Analytics Vidhya

LLM Generation Parameters

Instructor

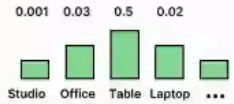
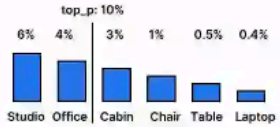
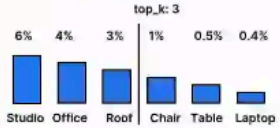
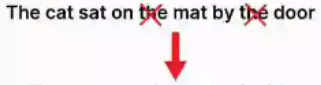
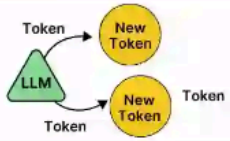
Dipanjan Sarkar

Head of Community & Principal AI Scientist at Analytics Vidhya

Google Developer Expert - ML & Cloud Champion Innovator

Published Author

7 LLM Generation Parameters

Parameters	Structure	Description	Range
max_tokens		Limits the number of tokens the model generates.	1 to ∞
temperature		Controls creativity; lower values = focused, higher values = more creative.	0 to 2
top_p		Sets the probability threshold for token diversity; considers predicting tokens whose probability adds up to top_p(higher = more variable)	0 to 1
top_k		Limits the number of top probable tokens considered when predicting the next token lower = more predictable, higher = more variable.	1 to ∞
frequency penalty		Reduces repeated tokens encouraging more unique and diverse tokens in the response	-2 to 2
presence penalty		Discourages reuse of already-present tokens and forces more generation of new tokens	-2 to 2
stop	Capital of India is Delhi [.]	Specifies when the model should stop generating further content.	Custom list of token identifiers

Note: *max_token value is now deprecated in favor of max_completion_tokens, and is not compatible with o1 series models.*

Max Tokens

The **max_tokens** parameter controls the length of the output generated by the model.

- A "token" can be as short as one character or as long as one word.
- By setting an appropriate **max_tokens** value, you can control whether the response is a quick snippet or an in-depth explanation.
- **Max_token** value is now deprecated in favor of **max_completion_tokens** (in OpenAI API).

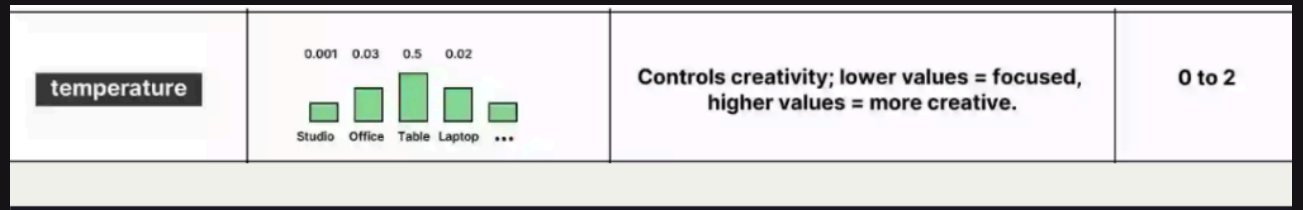

```
import openai
client = openai.OpenAI(api_key='Your_api_key')
max_tokens=10
temperature=0.5
response = client.chat.completions.create(
          model="gpt-4o",
          messages=[
              {"role": "user",
               "content": "What is the capital of India? Give 7 places to Visit"}
          ],
          max_tokens=max_tokens,
          temperature=temperature,
          n=1,
          )
print(response.choices[0].message.content)
```

Analytics Vidhya

Temperature

The **temperature** parameter influences how deterministic or random and creative the model's responses are.

- It's essentially a measure of how deterministic the responses should be:
  - **Low Temperature (e.g., 0.1)**: The model will produce more focused and predictable responses.
  - **High Temperature (e.g., 0.9)**: The model will produce more creative, varied, or even "wild" responses.
- Use low temperatures for tasks like generating technical answers, where precision matters, and higher temperatures for creative content generation tasks.

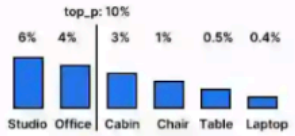



```
import openai
client = openai.OpenAI(api_key=api_key)
max_tokens=500
temperature=0.1
response = client.chat.completions.create(
            model="gpt-4o",
            messages=[
                {"role": "user",
                 "content": "What is the capital of India? Give 7 places to Visit"}
            ],
            max_tokens=max_tokens,
            temperature=temperature,
            n=1,
            stop=None
        )
print(response.choices[0].message.content)
```

Top-p - Nucleus Sampling

The **top_p** parameter, also known as nucleus sampling, helps control the diversity of responses.

