MIROSTAT: A NEURAL TEXT DECODING ALGORITHM THAT DIRECTLY CONTROLS PERPLEXITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural text decoding algorithms strongly influence the quality of texts generated using language models, but popular algorithms like top-k, top-p (nucleus), and temperature-based sampling may yield texts that have objectionable repetition or incoherence. Although these methods generate high-quality text after ad hoc parameter tuning that depends on the language model and the length of generated text, not much is known about the control they provide over the statistics of the output. This is important, however, since recent reports show that humans prefer when perplexity is neither too much nor too little and since we experimentally show that cross-entropy (log of perplexity) has a near-linear relation with repetition. First we provide a theoretical analysis of perplexity in top-k, top-p, and temperature sampling, under Zipfian statistics. Then, we use this analysis to design a feedback-based adaptive top-k text decoding algorithm called *mirostat* that generates text (of any length) with a predetermined target value of perplexity without any tuning. Experiments show that for low values of k and p, perplexity drops significantly with generated text length and leads to excessive repetitions (the boredom trap). Contrarily, for large values of k and p, perplexity increases with generated text length and leads to incoherence (confusion trap). Mirostat avoids both traps. Specifically, we show that setting target perplexity value beyond a threshold yields negligible sentence-level repetitions.

1 INTRODUCTION

Large-scale generative language models have received recent attention due to their high-quality open-ended text generation ability (Brown et al., 2020; Radford et al., 2019). Generating texts from these models usually relies on some form of random sampling. Pure sampling often leads to incoherent and low-quality texts (Holtzman et al., 2018), whereas greedy decoding leads to excessive repetitions, another form of low quality. The right decoding algorithm is needed to generate high-quality texts with controlled attributes (Ippolito et al., 2020; Zhang et al., 2020; Ippolito et al., 2019).

We introduce mirostat,[1] a neural text decoding algorithm that *actively controls* the generative process to maintain the perplexity of generated text at a certain desired value. Mirostat uses an adaptive top-k sampling algorithm to actively tune the value of k which helps maintain the overall perplexity of the text; recall that top-k sampling (Holtzman et al., 2018; Fan et al., 2018) is where the next word is sampled from the top k most probable choices.

Top-k sampling and several other recent sampling methods are motivated by suppressing an unreliable tail in the probability distribution of trained language models. Another sampling method is top-p, also known as *nucleus sampling*, where the next word is chosen from the top x probable choices, where x is the smallest integer such that their cumulative probability mass is at least p (Holtzman et al., 2020). While top-k sampling involves a fixed number of most probable choices, top-p sampling involves a dynamic number of choices based on a fixed p value and shows better statistical and human-evaluated performance. For small values of k and p, these sampling methods unfortunately repeat phrases in generated text. This can be handled by penalizing repetitions and using appropriate *temperature* values (Keskar et al., 2019) or adding diversity to the generated text (Zhang et al., 2020; Vijayakumar et al., 2018). On the other hand, large values of k and p can lead to

[1]The word mirostat is derived from *mirum* which is Latin for *surprise* and *stat* meaning control.

incoherent texts similar to pure sampling. Although choosing appropriate values of p or k can avoid repetition and incoherence, this involves ad hoc tuning of parameters. Even for a fixed value of p or k, the generated text can have varying statistical properties.

Intriguingly, as we demonstrate via Example 1 in Appendix A, small values of a certain perplexity statistic of generated texts called *surprise* (Def. 1) are closely linked to repetitions and large values of surprise are linked to incoherence. Perplexity is a statistical metric used to evaluate quality of neural text generation, and is closely related to average surprise as shown in Fig. 5 in Appendix A and formalized in Sec. 2. A large-scale human subject experiment by Zhang et al. (2020) showed human-evaluated quality is closely related to the likelihood of the generated text for fixed number of tokens. In particular, reducing perplexity increases quality upto some point before the quality starts dropping. This implies that good control over perplexity of the generated text would give direct control over the quality of generated text (as evaluated by humans). Generating texts with an appropriately chosen target perplexity value may maximize quality of generated text. *Ergo mirostat*.

Now we summarize our key contributions. Sec. 3 shows theoretically how cross-entropy and hence perplexity grows in top-k and top-p sampling as a function of k and p respectively, which was previously unknown. Sec. 4.1 introduces mirostat sampling, which outputs texts with predetermined target perplexity value. Although perplexity may not fully capture the quality of text (Hashimoto et al., 2019), much literature discusses its correlation to quality (Zhang et al., 2020). Hence, our algorithm to control perplexity would help generate high-quality text. Sec. 5.1 experimentally shows much fluctuation in cross-entropy rates in top-k and top-p sampling as a function of their input parameters, indicating their inability to control perplexity of output text. Sec. 5.2 demonstrates that repetition is closely related to perplexity of the generated texts, mostly independent of the sampling method, but slightly dependent on the model used. Sec. 5.3 experimentally shows that mirostat sampling avoids both boredom and confusion traps for a wide range of target perplexity values.

1.1 Related Work

Sampling from distorted probability distribution Pure sampling from language models often leads to incoherent text whereas greedy decoding leads to repetitions. Distorting probability distributions, as in top-k, top-p, or temperature sampling help improve quality of generated texts, if parameters are properly tuned (Holtzman et al., 2018; Fan et al., 2018; Holtzman et al., 2020). Tuning these methods, however, is ad hoc and does not provide good control over the statistics of the output. Our method uses statistics of previously-generated tokens as input to generate the next token, by distorting the probability distribution so it helps control the overall statistics of the generated text. This ability to control the perplexity of the output is a key advantage of our method over previous work. This, when used with the relation between perplexity and human-evaluated quality observed by Zhang et al. (2020), can yield text that has better quality control.

Controllable text generation Controllable text generation has commonly focused on controlling semantics of the output text, as in new language models like CTRL (Keskar et al., 2019), and sampling algorithms like plug-and-play language model (Dathathri et al., 2020) and constrained sentence generation by Metropolis-Hastings (Miao et al., 2019). Contrarily, our approach is purely statistical. We try to solve the statistical problems of pure sampling or greedy decoding by guiding the decoder along a desired statistical path.

Quality-diversity tradeoff Top-k, top-p, and low-temperature sampling improve the quality of the text, but at the cost of reduced diversity. Applications like question-answering only demand high-quality generation, but open-ended tasks such as story generation demand diversity too. However, Zhang et al. (2020) observe a tradeoff between quality and diversity; they further observe diversity is closely related to entropy whereas quality is maximized in a certain range of observed likelihood values for fixed-length sentences. Our algorithm well-controls observed cross-entropy, the observed likelihood per token of generated text. Hence, by maintaining the observed cross-entropy in a certain range, we can ensure high-quality text generation.

Repetitions Greedy decoding from language models often lead to texts with excessive repetitions both at token- and sentence-levels. Several techniques have been proposed to solve this problem. Token loss dynamic reweighting (TLDR) hypothesizes that some tokens are more difficult to learn

than others and so reweighting tokens during learning can balance things to reduce repetitions (Jiang et al., 2020). Keskar et al. (2019) use a repetition penalty in decoding to reduce repetition of tokens. Welleck et al. (2020) suggest the cause for repetitions is a flaw in the training objective itself and use a new objective that gives less probability to unlikely sequence including texts with high repetitions. Li et al. (2016); Vijayakumar et al. (2018); Kulikov et al. (2019) propose variants of beam search to induce diversity in generated text, which in turn alleviates the problem of repetition. Variants of top-k sampling and repetition penalty from (Keskar et al., 2019) were also used by Foster & White (2007) to reduce repetitions. Here, we demonstrate a near-linear relation between repetitions and observed cross-entropy and so we directly control repetitions by controlling observed cross-entropy.

2 Surprise, Cross-Entropy, and Perplexity

Here we formally define surprise, cross-entropy, and perplexity. For a random variable $X \in \mathcal{X}$ distributed as P, the surprisal associated with an instance x of X is defined as $-\log P(x)$ (Han & Kobayashi, 2007). Hence, less probable instances are more surprising than more probable instances. Extending the definition to conditional random variables, we next define the surprise associated with tokens and sentences with respect to generated text for a fixed model distribution P_M.

Definition 1. The *surprise* value of a token X with respect to generated text $X_{<i}$ and model distribution P_M for some fixed model M is $\mathfrak{S}_M(X|X_{<i}) = -\log P_M(X|X_{<i})$.

We will soon see this quantity is directly related to perplexity. Now we define the average surprise for a sentence X with n tokens.

Definition 2. For a sentence $X^n = (X_1, \ldots, X_n)$ with n tokens, the *surprise rate* with respect to a probability distribution P_M for some model M is $\overline{\mathfrak{S}}_M(X^n) = -\frac{1}{n}\sum_{i=1}^{n} \log P_M(X_i|X_{<i})$.

