Consistency of a Recurrent Language Model With Respect to Incomplete Decoding

Sean Welleck[1*] **Ilia Kulikov**[1*] **Jaedeok Kim**[2†]
Richard Yuanzhe Pang[1] **Kyunghyun Cho**[1,3]
[1] New York University [2] Samsung Research [3] CIFAR Associate Fellow

Abstract

Despite strong performance on a variety of tasks, neural sequence models trained with maximum likelihood have been shown to exhibit issues such as length bias and degenerate repetition. We study the related issue of receiving infinite-length sequences from a recurrent language model when using common decoding algorithms. To analyze this issue, we first define inconsistency of a decoding algorithm, meaning that the algorithm can yield an infinite-length sequence that has zero probability under the model. We prove that commonly used incomplete decoding algorithms – greedy search, beam search, top-k sampling, and nucleus sampling – are inconsistent, despite the fact that recurrent language models are trained to produce sequences of finite length. Based on these insights, we propose two remedies which address inconsistency: consistent variants of top-k and nucleus sampling, and a self-terminating recurrent language model. Empirical results show that inconsistency occurs in practice, and that the proposed methods prevent inconsistency.

1 Introduction

Neural sequence models trained with maximum likelihood estimation (MLE) have become a standard approach to modeling sequences in a variety of natural language applications such as machine translation (Bahdanau et al., 2015), dialogue modeling (Vinyals et al., 2015), and language modeling (Radford et al., 2019). Despite this success, MLE-trained neural sequence models have been shown to exhibit issues such as length bias (Sountsov and Sarawagi, 2016; Stahlberg and Byrne, 2019) and degenerate repetition (Holtzman et al., 2019). These issues are suspected to be related to the maximum likelihood objective's local normalization, which results in a discrepancy between the learned model's distribution and the distribution induced by the decoding algorithm used to generate sequences (Lafferty et al., 2001; Andor et al., 2016). This has prompted the development of alternative decoding methods (Wu et al., 2016; Holtzman et al., 2019) and training objectives (Murray and Chiang, 2018; Welleck et al., 2019). In this paper, we formalize and study this discrepancy between the model and the decoding algorithm.

We begin by formally defining *recurrent neural language models*, a family that encompasses neural models used in practice, such as recurrent neural networks (Elman, 1990; Cho et al., 2014; Hochreiter and Schmidhuber, 1997), and transformers (Vaswani et al., 2017). Next, we formally define a decoding algorithm – a function that induces a distribution over sequences given a recurrent language model and a context distribution – which is used to obtain probable sequences from a model. In this paper, we show that the distribution induced by a decoding algorithm can contradict this intended use; instead, the decoding algorithm may return improbable, infinite-length sequences.

Our main finding is that a sequence which receives zero probability under a recurrent language model's distribution can receive nonzero probability under the distribution induced by a decoding algorithm. This occurs when the recurrent language model always ranks the sequence termination token outside of the set of tokens considered at each decoding step, yielding an infinite-length, zero probability sequence. This holds whenever the decoding algorithm is *incomplete*, in the sense that the algorithm excludes tokens from consideration at each step of decoding, which is the case for common methods such as greedy search, beam search, top-k sampling (Fan et al., 2018), and nucleus sampling

[*] Equal contribution. Correspondence to: Sean Welleck wellecks@nyu.edu.
[†] Work done at New York University.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 5553–5568,
November 16–20, 2020. ©2020 Association for Computational Linguistics

(Holtzman et al., 2019). We formalize our main finding using the notion of *consistency* (Chen et al., 2017) – whether a distribution assigns probability mass only to finite sequences – and prove that a consistent recurrent language model paired with an incomplete decoding algorithm can induce an inconsistent sequence distribution.

Based on the insight that inconsistency occurs due to the behavior of the termination token under incomplete decoding, we develop two methods for addressing inconsistency. First, we propose *consistent sampling methods* which guarantee that the termination token is not excluded from selection during decoding. Second, we introduce a *self-terminating recurrent language model* which ensures that the termination token is eventually ranked above all others, guaranteeing consistency under incomplete decoding.

To empirically measure inconsistency, we decode sequences from trained recurrent language models and measure the proportion of sequences with lengths far exceeding the maximum training sequence length. Our experiments on the Wikitext2 dataset (Merity et al., 2016) suggest that inconsistency occurs in practice when using incomplete decoding methods, while the proposed consistent sampling methods and self-terminating model parameterization prevent inconsistency and maintain language modeling quality.

The theoretical analysis reveals defects of existing decoding algorithms, providing a way to develop future models, inference procedures, and learning algorithms. We present methods related to sampling and model parameterization, but there are more directions for future investigation; we close with directions related to sequence-level learning.

2 Background

We begin our discussion by establishing background definitions. First, we define a sequence which is the main object of our investigation.

Definition 2.1 (Sequence)**.** *A sequence Y is an ordered collection of items from a predefined finite vocabulary V. A sequence of finite length always ends with a special token $\langle eos \rangle \in V$ that only appears at the end of a sequence.*

Each model we consider generates a sequence conditioned on context information, such as a prefix in sentence completion. To consider this, we define a context distribution.

Definition 2.2 (Context distribution)**.** *A context distribution $p(C)$ is a probability distribution defined over a set $\mathcal{C}$. An element $C \in \mathcal{C}$ is called a context.*

2.1 Recurrent Language Models

A recurrent language model is an autoregressive model of a sequence distribution, where each conditional probability is parameterized with a neural network. Importantly, we assume that all tokens in a sequence are dependent on each other under a recurrent language model. This allows us to avoid cases in which the model degenerates to a Markovian language model, such as an n-gram model with a finite n.

Definition 2.3 (Recurrent language model)**.** *A recurrent language model p_θ is a neural network that computes the following at each time step:*

$$p_\theta(y_t = v \mid y_{<t}, C) = \frac{\exp(u_v^\top h_t + c_v)}{\sum_{v' \in V} \exp(u_{v'}^\top h_t + c_{v'})},$$

where $h_t = f_\theta(y_t, h_{t-1})$ and $h_0 = g_\theta(C)$, and u, c, θ are parameters. A recurrent language model thereby computes the probability of a sequence $Y = (y_1, \ldots, y_T)$ by

$$p_\theta(Y \mid C) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, C),$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$. This distribution satisfies $y_i \not\perp y_j \mid C, \ \forall i < j$.

Practical variants of the recurrent language model differ by the choice of transition function f_θ (Elman, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Vaswani et al., 2017). The use of softmax (Bridle, 1990) implies that every unique token in the vocabulary is considered at every location of a sequence.

Remark 2.1. *Under the conditional distribution of a recurrent LM, every token $v \in V$ is assigned a positive probability, implying that $0 < p_\theta(v \mid y_{<t}, C) < 1$. Any finite sequence is probable under a recurrent LM under any context, i.e., $p_\theta(Y \mid C) > 0$ for any sequence Y of finite length.*

2.2 Decoding Algorithms

Because it is intractable to decode the most probable sequence, it is necessary in practice to use an approximate decoding algorithm.

Definition 2.4 (Decoding algorithm)**.** *A decoding algorithm $\mathcal{F}(p_\theta, C)$ is a function that generates*

a sequence $\tilde{Y}$ given a recurrent language model p_θ and context C. Let $q_\mathcal{F}$ denote the distribution induced by the decoding algorithm $\mathcal{F}$.

We consider two families of decoding algorithms. In our analysis we only consider algorithms that decode in a single pass, forward in time, without modifying previously selected tokens.

