LLM Decoding

AI with Deep Learning
EE4016

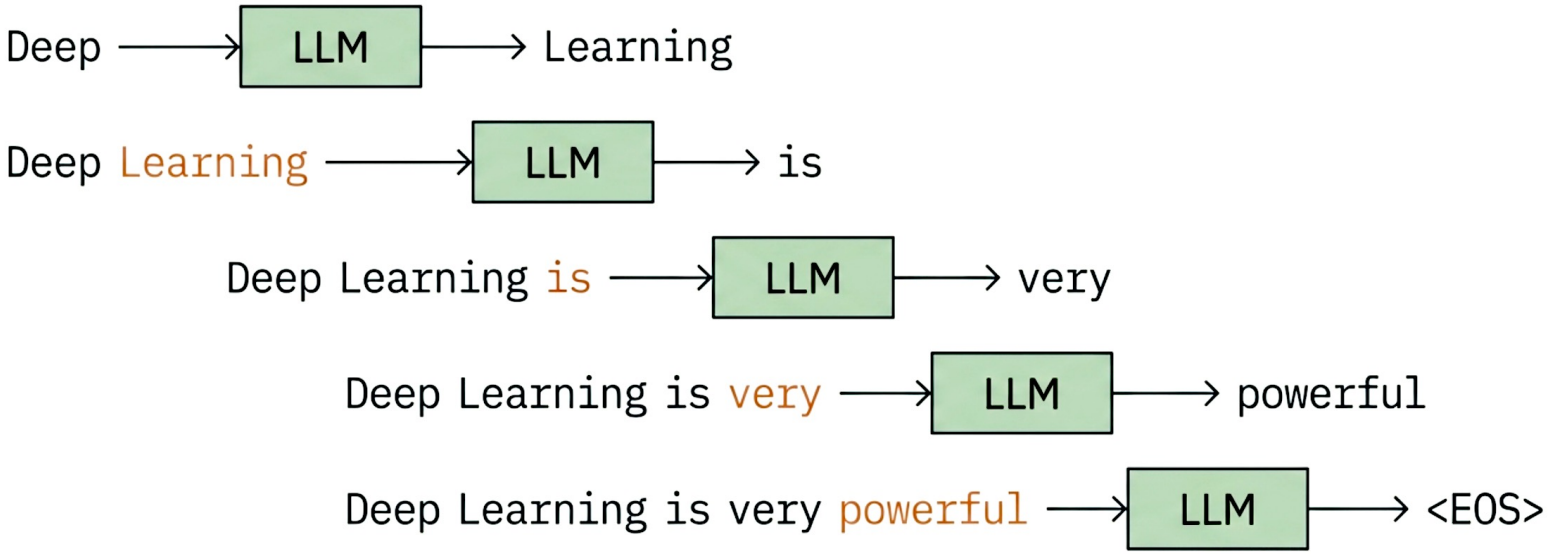
Prof. Lai-Man Po
Department of Electrical Engineering
City University of Hong Kong

Content

- **LLM Decoding Strategies**
  - Greedy Search, Beam Search, Pure Sampling, Temperature, Top-k, Top-p
- **Basic Prompt Engineering**
- **Advanced Prompt Engineering**
  - Chain-of-Though (CoT)
  - Zero-Shot CoT
  - Automatic Prompt Engineer (APE)
  - Emotional Blackmail Prompt

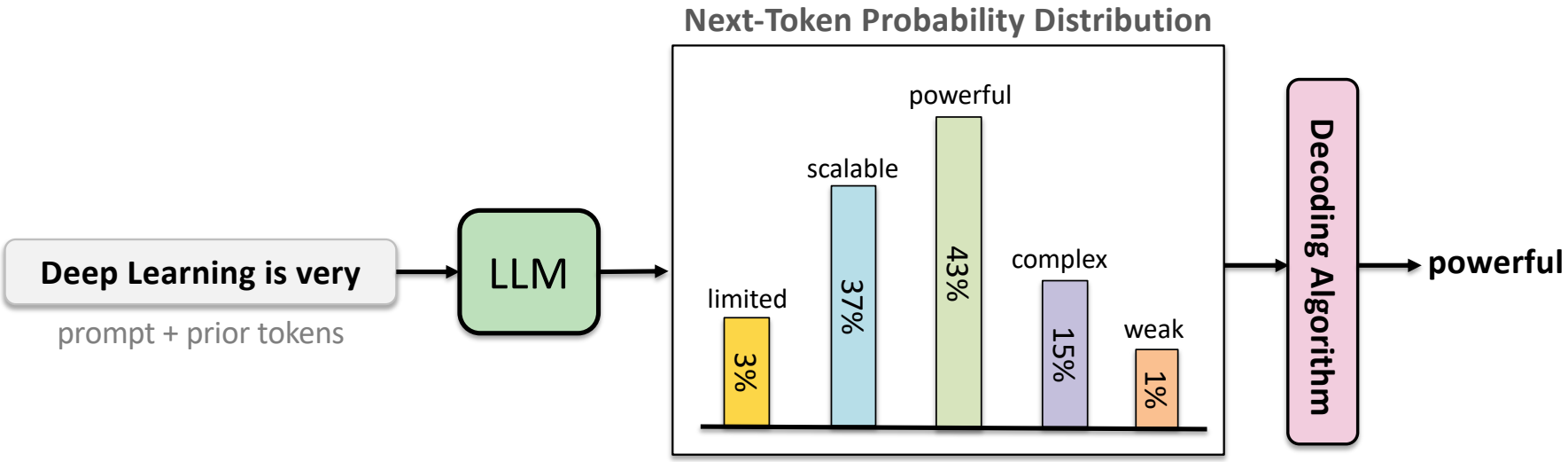
The Autoregressive Engine

Large Language Models construct sequences sequentially. At each step, the model predicts the next token based on all previously generated tokens, appending it to the context and repeating the process until reaching an End-of-Sentence (EOS) token or maximum length.