The cross-entropy of a discrete random variable $X \in \mathcal{X}$ distributed as P_M with respect to a discrete random variable $Y \in \mathcal{Y}$ distributed as P_N such that $\mathcal{Y} \subseteq \mathcal{X}$ is

$$H(P_N, P_M) = -\sum_{y \in \mathcal{Y}} P_N(y) \log P_M(y) = \mathbb{E}_{P_N}[\mathfrak{S}_M(Y)]. \tag{1}$$

The cross-entropy rate of a stochastic process $\mathcal{X} = \{X_i\}, X_i \in \mathcal{X}$ distributed as P_M with respect to a stochastic process $\mathcal{Y} = \{Y_i\}, Y_i \in \mathcal{Y}$ distributed as P_N and $\mathcal{Y} \subseteq \mathcal{X}$ is defined as

$$\mathcal{H}(P_N, P_M) = \lim_{n \to \infty} \mathbb{E}_{P_N}[\overline{\mathfrak{S}}_M(Y^n)], \tag{2}$$

when the limit exists. Further, if Y^n is sampled from P_N and if P_N is a stationary ergodic source, then by the Shannon-McMillan-Breiman theorem (Cover & Thomas, 2006, Thm. 16.8.1), we have

$$\lim_{n \to \infty} \overline{\mathfrak{S}}_M(Y^n) = \mathcal{H}(P_N, P_M), \tag{3}$$

when the limit exists. Now, the perplexity corresponding to $\mathcal{H}(P_N, P_M)$ is simply

$$\text{PPL}(P_N, P_M) = 2^{\mathcal{H}(P_N, P_M)}, \tag{4}$$

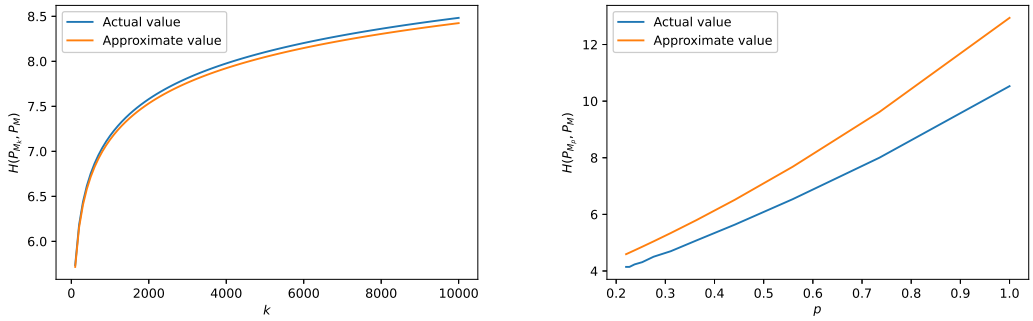
following Brown et al. (1992); Varshney et al. (2020). For experiments, when the text is generated using P_N, we approximate $\mathcal{H}(P_N, P_M)$ by $\overline{\mathfrak{S}}_M(Y^n)$ for a sentence of length of n. Perplexity denotes how close P_N is to P_M. The lower the perplexity, the closer the distributions P_N and P_M.

3 Theoretical Analysis of Sampling Methods

Here, we summarize our theoretical results for different sampling methods. Many more details and proofs are in Appendix B.

Zipf's law states that the frequency of occurrence of any word in the vocabulary is inversely proportional to its rank in the frequency table (Zipf, 1949; Powers, 1998). More precisely, for a vocabulary of size $N = |\mathcal{V}|$ the frequency of the ith most probable word is

$$p(i; s, N) = 1/(i^s H_{N,s}), \tag{5}$$



(a) (b)

Figure 1: Theoretical analysis of cross-entropy in top-k and top-p sampling: (a) approximation to $H(P_{M_k}, P_M)$ obtained in Thm. 2, (b) approximation to $H(P_{M_p}, P_M)$ obtained in Thm. 4 for $s = 1.1$, $N = 50,000$. Note that $H(P_{M_k}, P_M)$ grows sharply for small values of k and the growth is negligible for large values of k, whereas $H(P_{M_p}, P_M)$ grows nearly linearly with p.

where s is an exponent characterizing the distribution and $H_{N,s} = \sum_{n=1}^{N} \frac{1}{n^s}$ is the Nth generalized harmonic number. Further, for human languages the exponent s is very close to 1. Hence, when required, we write $s = 1 + \epsilon$, for some small $\epsilon > 0$. For all of our theoretical analysis, we assume the sampled words follow Zipf's law.

First we summarize results for top-k sampling. Thm. 1 shows that $\mathfrak{S}(k)$ grows steeply for small values of k, but grows very slowly for large values of k. Thm. 2 computes an approximation for $H(P_{M_k}, P_M)$; Fig. 1a shows this approximation is very close to the actual value. Note that since $H(P_{M_k}, P_M)$ does not grow much beyond $k = 2000$, it makes sense to tune k between 1 to 2000 to get a desired cross-entropy.

Now we summarize the results for top-p sampling. Thm. 3, proves that $\mathfrak{S}(p)$ behaves near-linearly in p. Further, Thm. 4 provides approximate expressions for $H(P_{M_p}, P_M)$ that show $H(P_{M_p}, P_M)$ grows approximately linearly with p; this approximate linearity is also shown in Fig. 1b. This is in contrast to top-k sampling where $H(P_{M_k}, P_M)$ is highly nonlinear.

Temperature is used to suitably distort the original distribution in so as to generate samples that avoid problems associated with pure sampling. In particular, lowering the temperature makes the sampling more greedy. For a given temperature $T > 0$, the frequency of the kth most probable word in (5) is given by $p(k; s, N, T) = 1/\left(k^{\frac{s}{T}} H_{N,\frac{s}{T}}\right) = p(k; \frac{s}{T}, N)$. Hence the effect of temperature in our analysis is captured simply by modifying s to $\frac{s}{T}$.

4 Perplexity-Controlled Text Generation

In this section we propose the mirostat algorithm to directly control the cross-entropy rate of the generated text. Mirostat works in two stages for generating each word. First it estimates the value of s assuming words follow Zipf's law. Then, it uses top-k sampling where k is a function of the estimated s and of the target surprise value of the output text.

We assume words follow Zipf's distribution (5). Further, we observe the probabilities produced by our language model as $\{p^{obs}(1), \ldots, p^{obs}(i), \ldots, p^{obs}(N)\}$. We use minimum mean-squared error (MMSE) estimation to find the value of s. However, s shows up in $p(k; s, N)$ both as an exponent of k and in $H_{N,s}$ which makes the computation difficult. Hence we estimate by minimizing MSE between logarithm of ratios of subsequent probabilities which eliminates $H_{N,s}$, i.e. we estimate s as

$$\hat{s} = \arg\min_{s} \sum_{i=1}^{N-1} (st_i - b_i)^2 = \frac{\sum_{i=1}^{N-1} t_i b_i}{\sum_{i=1}^{N-1} t_i^2}, \tag{6}$$

where $t_i = \log \frac{i+1}{i}$ and $b_i = \log \frac{p^{obs}(i)}{p^{obs}(i+1)}$. When N is large, we estimate s using the most probable m tokens for m around 100 to improve time complexity, which gives a practically good estimate.

4.1 ALGORITHM

Alg. 1 lists the details of mirostat,which generates texts with predetermined average surprise value. The input is a target surprise value τ, which in turn initializes a variable $\mu = 2\tau$. Each word is sampled by first estimating s from (6) as $\hat{s}$, then using top-k sampling by approximating k as a function of $\hat{s}$ and μ according to $H_{N,s} \approx \int_1^N \frac{1}{t^s} dt = \frac{(1-N^{s-1})}{s-1}$ and using (8) to get

$$k = \left(\frac{\hat{\epsilon} 2^{\mu}}{1 - N^{-\hat{\epsilon}}} \right)^{\frac{1}{\hat{s}}}, \tag{7}$$

where $\hat{\epsilon} = \hat{s} - 1$. We initialize k corresponding to surprise value 2τ and not τ since we are sampling from top-k and not computing the surprise value at k itself. This 2τ initialization works well in practice, and the rest is taken care of by feedback: An error term e is computed as the difference between the observed surprise $\mathfrak{S}(X)$ of the sampled word X and τ, which then is used to update μ.

Algorithm 1: Mirostat sampling for perplexity control

Target cross-entropy τ, maximum cross-entropy $\mu = 2\tau$, learning rate η, $m = 100$
while *more words are to be generated* **do**
  Compute $\hat{s}$ from (6): $\hat{s} = \sum_{i=1}^{m-1} t_i b_i / \sum_{i=1}^{m-1} t_i^2$
  Compute k from (7): $k = \left(\hat{\epsilon} 2^{\mu} / (1 - N^{-\hat{\epsilon}})\right)^{(1/\hat{s})}$
  Sample the next word X using top-k sampling
  Compute error: $e = \mathfrak{S}(X) - \tau$
  Update μ: $\mu = \mu - \eta e$
end

Note that we can use an alternate algorithm to tune k in Alg. 1 by iterating through the most probable tokens to set k corresponding to a token that has a suitable amount of surprise. This alternate algorithm would have varying computational complexity depending on the target surprise value and can be slow for large surprise values. Hence, in Alg. 1, we first estimate s using a fixed number of tokens and then directly use top-k sampling, which is computationally efficient for all target surprise values and provides good control over perplexity.