Stochastic decoding. The first family consists of stochastic algorithms. Among them, ancestral sampling is asymptotically unbiased and can be used for finding the most probable sequence, although with high variance.

Definition 2.5 (Ancestral sampling)**.** *Ancestral sampling $\mathcal{F}_{anc}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from $p_\theta(y_t \mid \tilde{y}_{<t}, C)$ until $\tilde{y}_t = \langle eos \rangle$: $\tilde{y}_t \sim p_\theta(y_t \mid \tilde{y}_{<t}, C)$.*

To avoid the high variance, two approximate stochastic decoding algorithms have recently been proposed and tested with recurrent language models. Top-k sampling considers only a subset of the k most probable tokens from the vocabulary at a time, while nucleus sampling considers only the minimal subset of most probable tokens whose total probability is higher than a predefined threshold.

Definition 2.6 (Top-k sampling (Fan et al., 2018))**.** *Top-k sampling $\mathcal{F}_{top\text{-}k}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from:*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \text{if } v \in V_k, \\ 0, & \text{otherwise.} \end{cases}$$

where $V_k = \arg \underset{v'}{\text{top-}k}\, p_\theta(v' \mid y_{<t}, C)$.

Definition 2.7 (Nucleus sampling (Holtzman et al., 2019))**.** *Nucleus sampling $\mathcal{F}_{nuc\text{-}\mu}$ generates a sequence from a recurrent language model p_θ given context C by recursively sampling from the following proposal distribution. Let $v_1, \ldots, v_{|V|}$ denote tokens in V such that $p_\theta(v_i \mid y_{<t}, C) \geq p_\theta(v_j \mid y_{<t}, C)$ for all $i < j$, and define*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \text{if } v \in V_\mu, \\ 0, & \text{otherwise,} \end{cases}$$

where $V_\mu = \{v_1, \cdots, v_{k_\mu}\}$ with

$$k_\mu = \min \left\{ k \;\middle|\; \sum_{i=1}^{k} p_\theta(v_i \mid y_{<t}, C) > \mu \right\}.$$

Deterministic decoding. The other family consists of deterministic decoding algorithms, where a token is selected deterministically according to a rule at each decoding step. The most naive algorithm, called greedy decoding, simply takes the most probable token at each step.

Definition 2.8 (Greedy decoding)**.** *Greedy decoding $\mathcal{F}_{greedy}$ generates a sequence from a recurrent language model p_θ given context C by recursively selecting the most likely token from $p_\theta(y_t|\tilde{y}_{<t}, C)$ until $\tilde{y}_t = \langle eos \rangle$:*

$$\tilde{y}_t = \arg \max_{v \in V} \log p_\theta(y_t = v \mid \tilde{y}_{<t}, C).$$

In contrast to greedy decoding, *beam search* with width k, $\mathcal{F}_{\text{beam-k}}$, operates on the level of partial sequences or prefixes. Starting from a set of empty prefixes, at each iteration a new prefix set is formed by expanding each prefix with each possible token, then choosing the k highest scoring expanded prefixes; refer to Appendix A for a formal definition.

Incompleteness. Other than ancestral sampling, the decoding algorithms above are *incomplete* in that they only consider a strict subset of the full vocabulary V at each time step, aside from the trivial case of $k = |V|$.[1]

Definition 2.9 (Incomplete Decoding)**.** *A decoding algorithm $\mathcal{F}$ is incomplete when for each context C and prefix $y_{<t}$, there is a strict subset $V'_t \subsetneq V$ such that*

$$\sum_{v \in V'_t} q_\mathcal{F}(y_t = v \mid y_{<t}, C) = 1.$$

3 Consistency of a Decoding Algorithm

Definition of consistency. A recurrent language model p_θ may assign a positive probability to an infinitely long sequence, in which case we call the model inconsistent. This notion of consistency was raised and analyzed earlier, for instance by Booth and Thompson (1973) and Chen et al. (2017), in terms of whether the distribution induced by p_θ is concentrated on finite sequences. We extend their definition to account for the context C.

Definition 3.1 (Consistency of a recurrent language model)**.** *A recurrent language model is consistent under a context distribution $p(C)$ if $p_\theta(|Y| = \infty) = 0$. Otherwise, the recurrent language model is said to be inconsistent.*

[1] Nucleus sampling is incomplete when for every context C and prefix $y_{<t}$, $\min_{v \in V} p_\theta(v|y_{<t}, C) < 1 - \mu$.

Any sequence decoded from a consistent model for a given context is guaranteed to terminate.

Lemma 3.1. *If a recurrent LM p_θ is consistent, $p_\theta(|Y| = \infty \mid C) = 0$ for any probable context C.*[2]

Next, we establish a practical condition under which a recurrent language model is consistent.

Lemma 3.2. *A recurrent LM p_θ is consistent if $\|h_t\|_p$ is uniformly bounded for some $p \geq 1$.*

Proof sketch. If $\|h_t\|_p$ is bounded, then each $u_v^\top h_t$ is bounded, hence $p_\theta(\langle \text{eos} \rangle \mid y_{<t}, C) > \xi > 0$ for a constant ξ. Thus $p_\theta(|Y| = \infty) \leq \lim_{t\to\infty}(1 - \xi)^t = 0$, meaning that p_θ is consistent. □

Although this condition is practical because layer normalization or bounded activation functions (Elman, 1990; Cho et al., 2014; Vaswani et al., 2017) result in bounded h_t, we show that even if a recurrent language model is consistent, a decoding algorithm may produce an infinite-length sequence. We formalize this discrepancy using the consistency of a decoding algorithm.

Definition 3.2 (Consistency of a decoding algorithm)**.** *A decoding algorithm $\mathcal{F}$ is consistent with respect to a consistent recurrent language model p_θ under a context distribution $p(C)$ if the decoding algorithm $\mathcal{F}$ preserves the consistency of the model p_θ, that is, $q_\mathcal{F}(|Y| = \infty) = 0$.*

When a consistent recurrent language model p_θ and a decoding algorithm $\mathcal{F}$ induce a consistent distribution $q_\mathcal{F}$, we say that p_θ paired with $\mathcal{F}$ is consistent. For instance, any consistent recurrent language model paired with ancestral sampling is consistent, because the induced distribution $q_{\mathcal{F}_\text{anc}}$ is the same as the distribution of the original model. We also have an analogue of Lemma 3.1.

Lemma 3.3. *A consistent decoding algorithm with respect to a consistent recurrent LM decodes only probable sequences. That is, if $q_\mathcal{F}(Y \mid C) > 0$, then $p_\theta(Y \mid C) > 0$ for any probable context C.*

Inconsistency of incomplete decoding. Any incomplete decoding algorithm (Definition 2.9) can be inconsistent regardless of the context distribution, because there is a recurrent LM that places $\langle \text{eos} \rangle$ outside of V_t' at every step of decoding. To show this, we construct a consistent recurrent language model whose distribution induced by an incomplete decoding algorithm is inconsistent.

[2]Proofs of Lemmas 3.1-3.3 are in Appendix B.

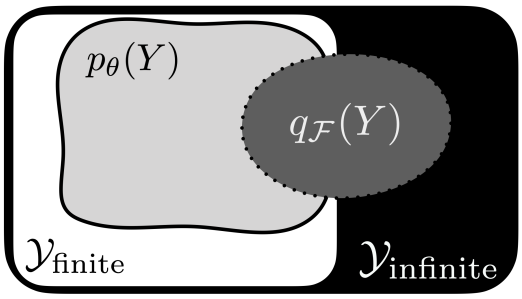

Figure 1: A depiction of the model's sequence distribution (light grey, solid border) and the decoder's induced sequence distribution (dark grey, dotted border). The white and black rectangles depict the set of all finite and infinite sequences, respectively. We prove that under practical conditions, any incomplete decoding algorithm may be inconsistent with respect to a consistent model, as depicted.

