Accelerating Entropy-Based Transformer Calibration

Michael Hu
Advisor: Prof. Karthik Narasimhan

Abstract

Current state-of-the-art language models do not accurately reflect long-term properties of the baseline language. In the language model GPT-2, we independently observe the phenomenon of entropy rate drift, where the entropy rates of generations from the model increase or decrease over time. For a model well-calibrated to the English language, we would expect to see no entropy rate drift. Furthermore, we demonstrate that several popular decoding heuristics, such as top-k sampling, nucleus sampling, and temperature scaling, do not protect against entropy rate drift. This motivates the use of local calibration, as proposed by Braverman et. al. [1]. We propose two techniques to accelerate local calibration: one by approximation, and the other by a reduction to temperature scaling. The first technique does not seem to work, and we provide some hypotheses as to why. The second technique seems more promising, inducing a drop in entropy rate increase.

1. Introduction

Language models estimate the likelihood of a certain word appearing in a sentence, conditional on the words that came before it. Recent architectural and training innovations have greatly advanced the state of the art for language modeling [11, 4, 19]. In particular, language models can now generate lucid and relatively coherent text. For an impressive example of a language model's generation capabilities, look no further than the *Ovid's Unicorn* passage, generated by GPT-2 [12].

In terms of model architecture, GPT-2 is an autoregressive (AR) language model. AR language models produce a conditional probability distribution, predicting the likelihood of a word based on the forward or backward context. Despite the improvements of AR language models in recent years, Braverman et. al. (2019) demonstrated that many AR language models are *miscalibrated*, in that repeated generations from these models tend to exhibit properties inconsistent with normal English

[1]. Concretely, AR language models tend to suffer from *entropy rate drift*, where the entropy rate of generations from the language models tend to increase or decrease over time.

Here, we extend the work of Braverman et. al. We focus on GPT-2, arguably the most successful autoregressive language model for natural language generation to date. In Section 3, we demonstrate that in addition to GPT-2 being miscalibrated, many common decoding heuristics for improving generation, such as top-k sampling, nucleus sampling, and temperature scaling, are not guaranteed improve GPT-2's calibration either. Thus, the calibration technique proposed by Braverman et. al., which we call *local calibration*, is the only technique that guarantees protections against entropy rate drift, both in theory and practice.

Local calibration only requires black-box access to the conditional probability distributions learned by a language model; thus, the technique can be applied to any AR language model. However, a drawback of local calibration is its computational complexity, which we analyze in Section 4. Next, we propose two methods for reducing the computational complexity of local calibration. The first method uses approximation, which we motivate from properties of English. The second method reduces local calibration to temperature scaling, a single-parameter technique used in both neural network calibration and decoding for natural language generation [17, 6]. Both methods provide significant speed increases to local calibration. Our first method does not work well in practice, and we discuss possible reasons why. Our second method results in a modest drop in entropy rate increase.

2. Background

2.1. Preliminaries: Information Theory and Local Calibration

Section 2.1 summarizes ideas in Braverman et. al. [1]. We begin with a crash course on information theory. Let $Pr(W_{1:T})$ be the true distribution of a T-length sentence composed of words W_1 through W_T. The words are capitalized to denote their identity as random variables.

$$H(Pr) := \mathop{\mathbb{E}}_{w_{1:T} \sim Pr} \left[\log \frac{1}{Pr(W_{1:T} = w_{1:T})} \right]$$

is the *entropy* of a distribution Pr. Entropy, first introduced by Claude Shannon [13], is a measure of how much information is telegraphed by an event. Intuitively, low probability events convey high information, and vice versa – if a low probability event occurs, that tends to betray more information about the underlying system. The *entropy rate* of a distribution is defined as $\text{EntRate}(Pr) := \frac{1}{T} H(Pr)$. Here, entropy rate corresponds to the average entropy per word. *Entropy rate drift* occurs when the entropy rate of a model's generations increase or decrease over time. $\textit{Perplexity} := e^{H(Pr(W_t|w_{<t}))}$ can be interpreted as as the number of plausible choices for the next word. Perplexity is the exponentiated entropy of the conditional distribution $Pr(W_t|w_{<t})$. Note that if perplexity increases over time, then the entropy rate is increasing as well.

Our language model learns a distribution $\widehat{Pr}$ over the possible T-length sequences. Minimizing *cross entropy loss* is the standard language model objective. Cross entropy loss is defined as:

$$CE(Pr||\widehat{Pr}) := \frac{1}{T} \mathop{\mathbb{E}}_{w_{1:T} \sim Pr} \left[\log \frac{1}{\widehat{Pr}(w_{1:T})} \right]$$

Last, we define the conditional one-step *lookahead entropy* $H(W_{t+1}|w_{\leq t})$. $H(W_{t+1}|w_{\leq t})$ is conditioned on the word w_t and its backward context $w_{<t}$.

$$H(W_{t+1}|w_{\leq t}) = \mathop{\mathbb{E}}_{w_{t+1} \sim \widehat{Pr}(\cdot\,|w_{\leq t})} \left[\log \frac{1}{\widehat{Pr}(w_{t+1}|w_{\leq t})} \right]$$

We are now ready to discuss Algorithm 2 on page 4, the method we hereby refer to as *local calibration* [1]. The input to Algorithm 2 is $\widehat{Pr}^{(\varepsilon)}$, a smoothed version of $\widehat{Pr}$, where a small amount of probability mass is distributed to each possible word. (This is a simplifying assumption, done to avoid dividing by zero when computing the cross entropy loss.) Local calibration guarantees that the returned distribution $\widehat{Pr}_\alpha$ improves in terms of entropy rate drift compared to $\widehat{Pr}^{(\varepsilon)}$.

Algorithm 2 Local Entropy Rate Calibration

1: Input: Model $\widehat{Pr}^{(\varepsilon)}$
2: Define $\widehat{Pr}_\alpha$, where:

$$\widehat{Pr}_\alpha(w_{1:T}) = \widehat{Pr}_\alpha(w_1) \cdot \widehat{Pr}_\alpha(w_2|w_1)...$$

and:

$$\widehat{Pr}_\alpha(w_t|w_{<t}) = \widehat{Pr}(w_t|w_{<t}) \cdot \exp\{-\alpha \cdot H(W_{t+1}|w_{\leq t})\}/Z_\alpha, \text{ where}$$
$$Z_\alpha := \sum_{w_{1:T}} \exp\{-\alpha \cdot H(W_{t+1}|w_{\leq t})\} \cdot \widehat{Pr}(w_{1:T})$$

3: Fit $\alpha^* : \alpha^* = \arg\min_\alpha CE(Pr||\widehat{Pr}_\alpha)$
4: **return** α^*

The intuition of local calibration is to apply a penalty or bonus to each probability $\widehat{Pr}(w_t|w_{<t})$ in the conditional distribution. This penalty or bonus is based on each word w_t's respective lookahead entropy $H(W_{t+1}|w_{\leq t})$ (Algorithm 2, step 2). Depending on α, lookahead entropies will induce a larger or smaller penalty. The algorithm then tunes α, such that the base objective of the language model is unaffected (Algorithm 2, step 3). For full details and theoretical properties of local calibration, the reader is deferred to the paper itself [1].