5 EXPERIMENTAL ANALYSIS

Here we provide experiments for the performance of top-k, top-p, and mirostat sampling. We use the GPT-2 language model with 117M parameters for all experiments (Radford et al., 2019) unless mentioned otherwise, and just refer to it as GPT-2. One main takeaway is that unlike other approaches, mirostat indeed provides direct control over the observed cross-entropy of the output text.

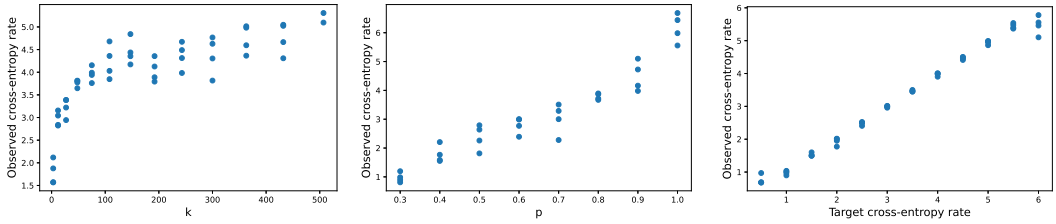
5.1 CROSS-ENTROPY RATE FOR DIFFERENT SAMPLING METHODS

Fig. 2 plots observed cross-entropy in generated texts versus several input parameters for different sampling methods. For each plot, we generate four output texts of 200 tokens corresponding to each value of input parameter in each sampling method with same context in each case.

Fig. 2a shows the observed surprise values in generated texts versus k in top-k sampling. Note that cross-entropy has a steep increase for small values of k and relatively slow increase in k for high values of k. Thus, for small values of k, cross-entropy is very sensitive to change in the k, but, for large values of k, cross-entropy hardly changes. Even though we can clearly see the increase in cross-entropy with increase in k, it is difficult to control cross-entropy using top-k sampling.

Fig. 2b plots the observed surprise values in generated text versus p in top-p sampling. Observe that cross-entropy grow essentially linearly with increase in p unlike top-k sampling.

Fig. 2c plots the observed cross-entropy in generated texts versus target cross-entropy in mirostat sampling, Alg. 1. Observe that mirostat sampling gives very good control over observed surprise



(a) (b) (c)

Figure 2: Cross-entropy rate in different sampling methods: (a) top-k, (b) top-p, and (c) mirostat. Note that cross-entropy rate is nonlinear in top-k sampling and nearly linear in top-p sampling. Also note that for a fixed input parameter in top-k and top-p sampling, observed cross-entropy rate fluctuates much, whereas mirostat shows negligible fluctuation in observed cross-entropy rate.

value, with low variance for surprise values less than five. For higher target surprise, Alg. 1 saturates in controlling observed surprise, since the algorithm truncates low probability words to control surprise value (and the baseline surprise without any truncation is around five). Thus, to get better control over observed surprise values, we must truncate some more probable words as well, which would reduce the quality of the generated text, hence not considered here.

The observation on different growth rate of surprise values in top-k and top-p sampling in Fig. 2 is not very intuitive on its own. Our theoretical analysis in Sec. 3 helps explain nonlinear growth in cross-entropy rate in top-k sampling and essentially linear growth in cross-entropy rate in top-p sampling. Note that our theoretical analysis in Sec. 3 deals with cross-entropy while our experiments deal with cross-entropy rate. However, for practical purposes cross-entropy helps us give an intuition about cross-entropy rate in different sampling methods. Further, the fluctuation in cross-entropy rate in Fig. 2a and Fig. 2b is likely because 200 tokens is not sufficient for convergence in (2). There is not much fluctuation in cross-entropy rate in Fig. 2c because we use feedback to control the cross-entropy rate more accurately, which gives accurate results even for a small number of tokens.

5.2 Perplexity and repetitions

Here, we present some experimental observations for percentage of repeated tokens across different sampling methods and language models. In Fig. 3, we generate texts with 200 tokens using different sampling methods and models with varying relevant input parameters such as k, p, or target surprise values, τ. We also consider the percentage of n-gram repetitions for different values of n for a fixed sampling method. We define percentage n-gram repetition as $\left(1 - \frac{\text{number of distinct } n\text{-gram tokens}}{\text{total number of } n\text{-gram tokens}}\right) \times 100$, where an n-gram token simply means concatenation of n contiguous tokens. Hence, for $n = 1$, n-gram repetitions capture word-level repetitions, whereas larger values of n capture sentence-level repetitions. For $n = 1$, we refer to percentage 1-gram repetition simply as percentage repetition.

In Fig. 3a, we observe that percentage repetition decreases with increase in cross-entropy and more importantly, for a fixed GPT-2 model, this relation is independent of the sampling method.

In Fig. 3b, we use top-k sampling with varying temperature values. We observe that repetitions for different temperature values and k follow the same curve as in Fig. 3a. This implies cross-entropy controls the percentage repetitions in generated texts. Moreover, it implies that once the model and cross-entropy are fixed, percentage repetition is not affected by the considered sampling methods.

In Fig. 3c, we capture n-gram repetitions for varying cross-entropy rate and different values of n. We note from Fig. 3c that for small values of n, the percentage n-gram repetitions drop almost linearly with increase in cross-entropy, whereas for larger values of n, the percentage n-gram repetitions is very close to zero for cross-entropy greater than 3. This indicates sentence-level repetitions disappear after a threshold of cross-entropy whereas word-level repetitions continue to appear for larger values of cross-entropy. Also, note that in human-generated text data, there are often common pronouns and conjunctions that are essential and are often repeated, hence we do not expect a good sampling algorithm to have absolutely zero 1-gram repetitions. But, we do expect a good

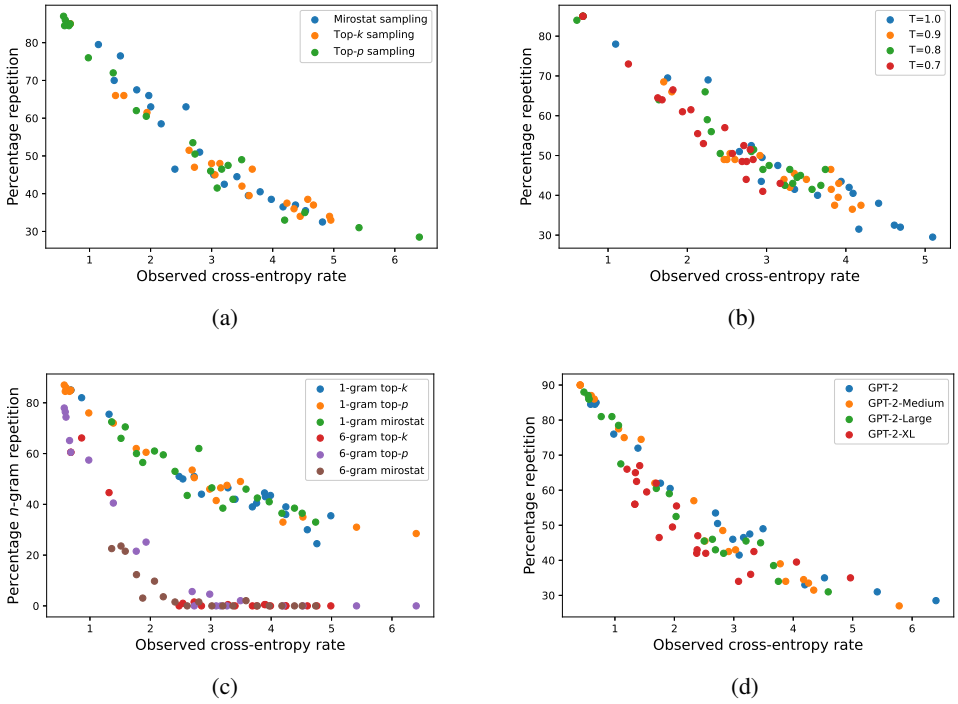


Figure 3: Percentage repetition vs. observed cross-entropy rate for (a) different sampling methods, (b) different temperature values, T, (c) for n-gram tokens and different sampling methods, (d) different language models. Note that repetitions vary heavily with observed cross-entropy rate in the generated text and mostly independent of sampling method used. Further, large models like GPT-2-XL with 1558M parameters seem to have slightly less repetitions for the same cross-entropy rate.

sampling algorithm to have minimum sentence-level repetitions, which all the sampling seems to show beyond a threshold of cross-entropy, which seems to be around 2.5 for GPT-2.

