LangChain LLM/ChatModel Parameters Cheat Sheet

Key parameters for controlling model behavior and output

Overview

- When calling LLMs or ChatModels in LangChain, parameters allow fine control over output.
- Common parameters include: `temperature`, `max_completion_tokens`, `top_p`, `stop`, `presence_penalty`, and `frequency_penalty`.
- Proper tuning improves output quality, creativity, and cost-efficiency.

1. Temperature

- Controls randomness of model output.
- Range: 0.0 (deterministic) to 1.0+ (creative/diverse).
- Use cases:
  - 0-0.3: Factual answers, code generation
  - 0.4-0.7: Balanced creativity and reliability
  - 0.8-1.0: Storytelling, brainstorming, poetry

2. Max Completion Tokens (`max_completion_tokens`)

- Limits the number of tokens the model generates in a single call.
- Helps manage cost and prevent overly long outputs.
- Recommendations:
  - Short responses: 50-100 tokens
  - Medium responses: 200-400 tokens
  - Long-form generation: 500+ tokens (depending on model context length)

3. Top-p (Nucleus Sampling)

- Controls diversity via probability mass.
- Only considers tokens whose cumulative probability mass is less than `top_p`.
- Range: 0.0 to 1.0
- Lower values = safer output; higher values = more creative/diverse.

4. Stop Sequences (`stop`)

- Defines token(s) at which the model should stop generating text.
- Useful to end responses at a specific point or avoid unwanted text.
- Can be a string or a list of strings.

5. Presence Penalty (`presence_penalty`)

- Encourages the model to introduce new topics or ideas.
- Positive value: discourages repetition of previously mentioned concepts.
- Range typically: 0.0 to 2.0

6. Frequency Penalty (`frequency_penalty`)

- Reduces likelihood of repeating the same tokens/words.
- Positive value: penalizes repeated token usage.
- Range typically: 0.0 to 2.0

7. LangChain Example

Python Code Example

```
from langchain_google_genai import ChatGoogleGenerativeAI

model = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash",
    temperature=0.7,            # randomness
    max_completion_tokens=150,   # output length
    top_p=0.9,                  # nucleus sampling
    stop=["\n"],                # stop sequence
    presence_penalty=0.5,       # encourage new topics
    frequency_penalty=0.3       # reduce repetition
)

prompt = "Write a short poem about Bangladesh in 2 lines."
result = model.invoke(prompt)

print(result.content)
```