Towards AI · Follow publication

Member-only story

# The 7 Secret Knobs That Control Every AI Response

Temperature, Top-P, and Max Tokens — the hidden parameters behind every ChatGPT answer. Learn how to master them all.

8 min read · Oct 19, 2025

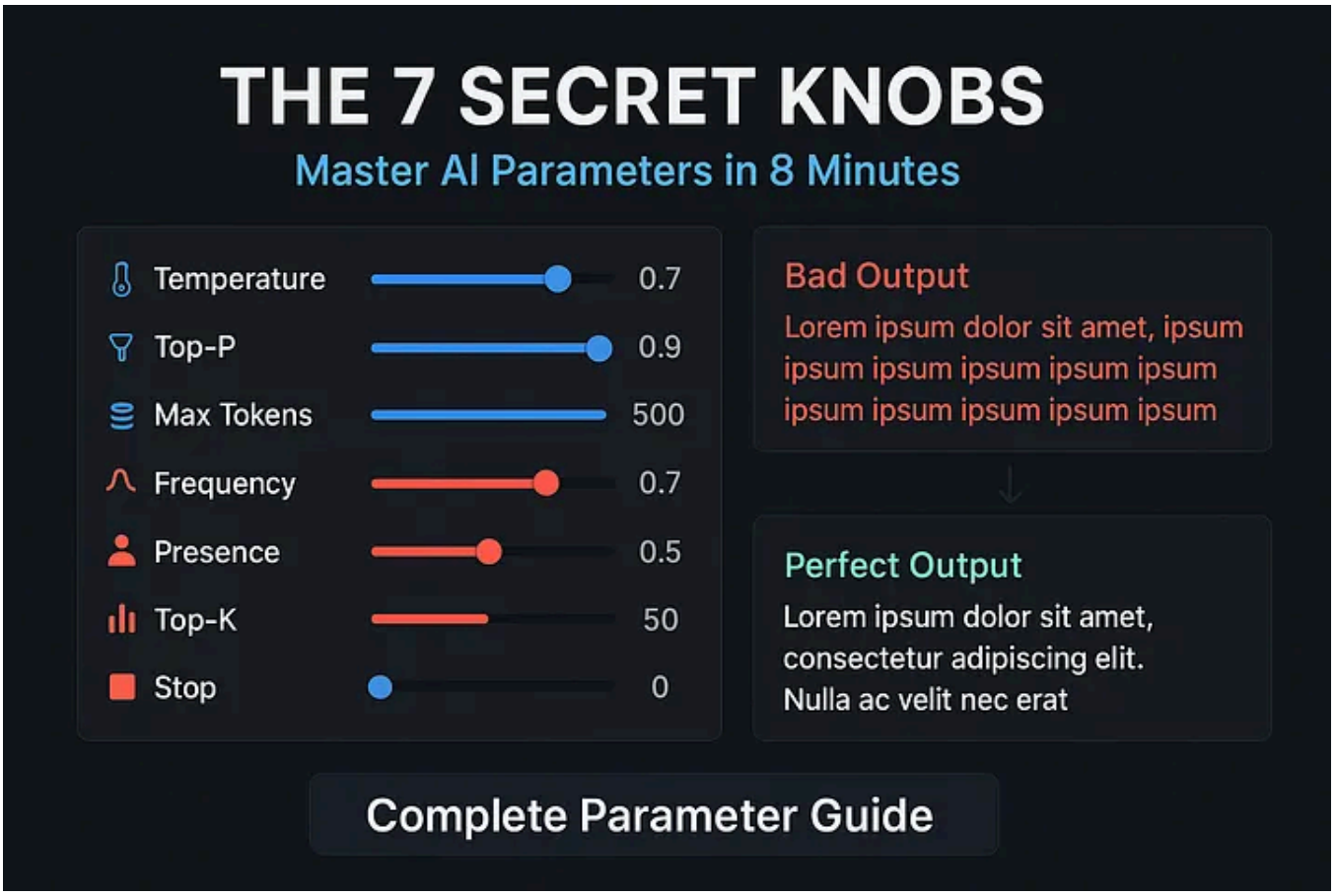
Rohan Mistry Follow

Listen Share More

Every time you hit “send” to ChatGPT, Claude, or any LLM, seven invisible parameters are silently shaping the response. Change one number, and you go from genius insights to nonsensical rambling.