Next Token Prediction of LLMs

- An LLM functions as a probabilistic model that estimates the **conditional probability distribution** $P(w_t|w_1, w_2, \dots, w_{t-1})$ of the next token w_t , given the preceding sequence of tokens $(w_1, w_2, \dots, w_{t-1})$.
- During text generation, LLMs determine the next output token using **decoding algorithms.**



https://medium.com/@lmpo/mastering-llms-a-guide-to-decoding-algorithms-c90a48fd167b

Decoding Algorithms in LLMs

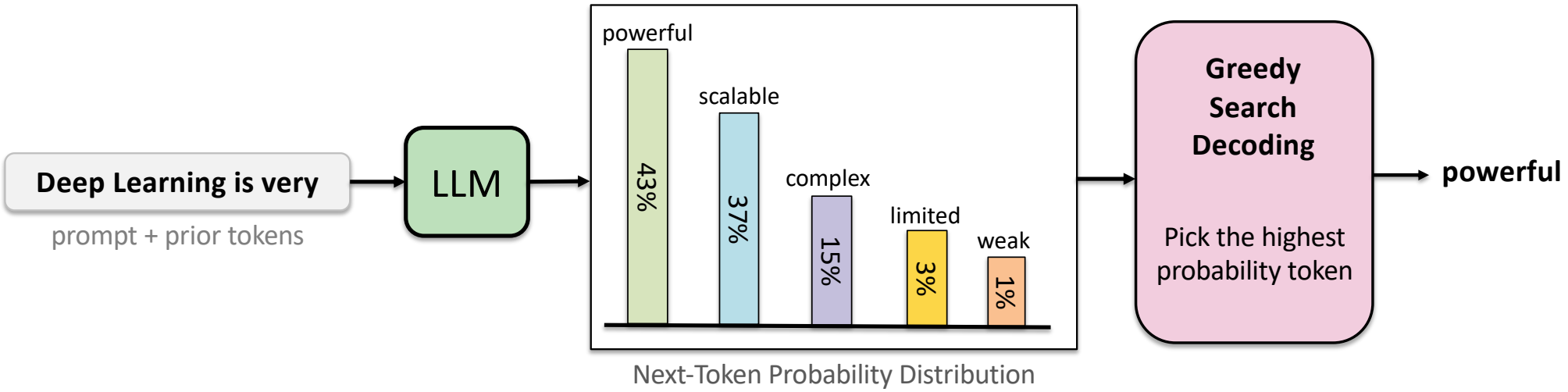
- Decoding algorithms are essential for converting the probability distributions generated by LLMs into coherent and meaningful text.
- The choice of decoding algorithm significantly influences the quality, diversity, and coherence of the generated text.
- Below, we delve into some of the most well-known decoding algorithms.

  - **Greedy Search Decoding**
  - **Beam Search Decoding**
  - **Sampling-based Methods**
    - Pure sampling
    - Temperature
    - Top-k sampling
    - Top-p (nucleus) sampling
  - **Generation Controlling**
    - Frequency Penalty
    - Presence Penalty

Greedy Search Decoding

Greedy Search decoding is the simplest and most straightforward decoding algorithm. At each step, it **selects the word with the highest probability**:

$$\hat{w}_t = \max_{w_t} P(w_t | w_1, w_2, \ldots, w_{t-1})$$



The Fatal Flaw of Greedy Exploitation

By failing to explore alternative paths, locally optimal choices quickly lead to global monotony.

Input: "I love to eat"
Greedy Search Output: "I love to eat food food food because food is food and food is great"

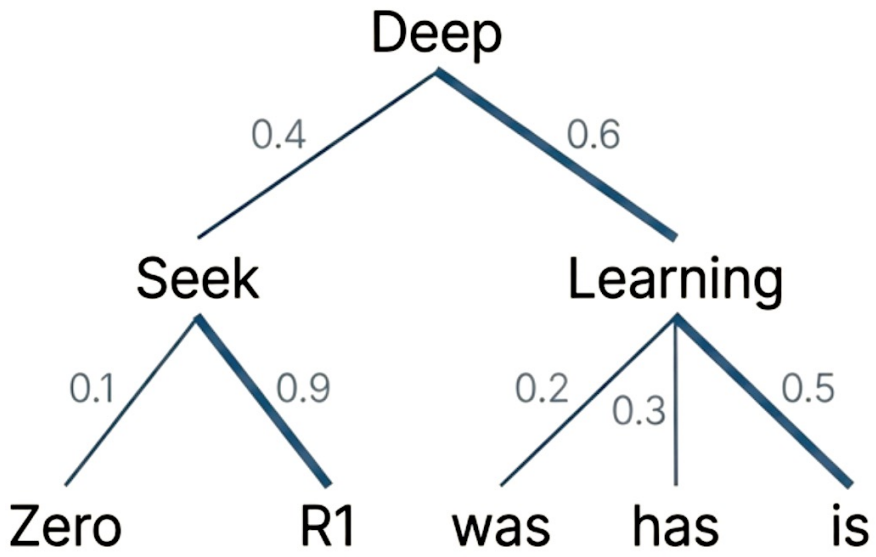
The model repeats "food" due to its high probability, resulting in unnatural, repetitive text.

Input: "My favorite color is"
Greedy Search Output: "My favorite color is blue blue blue blue is blue and blue is my favorite color blue"

This highlights how greedy decoding can produce monotonous, meaningless repetition of "blue."

Beam Search: Expanding the View

- Beam Search explores different paths through the search space simultaneously, selecting the path with the highest cumulative probability.
- The Beam Width (k) determines how many hypotheses are explored at once.
- Balances exploration and exploitation, leading to more globally optimal sequences than greedy decoding.

