The Curious Case of Neural Text *De*generation

Ari Holtzman[†‡] **Jan Buys**[†] **Maxwell Forbes**[†] **Yejin Choi**[†‡]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
{ahai,jbuys,mbforbes,yejin}@cs.washington.edu

Abstract

Despite considerable advancements with deep neural language models, the enigma of neural text ***de****generation* persists when these models are tested as text generators. The counter-intuitive empirical observation is that even though the use of likelihood as training objective leads to high quality models for a broad range of language understanding tasks, using likelihood as a *decoding objective* leads to text that is bland and strangely repetitive.

In this paper, we reveal surprising distributional differences between human text and machine text. In addition, we find that decoding strategies alone can dramatically effect the quality of machine text, even when generated from exactly the same neural language model.

Our findings motivate *Nucleus Sampling*, a simple but effective method to draw the best out of neural generation. By sampling text from the dynamic *nucleus* of the probability distribution, which allows for diversity while effectively truncating the less reliable tail of the distribution, the resulting text better demonstrates the quality of human text, yielding enhanced diversity without sacrificing fluency and coherence.

1 Introduction

Randomization over Maximization? On February 14th 2019, OpenAI surprised the scientific community with an impressively high-quality article about Ovid's Unicorn, written by GPT-2, the largest neural language model reported to date.[1] Notably, the top-quality generations obtained from the model rely on *randomness* in the decoding method, in particular through *top-k* sampling that samples the next word from the top k most probable choices (Fan et al., 2018; Holtzman et al., 2018; Radford et al., 2019), instead of aiming to decode text that *maximizes* likelihood.

Context:
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation (BeamSearch, b=10):
"The unicorns were able to communicate with each other, they said unicorns. a statement that the unicorns. Professor of the Department of Los Angeles, the most important place the world to be recognition of the world to be a of the world to be a of the world to be a of the world to be a of the world to be a of the world to be a of the world to be a of the…

Figure 1: Beam search leads to degenerate text, even when generated from GPT-2-117M, in stark contrast with the admirable quality of the text decoded using *top-k* sampling (Radford et al., 2019). The *continuation* is machine generated, conditioned on the *context* provided by a human. Blue text highlights decoded words that have occurred previously in the text.

In fact, decoding strategies that optimize for output with high probability, such as beam search, lead to text that is incredibly degenerate, even when using state-of-the-art models such as GPT-2-117M, as shown in Figure 1. This is counter-intuitive, as one would expect that good models would assign higher probability to more human-like, grammatical text.

Natural Language Distribution has Spikes The key to our findings is the striking distributional differences between human text and

[1] https://openai.com/blog/better-language-models/

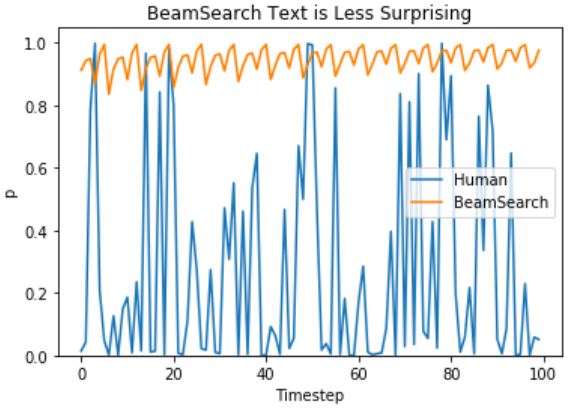

Figure 2: The probability assigned to tokens generated by humans and beam search using GPT-2-117M. Note the increased variance that characterizes the richness of human text.

machine-generated text: Figure 2 shows that the natural distribution of human text exhibits considerable fluctuations in the per-token perplexity while the distribution of machine text produced from maximum likelihood decoding leads to unnaturally flat and high per-token probability. These differences provide important clues as to why decoding strategies alone can dramatically effect the quality of machine text, even when generated from exactly the same neural language model.

Nucleus Sampling In this paper, we introduce *Nucleus Sampling*, a simple but surprisingly effective method that addresses the limitations of existing decoding methods. The key intuition is that the vast majority of probability mass is concentrated in the *nucleus* of the distribution, a small subset of the vocabulary that spans across anywhere between one to a few hundred candidates. Instead of relying on a fixed top k, we propose sampling from the top p portion of the probability mass, expanding and contracting the candidate pool dynamically. Nucleus sampling effectively reduces the risks of drawing words from the unreliable tail distribution (the origin of many awkward phrasings in machine text), while allowing for more diversity than likelihood maximization decoding methods.