2.2. Preliminaries: Natural Language Generation and Decoding Heuristics

Language models generally predict tokens, or subwords, instead of words themselves. Suppose we wish to generate a T-token passage of text from an AR language model, given some context $w_{<t}$. One possible procedure is to seed the language model with context $w_{<t}$, obtain the conditional distribution $Pr(W_t|w_{<t})$, sample a token w_t from the conditional distribution, concatenate w_t to the context, and repeat.

Naively sampling in the manner above can lead to nonsensical output [8]; in practice, we rarely sample tokens from $Pr(W_t|w_{<t})$ directly. Instead, we usually pass $Pr(W_t|w_{<t})$ through a decoding heuristic, such as top-k sampling, nucleus sampling, or temperature scaling [12, 8, 6]. These techniques tend to improve generations, and are discussed further in Section 3.

Other popular decoding techniques include greedy decoding and beam search [9]. However, these methods are not well-suited for open-ended language generation, our current setting – greedy

decoding and beam search are more commonly used in machine translation. For example, beam search tends to work better when the length of output is predicted in advance [18]. But for open-ended generation, the length of output can be arbitrary, as long as the output is cogent. Furthermore, Holtzman et. al. (2019) showed that greedy decoding and beam search can cause GPT-2 to repeatedly output the same phrase, a failure state [8]. For these reasons, we do not further analyze greedy decoding and beam search as decoding strategies in this study.

In a way, this independent work can be interpreted as building decoding heuristics influenced by local calibration. We propose two such heuristics in Sections 4 and 5. We do not call local calibration itself a heuristic, as the technique is derived from first principles and leads to provable improvements over a baseline.

2.3. Other work

Methods like local calibration treat the language model as a black-box accessor of conditional distributions $Pr(W_t|w_{<t})$. This would not be practical without language models that are quite accurate already. The Transformer architecture has found wide applicability in language modeling [14], and many current language models contain Transformer or attention modules, including GPT-2 [4, 12, 19, 3]. Another influential idea in language modeling is the "pre-train, fine-tune" paradigm, where language models are first pre-trained on a large corpus of data, then selectively fine-tuned for a specific task. Unsupervised pre-training induces language models to improve on a wide variety of tasks without explicit instruction [11].

Outside of autoregressive language modeling, another type of language model is the autoencoding (AE) language model. Autoencoding language models seek to recover ground truth values of compressed or corrupted input. In other words, these models "fill in the blanks." Popular AE models include BERT and its variants [4], and recent work from Wang. et. al. demonstrates how to generate text from BERT [15]. However, since AE models do not produce an explicit probability distribution, local calibration cannot be applied to AE language models as is. Local calibration is currently limited to AR language models.

Guo et. al. demonstrated that temperature scaling is an effective way of calibrating neural networks on standard computer vision and NLP datasets [6]. Temperature scaling is a single parameter version of Platt Scaling [10].

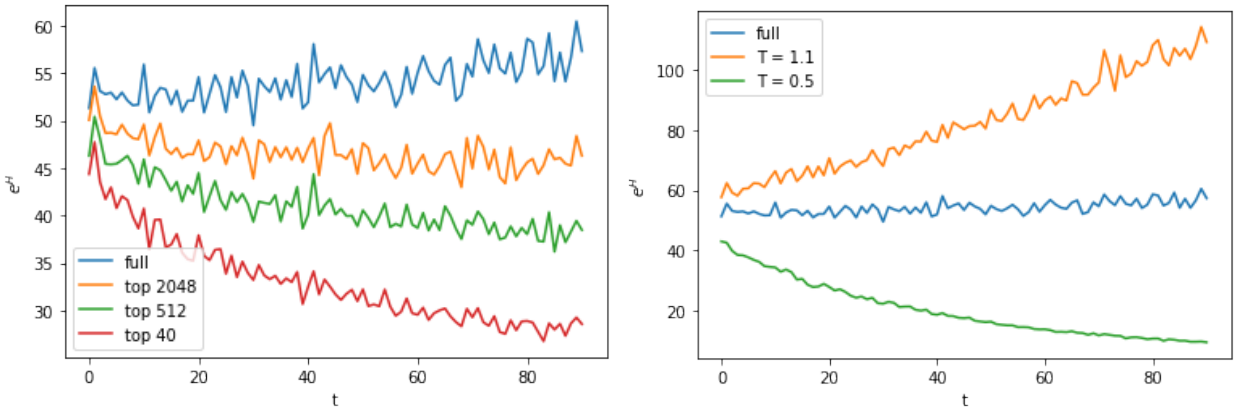
3. Why Calibration is Necessary

Braverman et. al showed that perplexity tends to increase with successive generations from AR language models [1]. We independently verify these results for GPT-2 (Figure 1, blue lines). For a well-calibrated model, we would expect to see little to no fluctuation in the perplexity, resulting in a horizontal line from left to right. However, these results are obtained by the naive sampling method described in Section 2.2, where we choose tokens from the entire conditional probability distribution $Pr(W_t|w_{<t})$. In practice, users do not sample from the entire distribution. Instead, they tend to alter the distribution using a decoding heuristic and sample from the altered distribution instead. Decoding heuristics help avoid issues such as repetition and hallucination: repetition, where the model repeatedly outputs the same phrase, and hallucination, where the model's output is nonsense [8].

Here, we provide a short exposition of three common decoding heuristics – top-k sampling, nucleus sampling, and temperature scaling – and show that entropy rate drift occurs for different hyperparameter values for all three heuristics. Mathematical formalisms are adapted from Holtzman et. al. [8]. In particular, two recent hyperparameter choices for generation are both miscalibrated with respect to entropy rate drift [12, 8].

3.1. Top-k Sampling

Top-k sampling truncates $\widehat{Pr}(W_t|w_{<t})$ to the top-k most likely tokens. It then renormalizes the remaining k tokens into a probability distribution and samples from the new distribution. Formally, let $\widehat{Pr}(W_t|w_{<t})$ be the distribution over the next token. Let V be the vocabulary set, and let $V_k := \arg\max_{V'} \sum_{w_t \in V'} \widehat{Pr}(w_t|w_{<t})$, where $|V'| = k$ and $V' \subseteq V$.



(a) Perplexity curves for top-k sampling.

(b) Perplexity curves for temperature scaling.

Figure 1: Perplexity curves of various decoding techniques for natural language generation. Current uses of top-k sampling tend to set $k \leq 40$, which would correspond to the red line in Figure 1(a). Figures 1(a) and 1(b) demonstrate that entropy rate may decrease as well as increase.

k	Δe^H
$k = \text{full}$	+5.999
$k = 2048$	-3.780
$k = 512$	-7.804
$k = 40$	-15.791

Table 1: Average change in perplexity for various k over 90 generations.

