Local and Global Decoding in Text Generation

Daniel Gareev,[uni.lu] **Thomas Hofmann,**[ETH] **Ezhilmathi Krishnasamy,**[uni.lu] **Tiago Pimentel**[ETH]
[uni.lu]University of Luxembourg, [ETH]ETH Zürich
daniel.gareev@gmail.com, ezhilmathi.krishnasamy@uni.lu
{thomas.hofmann, tiago.pimentel}@inf.ethz.ch

Abstract

Text generation, a key component in applications such as dialogue systems, relies on decoding algorithms that sample strings from a language model distribution. Traditional methods, such as top-k and top-π, apply local normalisation to the model's output distribution, which can distort it. In this paper, we investigate the effect of this distortion by introducing globally-normalised versions of these decoding methods. Additionally, we propose an independent Metropolis-Hastings algorithm to approximate sampling from globally-normalised distributions without explicitly computing them. Our empirical analysis compares the performance of local and global normalisation across two decoding algorithms (top-k and top-π) with various hyperparameters, using Pythia language models. Results show that, in most configurations, global decoding performs worse than the local decoding version of the same algorithms—despite preserving the distribution's integrity. Our results suggest that distortion is an important feature of local decoding algorithms.

lowlypalace/global-decoding

1 Introduction

Text generation is increasingly used in everyday applications, serving as a key component in dialogue agents like GPT-4 (OpenAI et al., 2024). Given a pre-trained language model (LM), text generation typically involves using **decoding algorithms**,[1] which extract a string $\mathbf{w}$ from a language model $p_{\boldsymbol{\theta}}$. In open-ended text generation, these algorithms are usually stochastic, allowing users to *sample* strings. Importantly, the goal of text generation is to produce **high-quality strings**, and a string's quality does not necessarily align with the probability mass assigned to it by a model (Holtzman et al., 2020; Zhang et al., 2021; Meister et al., 2022).

[1]See Welleck et al. (2024) for a review.

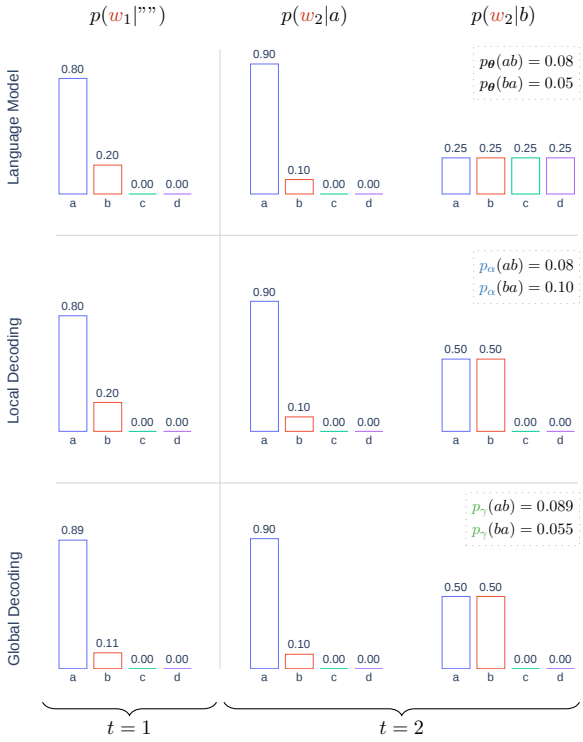

Figure 1: A simple language model over a four-symbol alphabet $\{a, b, c, d\}$ under local and global decoding. For this model, the probability of ab is higher than ba; however, the opposite is true for local decoding. In this example, top-k is 2 and maximum lengths are $T = 2$.

Over recent years, several decoding algorithms have been proposed (e.g., top-k, top-π; Fan et al., 2018; Holtzman et al., 2020; Basu et al., 2021; Hewitt et al., 2022; Meister et al., 2023b). Most of these methods operate in two steps: (i) they first prune the model's output, assigning zero probability to a large set of strings, and (ii) they then sample from this pruned distribution. However, before sampling, the distribution must be renormalised, i.e., its values must be adjusted to sum to one. This renormalisation is typically performed *locally* for each context, meaning that each conditional $p_{\boldsymbol{\theta}}(\cdot \mid \mathbf{w}_{<t})$ is renormalised independently; a process known as **local normalisation**. Importantly, by renormalising contexts independently, local normalisation can distort distribution $p_{\boldsymbol{\theta}}$ (see Fig. 1).

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 14577–14597

This paper explores the effects of such distortion on decoding algorithms. Specifically, we propose globally-normalised versions of classical decoding methods such as top-k and top-π. Globally-normalised methods prune the model's output identically to the locally-normalised version. However, instead of renormalising each context separately, they normalise the entire distribution $p_{\boldsymbol{\theta}}(\cdot)$ at once; a process known as **global normalisation**. Unlike local normalisation, global normalisation does not distort the distribution, allowing us to examine how distortion affects decoding performance.

However, global normalisation is generally intractable as it requires computing a sum over an infinite set of strings. To address this issue, we use a Markov chain Monte Carlo (MCMC) method to sample strings, adapting the independent Metropolis-Hastings (IMH) algorithm to the decoding setting. As IMH only requires unnormalised probabilities, this method allows us to approximately sample from the globally-normalised distribution without explicitly computing it.

In our experiments, we compare locally and globally normalised versions of two decoding algorithms: top-k and top-π. We run these algorithms on Pythia models (ranging from 70m to 2.8b in size) with several hyperparameter configurations (8 settings for each algorithm; with k spanning from 5 to 10,000 and π from 0.01 to 0.99). Our results show that globally-normalised methods generally perform worse than their locally-normalised counterparts, as evaluated by MAUVE scores. Additionally, our results suggest that local normalisation leads to longer, less repetitive and overall higher-quality text. We conclude that the distortion introduced by local decoding is an important component contributing to its performance.

2 Language Modelling

A language model, denoted as $p_{\boldsymbol{\theta}}(\mathbf{w})$ with parameters $\boldsymbol{\theta}$, defines a probability distribution over the set of all finite strings $\mathbf{w} = \langle w_1, w_2, \ldots, w_{|\mathbf{w}|} \rangle \in \Sigma^*$, where Σ represents an alphabet of subword units. These models are generally defined autoregressively as:

$$p_{\boldsymbol{\theta}}(\mathbf{w}) = \prod_{t=1}^{|\mathbf{w}|+1} p_{\boldsymbol{\theta}}(w_t \mid \mathbf{w}_{<t}) \tag{1}$$

Here, $w_{|\mathbf{w}|+1}$ represents a special end-of-sequence symbol ($\text{eos} \notin \Sigma$). While $p_{\boldsymbol{\theta}}(\mathbf{w})$ represents a global distribution over Σ^*, the conditionals $p_{\boldsymbol{\theta}}(w_t \mid \mathbf{w}_{<t})$ describe local distributions over an eos-augmented set of subword units $\overline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup \{\text{eos}\}$. Importantly, sampling from $p_{\boldsymbol{\theta}}(\mathbf{w})$ is straightforward, as it reduces to sampling iteratively from $p_{\boldsymbol{\theta}}(w_t \mid \mathbf{w}_{<t})$ until eos is selected.

In practice, users of language models typically impose a maximum string length T to constrain the output of $p_{\boldsymbol{\theta}}$. To facilitate the upcoming discussion, we formally define such T-maxlength language models here.

Definition 1. *A **T-maxlength language model** is a LM for which any string $\mathbf{w}$ longer than T has a probability of zero. Formally:*

$$|\mathbf{w}| > T \implies p_{\boldsymbol{\theta}}(\mathbf{w}) = 0 \tag{2}$$

This implies that for any T-length prefix $\mathbf{w}_{\leq T}$ with non-zero probability, $p_{\boldsymbol{\theta}}(\text{eos} \mid \mathbf{w}_{\leq T}) = 1$.

3 Decoding Algorithms

In this section, we first introduce both local and global decoding algorithms. We then discuss how these distributions compare to each other.

3.1 Local Decoding

Most **decoding algorithms** define a **pruning function** $\mathcal{D} : \Sigma^* \to \mathcal{P}(\overline{\Sigma})$ which, given a prefix $\mathbf{w}_{<t}$, returns a subset of $\overline{\Sigma}$ to retain in the distribution.[2] Subwords which are not in $\mathcal{D}(\mathbf{w}_{<t})$ are then assigned a probability of zero.[3] The pruning function is used to modify the LM's output distribution as follows:

$$\widehat{p}_u(w \mid \mathbf{w}_{<t}) = p_{\boldsymbol{\theta}}(w \mid \mathbf{w}_{<t}) \, \mathbb{1}\{w \in \mathcal{D}(\mathbf{w}_{<t})\} \tag{3}$$

Here, $\mathbb{1}\{\cdot\}$ is an indicator function that returns 1 if the condition holds and 0 otherwise. Thus, $\widehat{p}_u(w \mid \mathbf{w}_{<t})$ represents an **unnormalised pruned distribution**, where each subword $w \notin \mathcal{D}(\mathbf{w}_{<t})$ is re-assigned zero probability. A **local decoding algorithm** then normalises this distribution as:[4]

$$p_\alpha(w \mid \mathbf{w}_{<t}) = \frac{\widehat{p}_u(w \mid \mathbf{w}_{<t})}{\sum_{w' \in \overline{\Sigma}} \widehat{p}_u(w' \mid \mathbf{w}_{<t})} \tag{4a}$$

$$p_\alpha(\mathbf{w}) = \prod_{t=1}^{|\mathbf{w}|+1} p_\alpha(w_t \mid \mathbf{w}_{<t}) \tag{4b}$$

[2] $\mathcal{P}(\overline{\Sigma})$ refers to the powerset of alphabet $\overline{\Sigma}$.

[3] Our description of local decoding algorithms is inspired by the sampling adapters framework (Meister et al., 2023a).

[4] We assume that $\mathcal{D}$ is defined such that the probabilities are well-defined for all contexts, i.e., $\forall \mathbf{w}_{<t} \in \Sigma^*$, we have $\sum_{w' \in \overline{\Sigma}} \widehat{p}_u(w' \mid \mathbf{w}_{<t}) > 0$.

where we use subscript α to denote these distributions as locally normalised. As with the original model $p_{\boldsymbol{\theta}}(\mathbf{w})$, sampling from $p_\alpha(\mathbf{w})$ is straightforward: one can simply iteratively sample from the conditional $p_\alpha(\cdot \mid \mathbf{w}_{<t})$ at each time step. Several popular decoding algorithms can be instantiated by defining the function $\mathcal{D}$. We provide two examples next.

Definition 2. ***Top-k decoding*** *(Fan et al., 2018) is a local decoding algorithm with pruning function:*

$$\mathcal{D}_{\mathrm{k}=k'}(\mathbf{w}_{<t}) = \operatorname*{argmax}_{\mathcal{D}' \subseteq \overline{\Sigma}} \sum_{w \in \mathcal{D}'} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}_{<t}), \quad \text{s.t. } |\mathcal{D}'| = k' \tag{5}$$

Definition 3. ***Top-π decoding*** *(Holtzman et al., 2020) is a local decoding algorithm with pruning function:*

$$\mathcal{D}_{\pi=\pi'}(\mathbf{w}_{<t}) = \operatorname*{argmin}_{\mathcal{D}' \subseteq \overline{\Sigma}} |\mathcal{D}'|, \quad \text{s.t.} \sum_{w \in \mathcal{D}'} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}_{<t}) \geq \pi' \tag{6}$$

3.2 Global Decoding

Most decoding algorithms operate over $p_{\boldsymbol{\theta}}(\mathbf{w})$ as described above: they select a subset of strings to assign zero probability and then re-normalise this distribution *locally*. However, this re-normalisation process can also be done *globally* instead. We define **global decoding algorithms** as:

$$\widehat{p}_u(\mathbf{w}) = \prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w_t \mid \mathbf{w}_{<t}) \tag{7a}$$

$$p_\gamma(\mathbf{w}) = \frac{\widehat{p}_u(\mathbf{w})}{\sum_{\mathbf{w}' \in \Sigma^*} \widehat{p}_u(\mathbf{w}')} \tag{7b}$$

where γ marks this distribution as globally normalised. Unlike local normalisation, global normalisation does not **distort** the distribution beyond the pruning process. For any unpruned string, we have $p_\gamma(\mathbf{w}) \propto p_{\boldsymbol{\theta}}(\mathbf{w})$.