We take a deep dive into the questions:

1. Why does decoding with beam search from a strong language model lead to such degenerate text?
2. Why does sampling from a truncated vocabulary distribution perform better than sampling from the whole distribution?
3. What is the most principled method of truncation currently available?

Experimental analysis confirms that Nucleus sampling, our answer to question 3, exhibits considerably better behavior than other decoding strategies.

The rest of the paper is organized as follows. In §2, we define the scope of our study to focus on *open-ended* text generation and contrast key differences with *non-open-ended* generation. In §3 — §4, we provide key insights on why maximum likelihood decoding leads to degenerate text, and why the inclusion of tail distribution leads to degenerate text. These insights motivate two prominent stochastic approaches in recent literature, *sampling with temperature* and *top-k sampling*, discussed in §5. Finally, we introduce *Nucleus (top-p) Sampling* in §6, followed by a comprehensive analysis. We discuss related work in §7 and conclude in §8.

2 The Scope of Our Study: Open-ended Generation

We define the scope of this paper (§2.1), and the language model and dataset used in all subsequent experiments (§2.2). *Open-ended* text generation is contrasted with *non-open-ended* generation. which is characterized by considerable semantic alignment between the input and the output (§2.3).

2.1 Open-ended Generation

Given an input text passage as context, the task of *open-ended* generation is to generate text that forms a coherent continuation from the given context. More formally, given a sequence of m tokens $x_1 \dots x_m$ as **context**, the task of open-ended language generation is to generate the next n **continuation** tokens to obtain the completed sequence

$x_1 \dots x_{m+n}$. We assume that models compute $P(x_{1:m+n})$ using the common left-to-right decomposition of the text probability:

$$P(x_{1:m+n}) = \prod_{i=1}^{m+n} P(x_i|x_1 \dots x_{m+n-1}). \quad (1)$$

which is then used to generate token-by-token starting with x_{m+1}.

Open-ended generation includes conditional story generation and contextual text continuation, which have recently become promising research directions due to significant advancements in deep neural language models (Holtzman et al., 2018; Radford et al., 2019; Dai et al., 2019). While the input context restricts the space of acceptable output generations, there still is a considerable level of freedom in plausible generations in this setting, in contrast to non-open-ended generation.

2.2 Language Model and Dataset

For all the analyses we perform in the remainder of this paper, we use the GPT language model, and generate text based on the WritingPrompts dataset.

Language Model While many neural network architectures have been proposed for language modeling, including LSTMs (Sundermeyer et al., 2012) and convolutional networks (Dauphin et al., 2017), the Transformer architecture (Vaswani et al., 2017) has been the most successful in the extremely large-scale training setups in recent literature (Radford et al., 2018, 2019), leading to substantially stronger generation quality. In this study we use the GPT model (Radford et al., 2018).[2]

Dataset For open-ended generation, we use the WritingPrompts dataset of (Fan et al., 2018). We extract examples from the start of the content of each story: Each example consists of a context of 5 sentences with a maximum of 200 tokens; the task is to continue the text by generating the 200 next tokens (the continuation). We also make some comparisons to the reference continuation.

2.3 Non-open-ended Generation

Many text generation tasks are defined through (input, output) pairs, such that the output is a close *transformation* of the input. Example applications include machine translation, data-to-text generation, and summarization. Non-open-ended generation using neural networks is typically modelled using variants of encoder-decoder architectures, enhanced with various attention mechanisms. Generation is most often performed using beam search. Because the scope of the output content of is tightly scoped by the input content, the degree of freedom in non-open-ended generation is substantially less than in the open-ended case. Our work addresses the challenges faced by neural text generation when faced with an increased level of freedom in generation or weak alignment between the input and the output, as in the case of open-ended generation.

Open- and non-open-ended generation are not a strict dichotomy, since some tasks may fall somewhere in between depending on the degree of freedom expected in the output generation or the degree of semantic alignment between the input and the output. For example, book-level summarization would be closer to the open-ended case, while sentence compression would be closer to the non-open-ended case.