T	Δe^H
$T = 1$	+5.999
$T = 0.5$	-33.514
$T = 1.1$	+51.532

Table 2: Average change in perplexity for various temperatures T over 90 generations.

We define a new probability distribution $\widehat{Pr}^{(k)}$:

$$\widehat{Pr}^{(k)} := \begin{cases} \frac{\widehat{Pr}(w_t|w_{<t})}{\sum_{w_t \in V_k} \widehat{Pr}(w_t|w_{<t})} & \forall w_t \in V_k \\ 0 & \text{otherwise} \end{cases}$$

Top-k sampling then chooses a token from $\widehat{Pr}^{(k)}$. Recent papers usually set k to around 10 or 40, and resulting generations are syntactically correct and cogent [5, 12]. But empirically, we observe that setting k to 40 causes perplexity to decrease over time (Figure 1), indicating that top-40

sampling is miscalibrated. Over 90 generations, top-40 sampling induces an average perplexity decrease of 15.791 (Table 1). All experimental details are in the appendix.

3.2. Nucleus Sampling

Nucleus sampling, or top-p sampling, truncates the probability distribution to the minimal subset of tokens that make up $100p\%$ of the probability mass ($p \in [0,1]$). Ceteris paribus as above, let $V_p \in V$ be the smallest set such that $\sum_{w_t \in V_p} \widehat{Pr}(w_t|w_{<t}) \geq p$.

We define a new probability distribution $\widehat{Pr}^{(p)}$:

$$\widehat{Pr}^{(p)} := \begin{cases} \frac{\widehat{Pr}(w_t|w_{<t})}{\sum_{w_t \in V_p} \widehat{Pr}(w_t|w_{<t})} & \forall w_t \in V_p \\ 0 & \text{otherwise} \end{cases}$$

Top-p sampling then chooses a token from $\widehat{Pr}^{(p)}$. Here, note that top-p sampling is simply a dynamic version of top-k sampling. Thus, top-p sampling suffers from similar issues. For example, Holtzman et. al. used $p = 0.9$ for their own generations [8]. Figure 3 demonstrates that $p = 0.9$ is equivalent to approximately $k = 512$. On average, $k = 512$ suffers from an average perplexity decrease of 7.804 (Table 1).

3.3. Temperature Scaling

Language models like GPT-2 do not directly output a probability distribution. To obtain a distribution, the outputs of the neural network $u_{1:|V|}$, called logits, are passed through a softmax function. Temperature scaling simply divides the logits by a constant $T \in (0,\infty)$.

Concretely:

$$\widehat{Pr}^{(T)}(w_t|w_{<t}) := \frac{\exp\{u_t/T\}}{\sum_{i=1}^{|V|} \exp\{u_i/T\}}$$

We then sample from $\widehat{Pr}^{(T)}$. Note that for $T = 1, \widehat{Pr}^{(T)} = \widehat{Pr}$. To increase diversity in word choice, one sets $T > 1$. To decrease diversity, one sets $T \in (0, 1)$. Figure 1(b) shows model perplexity for $T = 0.5, 1$, and 1.1. All three choices are miscalibrated.

3.4. Analysis

Calibration with respect to entropy rate may be possible using one or more of top-k sampling, top-p sampling, or temperature scaling. But it is unclear how to choose a hyperparameter for these generation heuristics, such that entropy rate drift does not occur. Furthermore, these hyperparameters are usually chosen for reasons other than mitigating entropy rate drift; they are not designed for this purpose. Entropy rate naturally increases without any generation heuristic, and many current usages of generation heuristics overcorrect entropy rate drift as a side effect, such that entropy rate actually decreases over time. From our data, we draw two conclusions. First, local calibration is the only technique, to our knowledge, that provably mitigates entropy rate drift. Second, there exists a "sweet spot" for calibration. A heavy-handed adjustment via decoding heuristic may actually cause entropy rates to decrease instead of increase, which is also undesirable.

4. Accelerating Calibration via Approximation

The main challenge of using local calibration, or Algorithm 2, is its computational complexity. As a baseline, let us consider the computational complexity of generating T successive tokens from GPT-2, given C tokens of context. If we were to use a generation heuristic as above, we perform a single inference per token to obtain $\widehat{Pr}$. Passing $\widehat{Pr}$ through a decoding heuristic to obtain $\widehat{Pr}^{(\cdot)}$ then takes constant time, for an overall complexity of $O(T)$ in the number of inferences to GPT-2.

The same process takes $O(VC + VT)$ inferences for local calibration, where V is the size of the vocabulary. We first consider calibration itself. Computing $\widehat{Pr}_\alpha$ takes $O(V)$ inferences. We perform one inference per lookahead entropy, so it takes $O(V)$ inferences to compute Z_α. To compute the cross entropy loss, we must compute $\widehat{Pr}_\alpha$ for all of the C possible contexts. Thus, the cost for calibration is $O(VC)$ inferences.

We obtain α^* from calibration. To generate T tokens, we must compute $\widehat{Pr}_{\alpha^*}$ each time, for a total cost of $O(VT)$ inferences during generation. The overall cost of generating T tokens from C tokens of context is then $O(VC+VT)$. The vocabulary size V of GPT-2 is 50,000, and most model vocabularies are in the range of 10^4 to 10^5 [12, 14]. Therefore, local calibration is around 50,000 times slower than sampling using a heuristic, depending on the language model. For local calibration to be usable in practice, we require some way to reduce the number of inferences performed.

4.1. Lookahead Entropy Approximation: Approach and Implementation

Here, we propose a method for speeding up calibration by calculating a few lookahead entropies and approximating the rest. As discussed on the previous page, the $O(V)$ factor in $O(VC+VT)$ comes from calculating lookahead entropies. Thus, instead of computing $H(W_{t+1}|w_{\leq t})$ for all tokens, we propose doing so for only the top k most likely tokens. We then calculate $\overline{H}$, the mean of these k lookahead entropies. For any token that does not fall within the top k, we replace its lookahead entropy with $\overline{H}$. Using this approximation reduces the number of required inferences by $\frac{|V|}{k}$, which dramatically improves runtimes in practice. For example, if we use $k = 500$ with GPT-2, then we perform $\frac{50{,}000}{500} = 100$-fold fewer inferences compared to local calibration, which is roughly equivalent with a 100-fold decrease in runtime. We formally describe this calibration method in Algorithm 3.

