LANGUAGE MODEL SAMPLING FROM THEORY

Kevin Yin
Anlatan
kevin@anlatan.ai

ABSTRACT

Language model samplers such as top-*p* and top-*k* transform the output logits of a token vocabulary. We find the principles behind sampling, then derive NovelAI's "Unified" sampler, documenting all our assumptions.

1 INTRODUCTION

Major advances in sampling were top-*k*, which showed that it was possible to do better than temperature-only sampling, and top-*p* (Holtzman et al., 2019), the first modern sampler. Further progress has been slow.

NovelAI permits more sampler choices than most services, and it allows chaining samplers into a mega-sampler. This increased freedom is popular with power users, but creates complexity and indecision. Institutions and experts have arisen to navigate this.

Despite this, user opinion of sampler quality is largely uncorrelated with sampler quality. Users estimate signal from noisy data, but without statistical methods and double-blind experiments, their estimates are poorly calibrated. The substantial freedom of sampler choice substantially consists of poor choices, which users are not avoiding. So I spent a few days to create the $n + 1$th sampler to replace all these sampler chains.

To construct our sampler, we will first describe the optimal sampler and its fundamental principles. Then we will continually lower our expectations until we reach something computable.

2 IDEALISTIC GOALS

The goal of text is to show something interesting that the reader has not seen before.[1] This is different from the goal of a language model, which is to replicate its training dataset. Most webcrawl training datasets have 10% spam. So there is no a priori reason to sample with temperature=1 from the language model.

The worth of text is its value over replacement, measured per unit of time/effort the reader spends. We will separate this value into quality and diversity, as is done in Zhang et al. (2020).

2.1 QUALITY

Quality represents how interesting text is. Some factors affecting this are:

Mistakes in grammar, style, and punctuation. Typos are generally absent from LLM output, because each person makes a different typo. The model only reproduces widespread mistakes, like switching `they're` and `their`.

Logical correctness. Text should avoid contradicting prior text, such as a person seeing something despite being blind, or mentioning a fact he doesn't know.

Content quality. Stories can be interesting or boring.

We will not be able to judge quality directly, and will instead resort to proxies.

[1] Slightly more accurately, the goal is to cause a useful mental update. There's lengthy theory delineating the definition and its many anomalies. This is intellectually fulfilling but does not impact samplers.

2.2 DIVERSITY

Text should present something the reader has not seen before. A distance function measures the conceptual difference between two texts, and the sampler should output text with high distance to the reader's past experiences. For example, sentences about different topics are meaningfully different, while sentences with rearranged word order are minimally different.

However, the sampler does not know what the person has already read. It can only maximize the variety of its output, to minimize the collision chance. **Diversity** represents the number of distinct conceptual ideas in a probability distribution of texts. To measure diversity, you would calculate the chance of collisions in this "conceptual idea space", which is impossible in practice. (Theoretically, you could use an embedding model.)

The separation into quality and "not being seen before" is not entirely proper. Novelty interacts with both. A work's value may derive from being unexpected (e.g. Monty Hall Problem), or being unexpected to society as a whole (e.g. new research). We will ignore this.

3 A SIMPLE THOUGHT EXPERIMENT

A typical (large) language model has a token vocabulary of size 10k-300k. Given a context, it generates a probability distribution for the next token. A **sampler** accepts this token probability distribution as input, and outputs a changed probability distribution for the same tokens.

Let p_t be the model's output probability of a token t, and s_t be its probability after the sampler. We'll calculate the expected worth of this token with the sampler's probability distribution. Assume the quality of t is $\ln p_t$, so the expected quality is $\sum_t s_t \ln p_t$. Measure diversity as entropy, $\sum_t -s_t \ln s_t$. The expected worth is their sum $\sum_t s_t \ln(p_t/s_t) = -D_{\mathrm{KL}}(S\|P)$, the negative KL-divergence between the before-sampler distribution P and the after-sampler distribution S. This is maximized when $S = P$, so temperature=1 sampling is optimal in this thought experiment.

If instead the quality is $\ln f(p_t)$, then the method of Lagrange multipliers gives

$$\ln s_t = \lambda - 1 + \ln f(p_t)$$
$$s_t = e^{\lambda-1} f(p_t)$$

We don't need to solve for λ, because $s_t \propto f(p_t)$ and $\sum_t s_t = 1$. So $f(p_t)$ is the unnormalized probability mass of s_t, giving $s_t = f(p_t)/\sum_i f(p_i)$.

3.1 ANALYSIS

In this thought experiment, a single token is implicitly used as a unit of reader time. This is inaccurate, since tokens can be short words (`_the`) or long words (`_developers`).