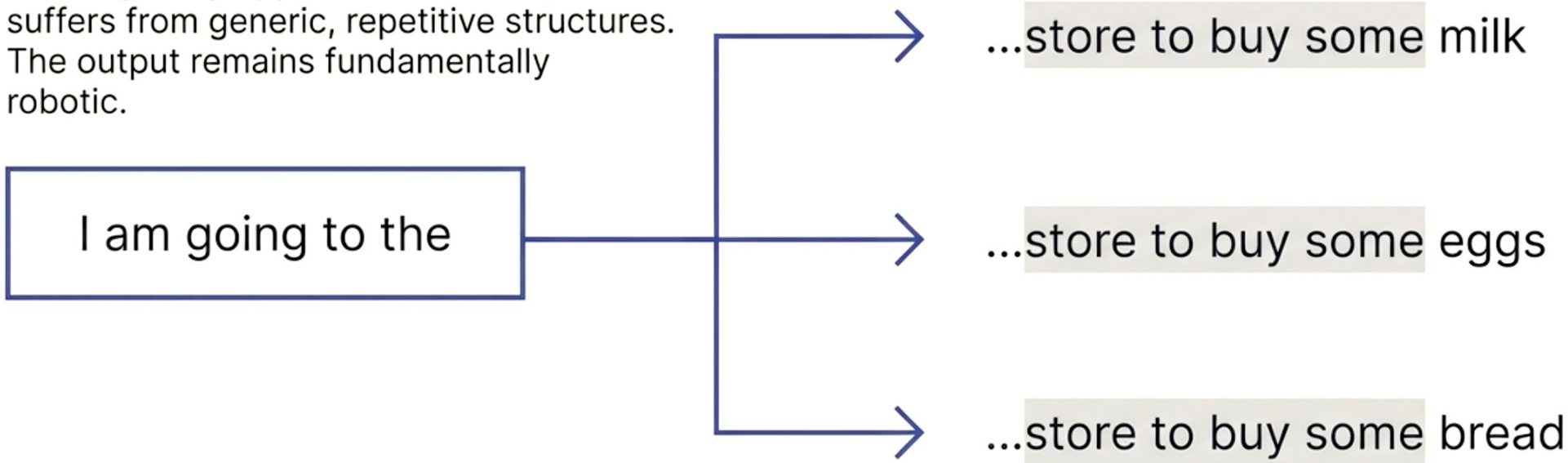


How beam search works:

1. **Initialization:** Begin with the most likely initial tokens as candidate sequences.
2. **Iterative Growth:** For each candidate sequence, append the top-k most probable next tokens, creating new sequences.
3. **Selection:** Assign a score to each new sequence based on its cumulative probability, and keep only the top-k scoring sequences.
4. **Termination:** Repeat this process until either a maximum token length is reached, or all candidate sequences end with an end-of-sequence token.

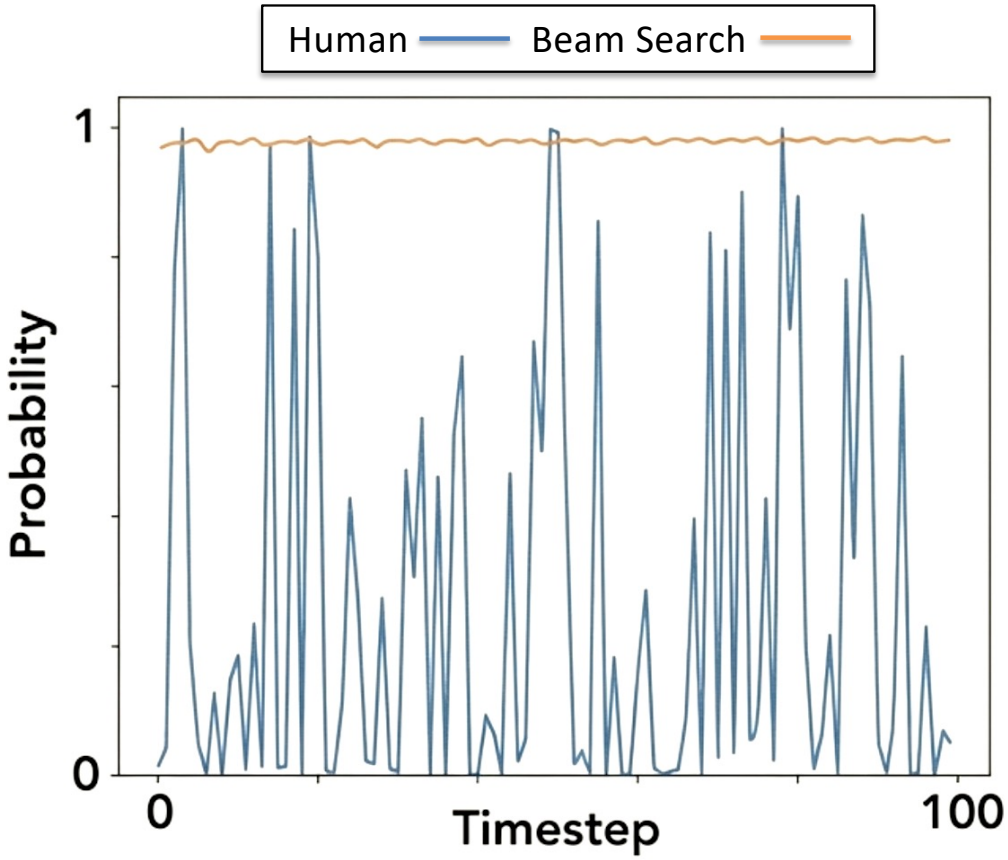
The Limits of Logic

A larger beam width explores more possibilities and usually yields better results than a greedy approach. However, it still suffers from generic, repetitive structures. The output remains fundamentally robotic.



The Unpredictability of Human Language

- Human-generated text exhibits a distribution with distinct peaks, reflecting high variability and creativity.
- Text generated by beam search algorithms features a high, flat distribution.
- The result is uniform, safe, and entirely devoid of human-like



Sampling-Based Decoding

- **Sampling-Based Decoding is a text generation technique that captures the variability and creativity of human language.**
- It generates text by randomly sampling from a probability distribution, resulting in more diverse and unpredictable outputs that resemble human-generated text.
- The Curious Case of Neural Text Degeneration (https://arxiv.org/abs/1904.09751.pdf)

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

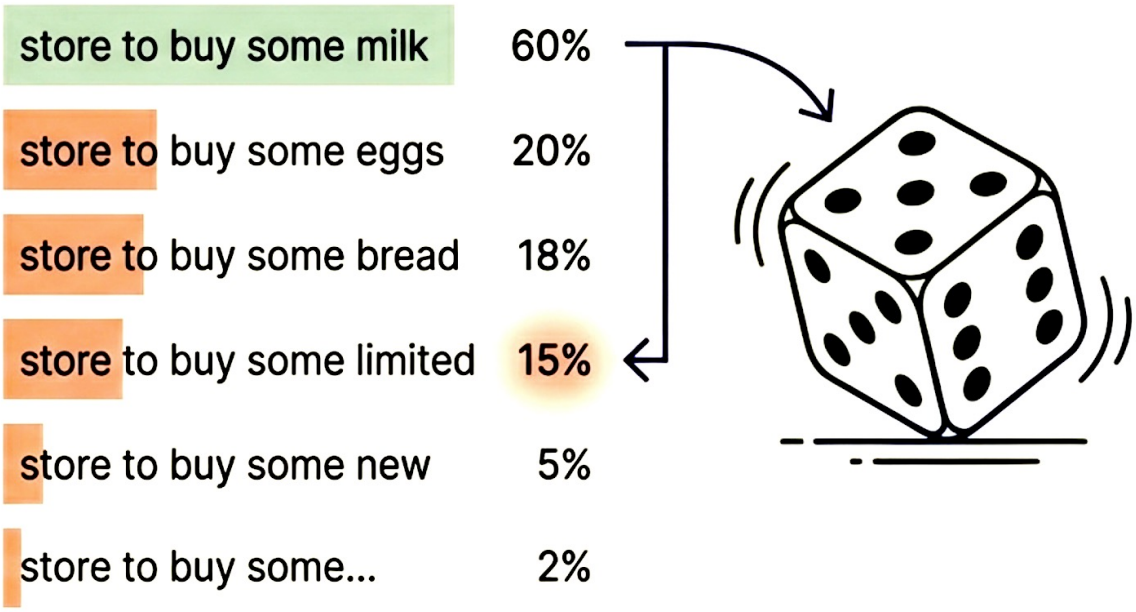
Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Figure 1: Even with substantial human context and the powerful GPT-2 Large language model, Beam Search (size 32) leads to degenerate repetition (highlighted in blue) while pure sampling leads to incoherent gibberish (highlighted in red). When $b \geq 64$, both GPT-2 Large and XL (774M and 1542M parameters, respectively) prefer to stop generating immediately after the given context.