As $k \to |V|$, $\widehat{Pr}'_\alpha \to \widehat{Pr}_\alpha$. For large k, Algorithm 3 scales most of the probability mass according to its respective lookahead entropy. Language tends to follow Zipf's Law [7], meaning that most conditional distributions $\widehat{Pr}(W_t|w_{<t})$ look like Figure 2, where probability mass is concentrated in the top 10 to 30 tokens or so. We quantify the level of concentration in Figure 3: after computing the proportion of probability mass captured by the top 512 tokens for 300,000 contexts in the Google Billion Words (GBW) corpus, we observe that on average, the top 512 tokens represent 90.7% of the probability mass (Figure 3). Therefore, we expect that for $k = 512$, Algorithm 3 properly scales around 90% of the probability mass. For the remaining 10%, we hope that $\overline{H}$ is a reasonable estimate for the true lookahead entropy.

Algorithm 3 Local Entropy Rate Calibration with Approximation

1: Input: Model $\widehat{Pr}^{(\varepsilon)}$, $k \in [1, |V|]$
2: Define $\widehat{Pr}'_\alpha$, where:

$$\widehat{Pr}'_\alpha(w_{1:T}) = \widehat{Pr}'_\alpha(w_1) \cdot \widehat{Pr}'_\alpha(w_2|w_1)...$$

3: Let $V_k := \arg\max_{V'} \sum_{w_t \in V'} \widehat{Pr}(w_t|w_{<t})$, where $|V'| = k$ and $V' \in V$. Z_α remains the normalization constant.

$$\overline{H} := \frac{1}{k} \sum_{w_t \in V_k} H(W_{t+1}|w_t, w_{<t})$$

$$\widehat{Pr}'_\alpha(w_t|w_{<t}) = \begin{cases} \widehat{Pr}(w_t|w_{<t}) \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w_{\leq t})\right)/Z_\alpha & \forall w_t \in V_k \\ \widehat{Pr}(w_t|w_{<t}) \cdot \exp\left(-\alpha \cdot \overline{H}\right)/Z_\alpha & \text{otherwise} \end{cases}$$

4: Fit $\alpha^* : \alpha^* = \arg\min_\alpha CE(Pr||\widehat{Pr}'_\alpha)$
5: **return** α^*

We implemented the minimization step in Algorithm 2 and 3 using Newton's method for fast convergence. In our experience, Newton's method converges to a minimum within 4 to 6 iterations. Fast convergence is crucial here, as an extra iteration of local calibration can take more than a day to compute. We initially tried gradient descent but found that it converged too slowly.

4.2. Results

Our approximation approach did not work in practice. In addition to $k = 512$, which we chose from empirical results, we tried several other values of k in powers of 2 greater and smaller than 512. (We use powers of 2 to better align virtual processes with physical processes: if we set $k = 65$, we may as well set $k = 128$, as we are using the same number of physical processes.) We consider α^* from local calibration as the ground truth α^* value. The α^* values we obtained for different values of k were within one decimal place of the ground truth value, at best. Ideally, we would like to see higher accuracy in the approximations. α^* values and runtimes are reported in Table 3. Runtimes for this approach were significantly faster than local calibration, but at the cost of accuracy. $k = 512$ was closest to the α^* returned by local calibration.

Generation from $\widehat{Pr}'_\alpha$ is formally described in Algorithm 4 on page 14. We kept k consistent from calibration to generation: if we chose to approximate using $k = 64$ during calibration, then we

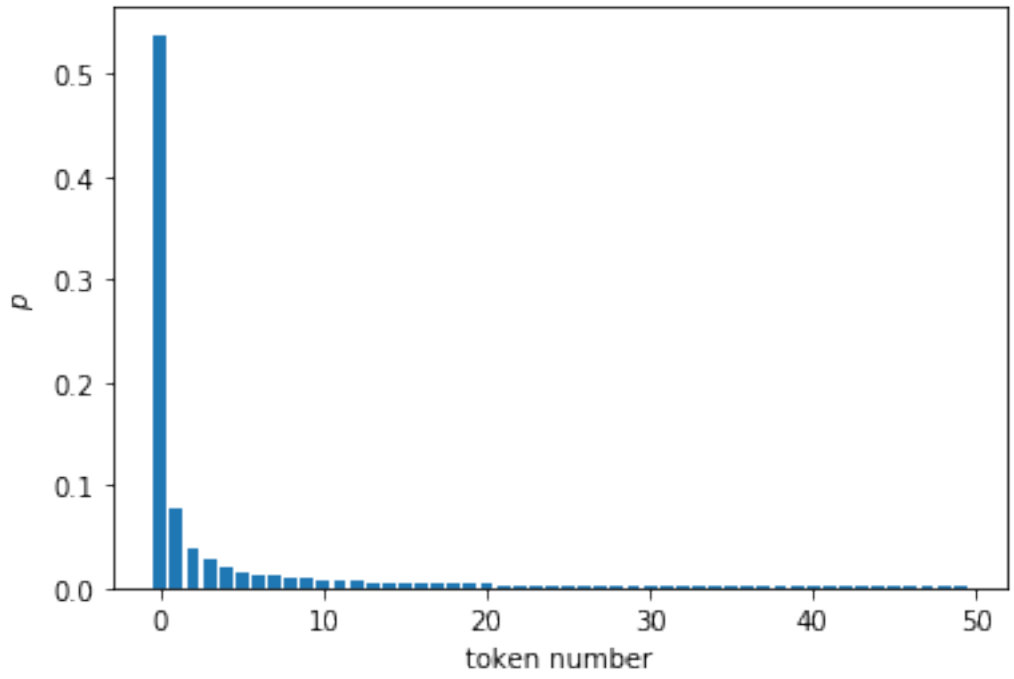


Figure 2: Likelihoods of the top 50 tokens of a learned distribution. To obtain the conditional distribution, a context $w_{<t}$ was chosen at random from the Google Billion Words corpus. The majority of the distribution's probability mass is captured within the top 10 tokens.

