02 LLM Settings-Parameters -KirkYagami 🧑‍💻 🕵️

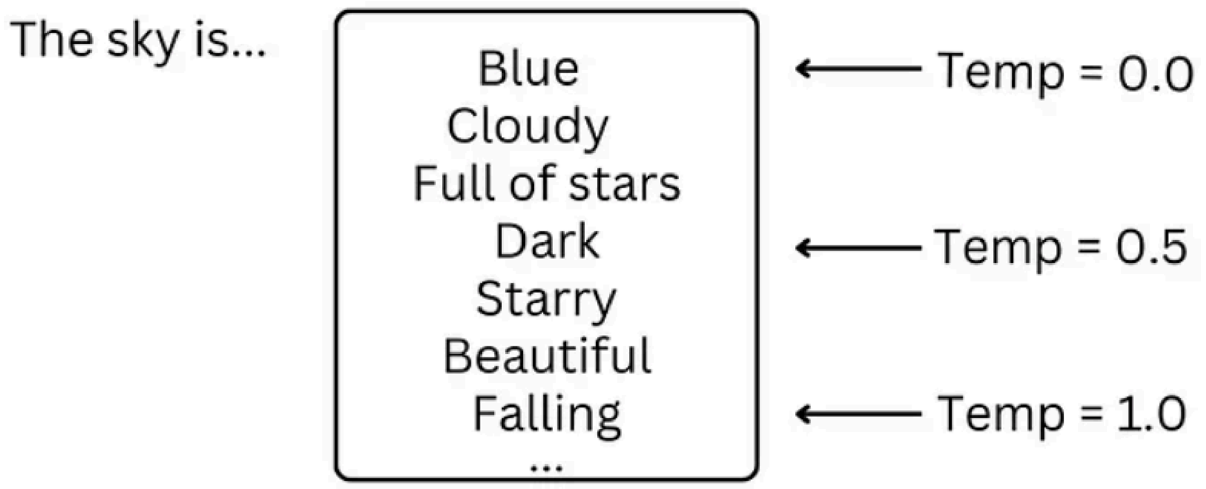
When designing and testing prompts, you typically interact with the LLM via an API. You can configure a few parameters to get different results for your prompts.

1. **Temperature** - In short, the lower the `temperature`, the more deterministic the results in the sense that the highest probable next token is always picked. Increasing temperature could lead to more randomness, which encourages more diverse or creative outputs. You are essentially increasing the weights of the other possible tokens.

    ```
    - Fact-Based QA - Lower `temp`
    - Poem Generation - Higher `temp`
    ```

    - Temperature = 0.1: The best way to learn programming is `to practice a lot and follow some online tutorials`.This is a very coherent and consistent text, but also very boring and predictable.
    - Temperature = 0.5: The best way to learn programming is `to experiment with different languages and frameworks`. This is a more diverse and interesting text, but still somewhat reasonable and logical.
    - Temperature = 1.0: The best way to learn programming is `to travel back in time and meet the inventors of programming languages`. This is a very diverse and unpredictable text, but also very imaginative and adventurous.

Tokens are sorted by the probability of their occurence



Lower Temp: model tends to pick from high tokens

Higher Temp: model tends to pick from low tokens

PYTHON

```python
from openai import OpenAI

# Initialize the OpenAI client with your API key
# It's recommended to set your API key as an environment variable (OPENAI_API_KEY)
# or load it securely.
client = OpenAI()

# Create a chat completion request
response = client.chat.completions.create(
    model="gpt-4o",  # Specify the desired model, e.g., "gpt-3.5-turbo", "gpt-4o"
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Tell me a fun fact about space."},
        {"role": "assistant", "content": "Did you know that a day on Venus is
longer than a year on Venus?"},
        {"role": "user", "content": "That's interesting! Why is that?"},
    ],
    max_tokens=150,  # Optional: Limit the length of the generated response
    temperature=0.7, # Optional: Control the randomness of the output (0.0 to 2.0)
)

# Extract and print the content of the assistant's response
print(response.choices[0].message.content)
```

PYTHON

```python
from google import genai
from google.genai import types

client = genai.Client()

response = client.models.generate_content(
    model="gemini-2.5-flash",
    contents=["Explain how AI works"],
    config=types.GenerateContentConfig(
        temperature=0.1
    )
)
print(response.text)
```

2. **The number of tokens:**

The number of tokens is a parameter that controls how long the generated text is. A higher number of tokens means more detail and information, while a lower number of tokens means more conciseness and simplicity.