Rolling the Dice: Pure Sampling

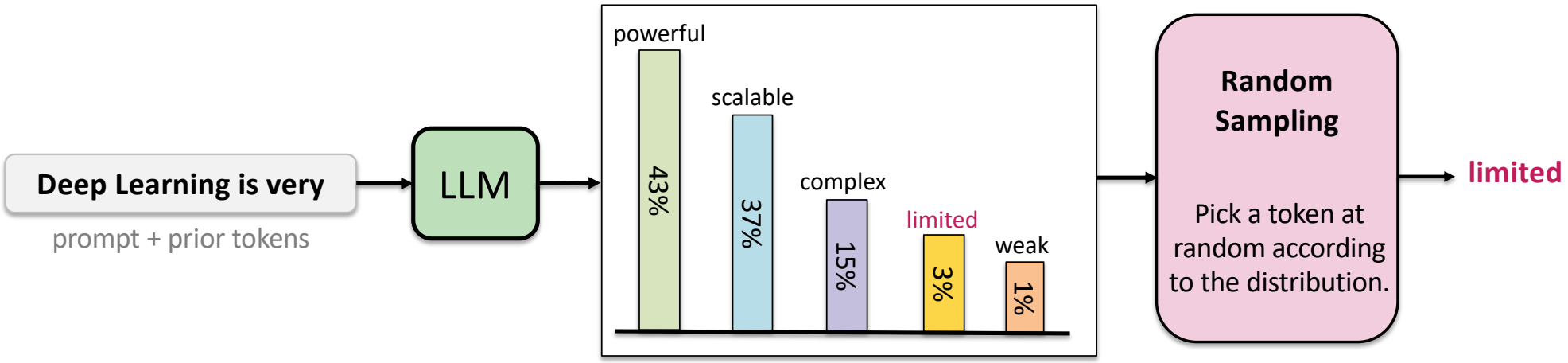
To sound human, the AI must take risks. Sampling-based decoding abandons strict highest-probability selection. Instead, it generates text by randomly selecting from the probability distribution based on the assigned weights. This captures human variability but introduces a severe new risk.



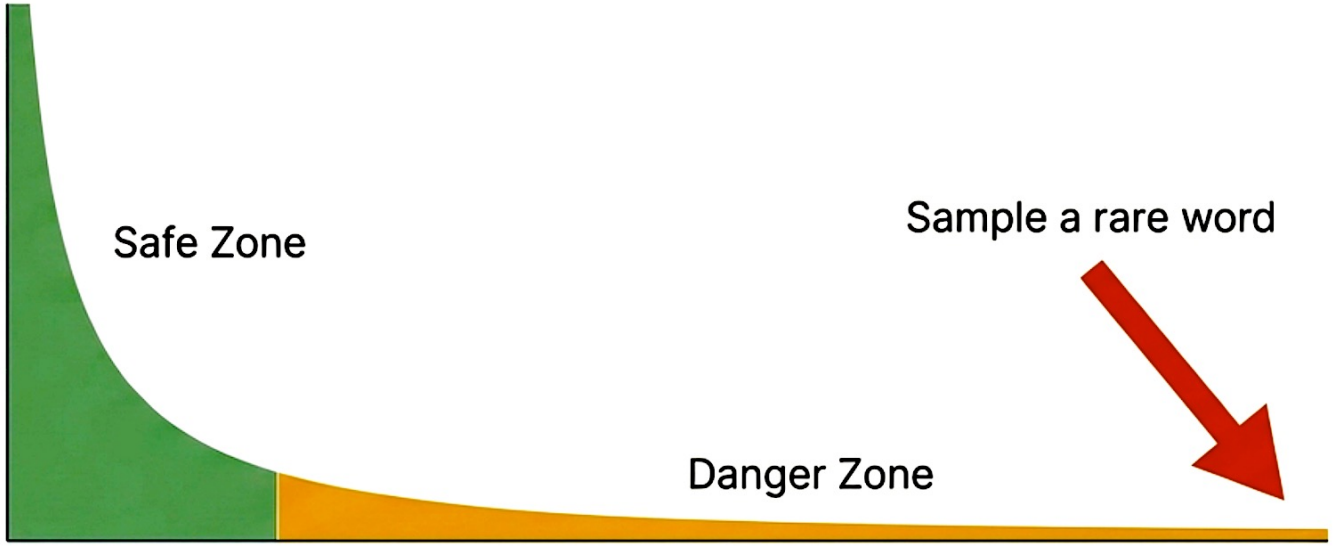
Random Sampling Decoding

- **Random sample** decoding generates text by randomly sampling from the model's output probability distribution, introducing randomness and uncertainty into the generation process. This approach allows for more diverse and creative text generation, enabling the exploration of different linguistic styles, tones, and ideas.

$$\hat{w}_t \sim P(w_t | w_1, w_2, \ldots, w_{t-1})$$



The Danger of Pure Random Sampling



Pure sampling picks proportionally based on probability.
This means the model will inevitably sample from the **Long Tail**, resulting in nonsensical, grammatically incorrect, or hallucinated text.

We must truncate the tail.