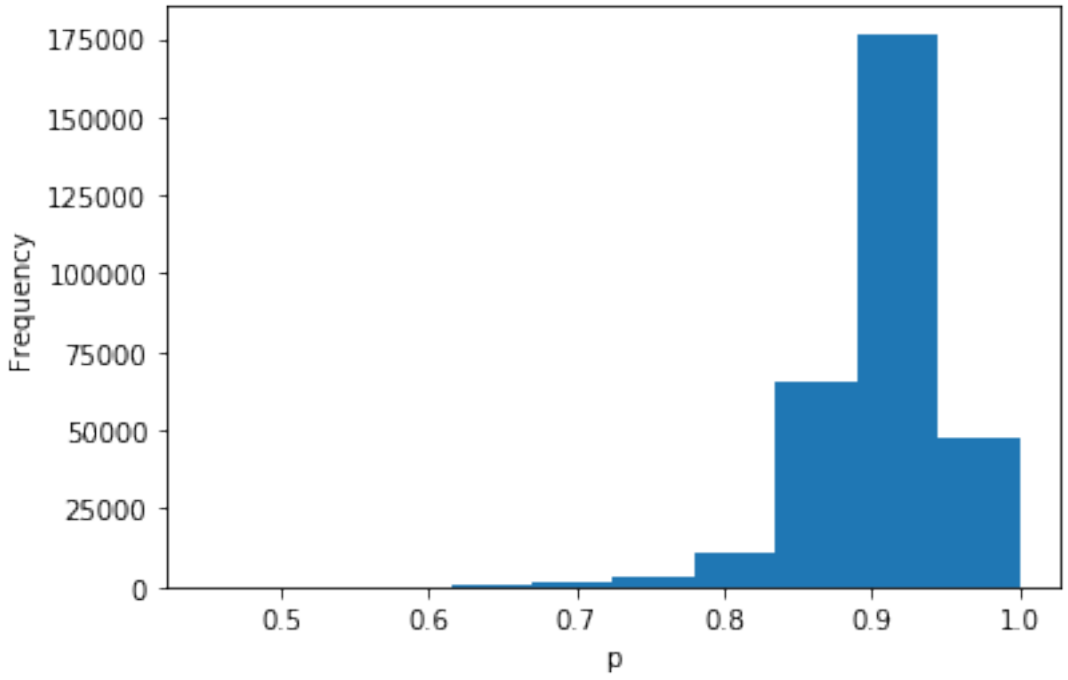
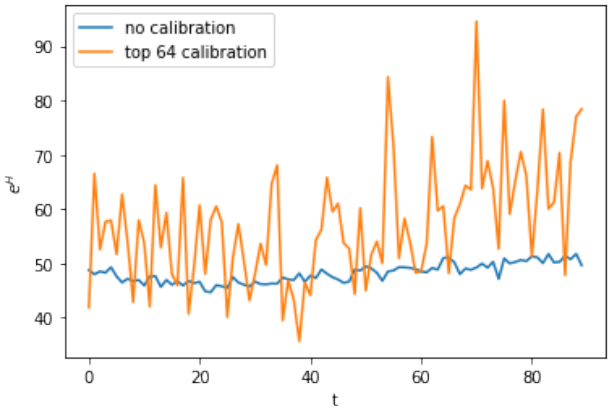
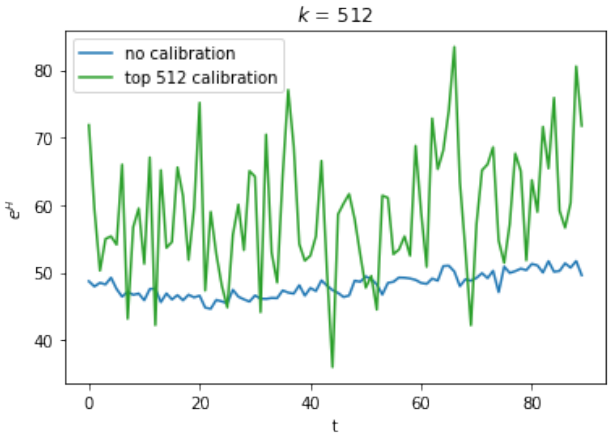


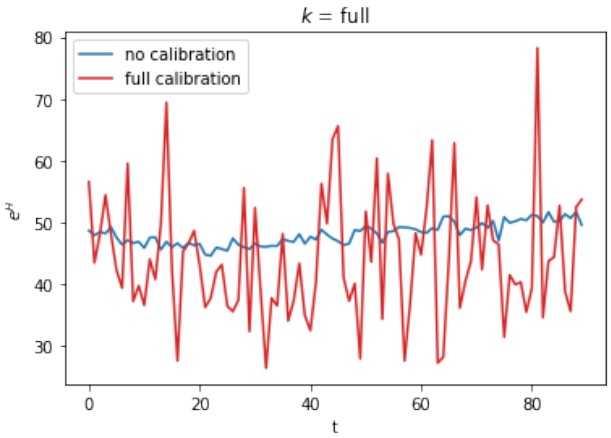
Figure 3: Sampling distribution of the proportion of probability mass captured by the top 512 most likely tokens. On average, the top 512 tokens capture 90.7% of the probability mass.



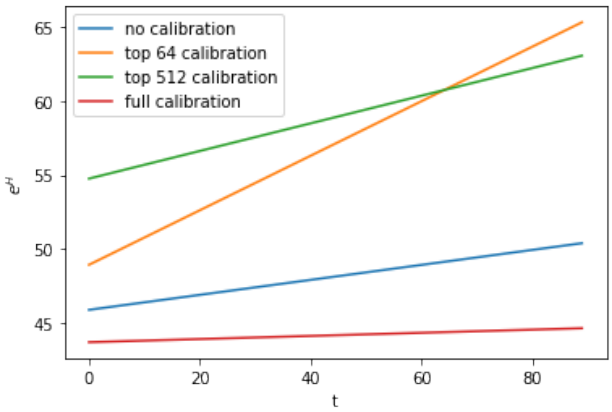
(a) Generation using k=64, α = 0.4082.



(b) Generation using k=512, α = 0.0859



(c) Full generation, α = 0.0339



(d) Lines of best fit for all curves. Full calibration works as expected, reducing entropy rate drift. Generation using $k = 64, 512$ increases upward entropy rate drift.

Figure 4: Calibration using different values of k. The "peakiness" of the calibration graphs are due to sample size – we averaged across 128 samples for each calibration trial and 10,000 samples for no calibration.

also used $k = 64$ during generation. The entropy behavior for $k = 64$, $k = 512$, and local calibration are shown in Figure 4. Local calibration decreases the upward entropy rate drift, while $k = 64$ and $k = 512$ increase the upward entropy rate drift.

We would define Algorithm 3 as successful, had it consistently approximated the α^* of Algorithm 2 to within 2 decimal places. We would define Algorithm 4 as successful, had it decreased the upwards entropy rate drift of the baseline (Figure 4, blue lines). As it stands, neither algorithm was successful. We discuss potential reasons for failure in Section 4.3.

k	α^*	Runtime
Local calibration (ground truth)	0.0339	5 days, 8 hours
$k = 64$	0.4082	19 minutes
$k = 512$	0.0859	1 hour, 12 minutes
$k = 1024$	-0.0206	2 hours, 32 minutes
$k = 2048$	-0.0232	4 hours, 56 minutes
$k = 4096$	-0.0298	9 hours, 43 minutes

Table 3: All calibrations performed on a 100-line corpus randomly sampled from GBW.

Algorithm 4 Generation with Approximation

Input: Model $\widehat{Pr}$, $k \in [1, |V|]$, generation length T,α^*
for $t = 1 : T$ **do**
    Let $V_k := \arg\max_{V'} \sum_{w_t \in V'} \widehat{Pr}(w_t|w_{<t})$, where $|V'| = k$ and $V' \in V$.

$$\overline{H} := \frac{1}{k} \sum_{w_t \in V_k} H(W_{t+1}|w_t, w_{<t})$$

$$\widehat{Pr}_{\alpha^*}(w_t|w_{<t}) = \begin{cases} \widehat{Pr}(w_t|w_{<t}) \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w_{\leq t})\right)/Z_\alpha & \forall w_t \in V_k \\ \widehat{Pr}(w_t|w_{<t}) \cdot \exp\left(-\alpha \cdot \overline{H}\right)/Z_\alpha & \text{otherwise} \end{cases}$$

    Sample $w_t \sim \widehat{Pr}_{\alpha^*}$
end for
return $w_1, w_2, ..., w_t$

4.3. Discussion

Here, we offer some hypotheses on why Algorithms 3 and 4 fail to improve entropy rate drift. Figure 5 on page 15 visualizes the lookahead entropies of each token for a given context, sorted from most likely to least likely. Although most lookahead entropy values are clustered in the 4 to 6 range, there exists significant density in the 0 to 4 range as well. Furthermore, lookahead entropy is positively correlated with token rank, with correlation coefficient of 0.262.