3.3 Local vs. Global Decoding

The distributions $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ can differ significantly. They might, for instance, rank strings in the opposite order; given two strings $\mathbf{w}$ and $\mathbf{w}'$, we might have: $p_\gamma(\mathbf{w}) < p_\gamma(\mathbf{w}')$ and $p_\alpha(\mathbf{w}) > p_\alpha(\mathbf{w}')$. Moreover, the Kullback-Leibler (KL) divergence between them can be arbitrarily large. We now prove two theorems about these distributions.

Theorem 1. *Let $\mathcal{V}_T$ be a set that includes all T-maxlength language models $p_{\boldsymbol{\theta}}(\mathbf{w})$ (see Defn. 1). There exist language models $p_{\boldsymbol{\theta}} \in \mathcal{V}_T$, for which the top-$k$ and top-π decoding versions $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ have* KL*s bounded below as:*

$$\sup_{p_{\boldsymbol{\theta}} \in \mathcal{V}_T} \mathrm{KL}\big(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})\big) \in \Omega(T) \tag{8a}$$

$$\sup_{p_{\boldsymbol{\theta}} \in \mathcal{V}_T} \mathrm{KL}\big(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})\big) \in \Omega(T) \tag{8b}$$

where Ω represents a lower bound in asymptotic notation.

Proof. See App. C. □

This theorem states that there is at least one choice of $p_{\boldsymbol{\theta}}(\mathbf{w})$ for which the KL between local and global decoding distributions grows linearly with the maximum string length T. Therefore, these distributions can differ considerably. In the next theorem, we also provide an upper bound for the divergence between the two distributions.

Theorem 2. *Let $p_{\min}$ be the minimum probability retained at each time step by either top-k (whose $p_{\min} = \frac{k}{|\Sigma|}$) or top-$\pi$ (whose $p_{\min} = \pi$). When using either of these decoding algorithms, both forward and reverse* KL*s between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ are upper bounded by:*

$$\mathrm{KL}\big(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})\big) \leq T \log \frac{1}{p_{\min}}, \tag{9a}$$

$$\mathrm{KL}\big(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})\big) \leq T \log \frac{1}{p_{\min}} \tag{9b}$$

where $p_{\boldsymbol{\theta}}(\mathbf{w})$ is a T-maxlength language model.

Proof. See App. D. □

This second theorem upper-bounds the KL between local and global decoding in terms of $p_{\min}$, the minimum probability kept by the pruning function for any context $\mathbf{w}_{<t}$. Notably, top-π typically uses hyperparameters that result in relatively large $p_{\min}$ values (e.g., $\pi \in [.8, .9, .95]$), which may explain why these settings are preferred—choosing smaller π could lead to larger distortions. In contrast, top-k is usually applied with hyperparameters resulting in smaller $p_{\min}$ values (e.g., $k \in [5, 10, 100]$). Could its reduced distortion be what gives top-π an advantage over top-k? Our experiments aim to explore this and related questions.

A Note on Prior Work. Previous research has also explored differences between local and global normalisation in text generation. For instance, Goyal et al. (2019) note that—while locally and globally normalised language models are equally expressive in general—local normalisation may impose constraints on the strings returned by beam search. Most similar to our work, Zhang et al. (2021) introduce a globally normalised version of temperature sampling and compare it to its locally normalised version.[5] However, their comparison between global and local decoding is less direct as they also impose a threshold on the maximum allowable probability of a sampled string. To the best of our knowledge, this paper is the first to systematically compare global and local normalisation in text generation, focusing on the widely used top-k and top-π decoding methods.

4 Sampling Strings with Independent Metropolis–Hastings

As noted in our introduction, sampling strings directly from $p_\gamma(\mathbf{w})$ is generally intractable. However, given access to the unnormalised $\widehat{p}_u(\mathbf{w})$, Markov chain Monte Carlo algorithms allow us to approximate this sampling process. Several such methods exist in the controllable text generation literature (Miao et al., 2019; Amini et al., 2023; Forristal et al., 2023; Lew et al., 2023; Du et al., 2024, *inter alia*).[6] Here, we propose a new approach based on independent Metropolis-Hastings. The method proceeds as follows: we first sample an initial string $\mathbf{w}^{(1)}$ from a **proposal distribution** $p_{\text{imh}}(\mathbf{w})$. Then, over N iterations, we sample new proposal strings $\mathbf{w}' \sim p_{\text{imh}}(\mathbf{w})$ and decide whether to accept the new string based on an **accept–reject distribution** $p_{\text{imh}}(\checkmark \mid \mathbf{w}', \mathbf{w}^{(n-1)})$. These steps are summarised in the following equations:

$$\mathbf{w}' \sim p_{\text{imh}}(\mathbf{w}) \tag{10a}$$

$$x \sim \texttt{Uniform}([0, 1]) \tag{10b}$$

$$\mathbf{w}^{(k)} = \begin{cases} \mathbf{w}' & \texttt{if } x \leq p_{\text{imh}}(\checkmark \mid \mathbf{w}', \mathbf{w}^{(n-1)}) \\ \mathbf{w}^{(n-1)} & \texttt{else} \end{cases} \tag{10c}$$

Finally, we take string $\mathbf{w}^{(N+1)}$ as our sample. Notably, if we define the accept–reject distribution as:

$$p_{\text{imh}}(\checkmark \mid \mathbf{w}', \mathbf{w}) \stackrel{\text{def}}{=} \min\left(1, \frac{\widehat{p}_u(\mathbf{w}')\, p_{\text{imh}}(\mathbf{w})}{\widehat{p}_u(\mathbf{w})\, p_{\text{imh}}(\mathbf{w}')}\right) \tag{11}$$

[5]We note this experiment is only present in the paper's arXiv version but not in the HumEval workshop version.

[6]Non-MCMC-based methods also exist for controllable text generation (e.g., Yang and Klein, 2021).

```
def ind_metropolis_hastings(p_α, p̂_u, N):
    p_imh = define_accept_reject(p_α, p̂_u)
    w = p_α.sample()
    for n in range(N):
        w' = p_α.sample()
        x = random.uniform(low=0.0, high=1.0)
        if u ≤ p_imh(✓ | w', w):
            w = w'
    return w
```

Figure 2: Pseudo-code for independent Metropolis–Hastings sampling.

this procedure approximates sampling from $p_\gamma(\mathbf{w}) \propto \widehat{p}_u(\mathbf{w})$ for large enough N. The convergence rate of IMH depends on how closely $p_{\text{imh}}(\mathbf{w})$ matches $p_\gamma(\mathbf{w})$ (Wang, 2021). As we do not have direct access to $p_\gamma(\mathbf{w})$, we instead use its locally normalised version as the proposal distribution:

$$p_{\text{imh}}(\mathbf{w}) \stackrel{\text{def}}{=} p_\alpha(\mathbf{w}) \tag{12}$$

We present a pseudo-code of this method in Fig. 2.

A Note on Prior Work. Previous methods in controlled generation (e.g., Miao et al., 2019) similarly sample strings according to Eq. (10), with the primary difference being their definition of $p_{\text{imh}}(\mathbf{w})$. These methods are based on Metroplis-Hastings and define a *conditional* proposal distribution $p_{\text{imh}}(\mathbf{w} \mid \mathbf{w}^{(n-1)})$. In contrast, we propose an *independent* Metropolis-Hastings method for multiple reasons. First, we can batch sample multiple strings from $p_{\text{imh}}(\mathbf{w})$ simultaneously. Second, we can compute both $\widehat{p}_u(\mathbf{w})$ and $p_{\text{imh}}(\mathbf{w})$ for a string in a single pass through the model, and simultaneously for multiple strings in a batch. Third, some choices of decoding algorithms (e.g., top-k with small k) assign zero probability to most strings in Σ^*. Depending on the choice of $p_{\text{imh}}(\mathbf{w} \mid \mathbf{w}^{(n-1)})$ in prior work, some sequences may become unreachable from others, potentially breaking the convergence guarantees of Metropolis-Hastings.

5 Experimental Setup

As discussed above, our experiments are designed to compare local and global decoding algorithms, specifically evaluating the effects of local normalisation's distortion on decoding performance. To achieve this, we generate a set of strings using pre-trained language models—particularly Pythia (Biderman et al., 2023)—with either local or global decoding strategies. Our experimental setup is flexible and can be applied to any local decoding algorithm. For this study, we focus on top-k and top-π decoding, exploring a broad range of configurations

for each. Below, we provide details on the key components of our experimental design, including the models we use, decoding algorithm configurations, IMH hyperparameters and evaluation protocols.

5.1 Models

For our experiments, we use Pythia (Biderman et al., 2023), a suite of open-source autoregressive (decoder-only) language models $p_{\boldsymbol{\theta}}(\mathbf{w})$. Pythia models are well-suited for performing scientific experiments with controlled settings—such as studying decoding behaviours and analysing their impact on open-ended text generation quality. Specifically, we use pythia-70m, pythia-410m, pythia-1.4b and pythia-2.8b. All models are run in double precision to avoid numerical instability.

5.2 Decoding Configurations

As mentioned above, we focus our experiments on top-k and top-π. We analyse several configurations of these methods. We study top-k with $k \in \{5, 10, 50, 100, 500, 1000, 5000, 10{,}000\}$ and we explore top-π with $\pi \in \{0.01, 0.05, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99\}$. Additionally, we run experiments without applying any pruning strategy, which is equivalent to running top-k while keeping the entire vocabulary $\mathcal{D}_{k=|\Sigma|}$ (composed of 50,432 tokens), or top-π while keeping all probability mass $\mathcal{D}_{\pi=1}$. This serves as a baseline decoding strategy and corresponds to using $\mathcal{D}(\mathbf{w}_{<t}) = \overline{\Sigma}$ in Eq. (3), as no words are pruned by the decoding algorithm.

5.3 Text Sampling

For text generation, Pythia models require at least one token in the input context. We use the eos token as a prompt, which is a common practice for generating open-ended text continuations. We generate 100,000 sequences for each pair of model–decoding configurations. At each time step, tokens w_t are sampled from $p_\theta(\cdot \mid \mathbf{w}_{<t})$ until either the maximum length of $T = 512$ tokens is reached, or until the eos token is sampled.

5.4 IMH Hyperparameters

To approximate sampling from $p_\gamma(\mathbf{w})$, we use independent Metropolis–Hastings. We run IMH algorithm 200 times for each model and decoding configuration, using independent string samples for each. Each IMH call is run for $N = 500$ iterations, thus requiring 100,000 sequences in total (as $100{,}000 = 500 \cdot 200$). We then select the last accepted sequence from each IMH call, resulting in 200 strings approximately sampled from p_γ.

5.5 Evaluation Methods

We use MAUVE (Pillutla et al., 2021) to evaluate the model-generated samples when using either local or global decoding. This metric measures the similarity between model-generated text and human-generated reference text. As human-generated reference text, we use the test set of the WebText dataset (Radford et al., 2019). Higher MAUVE scores imply that model-generated strings are more similar to the human-generated reference samples, suggesting higher text quality. MAUVE scores are reported in Fig. 3, as well as in Tab. 1 (in the appendix).[7] Additionally, we compute self-BLEU scores (Zhu et al., 2018) to measure the diversity of generated texts. Higher self-BLEU scores indicate lower diversity, meaning the generated samples are more similar to each other. Lower self-BLEU scores thus typically correspond to higher-quality text. Self-BLEU scores are shown in Fig. 4, as well as in Tab. 2 (in the appendix).