Theorem 3.4 (Inconsistency of an incomplete decoding algorithm)**.** *There exists a consistent recurrent LM p_θ from which an incomplete decoding algorithm $\mathcal{F}$, that considers only up to $(|V| - 1)$-most likely tokens according to $p_\theta(y_t \mid y_{<t}, C)$ at each step t, finds an infinite-length sequence $\tilde{Y}$ with probability 1, i.e., $q_\mathcal{F}(|Y| = \infty) = 1$.*

Proof. We prove this theorem by constructing a $\tanh$ recurrent network. We define the recurrent function f_θ as

$$\begin{aligned} h_t &= f_\theta(y_t, h_{t-1}) \\ &= \tanh\left(\left[\begin{array}{c|c} W_h & \mathbf{0} \\ \hline \mathbf{0} & I \end{array}\right] h_{t-1} + \left[\begin{array}{c} \mathbf{0} \\ \hline e(y_t) \end{array}\right]\right), \end{aligned}$$

where $e(y_t) \in \mathbb{R}^{|V|}$ is a one-hot representation of y_t, $W_h \in \mathbb{R}^{d\times d}$ where every entry is positive, and I is an identity matrix of size $|V| \times |V|$. $h_0 = g_\theta(C)$ is constructed to consist of positive values only. Because each element of $|h_t|$ is bounded by 1, the constructed recurrent language model p_θ is consistent by Lemma 3.2.

We set u_v (see Definition 2.3) to

$$u_v = \left[\begin{array}{c} \bar{u}_v \\ \hline e(v) \end{array}\right], \quad u_{\langle \text{eos} \rangle} = \left[\begin{array}{c} \bar{u}_{\langle \text{eos} \rangle} \\ \hline e(\langle \text{eos} \rangle) \end{array}\right],$$

where $v \neq \langle \text{eos} \rangle$, all elements of $\bar{u}_v$ are positive, all elements of $\bar{u}_{\langle \text{eos} \rangle}$ are negative, and $e(v)$ is a one-hot representation of v. c_v is set to zero.

This defines a valid recurrent language model (Definition 2.3), since the conditional distribution at each time t is influenced by all the previous tokens. More specifically, the logit of a token v

depends on $\sum_{t'=1}^{t} \mathbb{1}(y_{t'} = v)$, where $\mathbb{1}$ is an indicator function.

This recurrent language model always outputs positive logits for non-$\langle$eos$\rangle$ tokens, and outputs negative logits for the $\langle$eos$\rangle$ token. This implies $p(\langle\text{eos}\rangle \mid y_{<t}, C) < p(v \mid y_{<t}, C)$ for all $v \in V \setminus \{\langle\text{eos}\rangle\}$. This means that $\langle$eos$\rangle$ is always ranked last at each time step, so an incomplete decoding algorithm that considers at most $(|V| - 1)$ most probable tokens at each time step from $p_\theta(y_t \mid y_{<t}, C)$ cannot decode $\langle$eos$\rangle$ and thus always decodes an infinitely long sequence $\hat{Y}$, i.e., $q_{\mathcal{F}}(|Y| = \infty \mid C) = 1$ for any context C. It yields $q_{\mathcal{F}}(|Y| = \infty) = 1$, while $p_\theta(|Y| = \infty) = 0$ due to consistency of the model p_θ. □

Greedy decoding, beam search, top-k sampling, and nucleus sampling are all inconsistent according to this theorem.

4 Fixing the inconsistency

In this section, we consider two ways to prevent inconsistency arising from incomplete decoding algorithms. First, we introduce consistent versions of top-k and nucleus sampling. Second, we introduce the *self-terminating recurrent language model*, which is consistent when paired with any of the decoding algorithms considered in this paper.

4.1 Consistent Sampling Algorithms

The proof of Theorem 3.4 suggests that the inconsistency of incomplete decoding algorithms arises from the fact that $\langle$eos$\rangle$ may be excluded indefinitely from the set of top-ranked tokens. We propose a simple modification to top-k and nucleus sampling that forces $\langle$eos$\rangle$ to be included at each step of decoding. First, we give a condition for when a particular model p_θ paired with a decoding algorithm $\mathcal{F}$ is consistent.

Theorem 4.1. *Suppose a recurrent LM p_θ has uniformly bounded $\|h_t\|_p$ for some $p \geq 1$. If a decoding algorithm $\mathcal{F}$ satisfies $q_{\mathcal{F}}(\langle eos\rangle \mid y_{<t}, C) \geq p_\theta(\langle eos\rangle \mid y_{<t}, C)$ for every prefix $y_{<t}$ and context C, then the decoding algorithm $\mathcal{F}$ is consistent with respect to the model p_θ.*[3]

We define consistent variants of top-k and nucleus sampling which satisfy this condition.

Definition 4.1 (Consistent top-k sampling)**.** *Consistent top-k sampling is top-k sampling with the following modified proposal distribution:*

$$q(v) \propto \begin{cases} p_\theta(v|y_{<t}, C), & \text{if } v \in V', \\ 0, & \text{otherwise}, \end{cases}$$

where $V' = \{\langle eos\rangle\} \cup \arg\operatorname*{top\text{-}k}_{v'} p_\theta(v' \mid y_{<t}, C)$.

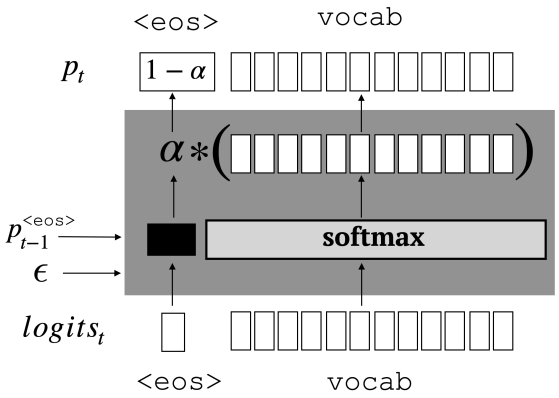

Figure 2: The self-terminating recurrent LM uses the layer shown in grey instead of the standard softmax layer. The layer takes logits ($u_\cdot^\top h_t$), the previous step's $\langle$eos$\rangle$ probability ($p_{t-1}^{\langle\text{eos}\rangle}$), and a hyper-parameter $\epsilon \in (0, 1)$. The layer computes α using Definition 4.3, which determines the $\langle$eos$\rangle$ probability ($p_t^{\langle\text{eos}\rangle} \in (\epsilon, 1)$), and guarantees that $p_t^{\langle\text{eos}\rangle} > p_{t-1}^{\langle\text{eos}\rangle}$. The remaining probability mass is allocated to the non-$\langle$eos$\rangle$ tokens.

Definition 4.2 (Consistent nucleus sampling)**.** *Consistent nucleus sampling is nucleus sampling with the following modified proposal distribution:*

$$q(v) \propto \begin{cases} p_\theta(v \mid y_{<t}, C), & \text{if } v \in V_\mu \cup \{\langle eos\rangle\}, \\ 0, & \text{otherwise}. \end{cases}$$

The induced probability of $\langle$eos$\rangle$ under these two algorithms is always equal to or larger than the model's probability. By Theorem 4.1, these algorithms are consistent with respect to any consistent recurrent language model.