Thus, perhaps using the mean as an approximation is a bad idea, given the data. We define the tail of the distribution as all the tokens not within the top k. Given the positive correlation, the population mean will be greater than the sample mean, since the sample is drawn from the initial k tokens. Tokens in the tail of the distribution will tend to have higher lookahead entropies, and this is not accounted for in the approximation.

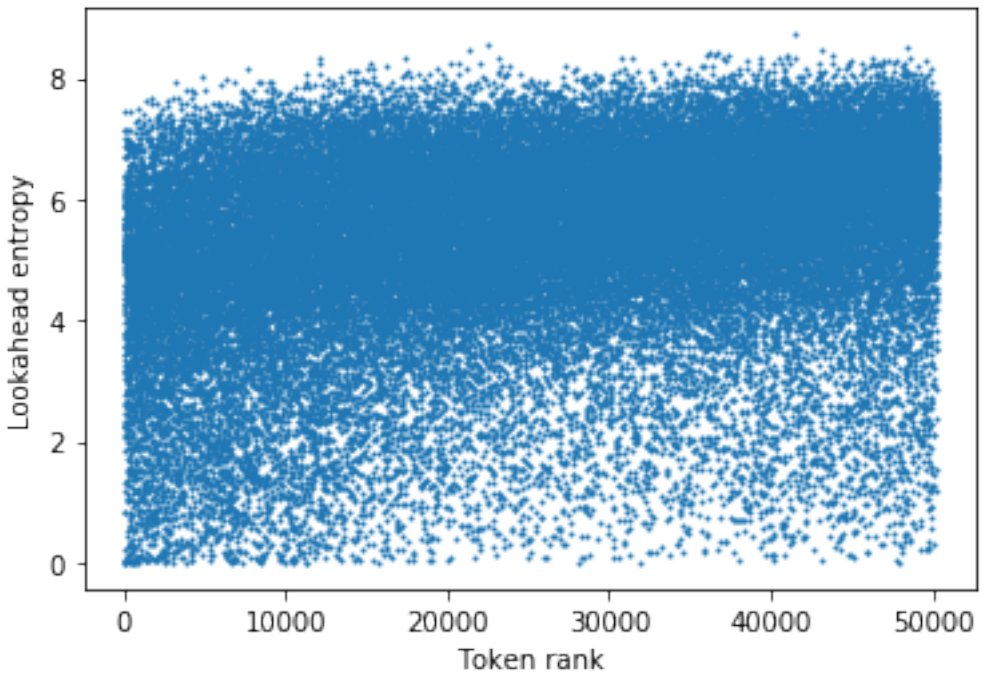


Figure 5: Lookahead entropy values for different tokens, sorted from most likely to least likely. Lookahead entropies are weakly correlated with token likelihood (0.262). Critically, tokens in the tail of the distribution continue to have low lookahead entropies.

Second, the approximation treats all values in the tail equally. But as we can see from Figure 5, there continue to exist tokens with very low (near zero) lookahead entropies in the tail. Penalizing these tokens the same amount as tokens with large lookahead entropies potentially muddies the effects of calibration. As a thought experiment, consider two tokens in the tail, both with the same probability. Token one has high lookahead entropy, and token two has low lookahead entropy. For $\alpha > 0$, Algorithm 2 assigns token one a higher penalty than token two, which is the desired behavior – we want to avoid token one, as it is more likely to lead to entropy rate drift. Conversely, Algorithm 3 assigns both tokens the same penalty, meaning that both tokens remain equally likely to be sampled.

Algorithm 3 assigns the same penalty to all tokens in the tail. Given the variability of lookahead entropies, this may not have been a good idea. The initial hope was that assigning dynamic penalties to the top k tokens would be sufficient, and calibration accuracy would not suffer much. Similarly, Algorithm 4 also suffers from a distributional tail that is poorly calibrated. This is problematic, as sampling from the tail becomes increasingly likely with successive generations. For example, if we

set $k = 512$, sampling from the tail is expected to occur within the first 10 generations about 65% of the time[1]. Sampling poorly calibrated tokens from the tail may have set off chain reactions of subsequently poor samplings, leading to exacerbated entropy rate drift.

5. Accelerating Calibrated Generation via Temperature Scaling

Here, we analyze the relationship between local calibration and temperature scaling. From this analysis, we derive a method that selects a temperature for generation. We seek to connect local calibration with temperature scaling, such that local calibration can be used to choose a temperature that mitigages entropy rate drift.

We can think of both local calibration and temperature scaling as applying some perturbation to the language model's output logits $u_{1:|V|}$ to obtain a new set of logits $u'_{1:|V|}$. Both techniques then softmax the logits $u'_{1:|V|}$ to obtain a probability distribution. From Section 3.3, we observe that temperature scaling applies the transformation $u'_{1:|V|} = u_{1:|V|}/T$.

By comparison, local calibration applies the transformation $u'_{1:|V|} = u_{1:|V|} - \alpha^* \cdot \vec{H}$, where $\vec{H}$ is the $|V|$-length vector of lookahead entropies. A short derivation is below.

$$\begin{aligned}
\widehat{Pr}_\alpha(w_t|w_{<t}) &= \frac{\widehat{Pr}(w_t|w_{<t}) \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w_{\leq t})\right)}{\sum_{w'_t \in V} \widehat{Pr}(w'_t|w_{<t}) \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w'_t, w_{<t})\right)} \\
&= \frac{\exp\{u_t\}) \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w_{\leq t})\right)}{\sum_{t'=1}^{|V|} \exp\{u_{t'}\} \cdot \exp\left(-\alpha \cdot H(W_{t+1}|w_{t'}, w_{<t})\right)} \\
&= \frac{\exp\{u_t - \alpha \cdot H(W_{t+1}|w_{\leq t})\}}{\sum_{t'=1}^{|V|} \exp\{u'_t - \alpha \cdot H(W_{t+1}|w_{t'}, w_{<t})\}} \qquad \square
\end{aligned}$$

Critically, $u'_{1:|V|} = u_{1:|V|} - \alpha^* \cdot \vec{H}$ is not a monotonically increasing transformation, while $u'_{1:|V|} = u_{1:|V|}/T$ is. Thus, local calibration can potentially change the argmax of the probability distribution, while temperature scaling will not. This is an important distinction. In the language of the previous discussion, local calibration applies a unique penalty for each logit, while temperature scaling applies the same proportion of penalty for all logits.