5.6 Bootstrapping

For each model-decoding configuration, we generate 100,000 sequence samples. These sequences are then (i) used to run IMH, producing 200 global decoding sequences, and (ii) subsampled without repetition to obtain 200 local decoding sequences. To compute confidence intervals, we apply bootstrapping to these samples. Specifically, we resample with replacement from the original 100,000 sequences to create a new set of 100,000 sequences. This resampling is done 10 times, after which we generate globally and locally decoded samples using the bootstrapped sets. We compute evaluation metrics (MAUVE and self-BLEU) based on these decoded sequences and report the mean values along with 95% confidence intervals, derived from the 10 bootstrapped evaluation runs.

6 Main Results

In this section, we present and analyse our results. First, we evaluate locally and globally decoded text using Mauve and self-BLEU metrics. Next, we examine the distortion caused by local normalisation and assess the impact of using IMH to sample from globally-normalised distributions. Lastly, we provide a qualitative analysis of the generated text.

[7] Although the values for $\mathcal{D}_{k=|\Sigma|}$ and $\mathcal{D}_{\pi=1.0}$ differ for local and global decoding, this is because the sequences for evaluation were sampled independently, even though the decoding methods coincide for this configuration.

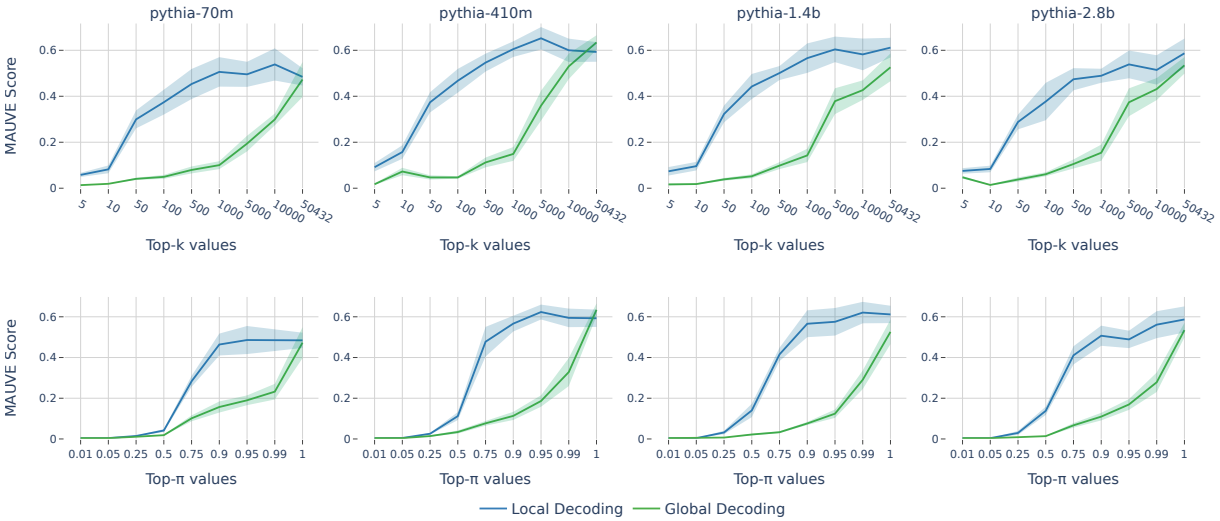

Figure 3: MAUVE evaluation scores when using local and global decoding with various top-k (Top) and top-π (Bottom) settings. Results are averaged over 10 runs, and 95% confidence intervals are shown.

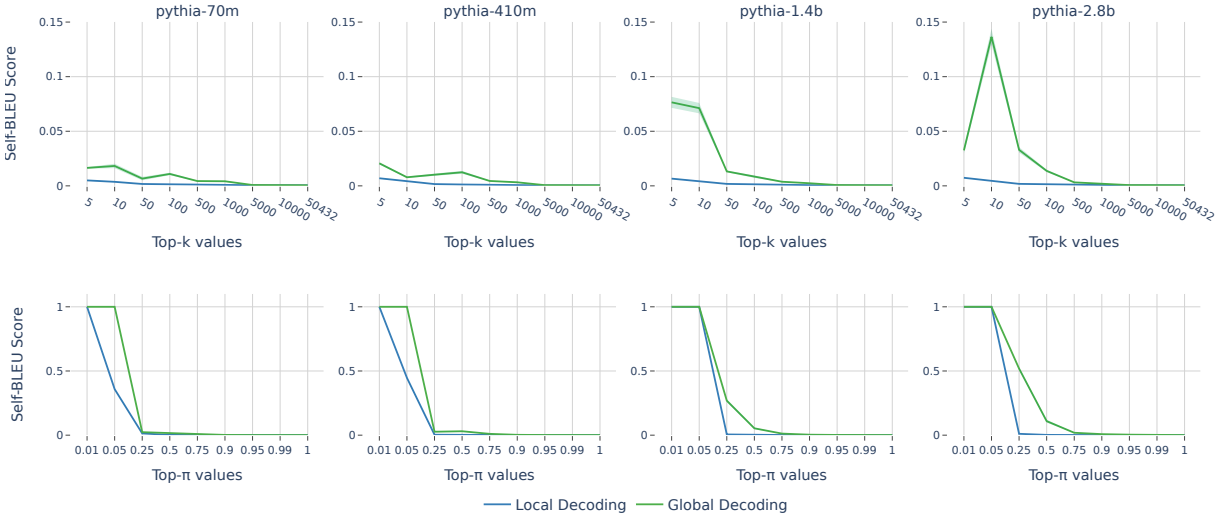

Figure 4: Self-BLEU evaluation scores when using local and global decoding with various top-k (Top) and top-π (Bottom) settings. Results are averaged over 10 runs, and 95% confidence intervals are shown.

6.1 Decoding Quality

Fig. 3 shows MAUVE scores for both local and global decoding. As shown in this figure, local decoding algorithms generally achieve higher MAUVE scores across most configurations compared to global decoding algorithms. This implies that local normalisation produces more human-like text, with the difference particularly noticeable in the mid-range settings of top-k or top-π. Notably, local and global decoding produce equivalent distributions when either $k = |\Sigma|$, or $\pi = 1.0$. They are also equivalent when $k = 1$ or $\pi \to 0$, where the pruning process retains only one string. However, global decoding scores drop rapidly as $k < |\Sigma|$ or $\pi < 1.0$, while local decoding scores even increase for some models. This suggests that, while global normalisation preserves the overall distribution, local decoding's distortion may enhance text quality rather than degrade it. Our results thus imply that these distortion artefacts introduced by local normalisation may improve text coherence and its resemblance to human-generated text.

6.2 Text Repetitions

Tab. 2 presents self-BLEU scores for text samples from both local and global decoding algorithms. As shown, local decoding generally produces more diverse text, reflected by its lower self-BLEU scores. Global normalisation thus appears to lead to *less* coherent sequences with *higher* repetition rates.

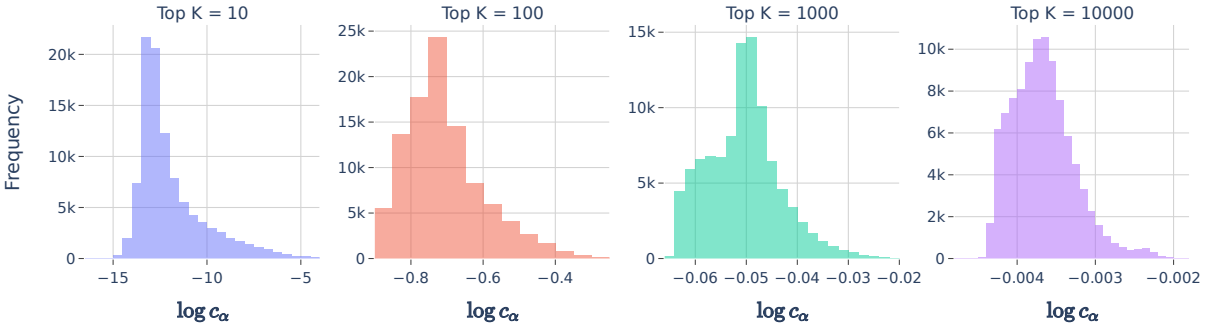

Figure 5: Histogram of local normalisation constants per sequence for pythia-2.8b when using different k values for top-k. These constants $c_\alpha(\mathbf{w})$ are defined in Eq. (13). The x-axis is log-scaled.

6.3 Impact of Local Normalisation

We now examine the distortion introduced by local normalisation, which highlights the differences between global and local decoding. To do this, we present the sequence-level local normalisation constants in Fig. 5. These constants are the product of all subword-level local normalisation constants in a sequence, defined as:

$$\overline{c_\alpha}(\mathbf{w}) \stackrel{\text{def}}{=} \prod_{t=1}^{|\mathbf{w}|} \underbrace{\sum_{w' \in \overline{\Sigma}} \widehat{p}_u(w' \mid \mathbf{w}_{<t})}_{c_\alpha(\mathbf{w}_{<t}) \text{ in Defn. 4}} \tag{13}$$

Since these constants vary across different strings, they distort the distribution: $p_\alpha(\mathbf{w}) = \frac{\widehat{p}_u(\mathbf{w})}{\overline{c_\alpha}(\mathbf{w})} \not\propto p_{\boldsymbol{\theta}}(\mathbf{w})$. Fig. 5 shows that these constants are more uniform for top-k with large k values (note that the x-axis have different scales and are logged).[8] This suggests that, as discussed in §3.3, locally normalised distributions with these configurations are less distorted. Combined with the results in Fig. 3, our findings suggest that a *moderate* level of distortion, may improve decoding performance.

6.4 Effect of IMH Sampling on Decoding

In this section, we examine the impact of IMH sampling on our results. As discussed in §4, IMH offers a way to *approximately* sample from distribution $p_\gamma(\mathbf{w})$, using only unnormalised scores $\widehat{p}_u(\mathbf{w})$. This method is approximate and converges to the target distribution as the number of iterations increases (i.e., as $N \to \infty$). On the other hand, running IMH with $N = 1$ is equivalent to sampling from the proposal distribution, which in our case is the local decoding distribution, $p_\alpha(\mathbf{w})$. In Fig. 6, we show how MAUVE scores change with

[8] Also note that $\log p_\alpha(\mathbf{w}) = \log \widehat{p}_u(\mathbf{w}) - \log \overline{c_\alpha}(\mathbf{w})$.

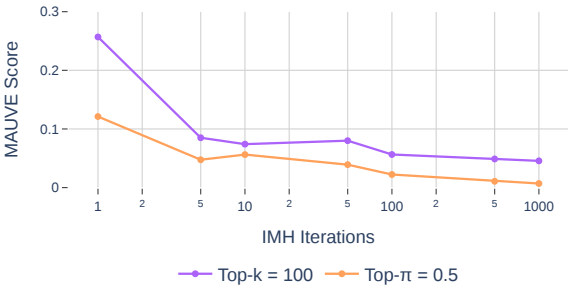

Figure 6: MAUVE Scores as the number of IMH iterations N changes. We evaluate global decoding with top-k ($k = 100$) and top-π ($\pi = 0.5$). We use 200 strings for each evaluation, except when $N = 1000$, in which case we use only 100 strings. Strings were generated using pythia-2.8b.

the number of IMH iterations used. As this figure illustrates, increasing N from 1 to 10 significantly affects the results, but scores stabilise for $N > 100$. Therefore, even with approximate samples, we believe that our overall results are representative of the global distribution $p_\gamma(\mathbf{w})$.

6.5 A Qualitative Analysis of Sampled Strings

In Tab. 3 (shown in App. A.3), we present samples generated by global and local decoding using pythia-1.4b. This table shows qualitative differences between the texts generated using the two methods. In the case of local decoding, the text samples are generally coherent, with well-formed sentences and logical flow. In contrast, global decoding often produces text with irrelevant symbols, code snippets, or disjointed phrases. Overall, the text generated by global decoding tends to lack the coherence and fluency observed in the local decoding samples.