4.2 Self-Terminating Recurrent LM

Although these consistent sampling algorithms can be used with any recurrent language model, their stochastic nature may not be suitable for finding a single, highly probable sequence. To avoid this limitation, we propose the *self-terminating recurrent language model* (STRLM).

Definition 4.3 (Self-terminating recurrent language model)**.** *A self-terminating recurrent language model computes the following conditional*

[3] See Appendix C for the proof.

probability at each time step:

$$p_\theta(v \mid y_{<t}, C) = \begin{cases} 1 - \alpha(h_t), & v = \langle eos \rangle \\ \frac{\alpha(h_t)\exp(u_v^\top h_t + c_v)}{\sum_{v' \in V'} \exp(u_{v'}^\top h_t + c_{v'})}, \end{cases}$$

$$\alpha(h_0) = \sigma(u_{\langle eos \rangle}^\top h_0),$$

$$\alpha(h_t) = \sigma(u_{\langle eos \rangle}^\top h_t)\left[1 - p_\theta(\langle eos \rangle \mid y_{<t-1}, C)\right],$$

with $\sigma : \mathbb{R} \to [0, 1-\varepsilon]$ *and* $\varepsilon \in (0, 1)$. h_t *is computed as in the original recurrent LM.*

The underlying idea is that the probability of ⟨eos⟩ increases monotonically, since

$$p_t^{\langle \text{eos} \rangle} = 1 - \prod_{t'=0}^{t} \sigma(u_{\langle \text{eos} \rangle}^\top h_{t'}).$$

Consequently, the STRLM is consistent when paired with greedy decoding or beam search; see Appendix C for formal statements and proofs.

5 Empirical Validation

The theoretical results rely on the existence of a model that results in inconsistency; it remains to be shown that inconsistency with respect to incomplete decoding occurs with recurrent language models encountered in practice. Moreover, while the proposed methods carry theoretical guarantees in terms of consistency, we must check whether they retain language modeling quality. To do so, we perform experiments using a sequence completion task. In each experiment, we use the beginning of a sequence as context, then decode continuations from a trained recurrent LM and measure the proportion of non-terminated sequences in order to approximately measure inconsistency. The first experiment (§5.1) shows that inconsistency occurs in practice, and the second experiment (§5.2) shows the effectiveness of the proposed approaches. Our third experiment (§5.3) shows that inconsistency also occurs frequently in GPT-2, a large-scale transformer language model.[4]

Sequence completion. We evaluate recurrent language models on a sequence completion task, which has previously been used to evaluate the effectiveness of sequence models, e.g., Sutskever et al. (2011); Graves (2013); Radford et al. (2019); Holtzman et al. (2019); Welleck et al. (2019). Sequence completion is a general setting for studying the behavior of language models, encompassing machine translation (Bahdanau et al., 2015), story generation (Fan et al., 2018), and dialogue modeling (Vinyals et al., 2015). The task consists of decoding a continuation $\hat{Y} \sim \mathcal{F}(p_\theta, C)$ given a length-k prefix $C = (c_1, \ldots, c_k)$, resulting in a completion $(c_1, \ldots, c_k, \hat{y}_1 \ldots, \hat{y}_T)$.

Dataset. Our first two experiments use Wikitext2 (Merity et al., 2016), which consists of paragraphs from English Wikipedia, since it has frequently been used to evaluate language models (Grave et al., 2017; Melis et al., 2018; Merity et al., 2018). We consider both word and BPE[5] tokenization. We split each paragraph into sentences using Spacy[6]. We split each sequence, using the first k tokens as a context and the remaining tokens as a continuation. To ensure that each sequence contains a prefix, we prepend padding tokens to make it length k. Special ⟨bos⟩ and ⟨eos⟩ tokens are inserted at the beginning and end of each sequence. We use $k = 10$. Table 7 contains dataset statistics.

Context distribution. We define empirical context distributions with prefixes from the train, valid, and test sets: $p(C; \mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{n=1}^{|\mathcal{D}|} \mathbb{1}(C = C^{(n)})$, where $\mathcal{D} = \{(C^{(n)}, Y^{(n)})\}_{n=1}^{N}$ is a dataset split.

Evaluation metrics. We use finite sequences to approximately measure the consistency of a model paired with a decoding algorithm, since decoding an infinite-length sequence is impossible. We use the proportion of decoded continuations that are longer than a predefined limit,

$$r_L = \frac{1}{|\mathcal{D}|}\sum_{n=1}^{|\mathcal{D}|} \mathbb{1}(|\hat{Y}^{(n)}| \geq L),$$

where $\hat{Y}^{(n)} \sim \mathcal{F}(p_\theta, C^{(n)})$ for each context $C^{(n)}$ in $\mathcal{D}$. We call r_L the *non-termination ratio* of the decoding algorithm $\mathcal{F}$ for an underlying model and context distribution. A value of r_L greater than zero means that some sequences did not terminate within L steps. When L is infinity, this implies that the model paired with the decoding algorithm is inconsistent. In practice, we use a finite L that is substantially larger than the maximum training sequence length, and we interpret a non-zero r_L as evidence that the model paired with the decoding algorithm is inconsistent. We use $L = 1500$, more than 10 times the max training sequence length.

[4] Code available at https://github.com/uralik/consistency-lm.

[5] github.com/huggingface/tokenizers

[6] https://spacy.io/

In each experiment, we report the mean and standard deviation of metrics across 10 independent initializations. Unless specified otherwise, we report metrics using the test context distribution, since the train, valid, and randomly generated context distributions had similar results.

Training. We train recurrent language models for sequence completion with maximum likelihood, using the loss $\mathcal{L}(p_\theta, Y) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, c_1, \ldots, c_k)$, where $Y = (c_1, \ldots, c_k, y_1, \ldots, y_T)$. This amounts to running the full training sequence through a recurrent model and zeroing the loss for the first k tokens, so that the first k steps correspond to learning a g_θ that encodes the context.

Models. We consider recurrent neural networks with hyperbolic tangent activations (tanh-RNN; Elman, 1990) and LSTM units (LSTM-RNN; Hochreiter and Schmidhuber, 1997). We perform an initial hyper-parameter sweep and select the best set of hyper-parameters for each of tanh-RNN and LSTM-RNN based on the validation perplexities.[7] With this best set of hyperparameters, we train each of these models with 10 different initializations. The choice of tanh and LSTM RNNs implies that all of the recurrent language models that we train are consistent according to Lemma 3.2. Our LSTM models achieve similar test perplexity (91.86 ± 0.4, word tokenization) to those reported in previous work (Merity et al., 2018); see Appendix D.

Additionally, we train self-terminating tanh-RNN and LSTM-RNN variants (Definition 4.3) at various values of ε, which controls a lower bound on the termination probability at each step. We use $\sigma(x) = (1 - \varepsilon) \cdot \text{sigmoid}(x)$. We use the hyper-parameters selected in the preceding grid search. Below, we consider BPE tokenization; similar conclusions held for word tokenization.[8]

5.1 Inconsistency of Recurrent LMs

In this experiment, we demonstrate evidence of inconsistency with incomplete decoding methods. Table 1 shows non-termination ratios for the recurrent language models using the decoding algorithms considered in this work. Decoding with ancestral sampling always resulted in sequences that terminated within L steps, since the induced distribution is the same as that of the consistent model.

