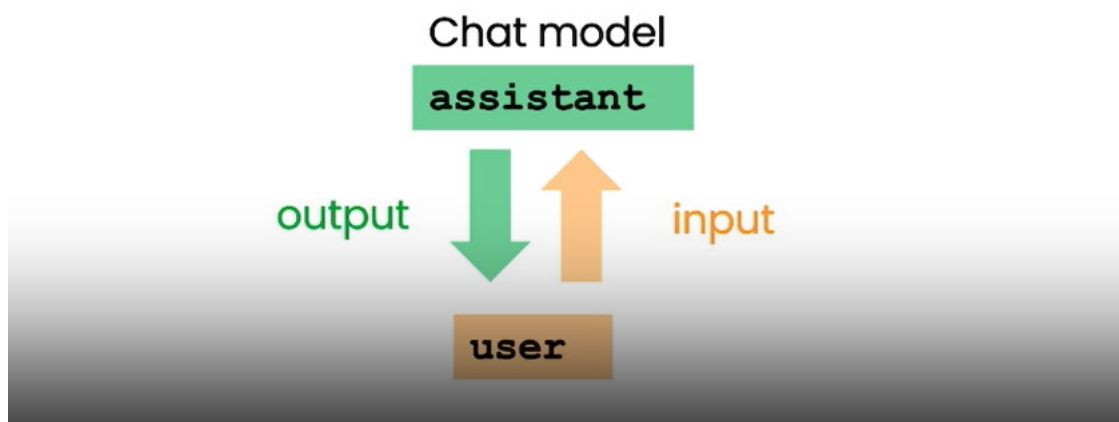
- Getting response from a single prompt/message :

OpenAI API call

```
def get_completion(prompt, model="gpt-3.5-turbo"):  
    messages = [{"role": "user",  
                  "content": prompt}]  
    response = openai.ChatCompletion.create(  
        model=model,  
        messages=messages,  
        temperature=0)
```



- Chat model



Setting multiple roles:

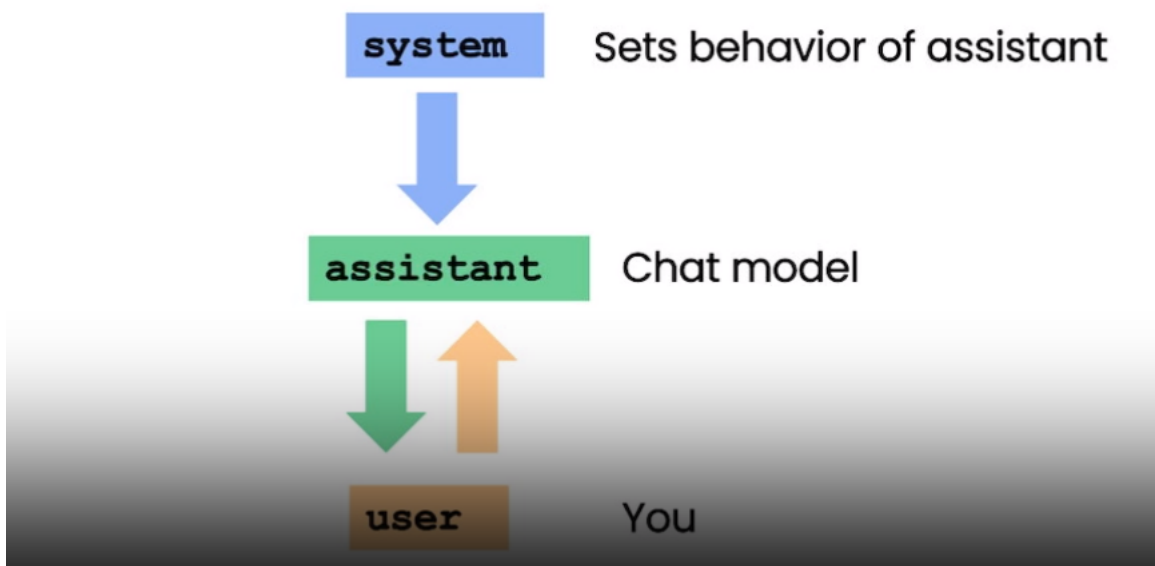
In this lesson we're going to actually use a different helper function and instead of kind of putting a single prompt as input and getting a single completion, we're going to pass in a `list of messages` and these messages can be kind of from a variety of different roles

```
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
    )
    # print(str(response.choices[0].message))
    return response.choices[0].message["content"]
```

Role

```
messages =
[
  {"role": "system",
   "content": "You are an assistant... "},
  {"role": "user",
   "content": "tell me a joke "},
  {"role": "assistant",
   "content": "Why did the chicken... "},
  ...
]
```

- The `system` message is a high-level instruction which sets the behavior of the assistant
- The `user` message give the user instruction
- The `Assistant` the model completion



Improvement interaction using context:

If you want the model to draw from, or quote, unquote remember earlier parts of a conversation, you must provide the earlier exchanges in the input to the model. And so we'll refer to this as `context`

```
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)
```

Building a OrderBot for a restaurant:

We can automate the collection of user prompts and assistant responses to build a OrderBot. The OrderBot will take orders at a pizza restaurant.

Transformer paper - Attention is all you need:

- <https://arxiv.org/pdf/1706.03762.pdf>