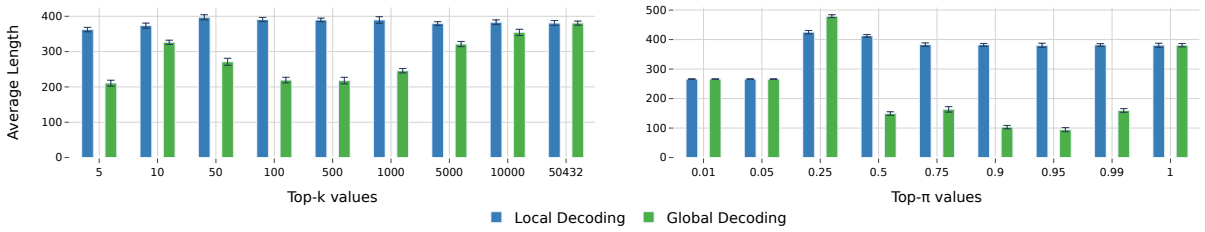

Figure 7: Average length of sequences using local and global decoding across top-k (Left) and top-π (Right) decoding configurations. The length of a sequence represents the number of tokens including eos. Sequences were generated with pythia-2.8b. Results are averaged over 10 runs, and 95% confidence intervals are shown.

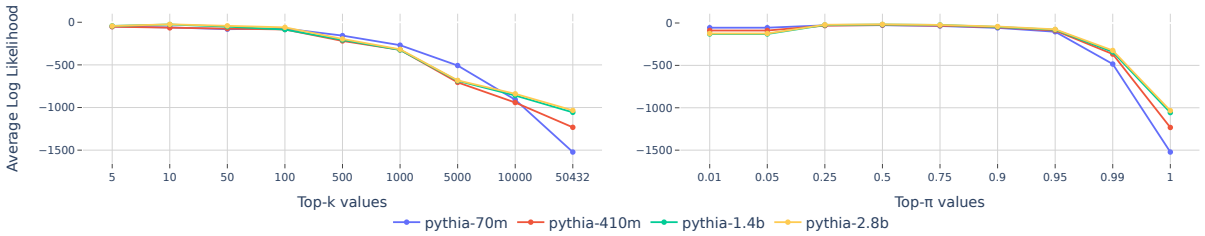

Figure 8: Log-likelihood of globally decoded samples under the original model's distribution $p_{\boldsymbol{\theta}}$. We sample sequences as: $\mathbf{w} \sim p_\gamma(\mathbf{w})$. We then evaluate their log-probabilities as: $\log p_{\boldsymbol{\theta}}(\mathbf{w})$. Results are averaged over 10 runs.

7 Discussion: Why Local Decoding Outperforms Global Decoding?

Interestingly, despite the theoretical advantages of global decoding, local decoding consistently outperforms it in most scenarios. We now propose three key reasons for this discrepancy.

First, sequences generated through global decoding tend to be shorter on average (see Fig. 7). As probabilities compound multiplicatively over the length of a sequence, and are constrained to values no greater than one, the overall probability of longer sequences decreases exponentially. In contrast, local normalisation constants (used to normalise contextual probabilities in local decoding) are at least one. When compounded, they can thus lead to exponentially higher probabilities for longer sequences under p_α. As a result, local decoding thus tends to favour longer sequences when compared to global decoding.

Second, global decoding suffers from the well-known **probability-quality paradox** in text generation (Holtzman et al., 2020; Zhang et al., 2021; Meister et al., 2022). This paradox states that the most likely sequences according to a language model are not always the highest quality ones. In fact, high-probability sequences frequently include repetitive or incoherent content. Prior research has shown, for instance, that maximising likelihood during decoding, such as in beam search, often leads to degenerate and repetitive outputs (Holtzman et al., 2020). In contrast, stochastic methods like top-π sampling produce more diverse, human-like text (Meister et al., 2022). Similarly, Stahlberg and Byrne (2019) show that beam search often does not return the most likely string under a language model, and that these "search errors" improve its generated text quality. These findings align with our results, which show that global decoding—by assigning higher probability to sequences which are already likely under the original model—tends to generate less coherent and lower-quality text.[9]

Lastly, language models often allocate significant probability mass to repetitive sequences. Even under local decoding, small values of k can cause the model to get stuck in repetitive loops. These loops typically have high local normalisation constants, as the repetitive continuations tend to be high probability and are often included top-k set (i.e., in $\mathcal{D}_{k=k'}$). In such cases, local decoding effectively discounts these sequences by adjusting their probabilities. In contrast, global decoding does not apply this discount, giving these repetitive sequences a higher chance of being sampled. This results in less coherent and more repetitive text under global normalisation.

[9] In Fig. 8, we show the log-likelihood of globally decoded sequences evaluated under the original language model's distribution $p_{\boldsymbol{\theta}}$. This figure shows that for lower values of k or π, the likelihoods under $p_{\boldsymbol{\theta}}$ are higher. Fig. 9 then shows the same is true when likelihoods are evaluated using the locally normalised distribution.

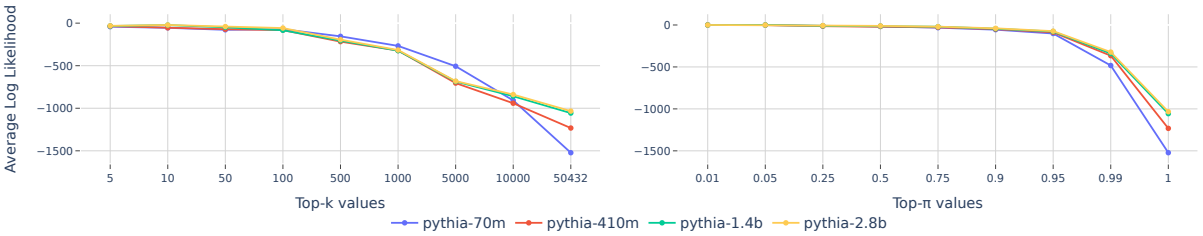

Figure 9: Log-likelihood of globally decoded samples under the local decoding distribution p_α. We sample sequences as: $\mathbf{w} \sim p_\gamma(\mathbf{w})$. We then evaluate their log-probabilities as: $\log p_\alpha(\mathbf{w})$. Results are averaged over 10 runs.

8 Conclusion

In this paper, we explored the effects of local and global normalisation on text generation quality, focusing on two popular decoding methods: top-k and top-π. We introduced the concept of global decoding algorithms and adapted local decoding algorithms to their global counterparts. Our empirical comparison revealed that, while global decoding preserves the original distribution, it often produces shorter, less coherent and more repetitive text than local decoding. In contrast, local decoding—despite introducing distortions to the original distribution—typically results in more coherent text. Our findings thus suggest that the distortion introduced by local decoding algorithms might be a beneficial feature rather than a flaw. Future research on decoding algorithms should explore and evaluate the effects of local normalisation alongside pruning strategies.

Limitations

As with any research project, our work has theoretical and empirical limitations. Theoretically, the lower bounds presented in §3.3 are asymptotic and omit key constants required to fully understand how different decoding algorithms and their hyperparameters influence performance. Additionally, we cannot directly sample from the global decoding distributions and rely on IMH for approximate sampling. Although we examine the impact of this choice in our experiments (see Fig. 6), it may still affect the results. Empirically, our experiments are limited to Pythia models ranging in size from 70m to 2.8b. Due to large number of samples required by our sampling methods and relatively large number of hyperparameter configurations analysed, we were limited to using smaller-sized models. As models continue to improve, global distributions may become better calibrated, potentially yielding stronger results. Additionally, we only run experiments in English. Extending this analysis to other languages is an important next step. Furthermore, our focus was limited to open-ended text generation and applying our approach to other natural language tasks, such as summarisation, could provide further insights. Finally, our evaluation primarily relies on automatic metrics such as MAUVE, and we do not perform human evaluations. Although MAUVE scores have been shown to correlate well with human judgements (Pillutla et al., 2021), they are not a definitive standard. Human evaluations would offer a more comprehensive understanding of the results. Moreover, incorporating other automatic metrics, such as Zipf Coefficient (Holtzman et al., 2020) and Generation Perplexity (Fan et al., 2018), could provide a more well-rounded assessment of decoding performance.

Acknowledgement

We thank Pietro Lesci, Gregor Bachmann, Yahya Emara, Clara Meister, Afra Amini and Sotiris Anagnostidis for their feedback on earlier versions of this paper. We also thank our anonymous reviewers.

References

Afra Amini, Li Du, and Ryan Cotterell. 2023. Structured voronoi sampling. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. 2021. Mirostat: A neural text decoding algorithm that directly controls perplexity. In *International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across

training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Li Du, Afra Amini, Lucas Torroba Hennigen, Xinyan Velocity Yu, Holden Lee, Jason Eisner, and Ryan Cotterell. 2024. Principled gradient-based MCMC for conditional sampling of text. In *Forty-first International Conference on Machine Learning*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Jarad Forristal, Fatemehsadat Mireshghallah, Greg Durrett, and Taylor Berg-Kirkpatrick. 2023. A block Metropolis-Hastings sampler for controllable energy-based text generation. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 403–413, Singapore. Association for Computational Linguistics.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2019. An empirical investigation of global and local normalization for recurrent neural sequence models using a continuous relaxation to beam search. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1724–1733, Minneapolis, Minnesota. Association for Computational Linguistics.

John Hewitt, Christopher Manning, and Percy Liang. 2022. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. 2023. Sequential Monte Carlo steering of large language models using probabilistic programs. *arXiv preprint*.

Clara Meister, Tiago Pimentel, Luca Malagutti, Ethan Wilcox, and Ryan Cotterell. 2023a. On the efficacy of sampling adapters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1455, Toronto, Canada. Association for Computational Linguistics.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023b. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121.

Clara Meister, Gian Wiher, Tiago Pimentel, and Ryan Cotterell. 2022. On the probability–quality paradox in language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 36–45, Dublin, Ireland. Association for Computational Linguistics.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. CGMH: Constrained sentence generation by Metropolis-Hastings sampling. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, et al. 2024. GPT-4 technical report. *arXiv preprint*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Guanyang Wang. 2021. Exact convergence rate analysis of the independent Metropolis–Hastings algorithms. *arXiv preprint*.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint*.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP*

Systems (HumEval), pages 25–33, Online. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.