[1] $k = 512$ captures approximately 90% of the probability mass (Figure 3). $1 - (0.9)^{10} = 0.65$.

Nevertheless, for a given α^*, there exists a temperature T that obtains the same adjustment for each logit, depending on the logit's respective lookahead entropy:

$$u_t/T = u_t - \alpha^* \cdot H(W_{t+1}|w_{\leq t})$$

$$\Rightarrow \qquad T = \frac{u_t}{u_t - \alpha^* \cdot H(W_{t+1}|w_{\leq t})}$$

With this in mind, we propose Algorithm 5, which generates T tokens given the output of Algorithm 2. (From here to the end of Section 5, T refers to generation length and T^* refers to temperature.) Algorithm 5 first computes T^*, the average expected temperature across all contexts. It then uses T^* as its temperature for generations.

Algorithm 5 Calibrated Generation via Temperature Scaling

1: Input: Model $\widehat{Pr}$, context of length C, α^*, generation length T
2:

$$T^* = \frac{1}{C}\sum_{k=1}^{C} \mathop{\mathbb{E}}_{w_t \sim Pr(\cdot|w_{<k})} \left[\frac{u_t}{u_t - \alpha^* \cdot H(w_{t+1}|w_t, w_{<k})} \right]$$

3: **for** $t = 1 : T$ **do**
4:

$$\widehat{Pr}^*(w_t|w_{<t}) = \frac{\exp\{u_t/T^*\}}{\sum_{i=1}^{|V|}\exp\{u_i/T^*\}}$$

5: Sample $w_t \sim \widehat{Pr}^*$
6: **end for**
7: **return** $w_1, w_2, ..., w_t$

T^* takes VC inferences to compute. After computing T^*, we perform $O(T)$ inferences to generate T tokens. Thus, composing Algorithm 2 with Algorithm 5 allows us to generate T tokens in $O(VC+T)$ time instead of $O(VC+VT)$ time. We identify two benefits of Algorithm 5, in addition to the speedup in computational complexity. First, Algorithm 5 also gives us another way of interpreting α^*. For example, on our 100-line corpus, Algorithm 2 computes $\alpha^* = 0.0339$. Algorithm 5 computes a corresponding $T^* = 0.998$. This indicates that $\alpha^* = 0.0339$

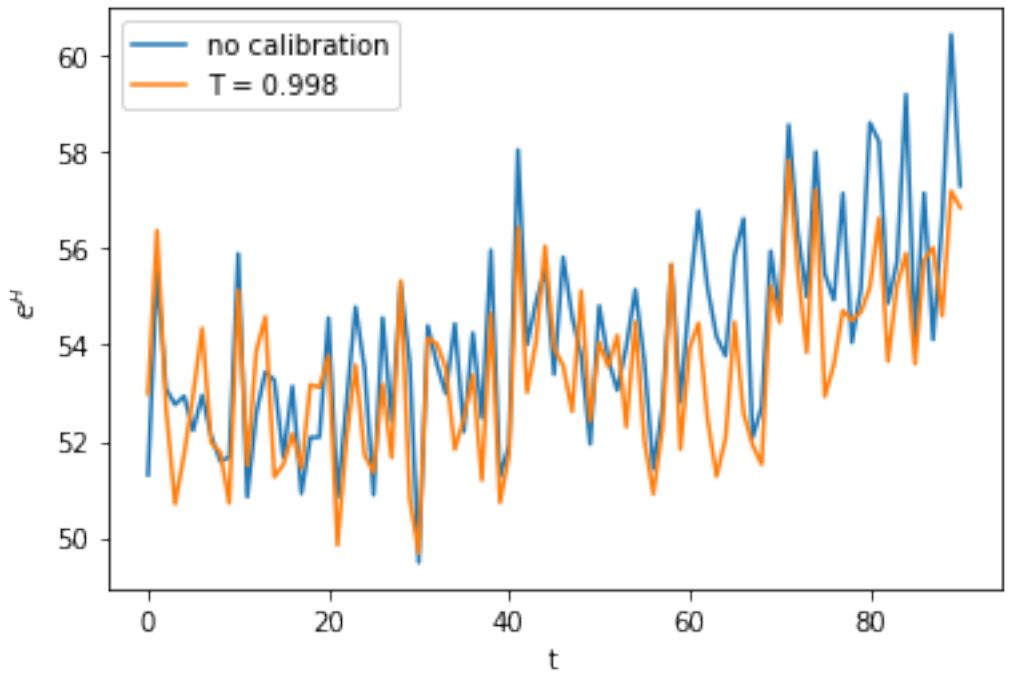


Figure 6: Perplexity curve for uncalibrated generation vs. using a temperature determined by calibration. Perplexity increases 5.999 over 90 generations in uncalibrated generation, compared to 3.868 for $T = 0.998$.

slightly suppresses diversity of generations (recall Section 3.3). Second, Algorithm 5 requires no hyperparameter tuning, unlike Algorithms 3 and 4.

Generating with $T^* = 0.998$ gives a modest improvement in perplexity, as shown in Figure 6. This modest improvement is likely due to the fact that temperature scaling, like Algorithm 4, applies the same penalty to each logit. Local calibration shines in that it imposes a differentiated penalty to each logit, thereby allowing us to sample from the entire distribution with greater confidence. Temperature scaling does not provide the same benefits, leading to a more modest improvement. On average, we observe a 36% decrease in entropy rate drift[2].

6. Conclusion

There currently does not exist a practical method for mitigating entropy rate drift in natural language generation. We have shown that entropy rate drift occurs in generations from GPT-2, both with and without the use of common decoding heuristics. Here, we propose and give empirical results

[2] $1 - \frac{3.868}{5.999} = 0.36$.

for two novel heuristics aimed at accelerating local calibration and subsequent generation. Our first heuristic seeks to approximate the lookahead entropies for tokens in the tail of the distribution, reducing the number of required inferences by several orders of magnitude. Our second heuristic seeks to reduce local calibration to temperature scaling by computing a unique temperature T^* to correspond to α^*. Heuristic 1, comprised by Algorithms 3 and 4, tends to make entropy rate drift worse. Performing the same adjustment to tokens within the distribution's tail may adjust the probabilities to be less reflective of our confidence in them. Heuristic 2, comprised by Algorithm 5, obtains a small drop in entropy rate drift.

Local calibration is a promising technique with reasonable computational complexity from a statistical perspective. However, the cost of repeatedly computing lookahead entropies makes the technique far slower than other generation-improving decoding techniques. Further work must be done to develop a version of local calibration that is usable in practice. We hope that this report illuminates both the necessity of local calibration and ways to improve the technique going forward.