Sampling Example

Initial: Dwight arose from his bed. He walked down stairs, He made his breakfast, and he sat at the finely crafted wooden dinner table. At his right, a cup of coffee. At his left, the news paper. The crossword puzzle was particularly interesting.

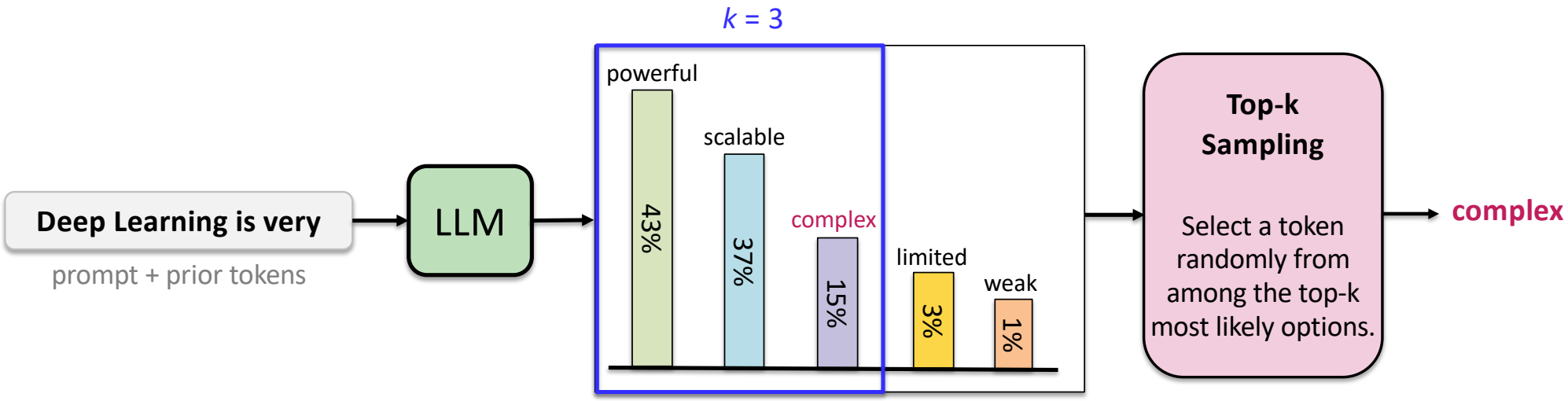
Continuation: He had opened the crossword puzzle and was pointing the newspaper from it. And the title: 12:50pm how happy has white rabbit been? why is They declining white rabbit?

The **long tail** of the distribution is where the quality of LMs become worse

Top-k Sampling: The Hard Cut-off

To mitigate the random sampling issue, **Top-k sampling** can be used. At each step *t*, the model randomly samples from the probability distribution, restricted to just the top-k most probable words.

$$\hat{w}_t \sim \text{Top}-\text{k}\left(P(w_t|w_1, w_2, \ldots, w_{t-1})\right)$$

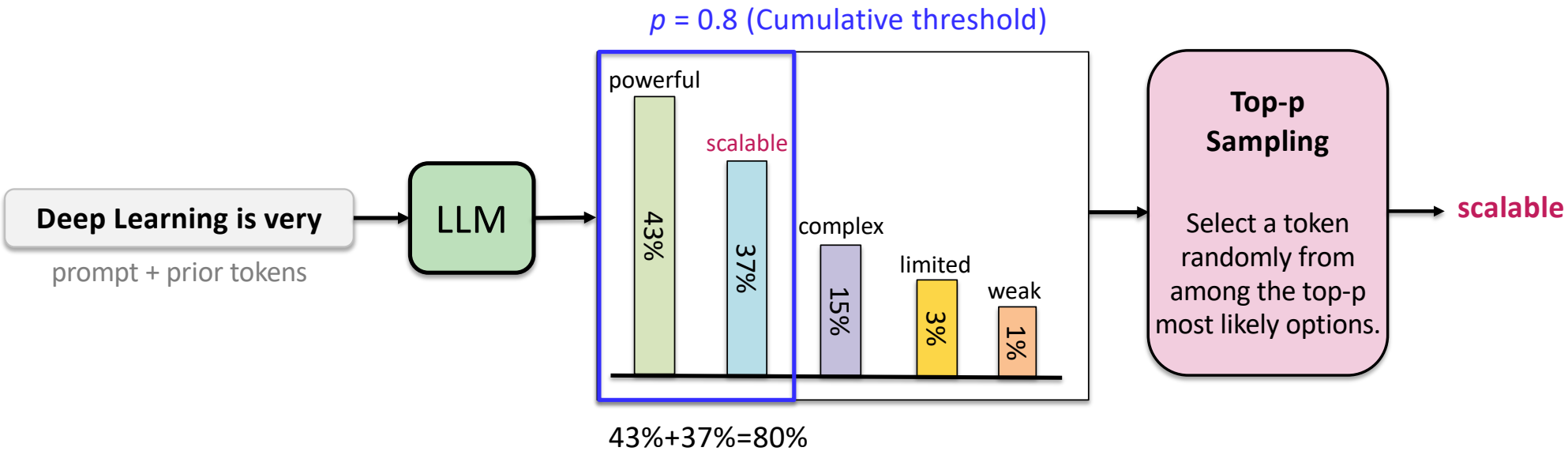


Limitations: Ignorant to context. Top-k amplifies bias and ignores the marginal difference between the k-th and (k+1)-th tokens.

Top-p (Nucleus) Sampling: The Dynamic Boundary

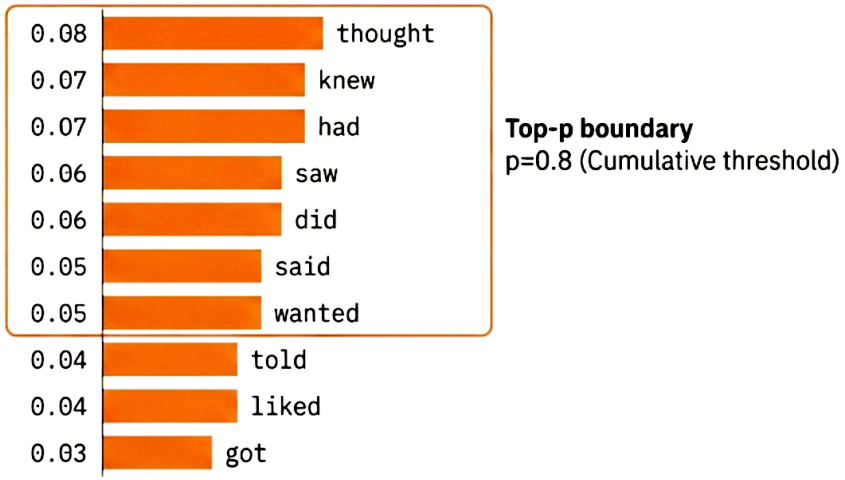
Top-p dynamically expands or shrinks the candidate pool based on the model's confidence. This is the dominant sampling strategy today (ChatGPT, Claude, Gemini, Llama).