[7] Refer to Appendix D for the hyper-parameter ranges.

[8] Refer to Appendix for results with word tokenization.

	tanh-RNN	LSTM-RNN
ancestral	0.00 ± 0.0	0.00 ± 0.0
greedy	12.35 ± 5.18	1.53 ± 1.41
beam-2	1.38 ± 0.95	0.07 ± 0.06
beam-4	0.25 ± 0.19	0.00 ± 0.01
topk-2	0.01 ± 0.01	0.01 ± 0.01
topk-4	0.00 ± 0.0	0.00 ± 0.01
nucleus-0.2	0.06 ± 0.02	0.13 ± 0.15
nucleus-0.4	0.04 ± 0.02	0.02 ± 0.01
consistent topk-2	0.00 ± 0.0	0.00 ± 0.01
consistent topk-4	0.00 ± 0.0	0.00 ± 0.0
consistent nucleus-0.2	0.04 ± 0.02	0.01 ± 0.01
consistent nucleus-0.4	0.02 ± 0.02	0.01 ± 0.01

Table 1: Non-termination ratio (r_L (%)) of decoded sequences using ancestral sampling, incomplete, and consistent decoding methods.

On the other hand, the non-zero non-termination ratios for the incomplete decoding algorithms suggest inconsistency with respect to each algorithm, providing evidence for Theorem 3.4.

Using greedy decoding, roughly 12% of all contexts resulted in a non-terminating continuation with the tanh-RNN, and roughly 1% with the LSTM-RNN. Nucleus sampling also produced non-terminating sequences with the tanh-RNN (0.06%, nuc-0.2) and LSTM-RNN (0.13%, nuc-0.2). Top-k sampling yielded a small number of non-terminating samples. In general, non-termination approaches zero as k and μ increase, since $\langle\text{eos}\rangle$ has a lower chance of being excluded.

Beam search produced non-terminating sequences with both the tanh-RNN and LSTM-RNN models. This means that $\langle\text{eos}\rangle$ was outside of the top tokens (determined by the beam width) considered at each step, since in our experiments we terminated the beam search when a single beam prefix contained $\langle\text{eos}\rangle$. Larger beam widths reduce non-termination, similar to increasing k or μ.

5.2 Consistency of the Proposed Methods

Consistent sampling. Table 1 shows that consistent nucleus and top-k sampling (§4.1) resulted in only terminating sequences, except for a few cases that we attribute to the finite limit L used to measure the non-termination ratio. Consistent nucleus paired with tanh-RNN did not reduce r_L as much as when it was paired with LSTM-RNN. Example continuations are shown in Table 2. On prefixes that led to non-termination with the baseline method, the quality tends to improve with the consistent variant since the continuation now termi-

Prefix	*One Direction delivered a performance of " Kiss You*
nucleus	" , and the album 's second album , " The X @-@ Files " , " The A. " , " The Preder " , " We 've Have You " , " I 've You Wanna Stay " , " The Dream " , " The Bide " , " My Achievement ", " The B. B. " , " A Life " . . .
c-nucleus	" , and " My Boo " was released on September 29 , 2010 . ⟨eos⟩
Prefix	*Boulter starred in two films in 2008 ,*
nucleus	and the band 's music , and " The Rise of Monkey " , " The One With the Way " , " The " Always " , " " Always Your " , " The Wift " , " The Baste " , " The Special With " , " The Way " , " The Special With You " . . .
c-nucleus	and the latter was released in the United States . ⟨eos⟩
Prefix	*This period of unhappiness was the making of*
Baseline	the " most important " of the " mad " , and the " " most important " of the " " " , " the most important " , and the " devil " , " The " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " , " The One " . . .
STRLM	the first commandment of the poem . ⟨eos⟩
Prefix	*Du Fu 's mother died shortly after he was*
Baseline	a member of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Order of the Republic of the Republic of the Republic of the Republic of the Republic of . . .
STRLM	a member of the Order of the British Empire . ⟨eos⟩

Table 2: Continuations with consistent nucleus sampling ($\mu = 0.2$) and self-terminating LSTM ($\epsilon = 10^{-3}$).

nates. Note that since the model's non-⟨eos⟩ token probabilities at each step are only modified by a multiplicative constant, the sampling process can still enter a repetitive cycle (e.g., when the constant is close to 1), though it is guaranteed to terminate.

Self-terminating RLM. As seen in Table 3, the self-terminating recurrent language models are consistent with respect to greedy decoding, at the expense of perplexity compared to the vanilla model. The value of ε from Definition 4.3, which controls a lower-bound on termination probability at each step, influences both r_L and perplexity. When ε is too large ($\varepsilon = 10^{-2}$), perplexity degrades. When ε is too small ($\varepsilon = 10^{-4}$), the lower-bound grows slowly, so ⟨eos⟩ is not guaranteed to be top-ranked within L steps, resulting in a positive r_L. An ε of 10^{-3} balanced consistency and language modeling quality, with a zero non-termination ratio and perplexity within 8 points of the baseline.

As shown in Figure 3, the self-terminating model matches the data length distribution better than the baseline. Example decoded sequences are shown in Table 2. For prefixes that led to non-termination with the baseline, the self-terminating models yields finite sequences with reasonable quality. The examples suggest that some cases of degenerate repetition (Holtzman et al., 2019; Welleck et al., 2019) are attributed to inconsistency.

5.3 Inconsistency of GPT-2

We perform a final experiment with GPT-2 117M, a transformer language model pre-trained with maximum likelihood on WebText, a collection of

	ST	ϵ	r_L (%)	**perplexity**
tanh-RNN	✓	10^{-2}	00.00 ± 0.0	229.09 ± 9.2
	✓	10^{-3}	00.00 ± 0.0	191.63 ± 1.4
	✓	10^{-4}	00.02 ± 0.02	188.36 ± 2.2
	✗	–	12.35 ± 5.2	186.44 ± 1.4
LSTM	✓	10^{-2}	0.00 ± 0.0	219.71 ± 9.2
	✓	10^{-3}	0.00 ± 0.0	186.04 ± 1.6
	✓	10^{-4}	0.18 ± 0.35	183.57 ± 2.3
	✗	–	1.48 ± 1.43	178.19 ± 1.3

Table 3: Non-termination ratio (r_L (%)) of greedy-decoded sequences and test perplexity for STRLMs.

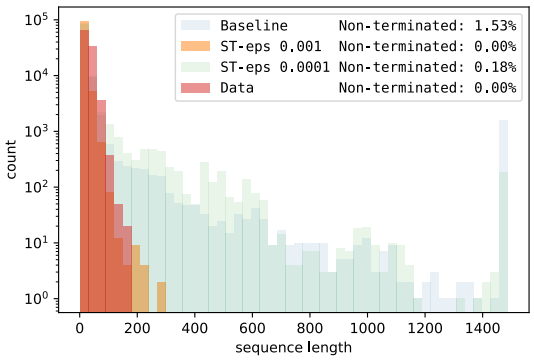

Figure 3: Lengths of generated sequences using greedy decoding from vanilla and self-terminating LSTMs.

scraped web pages (see Radford et al. (2019)). GPT-2 has been observed to produce repetitive text with greedy and beam search (Holtzman et al., 2019).