In Fig. 3d, we plot percentage repetition versus cross-entropy for different language models using top-p sampling for varying values of p. We observe that larger language models such as GPT-2-XL with 1558M parameters have slightly less repetitions for a fixed value of cross-entropy compared to smaller models such as GPT-2 with 117M parameters.

From these observations, we conclude that to control percentage repetition in generation we must control the cross-entropy of the output text, exactly like in mirostat.

5.3 Boredom and Confusion Traps

Here we show top-k and top-p sampling cannot control output and therefore get trapped into low-quality generation, for a wide range of k and p. We generated 10 samples of 900-token texts on the same context and averaged their observed cross-entropy at various points of generation in Fig. 4, except for the single-sample human-generated text (the tokens following the context in the corpus).

Fig. 4a illustrates that for small values of k and p, both top-k and top-p sampling methods fall into low cross-entropy regions—boredom traps—which results in increase in repetitions as the length of the generated text increases, as illustrated in Sec. 5.2. Hence, lack of control over output statistics in these methods leads to degradation of quality in generated texts for longer texts.

Fig. 4b shows that for high values of k and p in top-k and top-p sampling methods respectively, the observed cross-entropy of the generated texts increases with the length of generated texts. This leads to increase in incoherence in the text as the token index increases—the confusion trap.

In Fig. 4c we choose certain values of k and p in an ad hoc manner and generate texts using top-k and top-p sampling methods respectively to observe that for these values of k and p, the generated texts

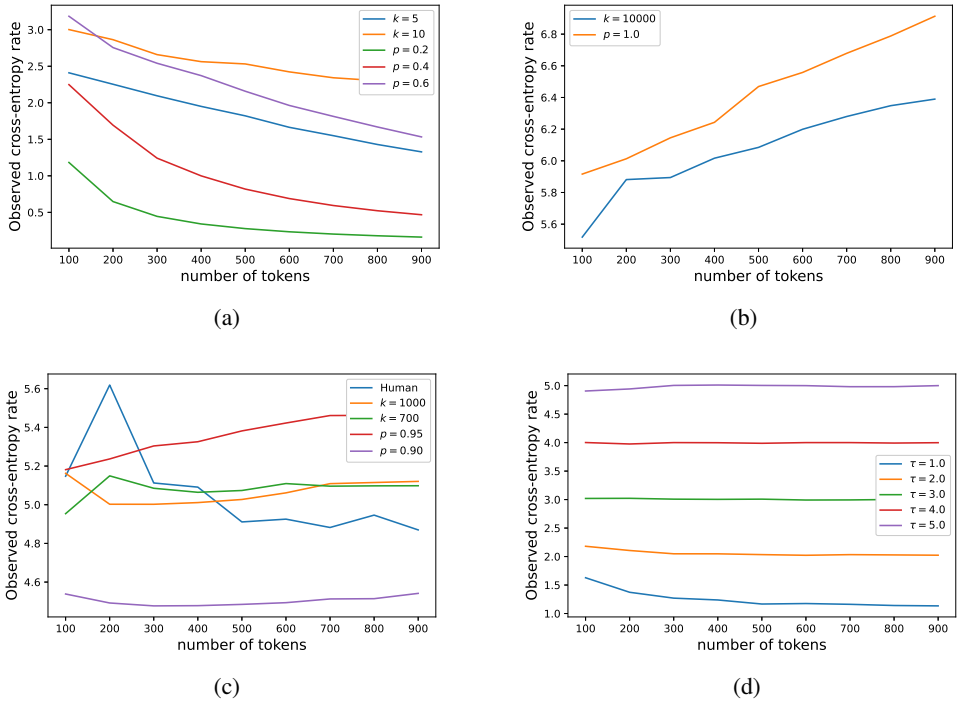


Figure 4: Cross-entropy rate vs. number of tokens generated for different sampling methods. We observe (a) boredom trap for small values of k and p, (b) confusion trap for large values of k and p, (c) human-like cross-entropy rate for moderate k and p, (d) mirostat sampling that shows control over cross-entropy over varying lengths of texts.

tend to have cross-entropy that seems to converge to a limiting value with increase in text length and not fall into either boredom or confusion traps. We also show how the observed cross-entropy varies with increase in text length in the human-generated text corresponding to the tokens following the context for these experiments. Human-generated text converges to some limiting value of cross-entropy when the generated text is long enough and does not fall into either boredom or confusion.

Finally, Fig. 4d shows cross-entropy for the mirostat-generated texts converges to the target cross-entropy within a few tokens and maintains the desired value of cross-entropy for long texts.

6 Conclusion

We provided a theoretical explanation of how perplexity varies as a function of input parameters in popular top-k and top-p neural text decoding algorithms, showing that log of perplexity varies nearly linearly as a function of p and a highly nonlinearly as a function of k. Building on this analysis, we developed mirostat, a neural text decoding algorithm that directly controls the perplexity of the generated text over a wide range of text length. Notably, for longer texts and certain ranges of input parameters, top-k and top-p sampling fall into boredom and confusion traps which cause low-quality texts; Mirostat avoids both traps. Further, recent large-scale human evaluation of neural generated text suggests that human-judged text quality is maximized for a certain range of perplexity of the output: mirostat provides direct control to stay in that perplexity range. There are also implications for data compression as given in Appendix A.2. As a takeaway, we find that mirostat with target surprise around 3.0, produces varying lengths of high-quality texts with minimal repetitions.

Additionally, we also analyze the relation between perplexity and repetitions in text. We find that for fixed model, repetitions vary linearly with perplexity and are independent of the sampling method used. We also find that larger models have less repetitions in them for any fixed amount of perplexity. Future work would include theoretically analyzing repetitions, boredom and confusion traps, and convergence properties of mirostat.

REFERENCES

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. An estimate of an upper bound for the entropy of English. *Comput. Linguist.*, 18(1): 31–40, 1992.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. arXiv:2005.14165v3 [cs.CL]., June 2020.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 2 edition, 2006.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: a simple approach to controlled text generation. In *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, April 2020.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proc. Assoc. Comput. Linguist. Annu. Meet. (ACL 2018)*, pp. 889–898, July 2018. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/P18-1082.

Mary Ellen Foster and Michael White. Avoiding repetition in generated text. In *Proc. Conf. Eleventh European Workshop Natural Language Generation. (ENLG'07)*, pp. 33–40, June 2007. URL https://www.aclweb.org/anthology/W07-2305.

Edgar N. Gilbert. Codes based on inaccurate source probabilities. *IEEE Trans. Inf. Theory*, IT-17 (3):304–314, May 1971. doi: 10.1109/TIT.1971.1054638.

Te Sun Han and Kingo Kobayashi. Mathematics of information and coding. *Ann. Math. Stat.*, 203, March 2007.

Tatsunori Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation. In *Proc. NAACL-HLT 2019*, pp. 1689–1701, June 2019.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. In *Proc. Assoc. Comput. Linguist. Annu. Meet. (ACL 2018)*, pp. 1638–1649, July 2018. doi: 10.18653/v1/P18-1152. URL https://www.aclweb.org/anthology/P18-1152.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, May 2020.

Daphne Ippolito, Reno Kriz, Maria Kustikova, João Sedoc, and Chris Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proc. Assoc. Comput. Linguist. Annu. Meet. (ACL 2019)*, pp. 3752–3762, July 2019.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of generated text is easiest when humans are fooled. In *Proc. Assoc. Comput. Linguist. Annu. Meet. (ACL 2020)*, pp. 1808–1822, July 2020.

Shaojie Jiang, Thomas Wolf, Christof Monz, and Maarten de Rijke. TLDR: Token loss dynamic reweighting for reducing repetitive utterance generation. arXiv:2003.11963 [cs.CL]., April 2020.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. arXiv:1909.05858v2 [cs.CL]., September 2019.

Ilia Kulikov, Alexander Miller, Kyunghyun Cho, and Jason Weston. Importance of search and evaluation strategies in neural dialogue modeling. In *Proc. 12th Int. Conf. Natural Language Generation (ICNLG 2019)*, pp. 76–87, October 2019.

Jiwei Li, Will Monroe, and Dan Jurafsky. A simple, fast diverse decoding algorithm for neural generation. arXiv:1611.08562 [cs.CL]., November 2016.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. CGMH: Constrained sentence generation by metropolis-hastings sampling. In *Proc. 33rd AAAI Conf. Artif. Intell.*, July 2019.

David M. W. Powers. Applications and explanations of Zipf's law. In *New Meth. Language Process. and Comp. Natural Language Learning*, pp. 151–160, 1998.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *Unpublished manuscript*, February 2019. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Jorma J. Rissanen and Glen G. Langdon. Arithmetic coding. *IBM J. Res. Develop.*, 23(2):149–162, March 1979.