A Detailed Results

A.1 MAUVE Scores

	pythia-70m		pythia-410m		pythia-1.4b		pythia-2.8b	
	MAUVE$_{\text{LOCAL}}$	MAUVE$_{\text{GLOBAL}}$	MAUVE$_{\text{LOCAL}}$	MAUVE$_{\text{GLOBAL}}$	MAUVE$_{\text{LOCAL}}$	MAUVE$_{\text{GLOBAL}}$	MAUVE$_{\text{LOCAL}}$	MAUVE$_{\text{GLOBAL}}$
$\mathcal{D}_{k=\lvert\Sigma\rvert}$	0.484 ± 0.075	0.473 ± 0.152	0.593 ± 0.085	0.634 ± 0.062	0.612 ± 0.085	0.526 ± 0.121	0.587 ± 0.128	0.534 ± 0.072
$\mathcal{D}_{k=5.0}$	0.058 ± 0.018	0.013 ± 0.004	0.091 ± 0.036	0.017 ± 0.007	0.074 ± 0.037	0.017 ± 0.008	0.075 ± 0.024	0.048 ± 0.011
$\mathcal{D}_{k=10.0}$	0.082 ± 0.034	0.019 ± 0.002	0.157 ± 0.059	0.073 ± 0.033	0.096 ± 0.037	0.018 ± 0.006	0.084 ± 0.029	0.014 ± 0.004
$\mathcal{D}_{k=50.0}$	0.299 ± 0.078	0.041 ± 0.009	0.374 ± 0.088	0.047 ± 0.018	0.323 ± 0.073	0.039 ± 0.012	0.288 ± 0.064	0.038 ± 0.018
$\mathcal{D}_{k=100.0}$	0.374 ± 0.107	0.049 ± 0.017	0.467 ± 0.103	0.047 ± 0.011	0.442 ± 0.109	0.052 ± 0.017	0.377 ± 0.163	0.061 ± 0.017
$\mathcal{D}_{k=500.0}$	0.453 ± 0.131	0.079 ± 0.03	0.546 ± 0.078	0.112 ± 0.043	0.501 ± 0.061	0.098 ± 0.03	0.474 ± 0.097	0.105 ± 0.04
$\mathcal{D}_{k=1000.0}$	0.506 ± 0.128	0.1 ± 0.035	0.605 ± 0.068	0.149 ± 0.06	0.566 ± 0.128	0.143 ± 0.058	0.489 ± 0.061	0.154 ± 0.07
$\mathcal{D}_{k=5000.0}$	0.495 ± 0.109	0.194 ± 0.068	0.652 ± 0.098	0.359 ± 0.13	0.604 ± 0.111	0.379 ± 0.113	0.539 ± 0.121	0.374 ± 0.121
$\mathcal{D}_{k=10000.0}$	0.538 ± 0.14	0.299 ± 0.051	0.6 ± 0.102	0.53 ± 0.111	0.582 ± 0.138	0.427 ± 0.085	0.515 ± 0.128	0.431 ± 0.096
$\mathcal{D}_{\pi=0.01}$	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0
$\mathcal{D}_{\pi=0.05}$	0.004 ± 0.001	0.004 ± 0.0	0.005 ± 0.001	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0	0.004 ± 0.0
$\mathcal{D}_{\pi=0.25}$	0.015 ± 0.004	0.011 ± 0.005	0.026 ± 0.008	0.014 ± 0.006	0.032 ± 0.017	0.007 ± 0.002	0.03 ± 0.02	0.009 ± 0.001
$\mathcal{D}_{\pi=0.5}$	0.041 ± 0.014	0.019 ± 0.005	0.112 ± 0.039	0.034 ± 0.015	0.14 ± 0.066	0.022 ± 0.007	0.138 ± 0.041	0.014 ± 0.004
$\mathcal{D}_{\pi=0.75}$	0.283 ± 0.054	0.101 ± 0.029	0.478 ± 0.146	0.077 ± 0.023	0.415 ± 0.066	0.033 ± 0.009	0.411 ± 0.09	0.067 ± 0.023
$\mathcal{D}_{\pi=0.9}$	0.464 ± 0.108	0.157 ± 0.055	0.566 ± 0.078	0.113 ± 0.04	0.566 ± 0.132	0.076 ± 0.015	0.507 ± 0.099	0.11 ± 0.037
$\mathcal{D}_{\pi=0.95}$	0.486 ± 0.138	0.19 ± 0.048	0.623 ± 0.073	0.186 ± 0.056	0.575 ± 0.135	0.124 ± 0.041	0.489 ± 0.086	0.169 ± 0.053
$\mathcal{D}_{\pi=0.99}$	0.484 ± 0.107	0.232 ± 0.077	0.595 ± 0.091	0.328 ± 0.138	0.621 ± 0.105	0.291 ± 0.091	0.561 ± 0.132	0.278 ± 0.095
$\mathcal{D}_{\pi=1.0}$	0.484 ± 0.075	0.473 ± 0.152	0.593 ± 0.085	0.634 ± 0.062	0.612 ± 0.085	0.526 ± 0.121	0.587 ± 0.128	0.534 ± 0.072

Table 1: MAUVE evaluation scores when using local and global decoding with various top-k and top-π settings. Results are averaged over 10 runs, and 95% confidence intervals are shown.

A.2 Self-BLEU Scores

	pythia-70m		pythia-410m		pythia-1.4b		pythia-2.8b	
	BLEU$_{\text{LOCAL}}$	BLEU$_{\text{GLOBAL}}$	BLEU$_{\text{LOCAL}}$	BLEU$_{\text{GLOBAL}}$	BLEU$_{\text{LOCAL}}$	BLEU$_{\text{GLOBAL}}$	BLEU$_{\text{LOCAL}}$	BLEU$_{\text{GLOBAL}}$
$\mathcal{D}_{k=\lvert\Sigma\rvert}$	0.0006 ± 0.0001	0.0005 ± 0.0002	0.0005 ± 0.0002	0.0005 ± 0.0001	0.0005 ± 0.0001	0.0005 ± 0.0002	0.0005 ± 0.0001	0.0005 ± 0.0002
$\mathcal{D}_{k=5.0}$	0.005 ± 0.0007	0.0164 ± 0.0014	0.0069 ± 0.0007	0.0206 ± 0.0021	0.0066 ± 0.0008	0.0765 ± 0.0101	0.0074 ± 0.0007	0.0325 ± 0.0056
$\mathcal{D}_{k=10.0}$	0.0037 ± 0.0007	0.0182 ± 0.0041	0.0044 ± 0.0008	0.0079 ± 0.0013	0.0044 ± 0.0003	0.0712 ± 0.0095	0.0047 ± 0.0004	0.1365 ± 0.0145
$\mathcal{D}_{k=50.0}$	0.0017 ± 0.0004	0.0066 ± 0.0032	0.0016 ± 0.0003	0.0102 ± 0.002	0.0018 ± 0.0004	0.0132 ± 0.0014	0.0018 ± 0.0003	0.033 ± 0.005
$\mathcal{D}_{k=100.0}$	0.0014 ± 0.0002	0.0109 ± 0.0022	0.0012 ± 0.0003	0.0124 ± 0.0026	0.0014 ± 0.0004	0.0084 ± 0.0022	0.0015 ± 0.0003	0.0137 ± 0.0021
$\mathcal{D}_{k=500.0}$	0.0008 ± 0.0001	0.0044 ± 0.0008	0.0008 ± 0.0002	0.0045 ± 0.0008	0.0008 ± 0.0001	0.0038 ± 0.0009	0.0009 ± 0.0002	0.0032 ± 0.0008
$\mathcal{D}_{k=1000.0}$	0.0008 ± 0.0002	0.0042 ± 0.001	0.0006 ± 0.0002	0.0033 ± 0.001	0.0007 ± 0.0001	0.0024 ± 0.0007	0.0008 ± 0.0002	0.0017 ± 0.0005
$\mathcal{D}_{k=5000.0}$	0.0006 ± 0.0001	0.0008 ± 0.0003	0.0005 ± 0.0002	0.0006 ± 0.0002	0.0005 ± 0.0001	0.0007 ± 0.0002	0.0005 ± 0.0001	0.0007 ± 0.0002
$\mathcal{D}_{k=10000.0}$	0.0006 ± 0.0001	0.0006 ± 0.0001	0.0005 ± 0.0002	0.0006 ± 0.0001	0.0005 ± 0.0002	0.0006 ± 0.0002	0.0006 ± 0.0002	0.0006 ± 0.0002
$\mathcal{D}_{\pi=0.01}$	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
$\mathcal{D}_{\pi=0.05}$	0.3584 ± 0.0104	1.0 ± 0.0	0.4466 ± 0.0128	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
$\mathcal{D}_{\pi=0.25}$	0.0135 ± 0.0024	0.0234 ± 0.0045	0.0035 ± 0.0006	0.0268 ± 0.0058	0.0056 ± 0.001	0.2687 ± 0.0218	0.0105 ± 0.0018	0.5187 ± 0.0204
$\mathcal{D}_{\pi=0.5}$	0.002 ± 0.0004	0.0165 ± 0.003	0.0014 ± 0.0003	0.0306 ± 0.0048	0.0016 ± 0.0003	0.0534 ± 0.0116	0.0021 ± 0.0003	0.1093 ± 0.0228
$\mathcal{D}_{\pi=0.75}$	0.0016 ± 0.0002	0.0132 ± 0.0041	0.0012 ± 0.0002	0.0103 ± 0.003	0.0012 ± 0.0001	0.0123 ± 0.0018	0.0013 ± 0.0002	0.0188 ± 0.004
$\mathcal{D}_{\pi=0.9}$	0.001 ± 0.0001	0.0019 ± 0.0004	0.0007 ± 0.0001	0.0026 ± 0.001	0.0009 ± 0.0001	0.0038 ± 0.0008	0.0008 ± 0.0001	0.008 ± 0.0014
$\mathcal{D}_{\pi=0.95}$	0.0007 ± 0.0001	0.0006 ± 0.0002	0.0006 ± 0.0002	0.0011 ± 0.0005	0.0006 ± 0.0001	0.0017 ± 0.0005	0.0008 ± 0.0002	0.004 ± 0.0015
$\mathcal{D}_{\pi=0.99}$	0.0006 ± 0.0001	0.0002 ± 0.0001	0.0005 ± 0.0003	0.0003 ± 0.0001	0.0005 ± 0.0001	0.0005 ± 0.0002	0.0006 ± 0.0001	0.0005 ± 0.0001
$\mathcal{D}_{\pi=1.0}$	0.0006 ± 0.0001	0.0005 ± 0.0002	0.0005 ± 0.0002	0.0005 ± 0.0001	0.0005 ± 0.0001	0.0005 ± 0.0002	0.0005 ± 0.0001	0.0005 ± 0.0002

Table 2: Self-BLEU evaluation scores when using local and global decoding with various top-k and top-π settings. Results are averaged over 10 runs, and 95% confidence intervals are shown.

A.3 Samples of Strings

	Local Decoding	Global Decoding
$\mathcal{D}_{k=\lvert\Sigma\rvert}$	Xmas is exciting this year with the discovery of the new Dave.\n \nBelow you can see the new Dave in a	Q:\n \nHow to start VLC using jars from android\n \nI am trying to start VLC using JPMS provided by Wizard
$\mathcal{D}_{k=5}$	Q:\n \nHow to use the new.NET 4.0 framework with Mono 2.8.7 on Ubuntu 11.10?\n \nHow can I use the new.NET	\n ##ifndef BOOST_MPL_AUX_TEMPLATE_ARITY_HPP_INCLUDED\n ##define BOOST_MPL_AUX_TEMPLATE_ARITY_HPP_INCLUDED
$\mathcal{D}_{k=10}$	\n \nI just read about the first time the UW football team beat Iowa, and I thought it was really cool:	1\n
$\mathcal{D}_{k=50}$	Hurricane Dorian has left at least 15 dead and hundreds missing in the Bahamas as it approaches the	##ifndef OSQA_CONFIG_H_\n ##define OSQA_CONFIG_H_\n \n /** @file\n *\n * Definitions shared by all platforms\n
$\mathcal{D}_{k=100}$	Tag: politics\n \nOn the afternoon of Wednesday, May 17th, I joined hundreds of thousands of other Amer	/*\nCopyright The Kubernetes Authors.\n \nLicensed under the Apache License, Version 2.0 (the "License")
$\mathcal{D}_{k=500}$	Results of hyperthermic intraperitoneal chemotherapy versus hyperthermic intraperitoneal chemotherap	<?xml version="1.0" encoding="UTF-8"?>\n <!– Copyright (C) 2013 The Android Open Source Project\n \n
$\mathcal{D}_{k=1000}$	Peace love, Hope and Fasting.\n \nThursday , November 10, 2008\n \nOkay ...A few weeks ago I had an interest	define("ace/mode/javascript_highlight_rules",["require","exports","module","ace/lib/oop","ace/mode/t
$\mathcal{D}_{k=5000}$	West Lafayette\n \nIndiana âĂŹs unexpected first visit to baseball crowncases its home teamâĂŹs success from	\n 624 P.2d 352 (1981)\n 105 Idaho 1002\nSue HANKS, Plaintiff-Appellant,\nv .\nHARTSDALE TOWNSHIP, Defendant
$\mathcal{D}_{k=10000}$	These algorithms have a "signature capture score"\nof about 3.\n	Beth Cooper\n \nNew York Times best-selling author Beth Cooper will deliver a lecture on poetry and cul
$\mathcal{D}_{\pi=0.01}$	Q:\n \nHow to get the value of a variable in a function?\n \nI have a function that is called from a butto	Q:\n \nHow to get the value of a variable in a function?\n \nI have a function that is called from a butto
$\mathcal{D}_{\pi=0.05}$	This invention relates to a semiconductor device and a method of manufacturing the same, and more pa	\n ##include "qabstractnetworkmodel.h"\n
$\mathcal{D}_{\pi=0.25}$	The present invention relates to a semiconductor device and a method of manufacturing the same, and	/*\n * Copyright (c) 2017-2018 THL A29 Limited, a Tencent company. All Rights Reserved.\n *\n * License
$\mathcal{D}_{\pi=0.5}$	K. T. P. S. B. P. C.\n \nK . T. P. S. B. P. C. is a small town in the Indian state of Uttar Pradesh. It	import { IconDefinition, IconPrefix, IconName } from "@fortawesome/fontawesome-common-types";\nexport
$\mathcal{D}_{\pi=0.75}$	YAML 1.1\n \%TAG!u! tag:unity3d.com,2011:\n — !u!21 &2100000\nMaterial :\n serializedVersion: 6\n m_Objec	export { default as Slider } from ’./Slider’;\n
$\mathcal{D}_{\pi=0.9}$	—\nabstract : ’We investigate the concept of the infinite divisibility of simple functions using som	Embodiments described herein relate to a device for testing nerve stimulation therapy for chronic pa
$\mathcal{D}_{k=0.95}$	Secretary of State Rex Tillerson, Defense Secretary Jim Mattis, CENTCOM Commander Army Gen. Lloyd Au	export * from ’./domain.module’;\nexport { useLinkTo, registerLinkTo } from ’./utils’;\n
$\mathcal{D}_{\pi=0.99}$	Associated Press\n \nSYRACUSE , N.Y. (AP) - A retired Northeastern University professor has been ordered	Along the dark, shadowy streets of ChicoMira moving south on Angioteque, cops have spotted three bro