3 Why Does Probability Maximization Lead to Degenerate Text?

In this section, we examine decoding strategies which assume that the model assigns higher probability to higher quality text, and therefore aim to find the output with the highest likelihood. More formally, these strategies define that decoding problem as

$$x_{m+1:n} = \operatorname*{argmax}_{x_{m+1:n}} P(x_{m+1:n}|x_{1:m}). \quad (2)$$

Computing the optimum argmax sequence from recurrent neural language models is not tractable, so consider two prominent decoding methods for approximating the argmax: *Beam search* is the most commonly used approximation in practice (Li et al., 2016c; Shen et al., 2017; Wiseman et al., 2017). *Greedy* decoding is a special case of beam search with beam size 1.

Beam search has been used successfully in numerous research into non-open-ended generation tasks such as machine translation, data-to-text generation, and summarization (Bahdanau et al., 2015; Luong et al., 2015; Cho et al., 2014). Thus, it would seem reasonable to expect that beam search would also work well for open-ended text generation. However, as illustrated in Figure 1, beam search for open-ended generation leads to

[2]The full GPT-2 model is not publicly available.

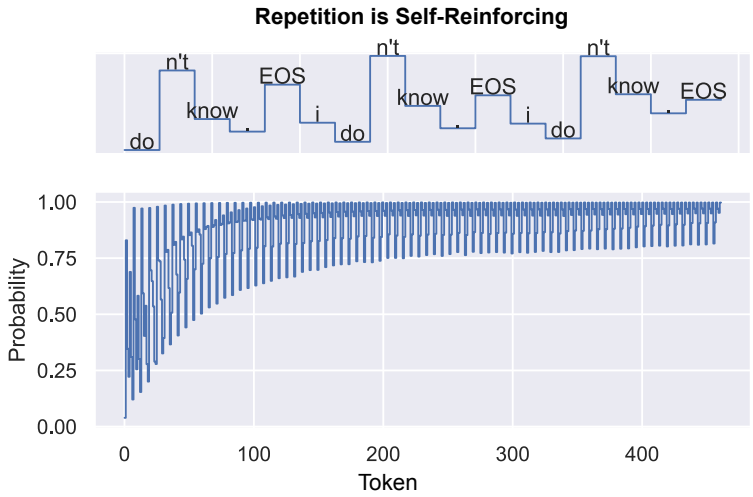

Figure 3: The probability of repetition increases with each instance of repetition, creating a positive-feedback loop.

strikingly degenerate text, even when generating from a state-of-the-art model.

One might wonder if the issue is a *search error*, i.e., there are higher quality sentences to which the model assigns higher probability than to the decodes ones, beam search has just failed to find them. However, we will show that the fundamental problem is not in the search error, but the maximum-likelihood decoding objective itself.

Our study reveals two surprising findings which provide new insights into why argmax decoding leads to degenerate text: (1) maximization naturally leads to repetition feedback loops (2) the distributional properties of maximum likelihood decoding differ strongly from human text, even from a language model's perspective.

3.1 The Gravitational Force of Repetition

Several previous studies on neural conversation models have reported that likelihood maximization approaches, such as beam search, tend to loop into repeating the same sentence, often a generic sentence such as "*I don't know.*" (Li et al., 2017, 2016a). What exactly happens when the neural text degenerates into such repetitions? The top chart of Figure 3 depicts how the per-token probability of GPT progresses through the repetition of *"I don't know. I don't know. I don't know."*, where the higher score on the Y axis indicates higher probability. What is striking is that for any fixed token, such as *"know"*, the following sequence of inequalities holds:

$$P(\text{“know”}|\text{“I don't”})$$
$$< P(\text{“know”}|\text{“I don't know. I don't”})$$
$$< P(\text{“know”}|\text{“I don't know. I don't know. I don't”})$$

In fact, this trend continues indefinitely as shown in the bottom chart of Figure 3, where the probability scores continue to rise to approach 1, as the sequence loops longer into a series of "*I don't know.*" statements. We found that this trend was true for every *any* string we looped, not just *"I don't know"*. This phenomenon may in part be an architectural side effect of transformers, where the prediction of the next word is influenced a great deal by the attention heads over the words in the previous immediate context. (Vig, 2018) [3]

3.2 The Turbulent Distribution of Natural Language

Another surprising observation is the striking difference between the probability distribution of human text and that of machine text, especially when the latter is generated using argmax decoding such as beam search. Figure 2, discussed briefly in §1, illustrates this point.

As a result, natural language rarely remains in the high probability zone for long, instead dipping into the low probability zone to give detail with content words. This explains the broken and repetitive text shown under "BeamSearch" in Figure 2.