$$\hat{w}_t \sim \text{Top}-\text{p}\left(P(w_t|w_1, w_2, \ldots, w_{t-1})\right)$$



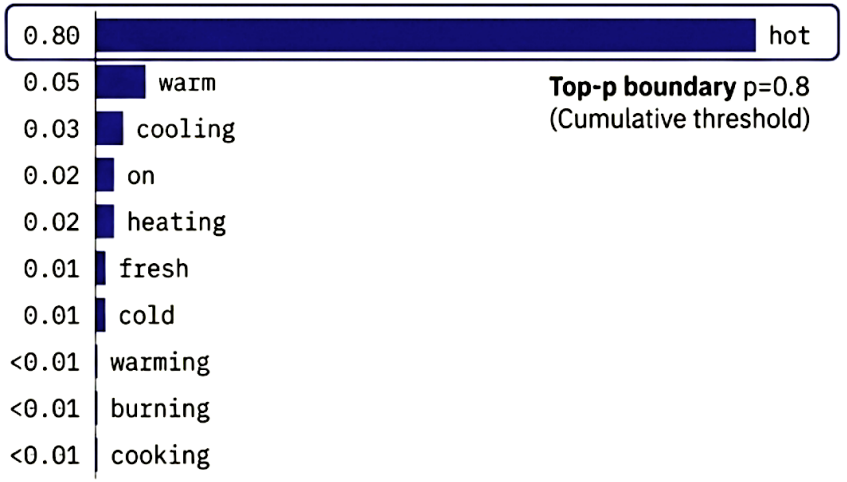
Why Top-p Wins: Adapting to Confidence

Broad Distribution (Model is unsure)



Top-p safely scoops up a large variety of sensible options.

Narrow Distribution (Model is confident)

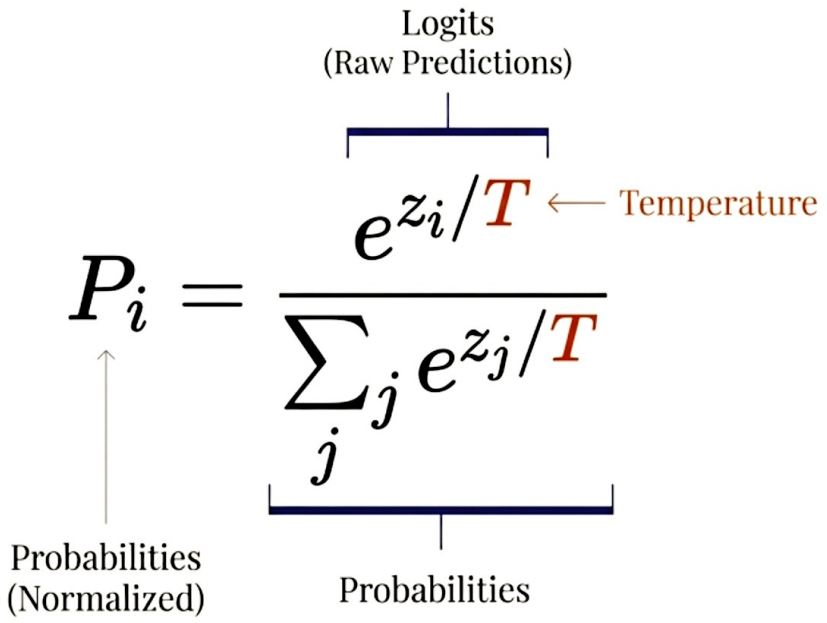


Top-p strictly isolates just the single top token, functioning like Greedy Search to ensure accuracy.

Takeaway: Top-k would blindly take 50 tokens from the narrow distribution, guaranteeing nonsense. Top-p protects the model when it is confident and unleashes it when it is exploring.

The Master Dial: Temperature

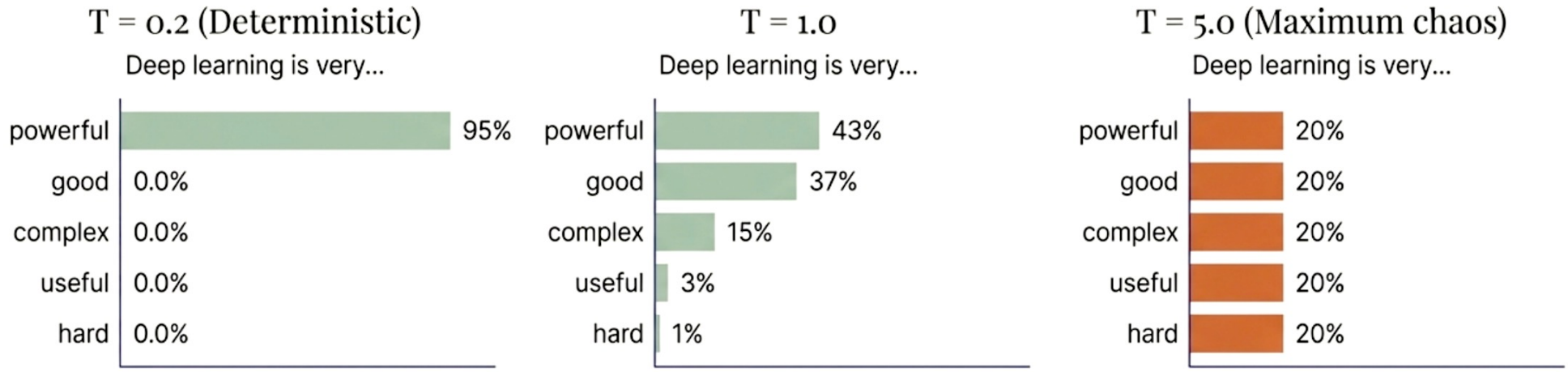
Softmax temperature is not a decoding algorithm itself, but a modifier that globally shapes the distribution before decoding even happens. It scales the raw logits (the model's unnormalized predictions) before they are converted into the final percentages, forcing a trade-off between exploitation and exploration.