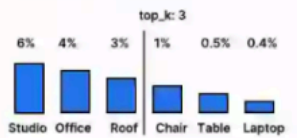
- It sets a threshold for the cumulative probability distribution of the next token generation choices:
  - **Low Value (e.g., 0.1)**: The model will only consider the top 10% of possible next tokens, limiting variation.
  - **High Value (e.g., 0.9)**: The model considers a wider range of possible next tokens (summing up to 90%), increasing variability.

top_p		Sets the probability threshold for token diversity; considers predicting tokens whose probability adds up to top_p(higher = more variable)	0 to 1

```
import openai
client = openai.OpenAI(api_key=api_key)
max_tokens=500
temperature=0.1
top_p=0.5
response = client.chat.completions.create(
          model="gpt-4o",
          messages=[
              {"role": "user",
               "content": "What is the capital of India? Give 7 places to Visit"}
          ],
          max_tokens=max_tokens,
          temperature=temperature,
          n=1,
          top_p=top_p,
          stop=None
      )
print(response.choices[0].message.content)
```

Top-k - Token Sampling

- The **top_k** parameter limits the model to only considering the top k most probable next tokens when predicting (generating) the next word.
  - **Low Value (e.g., 10)**: Limits the model to more predictable and constrained responses.
  - **High Value (e.g., 100)**: Allows the model to consider a larger number of tokens, increasing the variety of responses.

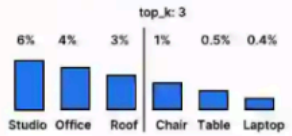
top_k		Limits the number of top probable tokens considered when predicting the next token lower = more predictable, higher = more variable.	1 to ∞

```
# set top_k to 50
sample_output = model.generate(
    **model_inputs,
    max_new_tokens=40,
    do_sample=True,
    top_k=50
)

print("Output:\n" + 100 * '-')
print(tokenizer.decode(sample_output[0], skip_special_tokens=True)
```

Top-k - Token Sampling

- The **top_k** parameter isn't directly available in the OpenAI API but is available in other platforms like Hugging Face transformers.

top_k		Limits the number of top probable tokens considered when predicting the next token lower = more predictable, higher = more variable.	1 to ∞

```
# set top_k to 50
sample_output = model.generate(
    **model_inputs,
    max_new_tokens=40,
    do_sample=True,
    top_k=50
)

print("Output:\n" + 100 * '-')
print(tokenizer.decode(sample_output[0], skip_special_tokens=True)
```

Analytics Vidhya

Frequency Penalty

The **frequency_penalty** parameter discourages the model from repeating previously used words.

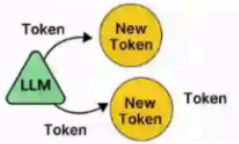
- It reduces the probability of tokens that have already appeared in the output.
  - **Low Value (e.g., 0.0)**: The model won't penalize for repetition.
  - **High Value (e.g., 2.0)**: The model will heavily penalize repeated words, encouraging the generation of new content.
- This is useful when you want the model to avoid repetitive outputs, like in creative writing, where redundancy might diminish quality.

frequency penalty	The cat sat on the mat by the door ↓ The cat rested on a rug inside	Reduces repeated tokens encouraging more unique and diverse tokens in the response	-2 to 2

```
import openai
# Initialize the OpenAI client with your API key
client = openai.OpenAI(api_key='Your_api_key')
max_tokens = 500
temperature = 0.1
top_p=0.25
frequency_penalty=1
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "user", "content": "What is the capital of India? Give 7 places to Visit"}
    ],
    max_tokens=max_tokens,
    temperature=temperature,
    n=1,
    top_p=top_p,
frequency_penalty=frequency_penalty,
                stop=None
   )
print(response.choices[0].message.content)
```

Presence Penalty

- Similar to the frequency penalty, but instead of penalizing based on how often a word is used, it penalizes based on whether a word has appeared at all in the response so far.
  - **Low Value (e.g., 0.0)**: The model won't penalize for reusing words.
  - **High Value (e.g., 2.0)**: The model will avoid using any word that has already appeared.
- Presence penalty helps encourage more diverse content generation.

presence penalty		Discourages reuse of already-present tokens and forces more generation of new tokens	-2 to 2

```python
import openai
# Initialize the OpenAI client with your API key
client = openai.OpenAI(api_key='Your_api_key')
# Define parameters for the chat request
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "user",
            "content": "What is the capital of India? Give 7 places to visit."
        }
    ],
    max_tokens=500,    # Max tokens for the response
    temperature=0.1,   # Controls randomness
    top_p=0.1,         # Controls diversity of responses
    presence_penalty=0.5,  # Encourages the introduction of new ideas
    n=1,               # Generate only 1 completion
    stop=None          # Stop sequence, none in this case
)
print(response.choices[0].message.content)
```

Stop Sequence

The **stop** parameter defines a sequence of characters or words that signals the model to stop generating future content.

- This allows you to cleanly end the generation at a specific point.
  - **Example Stop Sequences**: Could be periods (.), newlines (\n), or specific delimiter like "<eot>"
- Useful especially if you teach the model to generate content until a specific special token when fine-tuning.

stop	Capital of India is Delhi [.]	Specifies when the model should stop generating further content.	Custom list of token identifiers

```python
import openai
# Initialize the OpenAI client with your API key
client = openai.OpenAI(api_key='Your_api_key')
max_tokens = 500
temperature = 0.1
top_p = 0.1
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "user", "content": "What is the capital of India? Give 7 places to Visit"}
    ],
    max_tokens=max_tokens,
    temperature=temperature,
    n=1,
    top_p=top_p,
    stop=[".", "End of list"]  # Define stop sequences
)
print(response.choices[0].message.content)
```

Thank You

Analytics Vidhya