Most people never touch these settings. They stick with defaults and wonder why AI sometimes feels “dumb.” Master these 7 parameters, and you’ll get better outputs than 99% of users.

Non-members? Read this story free with this link 👉 Non-members link




Source: Image by author.

## Table of Contents



. . .

## The Probability Machine You're Actually Using

Here's what really happens when you ask AI a question:

**You type:** “Write a product description”

**What actually happens:**

1. Your text becomes tokens (numbers)
2. Model calculates probability for every possible next token
3. Seven parameters decide which token gets chosen
4. Repeat thousands of times until done

**You’re not having a conversation. You’re configuring a sampling algorithm.**

Let me show you each knob.

. . .

## Parameter 1: Max Tokens (The Length Controller)

**What it does:** Hard limit on how many tokens (≈words) the model can generate.

**Critical distinction:**

- Context window = Total input + output capacity
- Max tokens = Output-only limit

**Why it matters:**

**Cost control:**

```
GPT-4 Turbo:
500 tokens = $0.015
4,000 tokens = $0.12
→ 8× cost difference
```

**Speed:**

```
500 tokens: 10-25 seconds
2,000 tokens: 40-100 seconds
```

**Practical values:**

- Chat: 300–500 tokens
- Code: 1,000–2,000
- Articles: 2,000–4,000
- JSON: 50–200

**Pro tip:** Always set this. Default unlimited wastes money.

. . .

## Parameter 2: Temperature (The Creativity Dial)

**What it does:** Scales logits before sampling. Controls randomness.

**The effect:**

**Temperature = 0.1** (Deterministic)

```
"good": 85% → Always picks this
"great": 10%
"excellent": 4%
```

**Temperature = 1.5** (Creative)

```
"good": 35%
"great": 30% → Much more variety
"excellent": 20%
"bad": 15%
```

**When to use what:**

**0.0–0.3** (Factual tasks)

- ✅ Code generation
- ✅ Data extraction
- ✅ Translation
- ✅ Classification

**0.7–1.0** (Balanced)

- ✅ General conversation
- ✅ Explanations
- ✅ Default for most tasks

**1.2–2.0** (Creative)

- ✅ Creative writing
- ✅ Brainstorming
- ✅ Story generation

**Danger zone:** Temperature > 2.0 often produces nonsense.

**Sweet spot:** 0.7 for most tasks.

. . .

### Parameter 3: Top-P / Nucleus Sampling (The Quality Filter)

**What it does:** Sample only from tokens whose cumulative probability ≥ P. Cuts off the "long tail."

**How it works:**

**Without Top-P:**

```
Every token (including absurd ones) is a candidate
```

**With Top-P = 0.9:**

```
Cumulative probability:
"the": 25% (total: 25%)
"a": 20% (total: 45%)
"an": 15% (total: 60%)
"this": 10% (total: 70%)
"that": 8% (total: 78%)
"those": 7% (total: 85%)
"these": 5% (total: 90%) ← STOP
```

```
Only sample from these top tokens
Ignore the nonsensical tail
```

**Why this matters:** Prevents neural text degeneration (repetition, rambling).

**Practical values:**

- **1.0** = No filtering (risky)
- **0.95** = Light filtering (creative)
- **0.9** = **Recommended default**
- **0.5** = Heavy filtering (conservative)

**Key insight:** Anthropic advises tuning EITHER temperature OR Top-P, not both (they compound).

**Recommendation:** Fix Top-P at 0.9, vary temperature.

. . .

## Parameter 4: Top-K (The Candidate Limiter)

**What it does:** Only consider the K highest-probability tokens at each step.

**Difference from Top-P:**

- Top-K = Fixed number of tokens
- Top-P = Variable number based on probability mass

**Top-K issues:**

- Sometimes K is too many (when answer is obvious)
- Sometimes K is too few (when multiple good options)

**Top-P advantage:** Adaptive to the situation.

**Modern recommendation:** Use Top-P instead of Top-K.

Most APIs (OpenAI, Anthropic) default to Top-P and ignore Top-K.

. . .