Lav R. Varshney, Nitish Shirish Keskar, and Richard Socher. Limits of detecting text generated by large-scale language models. In *Proc. 2020 Inf. Theory Appl. Workshop*, February 2020.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. In *Proc. 32nd AAAI Conf. Artif. Intell.*, April 2018.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, May 2020.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, June 1987.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. Trading off diversity and quality in natural language generation. arXiv:2004.10450v1 [cs.CL]., April 2020.

George K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.

A EXAMPLES

Here we describe two examples. The first example demonstrates the relation between surprise, repetitions, and incoherence. The second example demonstrates the relation between text compression and cross-entropy.

A.1 RELATION BETWEEN SURPRISE, REPETITIONS, AND INCOHERENCE

Example 1. *We generate four samples of texts with 200 tokens (which are words or subwords that are part of the vocabulary of the model) using GPT-2 model with 117M parameters Radford et al. (2019). For the first two samples we use top-p sampling with $p = 0.4$, for the third sample we use top-p sampling with $p = 1.0$, and for the fourth sample we use our proposed mirostat sampling algorithm. The fifth example is human-generated, taken from the same corpus as the context. The context used is as follows.*

Context: *"For two months early in 1943, Shannon came into contact with the leading British mathematician Alan Turing. Turing had been posted to Washington to share with the U.S. Navy's cryptanalytic service the methods used by the British Government Code and Cypher School at Bletchley Park to break the ciphers used by the Kriegsmarine U-boats in the north Atlantic Ocean. He was also interested in the encipherment of speech and to this end spent time at Bell Labs."*

1. ***Observed average surprise value = 1.471 p = 0.4***

   Generated text: *"Turing's work on the cryptography of the Kriegsmarine was based on the cipher used by the Kriegsmarine's famous E-2. The Kriegsmarine code was based on the Cryptographic*

Algorithm (CAT) of the Royal Navy. It was a new method of solving the problem of encryption, which was a mathematical problem which was the basis of the cryptanalysis of the Japanese naval cipher.

Turing's work on the cipher was based on the algorithm used by the Kriegsmarine's famous E-2. The cipher was based on the algorithm used by the Kriegsmarine's famous E-2.

Turing's work on the cipher was based on the algorithm used by the Kriegsmarine's famous E-2.

Turing's work on the cipher was based on the algorithm used by the Kriegsmarine's famous E-2.

Turing's work on the cipher was based on"

2. ***Observed average surprise value = 1.887, p = 0.4***

   Generated text: *"Turing was a great mathematician, and he was not a professional cryptographer. But he was a very talented cryptographer. He was also a highly intelligent man. He was a great man who was very well-versed in the law of probability. He was also very well-versed in the mathematical methods of the United States Army. He was also very well-versed in the mathematics of the Russian Academy of Sciences. He was also very well-versed in the mathematical methods of the German Academy of Sciences.*

   In 1943, Shannon traveled to Moscow to meet with the Director of the NSA. He was told that the FBI had obtained information from the NSA about the German secret police. He was told that the FBI had obtained information from the NSA about the German secret police. He was told that the FBI had obtained information about the German secret police.

   The NSA, on the other hand, had been unable to get information from the NSA about the German secrete"

3. ***Observed average surprise value = 6.342 p = 1.0***

   Generated text: *"In January of 1942, in an open letter delivered to 8 persons and addressed to Sherwood, Shannon took the opportunity to address skeptics about the little out-of-print $128,000 unreported race as "cryptanalysis apparatus of extraterrestrial intelligence." His reply was also reprinted in 6 South Carolina publications (Atlanta Press, Fall Publishing, Miami, New House Press, Yorkshire Press, Sydney Press, Weatherby Press, Winter Press, and Thames & Hudson, to name a few), and eventually accepted as the final subscribed FHAT article. (Summary, were whom)*

   Birmingham City Police concludes that [Turing's][new replace Robert] Symington is not guilty of breach of OKF law. Further, his suggestions for preventing his sentencing Should he reside anywhere near the city he defies extradition to Bavaria closer to dawn minimal? Lylimud Simpson (Ladies & Gentlemen, Lawrence Livermore University Press, Philadelphia): Jim Gilmore and its wife, Eva Civilsky,"

4. ***Observed average surprise value = 1.828 mirostat target surprise = 1.8***

   Generated text: *"In April 1943, Shannon was assigned to the U.S. Naval Intelligence Center at Naval Air Station Norfolk, Virginia, where he was assigned to work on a project to develop a new cipher for the U.S. Navy's cryptanalysis program. The project was to be conducted by the U.S. Navy's cryptanalysis team.*

   In early April, the U.S. Navy was informed that Shannon had been assigned to work on an encrypted cipher for the Kriegsmarine's U-boat. The project was to be conducted by a group of cryptographers from the U.S. Naval Intelligence Center. The project was to be conducted by a group of cryptographers from the U.S. Naval Intelligence Center in a project that could be executed by the cryptanalysts.

   In early April, Shannon was ordered by the U.S. Navy to provide the cryptanalysts with information on a cipher for the U-boat's U-boats that could"

5. ***Observed average surprise value = 5.285 human-generated***

Generated text: *"Shannon and Turing met at teatime in the cafeteria. Turing showed Shannon his 1936 paper that defined what is now known as the 'Universal Turing machine'. This impressed Shannon, as many of its ideas complemented his own.*

In 1945, as the war was coming to an end, the NDRC was issuing a summary of technical reports as a last step prior to its eventual closing down. Inside the volume on fire control, a special essay titled Data Smoothing and Prediction in Fire-Control Systems, coauthored by Shannon, Ralph Beebe Blackman, and Hendrik Wade Bode, formally treated the problem of smoothing the data in fire-control by analogy with 'the problem of separating a signal from interfering noise in communications systems'. In other words, it modeled the problem in terms of data and signal processing and thus heralded the coming of the Information Age. Shannon's work on cryptography was even more closely related to his later publications on communication theory. At the close of the war"

Figure 5 shows plots of surprise values against indices of tokens in each of the samples in Ex. 1. The blue plot corresponds to surprise values of each token, while the red plot corresponds to average surprise values over a window of size 10 at each token index. Note that the surprise values drop drastically in Fig. 5a as the repetitions increase in Ex. 5.1. Similarly, in Fig. 5b, we observe a dip in surprise values wherever there is a repetition in Ex. 5.2. Clearly, there is a correlation between small average surprise values and repetitions. Further, in Fig. 5a note that the generating model seems to get trapped into a small surprise repetition region. We call this region of small surprise as *boredom trap*. We observe that these models tend to fall into a boredom trap for small values of p. Figure 5c corresponds to Ex. 5.3, where we choose $p = 1.0$ and illustrate that for large values of p, the average surprise value of the generated text tends to increase with the number of generated tokens, which leads to incoherence. We call this region of large surprise a *confusion trap*. Figure 5d shows surprise values corresponding to Ex. 5.4 which is generated using our proposed sampling algorithm, mirostat. We observe in Fig. 5d that mirostat increases the surprise value when when falling into a boredom trap and, thereby maintaining the average surprise value. By doing so, it not only helps generate high-quality text with predetermined average surprise value, but also helps avoid small surprise repetition regions and large surprise incoherent regions. In Fig. 5e, we show the surprise values in human-generated text that followed this context as shown in Ex. 5.5. We observe that human-generated text has average surprise value that is between values using top-p sampling for $p = 0.4$ and $p = 1.0$. More importantly, human-generated text does not fall into either of the traps.

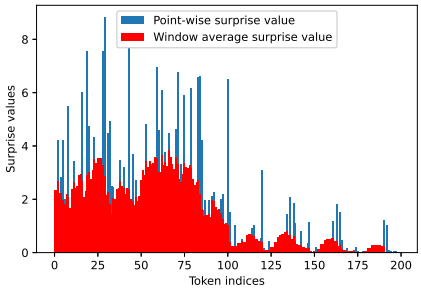
A.2 Text generation and compression

Here we will look at texts generated for various target surprise values using mirostat sampling with GPT-2 with 117M. We also observe the well-known relation between cross-entropy and data compression (Cover & Thomas, 2006, Ch. 5), (Gilbert, 1971). In particular, it is known that when the actual probability distribution of the generated text is not known, then the minimum lossless compression rate achievable is equal to the cross-entropy of the assumed distribution, which is the language model here, with respect to the actual unknown distribution, which is obtained from adaptive top-k sampling here.