Why is naturally existing human text *not* the most probable text? Rather than a modeling deficiency, we conjecture that this is an intrinsic property of human language. For instance, Grice's

[3] In fact, we observe the same trend with LSTM-based language models without self-attention. We conjecture that the phenomenon is again likely to be an architectural side effect, here of the recurrent parameterization.

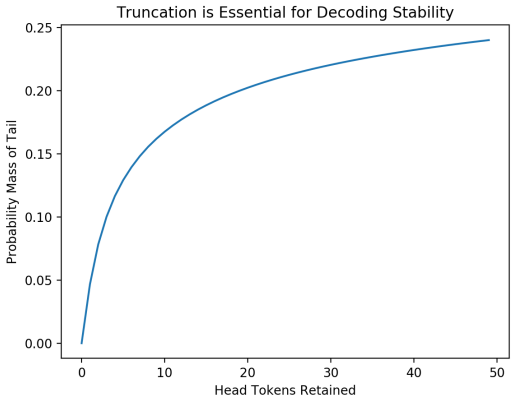

Figure 4: The chart shows the probability mass in the tail (approximated as the sum of all candidates with lower probability than the ground truth token) when only highest probability x tokens are considered; this is equivalent to asking how much of the tail is "left" when using top-k sampling where $k = x$.

Maxims of Communication (Grice, 1975) have established that people optimize against stating the obvious, making highly predictable text unlikely to occur in practice.

In sum, decoding based on *maximization* leads to text with unnaturally high probability and too little variance, which leads to distinctly unnaturally looking output. This motivates the use of *randomization* over *maximization*, which allows us to *sample* from the model's approximation of the data distribution rather than to *optimize* output probability.

4 Why Does Sampling from the Full Distribution Lead to Degenerate Text?

The findings from the previous section motivate decoding methods for open-ended generation which involves some element of *randomization*, instead of only aiming to *maximize* output probability. More formally, in sampling-based generation, at each timestep we sample the next word x_i by drawing a word from the conditional language model:

$$x_i \sim P(x|x_{1:i-1}) \tag{3}$$

While text generated using this process manage to avoid the gravitational force towards spurious repetitions, is still degenerate as it easily becomes incoherent (see the sampling example generation shown in Figure 10). We identify the unreliability of the tail of the distribution, where the quality of the learned model is relatively less robust, as the culprit. We here use "tail" to describe the large majority of tokens, which are assigned probability that is within some small ϵ of 0 because they simply don't fit. Concretely, there are two important ways the tail of the distribution is responsible for problematic generations obtained through sampling:

1. One bad sample can start a downward spiral Even one non-sensical token can start a downward spiral, thwarting the coherence of the rest of the generation. This is in part due to the *recency bias* and *explanation-away* problem, where language models have the tendency to rely overly on the short-term context that can easily explain away the longer-term context (Yu et al., 2017b).

2. Sampling from the tail is extremely likely Still, one could postulate that the probability of the words in the tail distribution is so low so that they would not in practice be sampled often enough to degrade the coherence significantly. However, the potential appearance of words from the tail distribution is extremely high during paragraph-level open-ended generation due to the fact that the probability of rare events goes up exponentially with length. Suppose the expected probability of sampling from the tail at timestep i is ϵ_i, then we have:

$$P(\text{avoid sampling from the tail}) = \prod_{i=n+1}^{n+m} 1 - \epsilon_i$$

For our analysis, we approximate the problematic tail of the distribution as the words that have lower probability than the gold token. According to this definition, in the full distribution the average probability mass assigned to the tail about 0.31. Therefore sampling from the tail is expected to happen *within the first three steps of decoding* and with $x > 99.96\%$ within 20 steps.

Truncating the Distribution The simplest solution to dealing with the tail of the distribution, is to only retain a fixed number of the highest probability tokens from the predicted distribution. The bottom of Figure 4 shows how the probability of sampling from the tail goes up the more tokens are retained, but even truncating 50 tokens still filters out a great deal of probability mass. In fact, retaining k tokens and sampling from the truncated distribution is precisely top-k sampling, bringing us to our next section.