Warping the Math

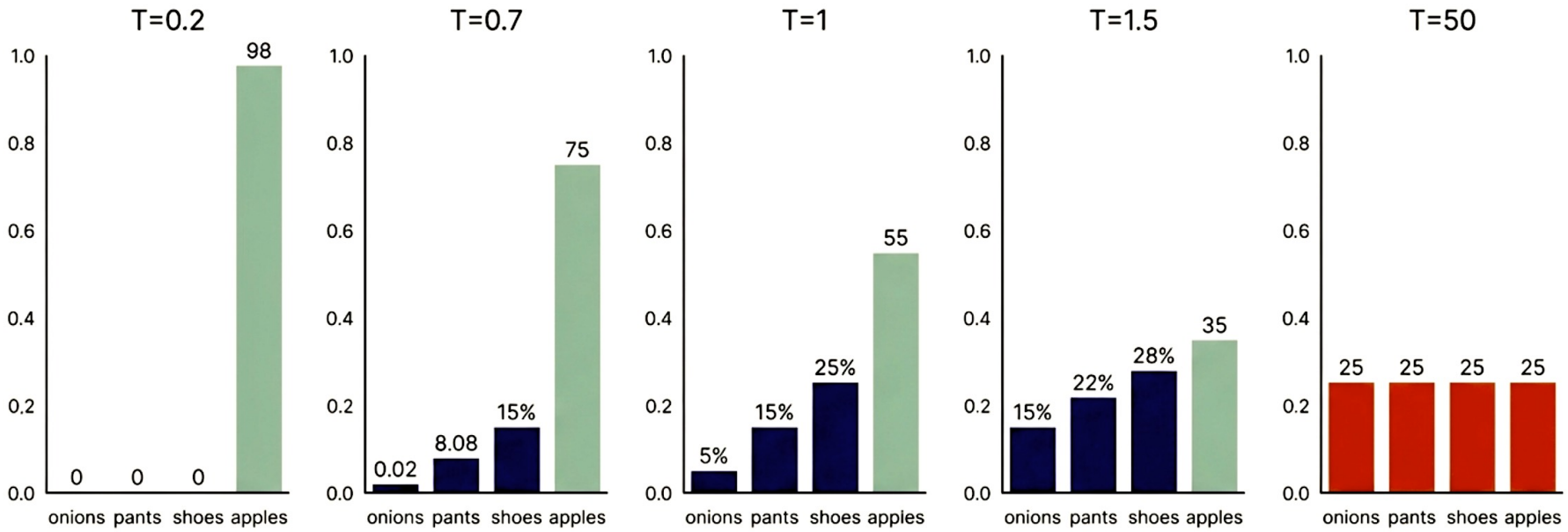
$$P_i = \frac{e^{\frac{z_i}{T}}}{\sum_{j=1}^{V} e^{\frac{z_j}{T}}}$$

- **Low Temperature (T < 1):** Sharpens the curve. It crushes lower probabilities toward zero, making the output highly deterministic and conservative.
- **Normal (T = 1):** The baseline, original distribution.
- **High Temperature (T> 1):** Flattens the curve. It pushes the model toward maximum chaos, giving rare words an equal shot.

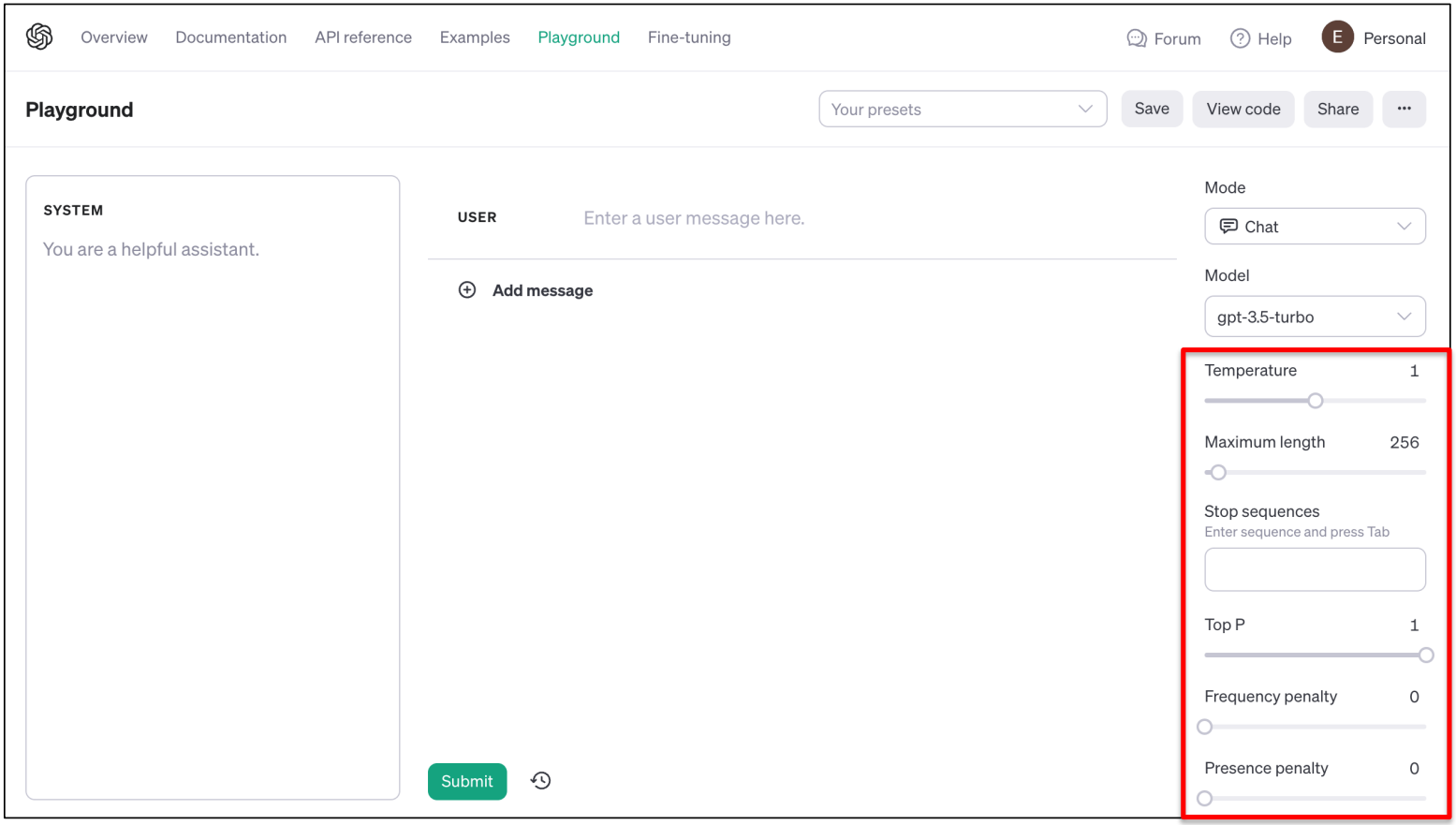


The Visual Impact of Temperature

As the dial turns up, certainty dissolves. A prompt that would deterministically output apples at a low temperature becomes a pure coin flip across all available vocabulary at a high temperature.