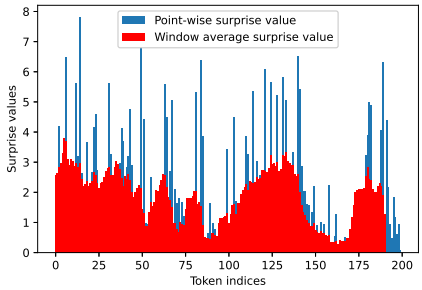
Example 2. *We generate 200 tokens for different values of target surprise values using the GPT-2 model with 117M parameters to show the quality of the text generated using Alg. 1 for different target surprise values. We also measure the compression rates obtained using arithmetic coding (Witten et al., 1987; Rissanen & Langdon, 1979) with the language model as the probability distribution. So, in a way, mirostat can generate text that has a predetermined minimum lossless compression rate for a given model.*

Context: *"For two months early in 1943, Shannon came into contact with the leading British mathematician Alan Turing. Turing had been posted to Washington to share with the U.S. Navy's cryptanalytic service the methods used by the British Government Code and Cypher School at Bletchley Park to break the ciphers used by the Kriegsmarine U-boats in the north Atlantic Ocean. He was also interested in the encipherment of speech and to this end spent time at Bell Labs."*

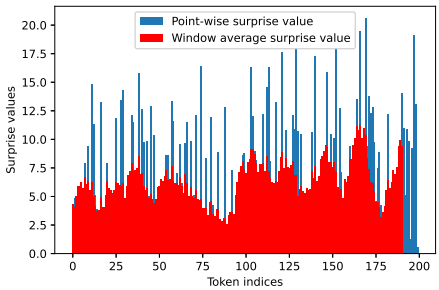
Target average surprise value = 1.0 Observed cross-entropy rate = 0.759 Average compressed size (per token) = 0.863 Percentage compression = 97.66%



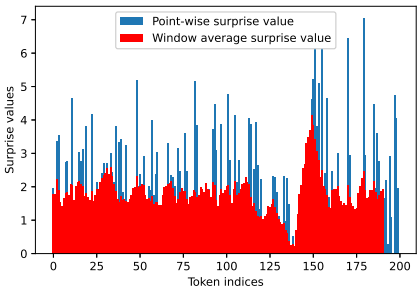
(a)



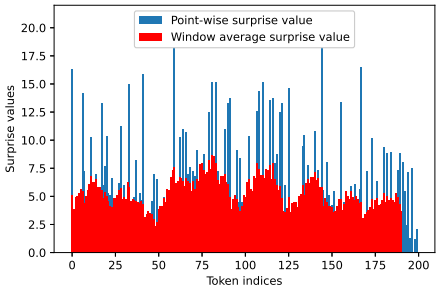
(b)



(c)



(d)



(e)

Figure 5: Relation between surprise values and attributes of generated text such as repetitions and incoherence: (a) top-p sampling with $p = 0.4$ and average observed surprise = 1.471, (b) top-p sampling with $p = 0.4$ and average surprise = 1.887, (c) top-p sampling with $p = 1.0$ and average observed surprise = 6.342, (d) mirostat sampling with target average surprise, $\tau = 1.8$ and average observed surprise = 1.828, (e) human-generated text, average observed surprise = 5.301. We observe that repetitions in generated texts are correlated with dips in surprise values, whereas incoherence is correlated with large and increasing surprise values with indices.

Generated text: *"In April 1943, Shannon was assigned to the U.S. Navy's cryptanalytic service, the Naval Intelligence Service. He was assigned to the cryptanalytic service's cryptanalysis unit, the Naval Intelligence Laboratory. He was also assigned to the cryptanalytic unit's cryptanalysis unit, the Naval Intelligence Laboratory.*

In May 1943, Shannon was assigned to the cryptanalytic unit's cryptanalysis unit, the Naval Intelligence Laboratory. He was also assigned to the cryptanalytic unit's cryptanalysis unit, the Naval Intelligence Laboratory.

In June 1943, Shannon was assigned to the cryptanalytic unit's cryptanalysis unit. He was assigned to the cryptanalytic unit's cryptanalysis unit, the Naval Intelligence Laboratory.

In July 1943, Shannon was assigned to the cryptanalytic unit's cryptanalytic unit. He was assigned to the cryptanalytic unit's cryptanalytic unit, the Naval Intelligence Laboratory.

In August 1943, Shannon was assigned"

Target average surprise value = 2.0 Observed cross-entropy rate = 1.987 Average compressed size (per token) = 2.07 Percentage compression = 95.13%
Generated text: *"He was a member of the British Intelligence Service, and was also a member of the British Intelligence Service's Intelligence Advisory Board.*

In early 1944, Shannon was assigned to the British Intelligence Service's Intelligence Advisory Board. He was also assigned the task of deciphering the ciphertext of the British Government Code. He was also assigned to the British Intelligence Service's Cryptanalytic Service. He was assigned to the British Intelligence Service's Intelligence Advisory Board at the end of 1944. He was assigned to the British Intelligence Service's Secret Intelligence Service at the end of 1944.

He was confirmed to work as the Director of the British Intelligence Service's Information Service at the end of 1944. He was the first British employee to be appointed as an Intelligence Advisory Officer at the end of the war. He was also confirmed as an Intelligence Advisory Officer to the United Kingdom's Intelligence Advisory Committee at the end of the war. (The United Kingdom is a member of the European Union.)

He was also appointed"

Target average surprise value = 3.0 Observed cross-entropy rate = 3.002 Average compressed size (per token) = 3.141 Percentage compression = 91.84%
Generated text: *"After a short stay at Bell Labs, Shannon was commissioned to report back to the Naval Intelligence Services (NISC) to learn more about his research. He made his way over to the Royal Naval Institute. He had already been working with a number of other cryptanalytic researchers, including the mathematician Alan Turing, for over eight years at Bell Labs, but this research was not yet done. He spent the next two months studying the cipher schemes of the various British cryptanalytic systems. The first step was to find out the actual cipher scheme for the cipher used by the HMS Queen Elizabeth on a submarine.*

By the time Shannon arrived at the Royal Naval Institute he had been working for Patagonia Scientific and Technology, the British cryptanalytic company, for eight years. Patagonia received a call at 9 A.D. on the evening of September 4, 1943, asking to check the cipher system for a cipher used by the HMS Queen Elizabeth, which is written as:"

Target average surprise value = 4.0 Observed cross-entropy rate = 3.943 Average compressed size (per token) = 4.06 Percentage compression = 89.98%
Generated text: *"He also began writing code to help with deciphering encrypted communications. During 1943, Shannon kept a diary to help decipher his instructions to the cryptanalytic service. Hearing of the story of Alan Turing sent Shannon excitedly into a panic of his own. He that he had only discovered Turing had made it clear to him that the great deception he had witnessed was due to the German government's efforts to conceal it. I believe he intended to reveal it. In the early days, the NSA was working with the National Security Agency (NSA) at different times against the Soviet Union in the early 90s and as recently as three years ago against the USSR in the early 1990s. It is believed that the four Russian spies were at the CIA while also an intermediary in the*

process of doing a disinformation campaign to manipulate the American public to support Bush Sr. by calling for American military intervention in Afghanistan against the Bush, Bush, and Cheney (and the Bush family) and by trying to persuade the public to"

Target average surprise value = 5.0 Observed cross-entropy rate = 4.929 Average compressed size (per token) = 5.116 Percentage compression = 85.41%
Generated text: *"Naval Contractors hired Shannon as a shortage specialist at the Navy. So she worked alone with Turing to decipher the British cipher for the northern German submarines. Shannon undertook the work initially on the S.G.F. dual cipher. Shannon spent the whole working days at the Bell Labs lab at Bletchley Park.*

After weeks of interrogations Shannon was able to break the great El-Fran jujito with a German accent. Shannon then calculated that this hydrocephalic consciousness alone would allow her to think the same words but without the huge amount of writing required to produce such a thing, and with millions of hours on board she was able to write some of the same phrases over a twenty-eight hours workweek.

Output: Individualist Hypothesis.

Miranda (alias Charlotte Reagmire, aka "The Lady in the Lake") (1945-2049) (wife of B.G. Lloyd) (Philadelphia, PA)"

Target average surprise value = 6.0 Observed cross-entropy rate = 5.735 Average compressed size (per token) = 5.834 Percentage compression = 85.55%
Generated text: *"The CIA trained a small band of cryptanalysts to do the maths again, this time using a UK number generator. A few days after the wars Belgium introduced Bermuda rugby as the appropriate sport for the National Guard. Notwithstanding there being no convention around using English for Rugby at the time, there would be no possible instance in the history of fencing in Europe. Flags for the Hurricanes had evolved recently using a Dutch Italian design called the Crazy Flag. These flag designs come largely of British origin and the date published of its introduction by the Royal Armouries of Lameucers is from 1638. The camouflage was recently added to the new Irish power orange flag. The design is based on the weapons bao mouèret Standard and has two coloured pouches connected to the rifle barrel by two checks along the top of the barrel with protection straps around the barrel to protect the cutouts. NATO hired a team of physicists to do the reconstruction. Readers who want to know more about this new"*

In Ex. 2 we can see that low value of surprise value results in repetitions and high value of surprise value results in incoherent generated texts. Moderate surprise values result in good quality, coherent text with no repetition. Also, note that the control does not work well when the target surprise value is greater then 5. This is because without any truncation, the average surprise of pure sampled text comes out to be around 5.4. Thus, in order to attain higher values of average surprise, we need to truncate from both sides of the distribution.

