Title: Top-p sampling

URL: https://en.wikipedia.org/wiki/Top-p_sampling

PageID: 74652867

Categories: Category:2019 in artificial intelligence, Category:Large language models, Category:Machine learning algorithms, Category:Natural language generation, Category:Natural language processing stubs

-----

Top- p sampling , also known as nucleus sampling , is a stochastic decoding strategy for generating sequences from autoregressive probabilistic models. It was originally proposed by Ari Holtzman and his colleagues in 2019 for natural language generation to address the issue of repetitive and nonsensical text generated by other common decoding methods like beam search . [1] The technique has since been applied in other scientific fields, such as protein engineering [2] and geophysics . [3]

In top- p sampling, a probability threshold p is set, and the next item in a sequence is sampled only from the smallest possible set of high-probability candidates whose cumulative probability exceeds p . This method adapts the size of the candidate pool based on the model's certainty, making it more flexible than top- k sampling , which samples from a fixed number of candidates. Due to its effectiveness, top- p sampling is a widely used technique in many large language model applications. [4]

Technique

At each step of the text generation process, a language model calculates a probability distribution over its entire vocabulary for the next token. While simply picking the token with the highest probability (greedy search) or a limited set of high-probability sequences (beam search) is possible, these deterministic methods often produce text that is dull, repetitive, or nonsensical. [1] Top- p sampling introduces randomness to avoid these issues while maintaining quality.

The core idea is to sample from a smaller, more credible set of tokens at each step, called the nucleus . This nucleus contains the most likely next tokens whose combined, or cumulative probability , just exceeds the threshold p . By sampling only from this dynamically-sized group, the model can adapt to different situations. When the model is confident about the next token (e.g., one token has a very high probability), the nucleus will be small. When the model is uncertain (the probabilities are more evenly distributed), the nucleus will be larger, allowing for more diversity.

The process at each step is as follows:

The model calculates the probabilities for all possible next tokens.

The tokens are sorted by their probability in descending order.

The nucleus is formed by selecting tokens from the top of the list until their cumulative probability exceeds the predefined threshold, p .

The probabilities of tokens within this nucleus are then rescaled so that they sum to 1. All tokens outside the nucleus are discarded (given a probability of 0).

The final next token is randomly sampled from this new, smaller distribution.

Formally, the nucleus, $V^{(p)} \subseteq V$ {\displaystyle V^{(p)}\subseteq V} , is defined as the smallest set of tokens satisfying: $\sum_{x \in V^{(p)}} P(x|x_1, \dots, x_{t-1}) \geq p$ {\displaystyle \sum _{x\in V^{(p)}}P(x|x_{1},\dots ,x_{t-1})\geq p} In this formula, $P(x|x_1, \dots, x_{t-1})$ {\displaystyle P(x|x_{1},\dots ,x_{t-1})} represents the probability of a token x {\displaystyle x} given the preceding tokens $x_1, \dots, x_{t-1}$ {\displaystyle x_{1},\dots ,x_{t-1}} .

Example

Imagine at a certain step, a language model has a vocabulary of five words: `[the, a, cat, dog, eats]` and produces the following probabilities:

the: 0.5

a: 0.2

cat: 0.1

dog: 0.1

eats: 0.1

If we set p = 0.8 {\displaystyle p=0.8} :

The tokens are sorted by probability: [the, a, cat, dog, eats].

The cumulative probability is calculated: the: 0.5 the + a: 0.5 + 0.2 = 0.7 the + a + cat: 0.7 + 0.1 = 0.8

the: 0.5

the + a: 0.5 + 0.2 = 0.7

the + a + cat: 0.7 + 0.1 = 0.8

The nucleus is the smallest set with cumulative probability ≥ 0.8, which is V (0.8) = { the, a, cat } {\displaystyle V^{(0.8)}=\{{\text{the, a, cat}}\}} .

The probabilities for this set are rescaled to sum to 1: P(the) = 0.5 / 0.8 = 0.625 P(a) = 0.2 / 0.8 = 0.25 P(cat) = 0.1 / 0.8 = 0.125

P(the) = 0.5 / 0.8 = 0.625

P(a) = 0.2 / 0.8 = 0.25

P(cat) = 0.1 / 0.8 = 0.125

The next token is then sampled from this new distribution, meaning dog and eats have a 0% chance of being chosen.