To illustrate this, let's imagine a LLM generating some texts based on the same prompt: "What is artificial intelligence?". I will use different "number of tokens" values and see how the texts change. Here are some examples:

- Number of tokens = 10: What is artificial intelligence? **It is a field of computer science**. This is a very concise and simple text, but also very vague and incomplete.
- Number of tokens = 50: What is artificial intelligence? **It is a field of computer science that studies how to create machines and systems that can perform tasks that normally require human intelligence**. This is a more detailed and informative text, but still somewhat general and broad.
- Number of tokens = 100: What is artificial intelligence? **It is a field of computer science that studies how to create machines and systems that can perform tasks that normally require human intelligence, such as reasoning, learning, planning, decision making, natural language processing, computer vision, speech recognition, etc. Artificial intelligence can be divided into subfields based on the goals, methods, or applications of the research**. This is a very detailed and informative text, but also very specific and comprehensive.

As you can see, the "number of tokens" parameter can affect the length, detail, and information of the generated texts. You can use different "number of tokens" values depending on your purpose and preference.

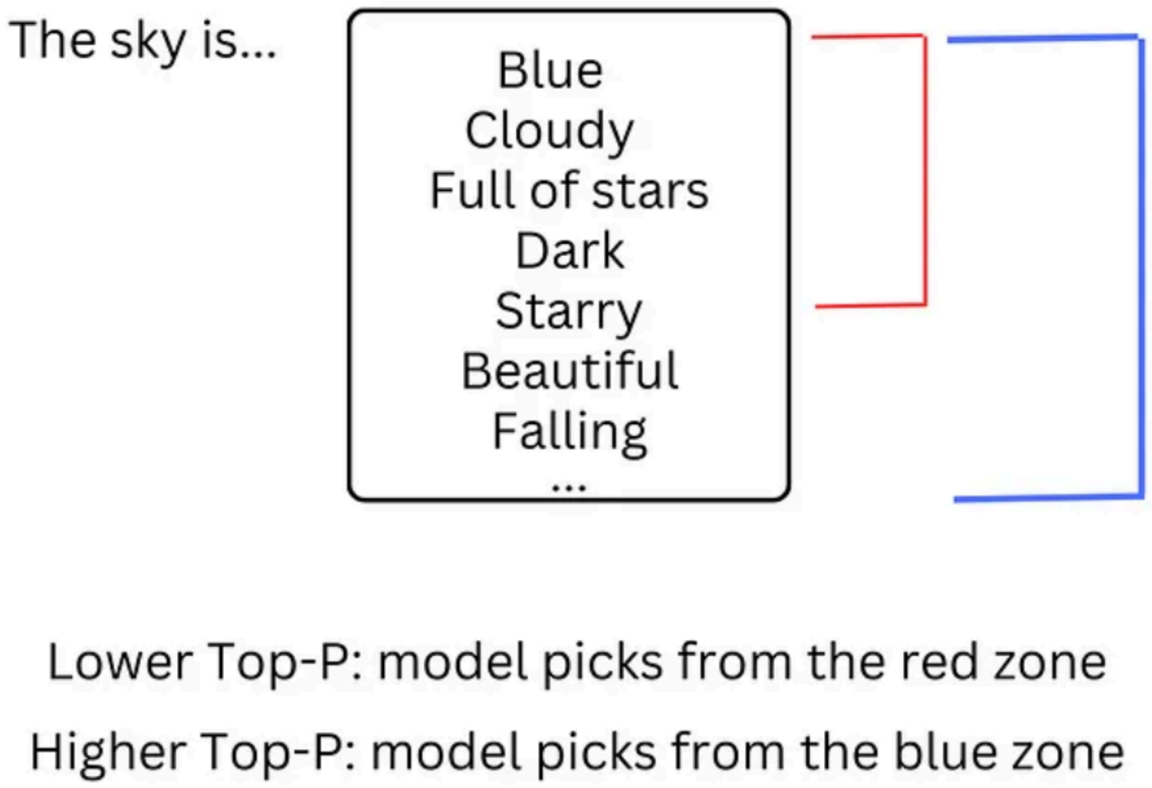
A lower "number of tokens" value is suitable for generating brief and uncomplicated texts. Alternatively, a higher "number of tokens" value will produce longer and more detailed texts. However, it is important to ensure that the "number of tokens" value is not set too low or too high, as this could result in irrelevant or redundant texts.

```python
from google import genai
from google.genai import types

client = genai.Client()

response = client.models.generate_content(
    model="gemini-2.5-flash",
    contents=["Explain how AI works"],
    config=types.GenerateContentConfig(
        maxOutputTokens=10
    )
)
print(response.text)
```