B THEORETICAL RESULTS

Theorem 1. *If words are sampled from the Zipf's distribution given by (5), then the surprise value of a word with rank k and its rate of increase are given by*

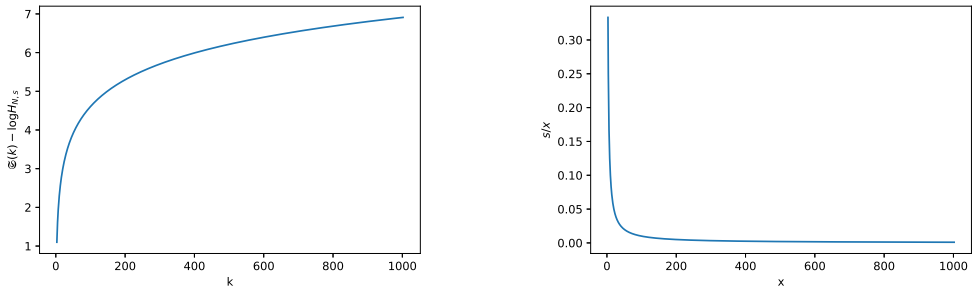
$$\mathfrak{S}(k) = s \log k + \log H_{N,s}, \tag{8}$$

$$\frac{d\mathfrak{S}(x)}{dx} = \frac{s}{x} \tag{9}$$

respectively, where $\mathfrak{S}(x)$ is a continuous function with the same expression as $\mathfrak{S}(k)$ with a continuous domain.

Proof. The expression of $\mathfrak{S}(k)$ follows directly from Def. 1 and (5). □

From Fig. 6, we note that $\mathfrak{S}(x)$ is highly sensitive to change in x for small values of x and its sensitivity to x decreases drastically with increase in x. Now, we analyze how cross-entropy varies



(a) Plot of $\mathfrak{S}(k) - \log H_{N,s}$ vs. k for $s = 1$, $N = 50,000$.

(b) Plot of s/x vs. x for $s = 1$, $N = 50,000$.

Figure 6: Theoretical analysis of surprise values in top-k sampling under Zipfian statistics. We note that surprise values increase sharply for small values of k, whereas surprise values hardly change for large values of k.

with k. Let P_M be the model distribution. Top-k sampling works by truncating the tail of the distribution P_M and samples from the most probable k tokens. Let the truncated distribution be denoted by P_{M_k}. In Prop. 1, we provide an expression for $H(P_{M_k}, P_M)$.

Proposition 1. *Let P_M be the model distribution satisfying (5) with vocabulary of size N and let P_{M_k} be the model distribution obtained by top-k sampling. Then $H(P_{M_k}, P_M)$ is given by*

$$H(P_{M_k}, P_M) = \frac{s}{H_{k,s}} \sum_{i=1}^{k} \frac{\log i}{i^s} + \log H_{N,s}. \tag{10}$$

Proof. The distribution P_M is given by (5) with vocabulary size N, and it is easy to check that the distribution P_{M_k} corresponding to top-k sampling is also given by (5) but with vocabulary size k. The rest follows directly from (1). □

It is difficult to get an intuition about the behavior of $H(P_{M_k}, P_M)$ directly from (10). Thus, in Thm. 2 we obtain an approximation to $H(P_{M_k}, P_M)$ that shows $H(P_{M_k}, P_M)$ is essentially of the form $c_1(1 - c_2 \frac{\ln k + c_3}{k^\epsilon - 1}) + c_4$ for $0 < \epsilon < \frac{1}{\ln 2}$, where c_1, c_2, c_3, c_4 are some constants. Hence we observe that $H(P_{M_k}, P_M)$ grows fast with small values of k and slows down for large values of k.

Theorem 2. *Let P_M be the model distribution satisfying (5) with vocabulary of size N. and let P_{M_k} be the model distribution obtained using top-k sampling. Then, for $1 < s \leq \frac{1}{\ln 2}$, $H(P_{M_k}, P_M)$ can be approximated as*

$$H(P_{M_k}, P_M) \approx \frac{b_1 \epsilon}{b_3} \left(1 - \frac{b_2 b_3 (\ln k + \frac{1}{\epsilon}) - b_1}{b_1 (b_3 k^\epsilon - 1)} \right) + \log H_{N,s}, \tag{11}$$

where $b_1 = s \left(\frac{\log 2}{2^{1+\epsilon}} + \frac{\log 3}{3^{1+\epsilon}} + \frac{1}{\epsilon (\ln 2) 3^\epsilon} \left(\ln 3 + \frac{1}{\epsilon} \right) \right)$, $b_2 = \frac{s}{\epsilon \ln 2}$, and $b_3 = 1 + 0.7\epsilon$ are constants.

Proof. From Prop. 1 we have $H(P_{M_k}, P_M) = \frac{s}{H_{k,s}} \sum_{i=1}^{k} \frac{\log i}{i^s} + \log H_{N,s}$. We start by finding bounds for the expression $\sum_{i=1}^{k} \frac{\log i}{i^s}$.

First note that the function $\frac{\log t}{t^s}$ is a decreasing function of t for $t > e^{\frac{1}{s}}$. Thus, for $1 \leq s \leq \frac{1}{\ln 2}$, we have the following inequalities

$$\frac{\log 2}{2^s} + \int_3^{k+1} \frac{\log t}{t^s} dt \leq \sum_{i=1}^{k} \frac{\log i}{i^s} \leq \frac{\log 2}{2^s} + \frac{\log 3}{3^s} + \int_4^{k+1} \frac{\log (t-1)}{(t-1)^s} dt. \tag{12}$$

Solving the above integration for $1 < s \leq \frac{1}{\ln 2}$ we get

$$\frac{a_1}{H_{k,s}} - \frac{a_2}{H_{k,s}(k+1)^{\epsilon}}\left(\ln(k+1) + \frac{1}{\epsilon}\right) \leq H(P_{M_k}, P_M) - \log H_{N,s} \leq \frac{b_1}{H_{k,s}} - \frac{b_2}{H_{k,s}k^{\epsilon}}\left(\ln k + \frac{1}{\epsilon}\right), \tag{13}$$

where $a_1 = s\left(\frac{\log 2}{2^{1+\epsilon}} + \frac{1}{\epsilon(\ln 2)3^{\epsilon}}\left(\ln 3 + \frac{1}{\epsilon}\right)\right)$, $a_2 = \frac{s}{\epsilon \ln 2}$, $b_1 = s\left(\frac{\log 2}{2^{1+\epsilon}} + \frac{\log 3}{3^{1+\epsilon}} + \frac{1}{\epsilon(\ln 2)3^{\epsilon}}\left(\ln 3 + \frac{1}{\epsilon}\right)\right)$, $b_2 = \frac{s}{\epsilon \ln 2}$.

Now, we bound $H_{k,s}$ as follows. Note that $\frac{1}{t^s}$ is a decreasing function in t for $t > 0$ and $s > 0$, hence, we have

$$\int_1^{k+1} \frac{1}{t^s} dt \leq \sum_{i=1}^{k} \frac{1}{i^s} \leq 1 + \int_2^{k+1} \frac{1}{(t-1)^s} dt \tag{14}$$

$$\frac{1-(k+1)^{-\epsilon}}{\epsilon} \leq \sum_{i=1}^{k} \frac{1}{i^s} \leq 1 + \frac{1-k^{-\epsilon}}{\epsilon}. \tag{15}$$

We empirically observed that $H_{k,s}$ can be approximated well as

$$H_{k,s} \approx 0.7 + \frac{1-k^{-\epsilon}}{\epsilon}, \tag{16}$$

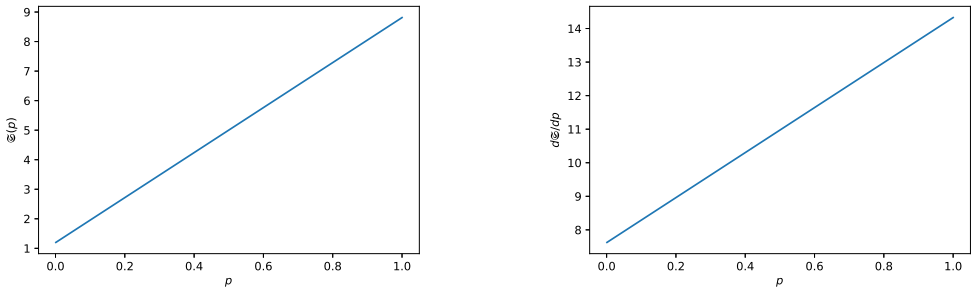
which lies between the bounds found in (15). Moreover, we approximate $H(P_{M_k}, P_M)$ using the upper bound obtained in (12) to get

$$H(P_{M_k}, P_M) \approx \frac{1}{H_{k,s}}\left(b_1 - \frac{b_2}{k^{\epsilon}}\left(\ln k + \frac{1}{\epsilon}\right)\right) + \log H_{N,s} \tag{17}$$

$$\approx \frac{\epsilon}{b_3(1-\frac{k^{-\epsilon}}{b_3})}\left(b_1 - \frac{b_2}{k^{\epsilon}}\left(\ln k + \frac{1}{\epsilon}\right)\right) + \log H_{N,s} \tag{18}$$

$$\approx \frac{b_1\epsilon}{b_3}\left(1 - \frac{b_2 b_3(\ln k + \frac{1}{\epsilon}) - b_1}{b_1(b_3 k^{\epsilon} - 1)}\right) + \log H_{N,s}, \tag{19}$$

where (19) follows by writing $\frac{1}{(1-\frac{k^{-\epsilon}}{b_3})}$ as an infinite series in (18), then simplifying the expression and writing the infinite series back as a fraction. □



(a) Plot of $\mathfrak{S}(p)$ vs. p for $s = 1.2$, $N = 50,000$.