6.1. Future Work

One possible way to accelerate local calibration is to use it in conjunction with other decoding heuristics. We hypothesize that Heuristic 1 does not work due to occasional samplings from the distribution's tail. Thus, we may be able to avoid the tail entirely, calibrating on and generating from the top k tokens only. This strategy makes sense if one is planning to eventually use top-k sampling – examining tokens outside the top k then becomes unnecessary.

Another method for entropy rate calibration may seek to avoid local calibration entirely. Here, one could adjust the hyperparameters for top-k sampling, nucleus sampling, or temperature scaling until entropy rate drift is mitigated on a validation set. To the author's knowledge, there currently does not exist a method for finding optimal hyperparameters to mitigate entropy rate drift.

Last, entropy rate drift can potentially be connected to other phenomena, such as repetition or hallucination. For example, hallucination may be related to low probability samplings that drive

up the entropy rate. Finding such a connection would be very fruitful, as mitigating hallucination would then mitigate entropy rate drift as a side effect.

6.2. Acknowledgements

Professor Karthik Narasimhan has been an incredible advisor throughout this entire semester. Professor, thank you for your patience, generosity, and willingness to guide and explain. I would like to thank Cyril Zhang for providing me with starter code, and my friends and family for their constant support.

References

[1] M. Braverman, X. Chen, S. M. Kakade, K. Narasimhan, C. Zhang, and Y. Zhang, "Calibration, entropy rates, and memory in language models," 2019. [Online]. Available: http://arxiv.org/abs/1906.05664

[2] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, and P. Koehn, "One billion word benchmark for measuring progress in statistical language modeling," *CoRR*, vol. abs/1312.3005, 2013. [Online]. Available: http://arxiv.org/abs/1312.3005

[3] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *CoRR*, vol. abs/1901.02860, 2019. [Online]. Available: http://arxiv.org/abs/1901.02860

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[5] A. Fan, M. Lewis, and Y. N. Dauphin, "Hierarchical neural story generation," *CoRR*, vol. abs/1805.04833, 2018. [Online]. Available: http://arxiv.org/abs/1805.04833

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *CoRR*, vol. abs/1706.04599, 2017. [Online]. Available: http://arxiv.org/abs/1706.04599

[7] L. Q. Ha, E. I. Sicilia-Garcia, J. Ming, and F. J. Smith, "Extension of zipf's law to words and phrases," in *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, ser. COLING '02. USA: Association for Computational Linguistics, 2002, p. 1–6. [Online]. Available: https://doi.org/10.3115/1072228.1072345

[8] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2019.

[9] I. Kulikov, A. H. Miller, K. Cho, and J. Weston, "Importance of a search strategy in neural dialogue modelling," *CoRR*, vol. abs/1811.00907, 2018. [Online]. Available: http://arxiv.org/abs/1811.00907

[10] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[13] C. E. Shannon, "Prediction and entropy of printed english," *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1951.tb01366.x

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[15] A. Wang and K. Cho, "BERT has a mouth, and it must speak: BERT as a markov random field language model," *CoRR*, vol. abs/1902.04094, 2019. [Online]. Available: http://arxiv.org/abs/1902.04094

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[17] Z. Xie, "Neural text generation: A practical guide," *CoRR*, vol. abs/1711.09534, 2017. [Online]. Available: http://arxiv.org/abs/1711.09534

[18] Y. Yang, L. Huang, and M. Ma, "Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation," *CoRR*, vol. abs/1808.09582, 2018. [Online]. Available: http://arxiv.org/abs/1808.09582

[19] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

7. Appendix

7.1. Experimental Details

All experimental code can be found at `https://github.com/mikkyhu/transformers`. The repo is a fork of the HuggingFace transformers repo [16], which conveniently reimplements many popular Transformer-based models in PyTorch.

We ran experiments on Princeton Natural Language Processing's ionic cluster. For GPUs, we used Nvidia RTX 2080Tis with 11GB memory. Experiments were conducted on GPT-2 "small," a version of GPT-2 with 124M parameters.

For calibration, generation, and additional properties of the English language, we used the Google Billion Words (GBW) corpus [2]. Table 4 provides instructions on how to reproduce the tables and figures in Sections 1-6.

7.2. Generation Samples

All generations in Table 5 were prompted with the following context:

> "It is a damaging blow," International Cycling Union (UCI) president Pat McQuaid told Reuters.

Happy to provide any additional experimental details or generation samples! I can be reached at myhu@princeton.edu.

7.3. Honor Code

I pledge my honor that this paper represents my own work in accordance with University regulations.

Figure	Experimental Details
Figure 1, Tables 1 and 2, Figure 6	Averaged entropies across 128 independent generations over 100 unrelated contexts. Contexts sampled randomly from GBW. Set generation length to 100. Entropy rate drift measured from token 10 to token 100. We started measurements from token 10 because we noticed some variability in the first 10 tokens. Since we are more interested in long-term trends, we wanted to discount this variability.
Figure 2	Randomly sampled a context from GBW. Plugged this context into GPT-2, sorted the probability distribution, and plotted the distribution for the top 50 tokens.
Figure 3	Obtained 300,000 contexts from GBW. For each context, computed the probability mass captured by the top 512 tokens. Constructed a histogram of these proportions.
Table 3	All calibrations performed on a 100-line, randomly chosen subset of GBW. We hereby refer to this corpus at GBW-100. Calibrated on GBW-100 using Algorithm 3 and different values of k. Ground truth α^* value obtained by running Algorithm 2 on GBW-100. We used GPU batching to make calibrations run faster.
Figure 4	Used Algorithm 4 for generations with calibration. Here, we used a single context exhibiting entropy rate blowup. The context is: ‘ “It is a damaging blow,” International Cycling Union (UCI) president Pat McQuaid told Reuters. ’ Call this context C. Due to the slowness of local calibration, we could not feasibly average across 100 unrelated contexts as in Figure 1. Averaged entropies across 128 independent generations. Entropy rate drift measured from token 10 to token 100.
Figure 5	Prompted GPT-2 with context C. Computed lookahead entropies for each token. Sorted tokens and plotted each token’s corresponding lookahead entropy.

Table 4: Instructions for reproducing tables and figures.

Settings	Generation
Algorithm 4, $k = 64$, $\alpha = 0.4082$	He added that the Paris Games would run into "intensity changes", with riders moving around "channeling learning towards training and mental growth after pooling", a planner said at the opening ceremony last week by the TUC.
Algorithm 4, $k = 512$, $\alpha = 0.0859$	Additionally, he said he felt many professional cyclists "are mired in uncertainty and ask tough questions, cause controversy and matter less" in the future.
No calibration	In an address to weight-room officials in London on Sunday, McQuaid said: "Today we approve the ban on law-abiding cyclists in a position to lose their cycling privileges by curtailing their source of funding."

Table 5: Outputs of GPT-2 given different calibration settings. The prompt: ' "It is a damaging blow," International Cycling Union (UCI) president Pat McQuaid told Reuters. '