ChatGPT Playground



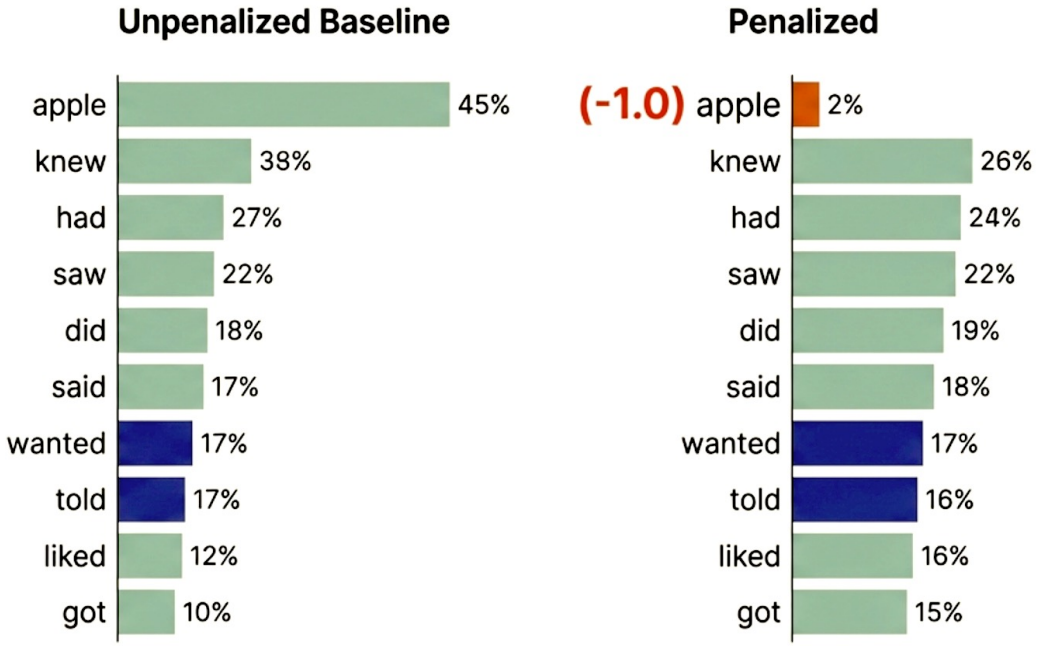
The Final Polish: Penalties

Even with great sampling, models can sometimes get stuck in loops. We apply mathematical penalties to fix this.

- **Repetition/Frequency Penalty**: Reduces the probability of redundant phrases by penalizing tokens that have already appeared.

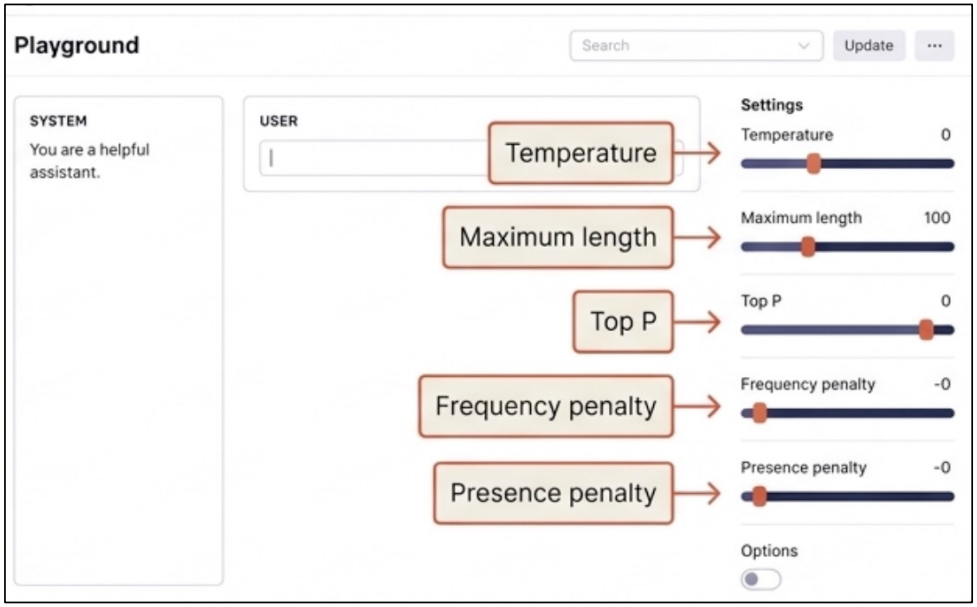
$$\log P(w) \leftarrow \log P(w) + \text{penalty}$$

- **Presence Penalty**: Encourages or discourages the introduction of completely new topics based on whether a token has appeared at all.

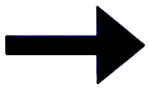
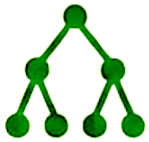


The Control Room

- These mathematical concepts are not abstract theories - they are the literal dials and switches available in developer APIs and Al playgrounds.
- Understanding the underlying distribution math is the key to fine-tuning model behavior to your exact requirements.



LLM Decoding & Parameters Cheat Sheet

	**Greedy Search:** Always picks the top token. Fast, local, but repetitive.		**Top-k:** Hard cutoff for randomness. (Ignores context).
	**Beam Search:** Explores multiple paths. Better structure, but generic.		**Top-p:** Dynamic cutoff for randomness. (Adapts to confidence).
	**Sampling:** Rolls the dice for human-like diversity.		**Temperature (0 to 1+):** The global master dial for predictability vs. chaos.
			**Penalties:** Surgical tools to banish repetition and force new topics.