(b) Plot of $\frac{d\mathfrak{S}(p)}{dp}$ vs. p for $s = 1.2$, $N = 50,000$.

Figure 7: Theoretical analysis of surprise values in top-p sampling. We observe that unlike top-k sampling, surprise values grow linearly with p in top-p sampling.

In Thm. 3, we provide approximate expressions for $\mathfrak{S}(p)$ and $\frac{d\mathfrak{S}(p)}{dp}$ that shows that $\mathfrak{S}(p)$ grows essentially linearly with p.

Theorem 3. *If words are sampled from the Zipf's distribution given by (5). If $\epsilon > 0$ is a small constant, then $\mathfrak{S}(p)$ and the rate of change of $\mathfrak{S}(p)$ with respect to p is given by*

$$\mathfrak{S}(p) \approx \frac{(1+\epsilon)}{b \ln 2} H_{N,s} p - \frac{(1+\epsilon)}{\epsilon} \log b + \log H_{N,s} \tag{20}$$

$$\frac{d\mathfrak{S}(p)}{dp} \approx \frac{(1+\epsilon) H_{N,s}}{b \ln 2} (1 + \frac{H_{N,s} \epsilon p}{b}), \tag{21}$$

where $b = 1 + 0.7\epsilon$.

Proof. The cumulative probability $p(k)$ for Zipf's distribution is given by $p(k) = \frac{H_{k,s}}{H_{N,s}}$. Using the approximation to $H_{k,s}$ in (16), we have

$$p(k) = \frac{b - k^{-\epsilon}}{\epsilon H_{N,s}}, \tag{22}$$

where $b = 1 + 0.7\epsilon$.

Now, writing k as a function of p, we get

$$k = (b - \epsilon p H_{N,s})^{-\frac{1}{\epsilon}}. \tag{23}$$

Using (23) in the equation $\mathfrak{S}(x) = s \log x + \log H_{N,s}$ from Thm. 1, we get

$$\begin{aligned} \mathfrak{S}(p) &= -\frac{1+\epsilon}{\epsilon} \log (b - H_{N,s} \epsilon p) + \log H_{N,s} \\ &= -\frac{1+\epsilon}{\epsilon \ln 2} \ln (1 - \frac{H_{N,s} \epsilon p}{b}) - \frac{1+\epsilon}{\epsilon} \log b + \log H_{N,s}. \end{aligned} \tag{24}$$

Further, taking ϵ small enough, we can approximate $\ln (1 - \frac{H_{N,s}\epsilon p}{b}) \approx -\frac{H_{N,s}\epsilon p}{b}$. Thus, we have

$$\mathfrak{S}(p) \approx \frac{(1+\epsilon)}{b \ln 2} H_{N,s} p - \frac{(1+\epsilon)}{\epsilon} \log b + \log H_{N,s}. \tag{25}$$

Now, $\frac{d\mathfrak{S}(p)}{dp}$ can be directly computed from (24) as

$$\frac{d\mathfrak{S}(p)}{dp} = \frac{H_{N,s}(1+\epsilon)}{\ln 2 (b - H_{N,s}\epsilon p)}. \tag{26}$$

For ϵ small enough, we can use the approximation $\frac{1}{1 - \frac{H_{N,s}\epsilon p}{b}} \approx 1 + \frac{H_{N,s}\epsilon p}{b}$ which gives

$$\frac{d\mathfrak{S}(p)}{dp} = \frac{H_{N,s}(1+\epsilon)}{b \ln 2} \left(1 + \frac{H_{N,s}\epsilon p}{b}\right). \tag{27}$$

□

In Fig. 7a, we plot the approximate expression for $\mathfrak{S}(p)$ obtained in Thm. 3 which is a linear function in p and has a slope approximately 10 for $s = 1.07$ and $N = 50,000$. In Fig. 7b, we plot the approximate expression for $\frac{d\mathfrak{S}(p)}{dp}$ from Thm. 3 which is also a linear function of p. This tells us that even though $\mathfrak{S}(p)$ can be approximated as essentially a linear function of p, it has a slightly increasing slope. Further, unlike the plot of $\frac{d\mathfrak{S}(x)}{dx}$ in Fig. 6b, which is decreasing with k, $\frac{d\mathfrak{S}(p)}{dp}$ in Fig. 7b has a positive slope. Next, we provide an approximate expression for $H(P_{M_p}, P_M)$ showing that it grows near-linearly with p.

Theorem 4. *Let P_M be the model distribution satisfying (5) with vocabulary of size N. and let $P_{M_{k(p)}}$ be the model distribution obtained using top-p sampling where $k(p)$ is the minimum value of k satisfying $\frac{1}{H_{N,s}} \sum_{i=1}^{k(p)} \frac{1}{i^s} \geq p$. Then, for $1 < s \leq \frac{1}{\ln 2}$, $H(P_{M_p}, P_M)$ can be approximated as*

$$H(P_{M_p}, P_M) \approx \frac{s}{2 \ln 2} \left(p H_{N,s} + \epsilon p^2 H_{N,s}^2\right) + \log H_{N,s}. \tag{28}$$

Proof. The cumulative probability $p(k)$ for (5) can be written as

$$p(k) = \frac{H_{k,s}}{H_{N,s}}. \tag{29}$$

We approximate $\sum_{i=1}^{k} \frac{\ln i}{i^s} \approx \int_1^k \frac{\ln t}{t^s} dt$ to get

$$H(P_{M_p}, P_M) \approx \frac{s}{pH_{N,s} \ln 2} \left(\int_1^k \frac{\ln t}{t^s} dt \right) + \log H_{N,s}, \tag{30}$$

$$= \frac{s}{pH_{N,s} \ln 2} \left(\frac{1}{\epsilon^2} - \frac{1}{\epsilon k^\epsilon} (\ln k + \frac{1}{\epsilon}) \right) + \log H_{N,s}, \tag{31}$$

$$\tag{32}$$

Approximating $p(k)$ from (29) as $p(k) = \frac{1}{H_{N,s}} \int_1^k \frac{1}{t^s} dt$, we get

$$k = (1 - \epsilon p H_{N,s})^{-\frac{1}{\epsilon}}. \tag{33}$$

Using (33) in (31), we have

$$\begin{aligned}
H(P_{M_p}, P_M) &\approx \frac{s}{pH_{N,s} \ln 2} \left(\frac{1}{\epsilon^2} - \frac{1}{\epsilon k^\epsilon} (\ln k + \frac{1}{\epsilon}) \right) + \log H_{N,s}, \\
&= \frac{s}{pH_{N,s} \ln 2} \left(\frac{1}{\epsilon^2} + \frac{(1 - \epsilon p H_{N,s})}{\epsilon^2} (\ln (1 - \epsilon p H_{N,s}) - 1) \right) + \log H_{N,s} \\
&= \frac{s}{\epsilon^2 p H_{N,s} \ln 2} \left(1 + (1 - \epsilon p H_{N,s})(\ln (1 - \epsilon p H_{N,s}) - 1) \right) + \log H_{N,s} \\
&= \frac{s}{\epsilon^2 p H_{N,s} \ln 2} \left(\ln (1 - \epsilon p H_{N,s}) - \epsilon p H_{N,s} \ln (1 - \epsilon p H_{N,s}) + \epsilon p H_{N,s} \right) + \log H_{N,s} \\
&\approx \frac{s}{2 \ln 2} \left(p H_{N,s} + \epsilon p^2 H_{N,s}^2 \right) + \log H_{N,s},
\end{aligned} \tag{34}$$

where (34) is obtained by taking the approximation $\ln (1 - \epsilon p H_{N,s}) \approx -\epsilon p H_{N,s} - \frac{(\epsilon p H_{N,s})^2}{2}$ for sufficiently small $\epsilon p H_{N,s}$. □