Table 3: Texts generated with pythia-1.4b using local and global decoding. We present one randomly selected text for each configuration. The sequences are trimmed to 100 characters for readability.

B Definitions and Useful Lemma

For mathematical convenience, we now define local and global normalisation constants.

Definition 4. *The **local normalisation constant** (denoted $c_\alpha : \Sigma^* \to [0, 1]$) is defined as:*

$$c_\alpha(\mathbf{w}) \stackrel{\text{def}}{=} \sum_{w \in \overline{\Sigma}} \widehat{p}_u(w \mid \mathbf{w}) = \widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w}) \tag{14}$$

Definition 5. *The **global normalisation constant** (denoted $c_\gamma \in [0, 1]$) and the **context-specific global normalisation constant** (denoted $\widetilde{c}_\gamma : \Sigma^* \to [0, 1]$) are defined as:*

$$c_\gamma \stackrel{\text{def}}{=} \sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}'|+1} \widehat{p}_u(w'_t \mid \mathbf{w}'_{<t}), \qquad \widetilde{c}_\gamma(\mathbf{w}) \stackrel{\text{def}}{=} \sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}'|+1} \widehat{p}_u(w'_t \mid \mathbf{w} \circ \mathbf{w}'_{<t}) \tag{15}$$

We note that $\widetilde{c}_\gamma(\mathbf{w})$ can be written recursively as:

$$\widetilde{c}_\gamma(\mathbf{w}) = \sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}'|+1} \widehat{p}_u(w'_t \mid \mathbf{w} \circ \mathbf{w}'_{<t}) \tag{16a}$$

$$= \widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w}) \sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}'|+1} \widehat{p}_u(w'_t \mid \mathbf{w} \circ w \circ \mathbf{w}'_{<t}) \tag{16b}$$

$$= \widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w})\, \widetilde{c}_\gamma(\mathbf{w} \circ w) \tag{16c}$$

and that $c_\gamma = \widetilde{c}_\gamma(\emptyset)$, where $\emptyset$ denotes an empty string.

Given these constants, we can rewrite local and global decoding algorithms, defined in (4) and (7), respectively, as:

$$p_\alpha(\mathbf{w}) = \prod_{t=1}^{|\mathbf{w}|+1} \frac{\widehat{p}_u(w \mid \mathbf{w}_{<t})}{\sum_{w' \in \Sigma} \widehat{p}_u(w' \mid \mathbf{w}_{<t})} = \frac{\prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w \mid \mathbf{w}_{<t})}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})} \tag{17}$$

$$p_\gamma(\mathbf{w}) = \frac{\prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w_t \mid \mathbf{w}_{<t})}{\sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w_t \mid \mathbf{w}_{<t})} = \frac{\prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w_t \mid \mathbf{w}_{<t})}{c_\gamma} \tag{18}$$

We now prove a recursively-defined lower bound on c_γ which will be useful later.

Lemma 1. *The global normalisation constant's value is lower bounded by:*

$$c_\gamma \geq \left(\min_{\mathbf{w} \in \Sigma^*} c_\alpha(\mathbf{w})\right)^T \left(\min_{\mathbf{w} \in \Sigma^T} \widetilde{c}_\gamma(\mathbf{w})\right) \tag{19}$$

Proof. We start this proof by lower bounding the context-specific global normalisation constant with a local normalisation constant's value:

$$\widetilde{c}_\gamma(\mathbf{w}) = \widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w})\, \widetilde{c}_\gamma(\mathbf{w} \circ w) \tag{20a}$$

$$\geq \widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w}) \left(\min_{w \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w)\right) \tag{20b}$$

$$\geq \left(\widehat{p}_u(\text{eos} \mid \mathbf{w}) + \sum_{w \in \Sigma} \widehat{p}_u(w \mid \mathbf{w})\right) \left(\min_{w \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w)\right) \tag{20c}$$

$$= c_\alpha(\mathbf{w}) \min_{w \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w) \tag{20d}$$

We can now recursively expand this lower bound:

$$\widetilde{c}_\gamma(\mathbf{w}) \geq c_\alpha(\mathbf{w}) \min_{w \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w) \tag{21a}$$

$$\geq c_\alpha(\mathbf{w}) \min_{w_1 \in \Sigma} \left(c_\alpha(\mathbf{w} \circ w_1) \min_{w_2 \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w_1 \circ w_2) \right) \tag{21b}$$

$$\geq c_\alpha(\mathbf{w}) \min_{w_1 \in \Sigma} \left(c_\alpha(\mathbf{w} \circ w_1) \min_{w_2 \in \Sigma} \left(c_\alpha(\mathbf{w} \circ w_1 \circ w_2) \min_{w_3 \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w_1 \circ w_2 \circ w_3) \right) \right) \tag{21c}$$

$$\geq c_\alpha(\mathbf{w}) \min_{w_1 \in \Sigma} \left(c_\alpha(\mathbf{w} \circ w_1) \cdots \right. \tag{21d}$$

$$\left. \min_{w_{T-1} \in \Sigma} \left(c_\alpha(\mathbf{w} \circ w_1 \circ \cdots \circ w_{T-1}) \min_{w_T \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w_1 \circ \cdots \circ w_{T-1} \circ w_T) \right) \right)$$

$$= \min_{w_1 \in \Sigma} \left(\cdots \min_{w_{T-1} \in \Sigma} \left(\right. \right. \tag{21e}$$

$$\left. \left. c_\alpha(\mathbf{w})\, c_\alpha(\mathbf{w} \circ w_1) \cdots c_\alpha(\mathbf{w} \circ w_1 \circ \cdots \circ w_{T-1}) \min_{w_T \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ w_1 \circ \cdots \circ w_{T-1} \circ w_T) \right) \right)$$

$$= \min_{\mathbf{w}' \in \Sigma^{T-1}} \left(\prod_{t=0}^{T-1} c_\alpha(\mathbf{w} \circ \mathbf{w}'_{\leq t}) \min_{w_T \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}' \circ w_T) \right) \tag{21f}$$

$$\tag{21g}$$

Finally, we get the bound above by noting that $\min_{w'' \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}' \circ w'') \leq \min_{\mathbf{w}'' \in \Sigma^T} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}'')$ and $c_\alpha(\mathbf{w} \circ \mathbf{w}'_{\leq t}) \leq \min_{\mathbf{w}'' \in \Sigma^*} c_\alpha(\mathbf{w} \circ \mathbf{w}'')$:

$$\widetilde{c}_\gamma(\mathbf{w}) \geq \min_{\mathbf{w}' \in \Sigma^{T-1}} \left(\prod_{t=0}^{T-1} c_\alpha(\mathbf{w} \circ \mathbf{w}'_{\leq t}) \min_{w_T \in \Sigma} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}' \circ w_T) \right) \tag{22a}$$

$$\geq \min_{\mathbf{w}' \in \Sigma^{T-1}} \left(\prod_{t=0}^{T-1} \min_{\mathbf{w}'' \in \Sigma^*} c_\alpha(\mathbf{w} \circ \mathbf{w}'') \right) \left(\min_{\mathbf{w}'' \in \Sigma^T} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}'') \right) \tag{22b}$$

$$= \left(\min_{\mathbf{w}'' \in \Sigma^*} c_\alpha(\mathbf{w} \circ \mathbf{w}'') \right)^T \left(\min_{\mathbf{w}'' \in \Sigma^T} \widetilde{c}_\gamma(\mathbf{w} \circ \mathbf{w}'') \right) \tag{22c}$$

Replacing $\widetilde{c}_\gamma(\mathbf{w})$ with $c_\gamma = \widetilde{c}_\gamma(\emptyset)$ completes the proof. □

C Proof of Lower-bound on Maximum Divergence between Global and Local Distributions (Thm. 1)

Theorem 1. *Let $\mathcal{V}_T$ be a set that includes all T-maxlength language models $p_{\boldsymbol{\theta}}(\mathbf{w})$ (see Defn. 1). There exist language models $p_{\boldsymbol{\theta}} \in \mathcal{V}_T$, for which the top-$k$ and top-π decoding versions $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ have* KL*s bounded below as:*

$$\sup_{p_{\boldsymbol{\theta}} \in \mathcal{V}_T} \mathrm{KL}\big(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})\big) \in \Omega(T) \tag{8a}$$

$$\sup_{p_{\boldsymbol{\theta}} \in \mathcal{V}_T} \mathrm{KL}\big(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})\big) \in \Omega(T) \tag{8b}$$

where Ω represents a lower bound in asymptotic notation.