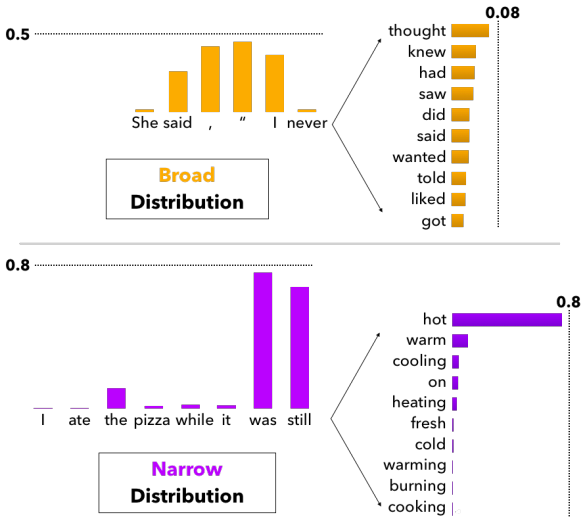

Figure 5: Examples of the probability mass assigned two partial human sentences by GPT, and the resulting **broad** and **narrow** distributions. Broad distributions lead to a large number of tokens with moderate shares of probability mass. In contrast, narrow confidence distributions (less common in open-ended generation) concentrate the overwhelming majority of probability mass into just a few tokens.

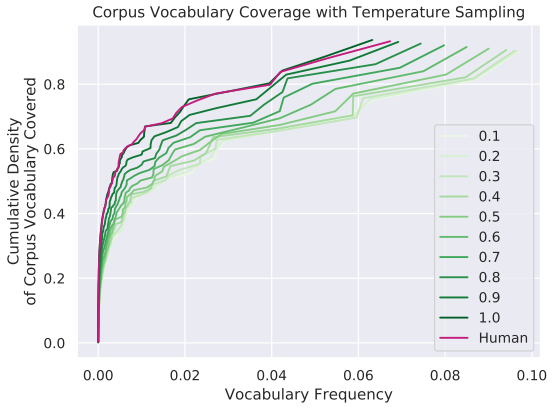

Figure 6: The fraction of the corpus covered by tokens that individually account for at most x proportion of tokens.

5 Sampling with a Truncated Tail

We now discuss two prominent methods used in literature — sampling with temperature (§5.1) and top-k sampling (§5.2) — that help attenuate the mass assigned to the tail of the distribution.

5.1 Sampling with Temperature

One common approach is to shape the distribution through temperature (Goodfellow et al., 2016; Ficler and Goldberg, 2017; Fan et al., 2018). Given the logits $u_{1:|V|}$ and temperature t, the softmax is re-estimated as

$$p(x = V_l|x_{1:i-1}) = \frac{\exp(u_l/t)}{\sum_{l'} \exp(u_l/t)}. \quad (4)$$

As $t \to 0$ this approaches greedy decoding, while $t \to \infty$ asymptotically approaches uniform sampling from the vocabulary. The use of temperature $t \in [0, 1)$ shapes the distribution to be more skewed towards high probability events, which has the implicit effect of weakening the tail distribution. Figure 5 shows how lowering the temperature increases the use of frequent words, driving down inter-generation diversity.

5.2 Top-k Sampling

Top-k sampling has recently become a popular alternative sampling procedure (Fan et al., 2018; Radford et al., 2019). At each time step, first the top k possible next tokens are selected (similar to expanding a candidate sequence in beam search). Then the next word is sampled from (only) those tokens, according to their relative probabilities.

More formally, given a distribution $P(x|x_{1:i-1})$, we define its top-k vocabulary $V^{(k)} \subset V$ as the set of size k which maximizes $\sum_{x \in V^{(k)}} P(x|x_{1:i-1})$. Let $p' = \sum_{x \in V^{(k)}} P(x|x_{1:i-1})$. The original distribution is re-scaled to a new distribution

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

from which we sample. Note that p' will be different at each time-step and there are no restrictions on its value.

While top-k sampling leads to considerably higher quality text, our investigation finds that the use of constant k is sub-optimal across varying contexts. As illustrated in the top chart of Figure 5, the next word distribution in some contexts can be flat across hundreds of reasonable options. In this case, there are many more than k reasonable candidates, and limiting sampling to only the top-k choices runs the risk of generating bland and potentially repetitive text. Figure 5 illustrates the opposite scenario, in which a model may not have k reasonable candidates because the probability mass is peaked for less than k words.