When the text is decomposed into tokens, each token's true quality is equal to the expected quality of the text as a whole, averaged over the distribution of future tokens, conditioned on that token being chosen.

The quality of a token is hard to estimate. The only information the sampler has is the input log-probabilities. It does not have access to future tokens, since those are expensive to compute.

Quality is a function of the log-probabilities rather than the logits, because the model's loss function is a function of solely log-probabilities. Unnormalized logits would be a valid function input if they were a side-channel for other information, such as z-scores measuring training exposure of a sample, which is plausible under z-loss. But we will ignore this effect. Other metrics like probability rank (as in top-*k*), sum of probabilities (as in top-*p*), or ratio to top probability (as in min-*p*) similarly should not be inputs to the quality function, because they are unrelated to the training objective, unless they are side channels for information.

Tokens with higher logprobs are higher quality because of consensus: text written by many people is more likely to be correct than text written by few people. There is also an effect from diversity of errors: there may be only a few correct options, but there are many ways to make mistakes.

In the absence of a meaningful parametrization of concept space, we calculate diversity as the entropy of the distribution of generated texts, normalized by token count. This measures the number of raw texts generated.

Furthermore, we only calculate entropy as the average entropy of the tokens that have been generated, since we do not have access to ungenerated texts. With this dual simplification, our estimate of diversity is poor, and significantly depends on the tokenizer. For example, if a token probability distribution consists of `_a` and `_an`, the maximum entropy for the token is achieved with a 50-50 distribution. However, the model's output probability for `_a` is typically 15x higher than for `_an`, representing a 15x larger pool of options in the next token. So maximizing entropy in a greedy fashion actually reduces overall entropy here.

Normalizing entropy by bytes would be more accurate than normalizing by tokens. But we're lazy and don't do that.

4 REPETITION

Side channels of information can be exploited to usefully distort the sampler behavior.

1. When the user rejects and retries a generation, a collision is more likely than normal, so diversity becomes more important. Another influence is that the model's prediction was rejected, so it should be more conservative in its judgment of quality - also reflecting a higher temperature.
2. Repetition of text has zero value. If text has already appeared, it should not appear again. Sometimes, models fall into loops, or near-loops. Weaker models have more repetition, as it is a valid inductive bias in the absence of training. Stronger models can successfully break out of it, and their repetitions are also more sensible.

   High repetition is generally low quality, often reflecting poor or simplistic writing. Repetition is in-distribution due to spam in the dataset, but even with a clean dataset, repetition detection would still be useful, as post-hoc verification.
3. The model is biased toward its training distribution, and will reach for the same strings in different situations. When a user uses the same model many times, these tendencies lead to collisions with past stories.

The temperature should increase in each situation.

1. When the user retries a generation, increase the temperature.
2. When text reappears within a context, increase the temperature (and also rewind).
3. When the user has more experience with the model (depending on similarity of the stories already generated), increase the temperature.

Alternatively, if history is maintained of former generations, repetition can be detected and avoided.

5 BIASES

- Texts consisting of consistently high-probability tokens are low quality. This is not just a quirk of autoregressive LM generation. High probability tokens are biased toward simplicity (because simple tokens are accessible to more writers), correctness, and repetition. When these tokens are chosen repeatedly, as in beam search, the bias will stack and degrade the output in the expected way.
- KL-divergence in the training loss does not capture the egregiousness of an error, only whether its label matches or not. User preferences have more grades than yes/no matching: tokens that contradict the training data may be fine (synonyms) or egregiously wrong (tokens from a different language). So the training loss and user judgment disagree.
- The model's output has variance from stochastic gradient descent. Annealing helps, but most LLMs are annealed with a cosine schedule to 10%, which is insufficient. For small

probabilities, the standard deviation scales with $\sqrt{p}$, which is much bigger than p. So unlikely token probabilities are dominated by noise.

This means optimal KL divergence will happen at temperature > 1, even though the model is trained with temperature 1. (The same behavior exists with ELO (Bradley-Terry) ratings: the expected winrate between two players is closer to 50% than their ratings predict.)

- If the model has insufficient capacity or insufficient training, its logits will be inaccurate. This deviation will not have clear trends or biases; the model can be very overconfident in the wrong tokens.

  Being under-trained is situation-dependent: some prompts have fewer entries in the training data or are out of distribution. This especially harms labels with lower probabilities, which receive less signal. (A binary distribution has the most information when p is near 0.5.)