## Parameter 5: Frequency Penalty (The Repetition Killer)

**What it does:** Reduces probability proportional to how often tokens appeared.

**Formula:**

```
modified_logit = original_logit - (frequency_penalty × token_count)
```

**The problem it solves:**

**Without frequency penalty:**

```
"The best product is the best choice.
The best quality ensures the best results..."
```

**With frequency penalty = 0.8:**

```
"The best product offers superior quality.
Its innovative design ensures excellent results..."
```

**When to use:**

- Long-form generation (articles, stories)
- Creative writing
- List generation
- Any time you see repetition

**When NOT to use:**

- Technical writing (terms should repeat)
- Code generation
- Structured output (JSON/XML)

**Practical values:**

- **0.0** = Default (no penalty)
- **0.3–0.7** = Mild variety
- **0.8–1.5** = Strong variety

**Recommended default: 0.0** (increase only if needed)

. . .

## Parameter 6: Presence Penalty (The Novelty Driver)

**What it does:** Penalizes tokens that appeared AT LEAST ONCE (binary, not proportional).

**Key difference from frequency:**

- Frequency = Penalizes based on count
- Presence = Binary (used = penalized equally)

**Effect:**

**With presence penalty = 1.0:**

```
Already used: "cat", "sat", "mat"
All get -1.0 penalty
Model heavily favors NEW words
```

**Frequency vs Presence:**

**Frequency penalty = 0.8:**

```
"This durable phone features excellent battery.
The display provides clarity. Camera captures photos."
→ Varies WORDS, stays on TOPIC
```

**Presence penalty = 0.8:**

```
"This durable phone features battery life.
The company makes laptops. Headquarters in California..."
→ Introduces NEW TOPICS (drifts away)
```

**When to use:**

- Brainstorming
- Exploratory writing
- Avoiding “getting stuck”

**When NOT to use:**

- Focused explanations
- Consistency matters
- Technical content

**Practical values:**

- **0.0** = Default
- **0.3–0.7** = Mild novelty
- **0.8–1.5** = Strong novelty

. . .

## Parameter 7: Stop Sequences (The Emergency Brake)

**What it does:** Forces model to halt when specific strings appear. Stop text is NOT included in output.

**Example:**

```
prompt = "List 3 benefits:\n1."
stop_sequences = ["\n4.", "\n\n"]
```

```
Output:
"1. Improves health
2. Boosts energy
3. Increases focus"
[Stops at "\n4."]
```

**Why this is critical:**

**JSON generation:**

```
stop_sequences = ["}"]
```

```
→ Stops exactly after closing brace
```

**Section boundaries:**

```
stop_sequences = ["###", "\n\nReviews:"]
→ Prevents unwanted sections
```

**Design tips:**

- Use unambiguous delimiters: `"<|END|>"`, `"\n---\n"`
- Multiple sequences for robustness
- Pair with max_tokens (belt-and-suspenders)

**Common stop sequences:**

- JSON: `"}"`, `"]"`
- Code: `"```"`
- Lists: `"\n\n"`
- Sections: `"###"`, `"---"`
- Chat: `"User:"`, `"Human:"`

. . .

## How Parameters Interact (The Critical Part)

**Most guides miss this: Parameters compound.**

### Interaction 1: Temperature × Top-P

```
Temperature = 1.5 (flattens distribution)
Top-P = 0.95 (keeps 95%)
```

```
Effect: Controlled creativity
Temperature spreads probability → Top-P crops tail
```

**Rule:** High temperature + Low Top-P = Controlled creativity

### Interaction 2: Penalties × Temperature

```
Temperature = 0.7
Frequency penalty = 1.0
```

```
Effect:
1. Temperature sets initial distribution
2. Penalties modify logits
3. Distribution shifts away from used tokens
→ Increasingly novel vocabulary
```

**Danger:** High temperature + High penalties = drift into gibberish

### Interaction 3: Both Penalties Together

```
Frequency penalty = 0.5
Presence penalty = 0.3
→ Variety without chaos
```

```
VS

Frequency = 1.5
Presence = 1.5
→ Often incoherent
```

**Rule:** If using both, keep combined total < 2.0

• • •

## Real Scenarios: What Settings to Use When

### Customer Support Chatbot

**Requirements:** Accurate, consistent, fast

```
{
    "max_tokens": 300,
    "temperature": 0.3,
    "top_p": 0.9,
    "frequency_penalty": 0.2,
    "presence_penalty": 0.0,
    "stop": ["\n\nUser:"]
}
```

**Why:** Low temp = factual, light frequency = natural language

. . .