Experimental setup. We use the Wikitext-103 dataset (Merity et al., 2016), a large-scale collec-

tion of Wikipedia articles with over 100 million words and 260 thousand unique tokens. We split the dataset into sequences according to the dataset's newline boundaries, then split each sequence into a context C and continuation Y, resulting in a dataset of (C, Y) pairs. Each continuation ends in a special $\langle \text{eos} \rangle$ token. We use a context size of $k = 10$ tokens, and discard sequences that are length k or shorter. The resulting dataset contains 874,556 training, 1,896 validation, and 2,162 test pairs.

We fine-tune the pre-trained GPT-2 model using maximum likelihood for 400k steps, and select the model state with the lowest validation perplexity (evaluated every 5k steps). Each training batch contains a maximum of 1024 total tokens. We use the implementation and default hyper-parameters from the `transformers` library (Wolf et al., 2019). We fine-tune the self-terminating GPT-2 models in a similar manner, starting from the pre-trained GPT-2 model and using the same hyper-parameters.

Each model is evaluated using greedy decoding with a maximum sequence length of 500, which was selected so that each decoded validation batch could fit in GPU memory. We define the non-termination ratio (r_L) using $L = 500$; this limit is more strict than the limit used in the preceding experiments (1500), yet still allows us to see large differences in generation behavior between the model and the ground truth (e.g. see Figure 4).

Results. Table 4 shows the non-termination ratio and perplexity of the baseline and self-terminating GPT-2 models. The self-terminating variant prevents non-termination, at the cost of perplexity. The model here uses $\epsilon = 2.5 \times 10^{-3}$, which we selected after observing that at higher values of ϵ, e.g. 1.0×10^{-3}, the self-terminating model generated sequences longer than the limit used to determine termination (500). Figure 4 shows the length distributions of the baseline GPT-2 continuations and those of the self-terminating GPT-2. The GPT-2 117M model generates many sequences at or near the maximum sequence length (500), unlike the ground-truth data. Introducing self-termination shifts the mass towards shorter sequences, whose lengths are also present in the ground-truth data.

6 Future Directions

The methods we proposed in this paper resolve inconsistency by changing the decoding algorithm or model parameterization. Another approach is to address inconsistency in the *learning* phase. One interesting direction is to investigate whether the lack of decoding in maximum likelihood learning is a cause of inconsistency. Maximum likelihood learning fits the model p_θ using the data distribution, whereas a decoded sequence from the trained model follows the distribution $q_{\mathcal{F}}$ induced by a decoding algorithm. *Sequence-level* learning, however, uses a decoding algorithm during training (e.g., Ranzato et al. (2016)), which we hypothesize can result in a good sequence generator that is consistent with respect to incomplete decoding.

	r_L (%)	**perplexity**
GPT2-117M	37.91	20.92
GPT2-117M ST	00.00	27.25

Table 4: Non-termination ratio (r_L (%)) of greedy-decoded sequences and perplexity for GPT2-117M and the self-terminating variant (ST) on Wikitext-103.

7 Conclusion

We extended the notion of consistency of a recurrent language model put forward by Chen et al. (2017) to incorporate a decoding algorithm, and used it to analyze the discrepancy between a model and the distribution induced by a decoding algorithm. We proved that incomplete decoding is inconsistent, and proposed two methods to prevent this: consistent decoding and the self-terminating recurrent language model. Using a sequence completion task, we confirmed that empirical inconsistency occurs in practice, and that each method prevents inconsistency while maintaining the quality of generated sequences. We suspect the absence of decoding in maximum likelihood estimation as a cause behind this inconsistency, and suggest investigating sequence-level learning as an alternative.

Acknowledgements

We thank Chris Dyer, Noah Smith and Kevin Knight for valuable discussions. This work was supported by NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science; Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI); and Samsung Research (Improving Deep Learning using Latent Structure). KC thanks eBay and NVIDIA for their support.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, volume 4, pages 2442–2452.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

T. L. Booth and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450.

John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer.

Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May, and Kevin Knight. 2017. Recurrent neural networks as weighted language recognizers. *arXiv preprint arXiv:1711.05408*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

John Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *ArXiv*, abs/1609.07843.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

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.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Pavel Sountsov and Sunita Sarawagi. 2016. Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, Texas. Association for Computational Linguistics.

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 3354–3360, Hong Kong, China. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

you need. In *Advances in Neural Information Processing Systems*.

Oriol Vinyals, Google Quoc, and V Le. 2015. A Neural Conversational Model. In *ICML Deep Learning Workshop*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

A Additional Definitions

In contrast to greedy decoding, beam search with width k, $\mathcal{F}_{\text{beam-k}}$, operates on the level of partial sequences or prefixes.

Definition A.1 (Prefix)**.** *A prefix ρ_t is an ordered collection of items from V. The score of a prefix is*

$$s(\rho_t) = \sum_{\tau=1}^{t} \log p_\theta(y_\tau = \rho_t[\tau] \,|\, \rho_t[< \tau], C),$$

where $\rho_t[\tau]$ is a token at time τ from ρ_t.

Starting from a set of empty prefixes, at each iteration a new prefix set is formed by expanding each prefix, then choosing the k highest scoring expanded prefixes.

Definition A.2 (Beam search)**.** *Beam search with width k, $\mathcal{F}_{beam-k}$, generates a sequence from a recurrent language model p_θ by maintaining a size-k prefix set P_t^{top}. Starting with $P_0^{top} = \varnothing$, at each iteration $t \in \{1, 2, \ldots\}$ beam search forms a new prefix set P_t^{top} by expanding the current set, $\mathrm{P}_t = \bigcup_{\rho \in \mathrm{P}_{t-1}^{top}} \{\rho \circ v \,|\, v \in V\}$ (where $\rho \circ v$ is concatenation), then choosing the k highest scoring elements: $\mathrm{P}_t^{top} = \arg\operatorname*{top\text{-}k}_{\rho \in \mathrm{P}_t} s(\rho)$. Any $\rho \in \mathrm{P}_t^{top}$ ending with $\langle eos \rangle$ is restricted from being expanded further, and is added to a set S. Beam search ends when S contains k sequences, and returns the highest scoring sequence in S.*

B Proof of Lemmas in Section 3

Lemma 3.1. *If a recurrent language model p_θ is consistent, $p_\theta(|Y| = \infty \,|\, C) = 0$ for any probable context C.*

Proof. Suppose there exists a probable context $\tilde{C}$ such that $p_\theta(|Y| = \infty \,|\, \tilde{C}) > 0$. Then

$$\begin{aligned} p_\theta(|Y| = \infty) &= \mathbb{E}\left[p_\theta(|Y| = \infty \,|\, C)\right] \\ &\geq p(\tilde{C}) p_\theta(|Y| = \infty \,|\, \tilde{C}) > 0, \end{aligned}$$

which contradicts the consistency of the model p_θ. □

Lemma 3.2. *A recurrent language model p_θ is consistent if $\|h_t\|_p$ is uniformly bounded for some $p \geq 1$.*