3. **Top P** - A sampling technique with temperature, called nucleus sampling, where you can control how deterministic the model is. If you are looking for exact and factual answers keep this low. If you are looking for more diverse responses, increase to a higher value. If you use Top P it means that only the tokens comprising the `top_p` probability mass are considered for responses, so a low `top_p`

value selects the most confident responses. This means that a high `top_p` value will enable the model to look at more possible words, including less likely ones, leading to more diverse outputs.

The general recommendation is to alter temperature or Top P but not both.


The sky is...
Blue
Cloudy
Full of stars
Dark
Starry
Beautiful
Falling
...
Lower Top-P: model picks from the red zone
Higher Top-P: model picks from the blue zone

4. **Max Length** - You can manage the number of tokens the model generates by adjusting the `max length`. Specifying a max length helps you prevent long or irrelevant responses and control costs.
5. **Stop Sequences** - A `stop sequence` is a string that stops the model from generating tokens. Specifying stop sequences is another way to control the length and structure of the model's response. For example, you can tell the model to generate lists that have no more than 10 items by adding "11" as a stop sequence.
6. **Frequency Penalty** - The `frequency penalty` applies a penalty on the next token proportional to how many times that token already appeared in the response and prompt. The higher the frequency penalty, the less likely a word will appear again. This setting reduces the repetition of words in the model's response by giving tokens that appear more a higher penalty.
   - Frequency penalty = 0.0: The best sport is soccer, **soccer is fun, and soccer is exciting.** This is a very repetitive and emphatic text, but also very boring and monotonous.
   - Frequency penalty = 0.5: The best sport is soccer, **soccer is fun and exciting.** This is a more varied and novel text with less repetitions, but still somewhat coherent and consistent.
   - Frequency penalty = 1.0: The best sport is soccer, **football is fun and exciting.** This is a very varied and novel text, but also very diverse and unpredictable, notice that it didn't repeat the word "soccer" present in the prompt.
7. **Presence Penalty** - The `presence penalty` also applies a penalty on repeated tokens but, unlike the frequency penalty, the penalty is the same for all repeated tokens. A token that appears twice

and a token that appears 10 times are penalized the same. This setting prevents the model from repeating phrases too often in its response. If you want the model to generate diverse or creative text, you might want to use a higher presence penalty. Or, if you need the model to stay focused, try using a lower presence penalty.

- Presence penalty = 0.0: The best sport is soccer, **soccer, soccer.** This is a very repetitive and emphatic text, but also very boring and monotonous.
- Presence penalty = 0.5: The best sport is **soccer, basketball, and tennis**. This is a more varied and novel text, but still somewhat coherent and consistent.
- Presence penalty = 1.0: The best sport is soccer, **rugby, and chess**. This is a very varied and novel text, but also very diverse and unpredictable.

Similar to `temperature` and `top_p`, the general recommendation is to alter the frequency or presence penalty but not both.