Top- k sampling

Top- k sampling is a similar technique where the pool of candidate tokens is restricted to the k {\displaystyle k} most likely tokens. The main advantage of top- p is its adaptability. When the model is very certain about the next token (a peaked distribution), the nucleus V (p) {\displaystyle V^{(p)}} can be very small. When the model is uncertain (a flat distribution), the nucleus can be much larger, allowing for more diversity. In contrast, top- k always samples from a fixed number of tokens, which may be too restrictive or too broad depending on the context. [1]

Applications

While top- p sampling is most famously used as a decoding strategy for large language models, the technique has also been adapted for use in other scientific domains that involve generating or analyzing sequential data from probabilistic models.

Natural language generation

In its original domain of natural language generation, top- p sampling is valued for its ability to produce more diverse and coherent text compared to deterministic methods. It has been shown to be beneficial in tasks like automatic question generation, where sample diversity is important for creating effective training data for question answering models. [5]

Drug and protein design

Top- p sampling is used in computational biology to generate novel molecular and protein sequences from specialized language models. In de novo drug design , chemical language models trained on molecular structures use nucleus sampling to generate focused libraries of new, valid drug candidates. By combining this generation with a predictive model for bioactivity , researchers have identified novel, potent kinase inhibitors . [6] Similarly, in protein engineering , the technique is used to sample protein language models to explore the vast space of possible amino acid sequences to find novel, functional candidates for use in therapeutics or new materials. [2]

Geophysics

The technique has also been applied in geophysics for denoising audio magnetotelluric (AMT) data. In one method, nucleus sampling is integrated into an attention mechanism to help identify and remove complex anthropogenic noise from geophysical signals. This improves the accuracy of AMT in interpreting the Earth's subsurface resistivity structure, which is critical for applications like mineral exploration . [3]

Limitations and alternatives

While top- p and top- k sampling address many of the issues found in deterministic methods like beam search , they are not without shortcomings. Research has shown that these stochastic methods can produce text with undesirable repetitions and may not fully capture the statistical properties of human language. [7] [8] [9]

A range of alternative sampling strategies have been proposed to address these limitations.

Factual-nucleus sampling was proposed to counter the tendency for the "uniform randomness" applied to the nucleus to harm the factuality of the generated text. It dynamically adapts the level of randomness to improve factual accuracy while maintaining text quality. [10]

Locally typical sampling frames text generation in an information-theoretic light. Instead of selecting only the highest-probability tokens, it samples from a set of tokens that are "locally typical" in an information-theoretic sense, which has been shown to reduce repetition and improve quality. [7]

Priority sampling is a deterministic alternative designed to address the issue of repeated or incoherent samples. It produces a set of unique samples ordered by the model's confidence and has been shown to outperform nucleus sampling in some compiler optimization tasks. [11]

See also

Beam search

References

v

t

e

Autoencoder

Deep learning

Fine-tuning

Foundation model

Generative adversarial network

Generative pre-trained transformer

Large language model

Model Context Protocol

Neural network

Prompt engineering

Reinforcement learning from human feedback

Retrieval-augmented generation

Self-supervised learning

Stochastic parrot

Synthetic data

Top-p sampling

Transformer

Variational autoencoder

Vibe coding

Vision transformer

Waluigi effect

Word embedding

Character.ai

ChatGPT

DeepSeek

Ernie

Gemini

Grok

Copilot

Claude

Gemini

Gemma

GPT 1 2 3 J 4 4o 4.5 4.1 OSS 5

1

2

3

J

4

4o

4.5

4.1

OSS

5

Llama

o1

o3

o4-mini

Qwen

Base44

Claude Code

Cursor

Devstral

GitHub Copilot

Kimi-Dev

Qwen3-Coder

Replit

Xcode

Aurora

Firefly

Flux

GPT Image 1

Ideogram

Imagen

Midjourney

Qwen-Image

Recraft

Seedream

Stable Diffusion

Dream Machine

Hailuo AI

Kling

Midjourney Video

Runway Gen

Seedance

Sora

Veo

Wan

15.ai

Eleven

MiniMax Speech 2.5

WaveNet

Eleven Music

Endel

Lyria

Riffusion

Suno AI

Udio

Agentforce

AutoGLM

AutoGPT
ChatGPT Agent
Devin AI
Manus
OpenAI Codex
Operator
Replit Agent
01.AI
Aleph Alpha
Anthropic
Baichuan
Canva
Cognition AI
Cohere
Contextual AI
DeepSeek
ElevenLabs
Google DeepMind
HeyGen
Hugging Face
Inflection AI
Krikey AI
Kuaishou
Luma Labs
Meta AI
MiniMax
Mistral AI
Moonshot AI
OpenAI
Perplexity AI
Runway
Safe Superintelligence
Salesforce
Scale AI
SoundHound
Stability AI
Synthesia
Thinking Machines Lab

Upstage

xAI

Z.ai

Category

v

t

e