Proof. This proof follows trivially from Lemmas 2 and 3 below. □

Lemma 2. *Let $\mathcal{V}_T$ be a set including all T-maxlength language models $p_{\boldsymbol{\theta}}(\mathbf{w})$ (see Defn. 1). There exist language models $p_{\boldsymbol{\theta}} \in \mathcal{V}_T$, whose decoding versions $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ have a reverse* KL *bounded below by:*

$$\sup_{p_{\boldsymbol{\theta}} \in \mathcal{V}_T} \mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) \in \Omega(T) \tag{23}$$

Proof. We prove this by construction. Let $\Sigma = \{a, b, c_1, ..., c_{|\Sigma|-2}\}$ and $p_{\boldsymbol{\theta}}(\mathbf{w})$ be defined such that:

$$p_{\boldsymbol{\theta}}(\mathbf{w}) = \begin{cases} x & \mathbf{w} = a \\ (1-x)\left(\frac{1}{|\Sigma|}\right)^{T-1} & \mathbf{w} \in b \circ \Sigma^{T-1} \\ 0 & \texttt{else} \end{cases} \tag{24}$$

where, when applied to sets, $\circ$ represents elementwise concatenation, i.e., $b \circ \Sigma^{T-1} = \{b \circ \mathbf{w}' \mid \mathbf{w}' \in \Sigma^{T-1}\}$. We can get a lower bound for this LM's reverse KL as:

$$\begin{aligned}
\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) &= \sum_{\mathbf{w}\in\Sigma^*} p_\alpha(\mathbf{w}) \log \frac{p_\alpha(\mathbf{w})}{p_\gamma(\mathbf{w})} && \text{(25a)}\\
&= \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{c_\gamma}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}\right] && \text{(25b)}\\
&= \log c_\gamma + \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{1}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}\right] && \text{(25c)}\\
&= \log c_\gamma + \sum_{\mathbf{w}\in b\circ\Sigma^{T-1}} p_\alpha(\mathbf{w}) \log \frac{1}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})} && \text{(25d)}\\
&= \log \underbrace{\left(x + \underbrace{\sum_{\mathbf{w}\in b\circ\Sigma^{T-1}} (1-x)\left(\frac{1}{|\Sigma|}\right)^{T-1}}_{\geq 0}\right)}_{c_\gamma} + (1-x) \log \frac{1}{\prod_{t=1}^{T+1} c_\alpha(\mathbf{w}_{<t})} && \text{(25e)}\\
&\geq \log x + (1-x) \log \frac{1}{\underbrace{c_\alpha(\emptyset)}_{=1} \cdot \prod_{t=2}^{T} c_\alpha(\mathbf{w}_{<t}) \cdot \underbrace{c_\alpha(\mathbf{w})}_{=1}} && \text{(25f)}\\
&= \log x + (1-x) \log \frac{1}{\prod_{t=2}^{T} c_\alpha(\mathbf{w}_{<t})} && \text{(25g)}\\
&= \log x + (1-x) \sum_{t=2}^{T} \log \frac{1}{c_\alpha(\mathbf{w}_{<t})} && \text{(25h)}\\
&= \log x + (1-x)\,(T-1) \log \frac{1}{c_\alpha(\mathbf{w}_{<t})} \in \Omega(T) && \text{(25i)}
\end{aligned}$$

The $\sup_{p_{\boldsymbol{\theta}}\in\mathcal{V}_T} \mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w}))$ is greater or equal to this LM's KL, and so is also bounded below. This completes the proof. □

Lemma 3. *Let $\mathcal{V}_T$ be a set including all T-maxlength language models $p_{\boldsymbol{\theta}}(\mathbf{w})$ (see Defn. 1). There exist language models $p_{\boldsymbol{\theta}} \in \mathcal{V}_T$, whose decoding versions $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ have a forward* KL *bounded below by:*

$$\sup_{p_{\boldsymbol{\theta}}\in\mathcal{V}_T} \mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \in \Omega(T) \tag{26}$$

Proof. We now prove this by construction for top-k, but note that a similar proof applies for top-π. Let $\Sigma = \{a, b, c_1, ..., c_{|\Sigma|-2}\}$ and $p_{\boldsymbol{\theta}}(\mathbf{w})$ be defined such that:

$$p_{\boldsymbol{\theta}}(\mathbf{w}) = \begin{cases} x^T & \mathbf{w} = a_1 \circ a_2 \circ \cdots \circ a_T \\ x^t\,(1-x)\left(\frac{1}{|\Sigma|}\right)^{T-t-1} & \mathbf{w} \in a_1 \circ \cdots \circ a_t \circ b_{t+1} \circ \Sigma^{T-t-1} \\ 0 & \texttt{else} \end{cases} \tag{27}$$

Further, let $1 > x > \frac{k}{|\Sigma|}$. First, we simplify the value of the global normalisation constant for this model:

$$c_\gamma = \sum_{\mathbf{w}' \in \Sigma^*} \prod_{t=1}^{|\mathbf{w}'|+1} \widehat{p}_u(w'_t \mid \mathbf{w}'_{<t}) \tag{28a}$$

$$= \prod_{t=1}^{|\mathbf{a}_{1:T}|+1} \widehat{p}_u(a \mid \mathbf{a}_{1:t-1}) \tag{28b}$$

$$+ \sum_{\mathbf{w} \in \mathbf{a}_{1:i} \circ b \circ \Sigma^{T-i-1}} \left(\left(\prod_{t=1}^{i} \widehat{p}_u(a_t \mid \mathbf{w}_{<t}) \right) \widehat{p}_u(b_t \mid \mathbf{w}_{\leq i}) \left(\prod_{t=i+1}^{T+1} \widehat{p}_u(w_t \mid \mathbf{w}_{<t}) \right) \right)$$

$$= x^T + \sum_{\mathbf{w} \in \mathbf{a}_{1:i} \circ b \circ \Sigma^{T-i-1}} x^i \, (1-x) \, \frac{1}{|\Sigma|}^{T-i-1} \prod_{t=1}^{T+1} \mathbb{1}\{w_t \in \mathcal{D}(\mathbf{w}_{<t})\} \tag{28c}$$

$$= x^T + \sum_{i=0}^{T-1} x^i \, (1-x) \, \frac{1}{|\Sigma|}^{T-i-1} k^{T-i-1} \tag{28d}$$

$$= x^T + (1-x) \, \frac{k}{|\Sigma|}^{T-1} \sum_{i=0}^{T-1} (\frac{x\,|\Sigma|}{k})^i \tag{28e}$$

$$= x^T + (1-x) \, \frac{k}{|\Sigma|}^{T-1} \frac{1 - (\frac{x\,|\Sigma|}{k})^T}{1 - \frac{x\,|\Sigma|}{k}} \tag{28f}$$

Now, we simplify the value of $\mathbb{E}_{\mathbf{w} \sim p_\gamma(\mathbf{w})} \left[\log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \right]$:

$$\mathop{\mathbb{E}}_{\mathbf{w} \sim p_\gamma(\mathbf{w})} \left[\log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \right]$$

$$= \sum_{\mathbf{w} \in \Sigma^*} p_\gamma(\mathbf{w}) \log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \tag{29a}$$

$$= \sum_{\mathbf{w} \in \mathbf{a}_{1:i} \circ b \circ \Sigma^{T-i-1}} p_\gamma(\mathbf{w}) \log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \tag{29b}$$

$$= \sum_{\mathbf{w} \in \mathbf{a}_{1:i} \circ b \circ \Sigma^{T-i-1}} \frac{x^i \, (1-x) \, \frac{1}{|\Sigma|}^{T-t-1}}{c_\gamma} \log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \tag{29c}$$

$$= \frac{\sum_{\mathbf{w} \in \mathbf{a}_{1:i} \circ b \circ \Sigma^{T-i-1}} x^i \, (1-x) \, \frac{1}{|\Sigma|}^{T-t-1} \log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}{c_\gamma} \tag{29d}$$

$$= \frac{(1-x) \, \frac{k}{|\Sigma|}^{T-1} \sum_{i=0}^{T-1} (\frac{x\,|\Sigma|}{k})^i \log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}{c_\gamma} \tag{29e}$$

$$= \frac{(1-x) \, \frac{k}{|\Sigma|}^{T-1} \sum_{i=0}^{T-1} (\frac{x\,|\Sigma|}{k})^i \log \left(\underbrace{\prod_{t=1}^{i} c_\alpha(\mathbf{w}_{<t})}_{=1} \prod_{t=i+1}^{T} c_\alpha(\mathbf{w}_{<t}) \underbrace{c_\alpha(\mathbf{w}_{<T+1})}_{=1} \right)}{c_\gamma} \tag{29f}$$

$$= \frac{(1-x) \, \frac{k}{|\Sigma|}^{T-1} \sum_{i=0}^{T-1} (\frac{x\,|\Sigma|}{k})^i \log \prod_{t=i+1}^{T} \frac{k}{|\Sigma|}}{c_\gamma} \tag{29g}$$

$$= \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \sum_{i=0}^{T-1} \left(\frac{x|\Sigma|}{k}\right)^i (T-i) \log \frac{k}{|\Sigma|}}{c_\gamma} \tag{29h}$$

$$= \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \left(\sum_{i=0}^{T-1} \left(\frac{x|\Sigma|}{k}\right)^i (T+1) - \sum_{i=0}^{T-1} \left(\frac{x|\Sigma|}{k}\right)^i (i+1)\right) \log \frac{k}{|\Sigma|}}{c_\gamma} \tag{29i}$$

$$= \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \left((T+1)\frac{1-\left(\frac{x|\Sigma|}{k}\right)^T}{1-\frac{x|\Sigma|}{k}} - \frac{T\left(\frac{x|\Sigma|}{k}\right)^{T+1} - (T+1)\left(\frac{x|\Sigma|}{k}\right)^T + 1}{\left(1-\frac{x|\Sigma|}{k}\right)^2}\right) \log \frac{k}{|\Sigma|}}{c_\gamma} \tag{29j}$$

$$= \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \left((T+1)\frac{1-\frac{x|\Sigma|}{k} - \left(\frac{x|\Sigma|}{k}\right)^T + \left(\frac{x|\Sigma|}{k}\right)^{T+1}}{\left(1-\frac{x|\Sigma|}{k}\right)^2} - \frac{T\left(\frac{x|\Sigma|}{k}\right)^{T+1} - (T+1)\left(\frac{x|\Sigma|}{k}\right)^T + 1}{\left(1-\frac{x|\Sigma|}{k}\right)^2}\right) \log \frac{k}{|\Sigma|}}{c_\gamma} \tag{29k}$$

$$= \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \left(\frac{\left(\frac{x|\Sigma|}{k}\right)^{T+1} - (T+1)\frac{x|\Sigma|}{k} + T}{\left(1-\frac{x|\Sigma|}{k}\right)^2}\right) \log \frac{k}{|\Sigma|}}{c_\gamma} \tag{29l}$$

Note that the KL we are interested in is defined as:

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) = \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\gamma(\mathbf{w})} \left[\log \frac{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}{c_\gamma}\right] \tag{30}$$

so we can fill in the values above into it. Now we prove this $\mathrm{KL} \in \Omega(T)$. To do so, we show that the $\lim_{T\to\infty} \frac{\mathrm{KL}}{T} = C$, for a $C > 0$. First, we isolate the terms dependent on T in the KL's equation.

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) = \log \frac{1}{c_\gamma} + \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\gamma(\mathbf{w})} \left[\log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})\right] \tag{31a}$$

$$= \log \frac{1}{x^T + (1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \frac{1-\left(\frac{x|\Sigma|}{k}\right)^T}{1-\frac{x|\Sigma|}{k}}} + \frac{(1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \left(\frac{\left(\frac{x|\Sigma|}{k}\right)^{T+1} - (T+1)\frac{x|\Sigma|}{k} + T}{\left(1-\frac{x|\Sigma|}{k}\right)^2}\right) \log \frac{k}{|\Sigma|}}{x^T + (1-x)\left(\frac{k}{|\Sigma|}\right)^{T-1} \frac{1-\left(\frac{x|\Sigma|}{k}\right)^T}{1-\frac{x|\Sigma|}{k}}} \tag{31b}$$

$$= \log \frac{x^{-T}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} + \frac{\frac{1-x}{\frac{k}{|\Sigma|}} \frac{1}{\left(1-\frac{x|\Sigma|}{k}\right)^2} \left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^{-1} - (T+1)\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^{T-1} + T\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T\right) \log \frac{k}{|\Sigma|}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} \tag{31c}$$

$$= \log \frac{x^{-T}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} + \frac{\frac{1-x}{\frac{k}{|\Sigma|}\frac{k}{|\Sigma|}\frac{1}{x}} \frac{1}{\left(1-\frac{x|\Sigma|}{k}\right)^2} \left(1 - (T+1)\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T + T\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^{T+1}\right) \log \frac{k}{|\Sigma|}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} \tag{31d}$$

$$= \log \frac{x^{-T}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} + \frac{\frac{(1-x)\,x}{\left(\frac{k}{|\Sigma|}-x\right)^2} \left(1 - (T+1)\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T + T\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^{T+1}\right) \log \frac{k}{|\Sigma|}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} \tag{31e}$$

$$= -T \log x - \log\left(1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)\right) + \frac{\frac{(1-x)\,x}{\left(\frac{k}{|\Sigma|}-x\right)^2} \left(1 - (T+1)\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T + T\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^{T+1}\right) \log \frac{k}{|\Sigma|}}{1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\left(\left(\frac{k}{|\Sigma|}\frac{1}{x}\right)^T - 1\right)} \tag{31f}$$