### Creative Story Writing

**Requirements:** Imaginative, varied, engaging

```
{
    "max_tokens": 3000,
    "temperature": 1.2,
    "top_p": 0.95,
    "frequency_penalty": 0.7,
    "presence_penalty": 0.4,
    "stop": ["### THE END"]
}
```

**Why:** High temp = creative, penalties = variety

. . .

### Code Generation

**Requirements:** Correct syntax, complete functions

```
{
    "max_tokens": 1500,
    "temperature": 0.2,
    "top_p": 0.9,
    "frequency_penalty": 0.0,
    "presence_penalty": 0.0,
    "stop": ["```", "\n\n#"]
}
```

**Why:** Very low temp = correct, no penalties = terms repeat properly

. . .

### Data Extraction (JSON)

**Requirements:** Strict format, deterministic

```
{
    "max_tokens": 200,
    "temperature": 0.0,
    "top_p": 1.0,
    "frequency_penalty": 0.0,
    "presence_penalty": 0.0,
    "stop": ["}"]
}
```

**Why:** Temp 0 = identical every time, stop at closing brace

. . .

## The Cheat Sheet You'll Actually Use

### Quick Decision Guide

**Task Type:**

- **Factual/Technical** → temp: 0.0–0.3, penalties: 0.0
- **General Chat** → temp: 0.7–0.9, freq: 0.2–0.4

- **Creative** → temp: 1.2–1.5, freq: 0.6–0.8, presence: 0.4–1.0

**Output Length:**

- **Short** (chat) → 200–500
- **Medium** (descriptions) → 500–1,000
- **Long** (articles) → 1,000–4,000

**Format Critical?**

- **Yes** (JSON) → temp: 0.0, stop: format delimiters
- **No** (natural text) → stop: section markers

**Repetition Problem?**

- **Yes** → frequency_penalty: 0.5–0.8
- **No** → frequency_penalty: 0.0

## The "Start Here" Defaults

```
{
    "max_tokens": 500,
    "temperature": 0.7,
    "top_p": 0.9,
    "frequency_penalty": 0.0,
    "presence_penalty": 0.0,
    "stop": []
}
```

**Then adjust ONE parameter at a time.**

. . .

## The Golden Rules

1. **Always set max_tokens** (control cost & length)
2. **Temperature is your primary dial** (0–0.3 factual, 0.7–1.0 balanced, 1.2+ creative)

3. **Keep Top-P at 0.9** unless specific reason
4. **Start penalties at 0.0** (increase only when seeing issues)
5. **Define stop sequences** for structured output
6. **Tune ONE parameter at a time** (test, measure, iterate)
7. **Different tasks need different configs** (code ≠ creative ≠ chat)

. . .

## Common Mistakes to Avoid

**Mistake 1:** Setting everything high

```
# DON'T DO THIS
{"temperature": 1.8, "top_p": 1.0,
 "frequency_penalty": 1.5, "presence_penalty": 1.5}
→ Incoherent nonsense
```

**Mistake 2:** Ignoring max_tokens

```
# DON'T DO THIS
{"max_tokens": None}
→ Unpredictable costs, slow responses
```

**Mistake 3:** Penalties on technical content

```
# DON'T DO THIS for code
{"frequency_penalty": 0.8}
→ Variable names change mid-code
```

**Mistake 4:** No stop sequences

```
# DON'T DO THIS for JSON
{"stop": []}
→ Continues past closing brace
```

. . .

## The Bottom Line

**You're not just "using AI." You're programming a probability sampling system.**

These 7 parameters are your programming interface:

1. **Max Tokens** = Length limit
2. **Temperature** = Creativity dial
3. **Top-P** = Quality filter
4. **Top-K** = Candidate limiter (use Top-P instead)
5. **Frequency Penalty** = Repetition killer
6. **Presence Penalty** = Novelty driver
7. **Stop Sequences** = Emergency brake

**Master these, and you'll get 10× better outputs than users who don't.**

**The meta-lesson:** Same prompt + different parameters = completely different results.

Now you know the control panel. Time to experiment.

. . .

## Your Next Steps

**Beginner?**

- Start with the defaults above

- Experiment with temperature first
- Practice on different tasks

**Intermediate?**

- A/B test on your actual use cases
- Measure impact of each parameter
- Build task-specific configs

**Advanced?**

- Dynamic parameter selection
- Track metrics and iterate
- Share what you learn

. . .