  If we can detect that the model is undertrained for a context, we could sharpen the distribution (forcing the model to choose only its most confident choices, in the hopes of avoiding error) or flatten the distribution (recognizing that the model's output has low correlation with quality).
- Most LLMs are trained with incorrect weight decay schedules (coupled 0.1 in PyTorch AdamW). This makes the distribution more uniform on less-seen training examples.
- Adam's beta2: if a token's prediction is near-0 and the label is false for many steps, then its gradient will be small, and the second moment will be small. If its label is activated, the second moment will suddenly jump. This is asymmetric: the upward force is divided by a larger value than the downward force. This creates a downward bias for rarely activated tokens. (And correspondingly, an upward bias for near-certain tokens.) Using the previous step's second moment in AdamW would fix this, but that has major disadvantages.

6 FORMULA BUILDING

Given a context, each possible next token has a **quality** number that is independent of the other proposed tokens. A possible formula is to fit an arbitrary function $\text{quality}(t) = f(\ln p_t)$, for tokens t with probability p_t. The asymptotics of this function are mainly driven by the first two biases in the previous section. The slope at 0 (high log-probability) is below 1, because high-probability tokens are biased toward simplicity and repetition. The slope for low-probability tokens is greater than 1, because of the mismatch between KL-divergence and user preference. These two behaviors are verified empirically for creative tasks (storytelling and chat). In some domains, like code or math, the first behavior may not result in a slope decrease.

However, log-probabilities depend on the population of other valid tokens. If quality is a property of a token, independent of other tokens, we need to correct our estimate by determining what the logprob of a baseline-quality token would be. If the entropy of the distribution is high, it suggests that there are many high-quality options, so the baseline quality token would have a lower logprob. This correction is specified as "Conf" in NovelAI's **Unified** sampler.

Some contexts will naturally have higher entropy: beginnings of sentences and new words will be higher entropy than the middle of words. The beginning of a context will have higher entropy than the end. These corrections would be captured by n-gram models and context length. However, we don't bother doing that.

We'll measure diversity as average entropy per token, knowing of its deficiencies.

Synthesizing these considerations, the quality function in NovelAI's Unified sampler is:

$$\text{quality}(t) = (c_1 + c_0 H(P))(\ln p_t) + c_2(\ln p_t)^2 + c_3(\ln p_t)^3 \tag{1}$$

The three coefficients c_1 (linear), c_2 (quad), and c_3 (cubic) are the first three coefficients of a Maclaurin series, intended to fit the quality function f. c_0 (conf) is an entropy adjustment for the logprob of a baseline-quality token.

Because of c_0, this formula does not fit the quality function in our thought experiment, since its function f only takes p_t as input, not $H(P)$. However, we ignore this and set $\ln s_t \propto \text{quality}(t)$ anyway, as a sampler which maximizes (quality) + (entropy). The deviation increases for larger s_t.

7 EXPERIMENTS

Aetherroom ran an A/B experiment on this parametrization. In an unspecified double-blind manner, subjects chose between two texts generated by samplers with different coefficients. The winning coefficients were scored 1 and the losing coefficients were scored 0. The heatmap in Figure 1 plots the average score of the 40 nearest coefficients (using k-NN), which is the expected winrate.