Proof. Let $B > 0$ be an upper bound such that $\|h_t\|_p < B$ for all t. Let q be the conjugate of p satisfying $1/p + 1/q = 1$. Then we have from Hölder's inequality, for all $v \in V$ and t,

$$u_v^\top h_t \leq \|u_v^\top h_t\|_1 \leq \|h_t\|_p \|u_v\|_q < B u^+,$$

where $u^+ = \max_{v \in V} \|u_v\|_q$. Note that

$$\begin{aligned} \log \sum_{v \in V} e^{u_v^\top h_t + c_v} &\leq \log \left(\max_{v \in V} e^{u_v^\top h_t + c_v} \times |V| \right) \\ &\leq \max_{v \in V} \{u_v^\top h_t + c_v\} + \log |V| \\ &< B u^+ + c^+ + \log |V|, \end{aligned}$$

where $c^+ = \max_{v \in V} c_v$. For a given $y_{<t}$ and context C,

$$\begin{aligned} &\log p_\theta(\langle \text{eos} \rangle \,|\, y_{<t}, C) \\ =&(u_{\langle \text{eos} \rangle}^\top h_t + c_{\langle \text{eos} \rangle}) - \log \sum_{v \in V} e^{u_v^\top h_t + c_v} \\ >&(-B u^+ + c_{\langle \text{eos} \rangle}) - (B u^+ + c^+ + \log |V|) > -\infty, \end{aligned}$$

and it follows that $p_\theta(\langle \text{eos} \rangle \,|\, y_{<t}, C) > \xi > 0$ for some strictly positive constant ξ. Then

$$\begin{aligned} p_\theta(|Y| = \infty) &= \lim_{t \to \infty} p_\theta(|Y| > t) \\ &= \lim_{t \to \infty} \mathbb{E}\left[p_\theta(|Y| > t \,|\, C)\right] \\ &= \mathbb{E}\left[\lim_{t \to \infty} p_\theta(|Y| > t \,|\, C)\right] \\ &\leq \mathbb{E}\left[\lim_{t \to \infty} (1 - \xi)^t\right] = 0, \end{aligned}$$

and hence p_θ is consistent. □

Lemma 3.3. *A consistent decoding algorithm with respect to a consistent recurrent language model decodes only probable sequences. That is, if $q_{\mathcal{F}}(Y \,|\, C) > 0$, then $p_\theta(Y \,|\, C) > 0$ for any probable context C.*

Proof. Suppose there exists a decoded sequence $\tilde{Y}$ by $\mathcal{F}$ and probable context $\tilde{C}$ such that $q_{\mathcal{F}}(\tilde{Y} \,|\, \tilde{C}) > 0$ but $p_\theta(\tilde{Y} \,|\, \tilde{C}) = 0$. By Remark 2.1, the sequence $\tilde{Y}$ is of infinite length and thus $q_{\mathcal{F}}(|Y| = \infty \,|\, \tilde{C}) \geq q_{\mathcal{F}}(\tilde{Y} \,|\, \tilde{C}) > 0$, which contradicts the consistency of $q_{\mathcal{F}}$ by Lemma 3.1. □

C Proofs for Section 4

Theorem 4.1. *Suppose a recurrent LM p_θ has uniformly bounded $\|h_t\|_p$ for some $p \geq 1$. If a decoding algorithm $\mathcal{F}$ satisfies $q_{\mathcal{F}}(\langle eos \rangle \,|\, y_{<t}, C) \geq p_\theta(\langle eos \rangle \,|\, y_{<t}, C)$ for every prefix $y_{<t}$ and context C, then the decoding algorithm $\mathcal{F}$ is consistent with respect to the model p_θ.*

Proof. By Lemma 3.2 the model p_θ is consistent and $p_\theta(\langle\text{eos}\rangle \,|\, y_{<t}, C) > \xi$ for some positive value ξ. Thus, $q_\mathcal{F}(\langle\text{eos}\rangle \,|\, y_{<t}, C) \geq p_\theta(\langle\text{eos}\rangle \,|\, y_{<t}, C) > \xi$. For $t \geq 1$,

$$\begin{aligned} &q_\mathcal{F}(|Y| > t \,|\, C) \\ &= q_\mathcal{F}(y_1 \neq \langle\text{eos}\rangle, \cdots, y_t \neq \langle\text{eos}\rangle \,|\, C) \\ &\leq (1-\xi)^t. \end{aligned}$$

Taking the limit $t \to \infty$ and expectation over C, we have

$$\begin{aligned} q_\mathcal{F}(|Y| = \infty) &= \mathbb{E}_C \left[\lim_{t\to\infty} q_\mathcal{F}(|Y| > t \,|\, C) \right] \\ &\leq \lim_{t\to\infty} (1-\xi)^t = 0, \end{aligned}$$

from which the decoding algorithm is consistent. □

Theorem 4.2. *Greedy decoding is consistent with respect to any self-terminating recurrent LM.*

Proof. Let $p_t^{\langle\text{eos}\rangle}$ denote $p_\theta(\langle\text{eos}\rangle \,|\, y_{<t}, C)$ and $a_t^{\langle\text{eos}\rangle}$ denote $u_{\langle\text{eos}\rangle}^\top h_t + c_{\langle\text{eos}\rangle}$. By Definition 4.3 we have

$$\begin{aligned} p_t^{\langle\text{eos}\rangle} &= 1 - \sigma(a_t^{\langle\text{eos}\rangle})(1 - p_{t-1}^{\langle\text{eos}\rangle}) \\ &= 1 - \prod_{t'=0}^{t} \sigma(a_{t'}^{\langle\text{eos}\rangle}) \geq 1 - (1-\epsilon)^{t+1}. \end{aligned}$$

Take $B = -\log 2 / \log(1-\epsilon)$. We then have $p_t^{\langle\text{eos}\rangle} > 1/2$ for all $t > B$, which implies that $\langle\text{eos}\rangle$ is always the most probable token after time step B. Hence, the sequence length is less than B with probability 1. □

Theorem 4.3. *Beam search with width k, $\mathcal{F}_{beam-k}$, is consistent with respect to any STRLM.*

Proof. Let $S(\rho)$ be the size-k set of sequences kept by $\mathcal{F}_{\text{beam}-k}$ that start with a prefix ρ.

Take $B = -\log 2 / \log(1-\epsilon)$ as in the proof of Theorem 4.2. Suppose that there exists at least one prefix $\hat{\rho} \in P_B^{\text{top}}$ which does not end with $\langle\text{eos}\rangle$.

We first want to show that $\hat{\rho}$ induces at most k more steps in beam search with width k, that is, $Y \in S(\hat{\rho})$ implies $|Y| \leq B + k$.

We know from the proof of Theorem 4.2 that an STRLM p_θ satisfies: for any context C and $v \in V \setminus \{\langle\text{eos}\rangle\}$,

$$p_\theta(\langle\text{eos}\rangle \,|\, \hat{\rho}, C) > p_\theta(v \,|\, \hat{\rho}, C).$$

For any subsequence $y = (y_1, \ldots, y_l)$ with $y_1 \neq \langle\text{eos}\rangle$,

$$\begin{aligned} p_\theta(\hat{\rho} \circ y \,|\, \hat{\rho}, C) &= \prod_{i=1}^{l} p_\theta(y_i \,|\, \hat{\rho} \circ y_{<i}, C) \\ &\leq p_\theta(y_1 \,|\, \hat{\rho}, C) \\ &< p_\theta(\langle\text{eos}\rangle \,|\, \hat{\rho}, C). \end{aligned}$$

Thus, $\hat{\rho} \circ \langle\text{eos}\rangle$ is the most probable sequence among sequences starting with the prefix $\hat{\rho}$, and it follows that $\hat{\rho} \circ \langle\text{eos}\rangle \in S(\hat{\rho})$.

Thus, in $S(\hat{\rho})$, there are $(k-1)$ sequences starting with $\hat{\rho} \circ v$ for $v \in V \setminus \{\langle\text{eos}\rangle\}$. By the same argument, at each step at least one sequence ending with $\langle\text{eos}\rangle$ is added to $S(\hat{\rho})$, and therefore at time step $(B+k)$, k sequences ending with $\langle\text{eos}\rangle$ are in $S(\hat{\rho})$.