## Quick Reference Card

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
PARAMETER QUICK REFERENCE
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

```
MAX TOKENS
├─ Chat: 300-500
├─ Code: 1000-2000
└─ Articles: 2000-4000
TEMPERATURE
├─ Factual: 0.0-0.3
├─ Balanced: 0.7-0.9
└─ Creative: 1.2-1.5
TOP-P (Default: 0.9)
├─ Conservative: 0.5-0.7
└─ Exploratory: 0.95-1.0
FREQUENCY PENALTY (Default: 0.0)
└─ If repetition: 0.3-0.8
PRESENCE PENALTY (Default: 0.0)
```

```
└─ If stuck on topic: 0.3-1.0
STOP SEQUENCES
├─ JSON: "}", "]"
├─ Code: "```"
└─ Custom: "<|END|>", "###"
═══════════════════════════════════
GOLDEN RULES
═══════════════════════════════════
1. Always set max_tokens
2. Tune temp OR top_p (not both)
3. Start penalties at 0.0
4. One parameter change at a time
5. Test, measure, iterate
═══════════════════════════════════
```

. . .

**Most AI tutorials teach you prompts. This taught you the control panel.**

Prompts are instructions. Parameters are the execution environment.

**Same prompt + different parameters = completely different results.**

Now go configure some probability distributions. 🎛️

. . .

*Found this helpful? Share it with someone frustrated with AI outputs.*

**Connect:** LinkedIn | GitHub

Artificial Intelligence ChatGPT Large Language Models Llm Ai Prompt Engineering

Follow

## Published in Towards AI

89K followers · Last published just now

Making AI accessible to 100K+ learners. Find the most practical, hands-on and comprehensive AI Engineering and AI for Work certifications at academy.towardsai.net - we have pathways for any experience level. Monthly cohorts still open—use COHORT10 for 10% off!

Follow

## Written by Rohan Mistry

145 followers · 22 following

I am a Master's student in Artificial Intelligence and Machine Learning, focused on real-world AI solutions.

## Responses (1)

Bgerby

What are your thoughts?

Fred
2 days ago

Thank you!

4 Reply

## More from Rohan Mistry and Towards AI

In Python in Plain English by Rohan Mistry

### 🐍 Python 3.14 — The Most Curious Python Yet

Hidden gems, new runtime experiments, and why this release quietly changes everything.

Oct 14 111 2

In Towards AI by Ashish Abraham

## No Libraries, No Shortcuts: LLM from Scratch with PyTorch

The no BS guide to build, train, and fine-tune a Transformer architecture from scratch

Oct 2 1.1K 14

In Towards AI by Devi

## Running Small Language Models (SLMs) on CPUs: A Practical Guide

The what, why, and how with a practical example, to solidify your learning!

Sep 29 271 4

In Towards AI by Rohan Mistry

## Your Model Has 95% Accuracy. It's Completely Useless.

You Built a Model That Predicts Everything as "No." It Has 95% Accuracy. You Just Shipped Garbage.

Oct 9 47 2

See all from Rohan Mistry

See all from Towards AI

## Recommended from Medium

In Artificial Intelligence in Plain English by DrSwarnenduAI

## The Superintelligence Letter: What Nobody's Telling You

Disclaimer: This is my understanding of what superintelligence means and why people are concerned. I might be wrong. I probably am missing...

4d ago 57 10

Will Lockett

## You Have No Idea How Screwed OpenAI Actually Is

When you find yourself in a hole, at what point do you stop digging?

4d ago 4.96K 148

Shirley Li

## MatFormer: Train Once and Deploy Many

The secret weapon of Gemma 3n to let one model fit all.

5d ago 47 2

In AI Software Engineer by Joe Njenga

## This Viral DeepSeek OCR Model Is Changing How LLMs Work

This DeepSeek OCR model hit an overnight success not seen in any other release—4k+ GitHub stars in less than 24 hours and more than 100k...

5d ago 208 4

In Generative AI by Avijnan Chatterjee

## This Free AI Tool Does What ChatGPT Can't

Discover Google AI Studio's hidden features: system prompts, Compare Mode, and real-time screen sharing. Free tutorial for analyzing...

Oct 19 134

In Level Up Coding by Vivedha Elango

## Your AI App Has Security Holes You Can't See (Here's How to Find Them)

Your Guide to Red teaming AI Apps with Comparison of open-source red teaming libraries and Implementation Examples

4d ago 251 7

See more recommendations