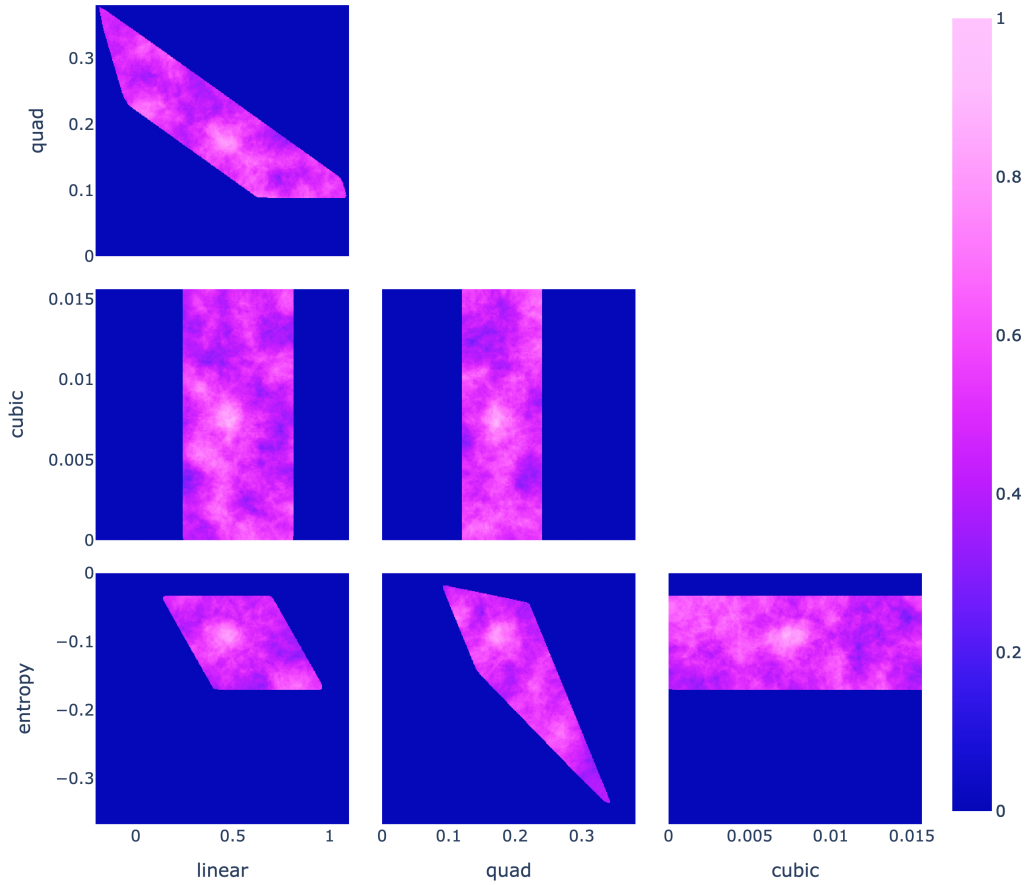

Figure 1: 2D slices of the 4D coefficient space. Note that bright regions crossing 3 dimensions would not be visible. The coordinates not specified by each 2D slice are of the claimed center.

The bright center is at approximately

$$\begin{aligned} \text{linear} &= c_1 = 0.48 \\ \text{quad} &= c_2 = 0.17 \\ \text{cubic} &= c_3 = 0.008 \\ \text{entropy (conf)} &= c_0 = -0.09 \end{aligned}$$

When win/loss is randomized, no bright regions appear anywhere, so there is signal in the brightness. However, we do not know the variance of the center estimate. When a quadratic function is fit to the data, its associated symmetric matrix has negative eigenvalues, so the resulting critical point means nothing. Maybe we have insufficient data or need a non-naive trust region approach.

We don't know how much these parameters generalize outside of Aetherroom.

The "Unified" sampler, omitting the cubic term, is used in NovelAI. It is popular among sampler chain designers and users, but this likely reflects positive messaging from NovelAI's user sampling guide, rather than independent judgments.

8 FUTURE WORK

The parametrization could have more terms, such as an entropy correction to the quadratic coefficient. The probability distribution could be adjusted nonlinearly before calculating entropy. The coefficient of the highest degree term should stay positive, to prevent very negative logits from gaining prominence.

This sampler has no history dependence: the quality of a token is not dependent on the quality of the previous token. There may be a side channel here that is not being exploited.

Per-token entropy can be adjusted to better estimate the overall entropy of the text. For example, adding special logic for `_a` and `_an` would solve that testcase immediately. This can be extended in obvious ways to other tokens.

If a model supports multi-token prediction, lookahead will give a better estimate of diversity and quality. This improves the output, on top of the inference speedup. However, the combinatorial explosion requires pruning, especially with byte models. The positive relationship between quality and logprobs breaks down over longer stretches of text; Gareev et al. (2024) and Zhang et al. (2020) describe this.

A context-aware sampler can do **repetition penalty**. If a token has previously appeared, its chance is lowered on future instances. Repetition penalty is extremely tokenizer-dependent and causes unpleasant distortions. A better form is **phrase repetition penalty**, which tracks text across multiple tokens. PRP works best with backtracking, which turns it into post-hoc verification. The most prominent weakness of PRP is long names. For example, "Hakurei Reimu" is 6 tokens, which would be a long repetition for normal words.

A token-aware sampler can use substring similarity, which can tell that `_developing` and `_developed` are related. This would also result in a better estimate of entropy.

I will pursue none of these ideas because I have more important things to do.

ACKNOWLEDGMENTS

For sampler A/B testing, Arthur Burgan did UI, Emre Agca did frontend, F. Johnson did middleware, and Luna did backend. J. A. Lievokäärme and F. Johnson helped understand NovelAI's existing samplers. Eren Doğan let me do whatever I want.

Writing this article took more time than the actual work.

REFERENCES

Daniel Gareev, Thomas Hofmann, Ezhilmathi Krishnasamy, and Tiago Pimentel. Local and global decoding in text generation, 2024. URL https://arxiv.org/abs/2410.10810.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019. URL http://arxiv.org/abs/1904.09751.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. Trading off diversity and quality in natural language generation. *CoRR*, abs/2004.10450, 2020. URL https://arxiv.org/abs/2004.10450.