We now analyse the limit: $\lim_{T\to\infty} \frac{\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w}))}{T}$. Note that, by construction, $1 > x > \frac{k}{|\Sigma|}$. We thus write:

$$\lim_{T\to\infty} \frac{\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w}))}{T}$$

$$= \lim_{T\to\infty} \frac{-T\,\log x - \log\left(1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\,\left((\frac{k}{|\Sigma|}\frac{1}{x})^T - 1\right)\right) + \frac{\frac{(1-x)\,x}{\left(\frac{k}{|\Sigma|}-x\right)^2}\left(1-(T+1)(\frac{k}{|\Sigma|}\frac{1}{x})^T + T\,(\frac{k}{|\Sigma|}\frac{1}{x})^{T+1}\right)\log\frac{k}{|\Sigma|}}{1+\frac{1-x}{\frac{k}{|\Sigma|}-x}\left((\frac{k}{|\Sigma|}\frac{1}{x})^T-1\right)}}{T} \tag{32a}$$

$$= \lim_{T\to\infty} \frac{-T\,\log x}{T} + \frac{\frac{(1-x)\,x}{\left(\frac{k}{|\Sigma|}-x\right)^2}\left(1-(T+1)(\frac{k}{|\Sigma|}\frac{1}{x})^T + T\,(\frac{k}{|\Sigma|}\frac{1}{x})^{T+1}\right)\log\frac{k}{|\Sigma|}}{T\left(1 + \frac{1-x}{\frac{k}{|\Sigma|}-x}\,\left((\frac{k}{|\Sigma|}\frac{1}{x})^T - 1\right)\right)} \tag{32b}$$

$$= -\log x > 0 \tag{32c}$$

This completes the proof. □

D Upper-bounding the Divergence between Global and Local Distributions (Thm. 2)

Theorem 2. *Let $p_{\min}$ be the minimum probability retained at each time step by either top-k (whose $p_{\min} = \frac{k}{|\Sigma|}$) or top-$\pi$ (whose $p_{\min} = \pi$). When using either of these decoding algorithms, both forward and reverse* KL*s between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ are upper bounded by:*

$$\mathrm{KL}\big(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})\big) \leq T\,\log\frac{1}{p_{\min}}, \tag{9a}$$

$$\mathrm{KL}\big(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})\big) \leq T\,\log\frac{1}{p_{\min}} \tag{9b}$$

where $p_{\boldsymbol{\theta}}(\mathbf{w})$ is a T-maxlength language model.

Proof. This proof follows trivially from Lemmas 6 and 7 below. □

D.1 A General Upper-bound

In this section, we prove an upper-bound on the KL between both decoding distributions. We then provide corollaries discussing how this bound is instantiated by top-k and top-π algorithms.

Lemma 4. *Let $p_{\boldsymbol{\theta}}(\mathbf{w})$ be a T-maxlength language model (see Defn. 1). Now let $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ be global and local decoding algorithms run on top of $p_{\boldsymbol{\theta}}(\mathbf{w})$. In this case, the forward* KL *between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ is bounded above by:*

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \leq T\,\log\frac{1}{\min_{\mathbf{w}\in\Sigma^*}\, c_\alpha(\mathbf{w})} \tag{33}$$

Proof. First, we use the definition of the KL to show a bound:

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) = \sum_{\mathbf{w}\in\Sigma^*} p_\gamma(\mathbf{w}) \log \frac{p_\gamma(\mathbf{w})}{p_\alpha(\mathbf{w})} \tag{34a}$$

$$= \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\gamma(\mathbf{w})} \left[\log \frac{\frac{\prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w_t|\mathbf{w}_{<t})}{c_\gamma}}{\frac{\prod_{t=1}^{|\mathbf{w}|+1} \widehat{p}_u(w|\mathbf{w}_{<t})}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}} \right] \qquad \text{definition of } p_\gamma(\mathbf{w}) \text{ and } p_\alpha(\mathbf{w}) \tag{34b}$$

$$= \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\gamma(\mathbf{w})} \left[\log \frac{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})}{c_\gamma} \right] \qquad \text{cancel terms} \tag{34c}$$

$$= \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\gamma(\mathbf{w})} \left[\log \prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t}) \right] + \log \frac{1}{c_\gamma} \qquad c_\gamma \text{ does not depend on } \mathbf{w} \tag{34d}$$

$$\leq \log \frac{1}{c_\gamma} \qquad \text{first term is } \leq 0 \tag{34e}$$

$$\leq \log \frac{1}{(\min_{\mathbf{w}\in\Sigma^*} c_\alpha(\mathbf{w}))^T \, (\min_{\mathbf{w}\in\Sigma^T} \widetilde{c}_\gamma(\mathbf{w}))} \qquad \text{apply lemma 1} \tag{34f}$$

$$= T \log \frac{1}{\min_{\mathbf{w}\in\Sigma^*} c_\alpha(\mathbf{w})} + \log \frac{1}{\min_{\mathbf{w}\in\Sigma^T} \widetilde{c}_\gamma(\mathbf{w})} \tag{34g}$$

Now note that $\widetilde{c}_\gamma(\mathbf{w}) = \widehat{p}_u(\mathrm{eos} \mid \mathbf{w}) + \sum_{w\in\Sigma} \widehat{p}_u(w \mid \mathbf{w}) \widetilde{c}_\gamma(\mathbf{w} \circ w)$. Since our language model is T-maxlengthed, then $p_{\boldsymbol{\theta}}(\mathrm{eos} \mid \mathbf{w}) = 1$, which implies that $\widehat{p}_u(\mathrm{eos} \mid \mathbf{w}) = 1$ for all $\mathbf{w} \in \Sigma^T$. We thus have that $\min_{\mathbf{w}\in\Sigma^T} \widetilde{c}_\gamma(\mathbf{w}) = 1$. This completes the proof. □

Lemma 5. *Let $p_{\boldsymbol{\theta}}(\mathbf{w})$ be a T-maxlength language model (see Defn. 1). Now let $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ be global and local decoding algorithms run on top of $p_{\boldsymbol{\theta}}(\mathbf{w})$. In this case, the reverse* KL *between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ is bounded above by:*

$$\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) \leq T \log \frac{1}{\min_{\mathbf{w}\in\Sigma^*} c_\alpha(\mathbf{w})} \tag{35}$$

Proof. For this proof, we start with the KL's definition and show the upper-bound:

$$\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) = \sum_{\mathbf{w}\in\Sigma^*} p_\alpha(\mathbf{w}) \log \frac{p_\alpha(\mathbf{w})}{p_\gamma(\mathbf{w})} \tag{36a}$$

$$= \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{c_\gamma}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})} \right] \tag{36b}$$

$$= \log c_\gamma + \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{1}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})} \right] \tag{36c}$$

$$\leq \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{1}{\prod_{t=1}^{|\mathbf{w}|+1} c_\alpha(\mathbf{w}_{<t})} \right] \tag{36d}$$

$$\leq \mathop{\mathbb{E}}_{\mathbf{w}\sim p_\alpha(\mathbf{w})} \left[\log \frac{1}{\prod_{t=1}^{T} \min_{\mathbf{w}'\in\Sigma^*} c_\alpha(\mathbf{w}')} \right] \tag{36e}$$

$$= \log \frac{1}{\prod_{t=1}^{T} \min_{\mathbf{w}\in\Sigma^*} c_\alpha(\mathbf{w})} \tag{36f}$$

$$= T \log \frac{1}{\min_{\mathbf{w}\in\Sigma^*} c_\alpha(\mathbf{w})} \tag{36g}$$

This concludes the proof. □

D.2 An Upper-bound for Top-k (Lemma 6)

Lemma 6. *When using top-k decoding, both* KL*s (forward and reverse) between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ are bounded above by:*

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \leq T \log \frac{|\overline{\Sigma}|}{k}, \tag{37a}$$

$$\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) \leq T \log \frac{|\overline{\Sigma}|}{k} \tag{37b}$$

where $p_{\boldsymbol{\theta}}(\mathbf{w})$ is a T-maxlength language model.

Proof. For convenience, we first rewrite the definition of the set of strings unpruned by top-k here:

$$\mathcal{D}(\mathbf{w}_{<t}) = \operatorname*{argmax}_{\mathcal{D}' \subseteq \overline{\Sigma}} \sum_{w \in \mathcal{D}'} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}_{<t}), \text{ s.t. } |\mathcal{D}'| = k \tag{38}$$

Top-k's $\mathcal{D}(\mathbf{w}_{<t})$ is thus defined as the largest probability k-sized subset of $\overline{\Sigma}$. This set clearly has at least probability $\frac{k}{|\overline{\Sigma}|}$. We can thus bound the local constant's value as:

$$c_\alpha(\mathbf{w}) = \sum_{w \in \overline{\Sigma}} \widehat{p}_u(w \mid \mathbf{w}) = \sum_{w \in \overline{\Sigma}} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}) \, \mathbb{1}\{w \in \mathcal{D}(\mathbf{w})\} \geq \frac{k}{|\overline{\Sigma}|} \tag{39}$$

We can now apply this inequality to the bound in lemma 4:

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \leq T \log \frac{1}{\min_{\mathbf{w} \in \Sigma^*} c_\alpha(\mathbf{w})} \tag{40a}$$

$$\leq T \log \frac{|\overline{\Sigma}|}{k} \tag{40b}$$

The same logic applies to the $\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w}))$. This completes the proof. □

D.3 An Upper-bound for Top-π (Lemma 7)

Lemma 7. *Let $p_{\boldsymbol{\theta}}(\mathbf{w})$ be a T-maxlength language model (see Defn. 1). Now let $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ be global and local decoding algorithms based on top-π (as in Defn. 3). In this case, both* KL*s (forward and reverse) between $p_\gamma(\mathbf{w})$ and $p_\alpha(\mathbf{w})$ are bounded above by:*

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \leq T \log \frac{1}{\pi}, \tag{41a}$$

$$\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w})) \leq T \log \frac{1}{\pi} \tag{41b}$$

Proof. For convenience, we first rewrite the definition of the set of strings unpruned by top-π here:

$$\mathcal{D}(\mathbf{w}_{<t}) = \operatorname*{argmin}_{\mathcal{D}' \subseteq \overline{\Sigma}} \mathcal{D}', \quad \text{s.t.} \sum_{w \in \mathcal{D}'} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}_{<t}) \geq \pi \tag{42}$$

Top-π's $\mathcal{D}(\mathbf{w}_{<t})$ is thus defined as the smallest subset of $\overline{\Sigma}$ which has at least probability π. We can thus bound the local constant's value as:

$$c_\alpha(\mathbf{w}) = \sum_{w \in \overline{\Sigma}} \widehat{p}_u(w \mid \mathbf{w}) = \sum_{w \in \overline{\Sigma}} p_{\boldsymbol{\theta}}(w \mid \mathbf{w}) \, \mathbb{1}\{w \in \mathcal{D}(\mathbf{w})\} \geq \pi \tag{43}$$

We can now apply this inequality to the bound in lemma 4:

$$\mathrm{KL}(p_\gamma(\mathbf{w}) \,||\, p_\alpha(\mathbf{w})) \leq T \log \frac{1}{\min_{\mathbf{w} \in \Sigma^*} c_\alpha(\mathbf{w})} \tag{44a}$$

$$\leq T \log \frac{1}{\pi} \tag{44b}$$

The same logic applies to the $\mathrm{KL}(p_\alpha(\mathbf{w}) \,||\, p_\gamma(\mathbf{w}))$. This completes the proof. □