Note that the result set S by $\mathcal{F}_{\text{beam}-k}$ (Definition 2.11) satisfies

$$S \subseteq \bigcup_{\rho \in P_B^{\text{top}}} S(\rho).$$

Since each $\rho \in P_B^{\text{top}}$ induces sequences of length at most $B + k$, we have

$$p_\theta(|Y| > B + k \,|\, C) = 0.$$

Taking the expectation over C yields the consistency of the model p_θ. □

Parameter	Values
Hidden Size	$\{256, \mathbf{512}, 1024\}$
Dropout	$\{0.1, \textit{0.3}, \mathbf{0.5}\}$
Embedding Weight Tying	$\{\textbf{\textit{True}}, \text{False}\}$

Table 5: Grid search specification. The values selected for the **LSTM-RNN** and tanh-*RNN* models are shown in bold and italics, respectively (word tokenization).

D Additional Results and Experiment Details

Training. Each model is trained on a single Nvidia P40 GPU for up to 100 epochs, stopping when validation perplexity does not decrease for 10 consecutive epochs.

Hyper-parameters. Tables 5,6 show the grid search specifications. All models were 2 layers and were trained with the Adam optimizer.

Model perplexities. Tables 10, 11 shows train and test perplexities for the tanh-RNN and LSTM-RNN models using word and BPE tokenization, respectively.

Additional example continuations. Table 12 shows additional greedy-decoded continuations using a self-terminating LSTM-RNN and the baseline LSTM-RNN with BPE tokenization.

GPT-2 length distributions. Figure 4 shows the length distributions of ground-truth continuations, continuations from GPT-2 117M, and continuations from the self-terminating GPT-2 117M.

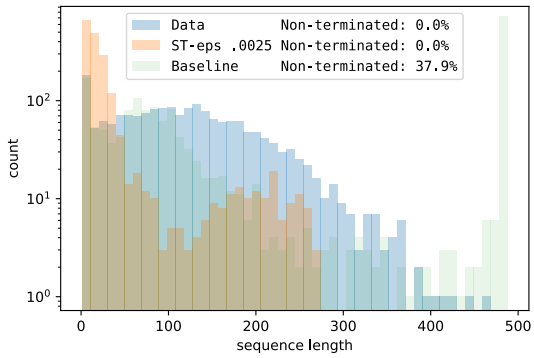

Figure 4: Lengths of ground-truth and greedy-decoded continuations from the baseline GPT-2 117M and self-terminating GPT-2 117M models ($\epsilon = 0.0025$).

Parameter	Values
Hidden Size	$\{\textbf{\textit{256}}, 512, 1024\}$
Dropout	$\{0.1, \mathbf{0.3}, \textit{0.5}\}$
Embedding Weight Tying	$\{\text{True}, \textbf{\textit{False}}\}$

Table 6: Grid search specification. The values selected for the **LSTM-RNN** and tanh-*RNN* models are shown in bold and italics, respectively (BPE tokenization).

Type	# Train	# Valid	# Test	$\lvert V \rvert$	Avg. len
Word	78274	8464	9708	33182	24
BPE	83344	8721	10156	19483	28

Table 7: Wikitext2 statistics.

	tanh-**RNN**	**LSTM-RNN**
ancestral	0.00 ± 0.0	0.00 ± 0.0
greedy	6.07 ± 5.6	1.03 ± 0.3
beam-2	1.21 ± 0.3	0.07 ± 0.1
beam-4	0.29 ± 0.1	0.00 ± 0.0
topk-2	0.84 ± 0.8	0.00 ± 0.0
topk-4	0.02 ± 0.0	0.00 ± 0.0
nucleus-0.2	2.49 ± 0.2	0.76 ± 0.3
nucleus-0.4	0.32 ± 0.1	0.22 ± 0.1

Table 8: Non-termination ratio (r_L (%)) of decoded sequences using ancestral sampling and incomplete decoding methods (word tokenization).

	ST	ϵ	r_L (%)	**perplexity**
tanh-RNN	✓	10^{-2}	0.00 ± 0.0	150.07 ± 2.7
	✓	10^{-3}	0.00 ± 0.0	138.01 ± 0.6
	✓	10^{-4}	1.04 ± 0.6	138.67 ± 1.8
	✗	–	6.07 ± 5.6	136.57 ± 1.8
LSTM	✓	10^{-2}	0.00 ± 0.0	101.24 ± 0.3
	✓	10^{-3}	0.00 ± 0.0	94.33 ± 0.6
	✓	10^{-4}	0.94 ± 0.5	94.15 ± 0.8
	✗	–	1.03 ± 0.3	91.86 ± 0.4

Table 9: Non-termination ratio (r_L (%)) of greedy-decoded sequences and test perplexity for self-terminating recurrent models (word tokenization).

model	**context**	**perplexity**
tanh-RNN	train	91.54 ± 7.9
tanh-RNN	test	136.57 ± 1.8
LSTM-RNN	train	45.80 ± 2.5
LSTM-RNN	test	91.86 ± 0.4

Table 10: Perplexities of trained recurrent language models (word tokenization).

model	context	perplexity
$\tanh$-RNN	train	61.20 ± 1.2
$\tanh$-RNN	test	186.44 ± 1.4
LSTM-RNN	train	72.72 ± 2.4
LSTM-RNN	test	178.39 ± 1.2

Table 11: Perplexities of trained recurrent language models (BPE tokenization).

Prefix	*Payne was quoted as saying : " With the album*
nucleus	's " album , " The Predious " , " The One With the Wind " , " I 've Ever 't Have You 's My Way " , " I 've Ever It 's Johnny " , " The Predes " , " " Always " , " The Predatory Was " , " The Dream " , " The Baste " , " " Always Boy " , " My Drum " , " The Simpsons " , " " Always Man ", " The " Sweet Night " , . . .
c-nucleus	's " album , " The Predious " , " The One With the Wind " , " I 've Ever 't Have You 's My Way " ⟨eos⟩
Prefix	*In a 2006 interview , fellow actor Ben Whish*
nucleus	's father , a young actor , and a romantic relationship with the show , " The One Where the The Simpsons " , " The Pape " , " The Next Generation " , " The Sixth Extinction " , " We 't You Wanna Stay " , " The Dream " , " The Predator " , " The Collection " , " The Big Lear " , " The Predor " , " The Predation " , " My Blue " , " The Simpsons " , " The Sixth Extinction " , " My Love " , " The Rise of the Year " , " The Simpsons " , " The Predator " , " My Dream " , . . .
c-nucleus	was the first time in the film , and was published in the same episode of the season . ⟨eos⟩
Prefix	*Most of what is known of Du Fu 's*
Baseline	" the " , the " " great " , the " " " , " the most important " , " the most important " , " Ode to the Nightingale " , " Ode to the Nightingale " , " Ode to the Nightingale " , " Ode to the Nightingale " , " Ode to the Nightingale " , " Ode to the Nightingale " , " Ode to the Nightingale " , . . .
STRLM	Coty , was a " one of the most important " of the American science fiction . ⟨eos⟩
Prefix	*He was relieved by Yan Wu , a friend and*
Baseline	the first wife of the Order of the Order of the Order of the Order of the Order of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic of the Republic . . .
STRLM	the wife of the Royal Navy . ⟨eos⟩

Table 12: More continuations with consistent nucleus sampling ($\mu = 0.2$) and self-terminating LSTM ($\epsilon = 10^{-3}$) with BPE tokenization.