6 Nucleus (Top-p) Sampling

We propose Nucleus Sampling, a principled alternative to top-k sampling, which uses the probabil-

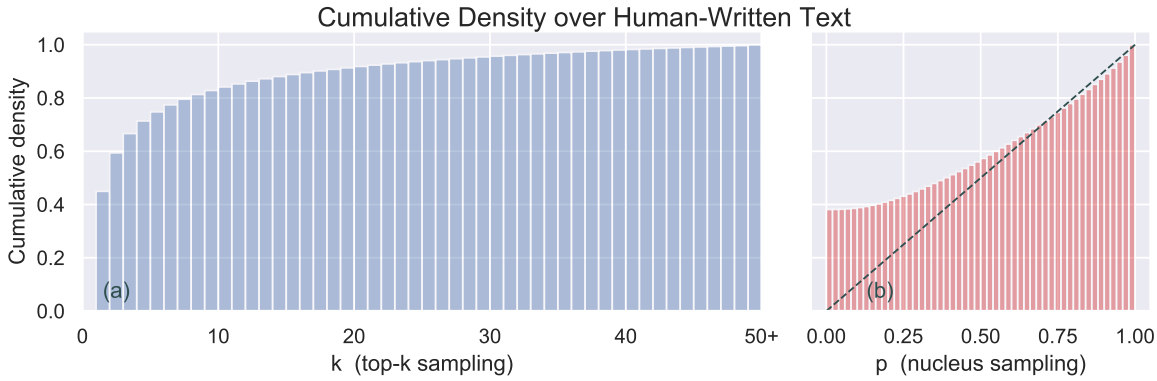

Figure 7: The left-hand side graph illustrates the diminishing returns received as the k increases in top-k, which contrasts with the increasing returns of Nucleus Sampling (right) that allows values of p close to 1 to act very similarly to pure sampling without risk of sampling from the low-confidence tail. The height of a bar encodes the cumulative density of the minimum value of k (for top-k sampling) or p (for Nucleus sampling) required to assign a non-zero probability to the *gold* next word prediction over a corpus of human-written text.

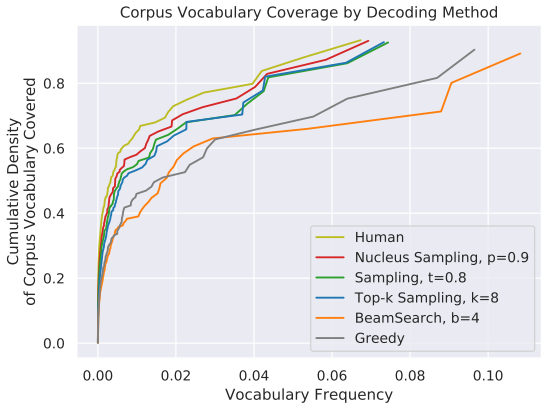

Figure 8: A chart describing the distributional differences between n-gram frequencies of human and machine text. The complete separation of likelihood maximization and stochastic methods, stochastic clearly closer to human, indicates an inherent issue with a likelihood maximization as a decoding objective.

ity distribution to determine the set of tokens to be sampled from. We define Nucleus Sampling as follows: given a distribution $P(x|x_{1:i-1})$, its top-p vocabulary $V^{(p)} \subset V$ is the smallest set such that

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) >= p. \quad (6)$$

In practice this means that we select the highest probability tokens whose cumulative probability mass exceeds our pre-chosen threshold p. Let $p' = \sum_{x \in V^{(k)}} P(x|x_{1:i-1})$. The original distribution is then re-scaled as in equation 5. However, in contrast to top-k sampling, p' will remain almost constant.

6.1 Relationship between Nucleus Sampling and Top-k Sampling

Nucleus Sampling and top-k both sample from truncated Neural LM distributions, differing only in the strategy of where to truncate. Choosing where to truncate can be interpreted as determining the generative model's confidence region.

Figure 7 helps to explain the difference between the two sampling strategies based on the cumulative density over the minimum value of p (for Nucleus Sampling) and k (for top-k sampling) required to assign a non-zero probability to the *gold* next word prediction over a corpus of human-written text. Put in other words, it represents the proportion of words in the corpus assigned a non-zero probability by the distribution for a given k or p value. For this analysis 6681 blocks of 200 words from the test set of WritingPrompts were used.

We see that the marginal increase in density becomes smaller as k gets larger, but larger for higher values of p. As top-k sampling is defined in terms of the number of candidates included, this naturally leads to diminishing returns as k increases.

For high values of p the model exhibits well-calibrated behaviour, as the proportion of the corpus covered is almost exactly equal to p (see Figure 7(b)). Therefore in this region of confidence the decoding strategy gives us fine-grained control over the relation between the model distribution and the distribution over samples. This is because this metric directly measures the increase in

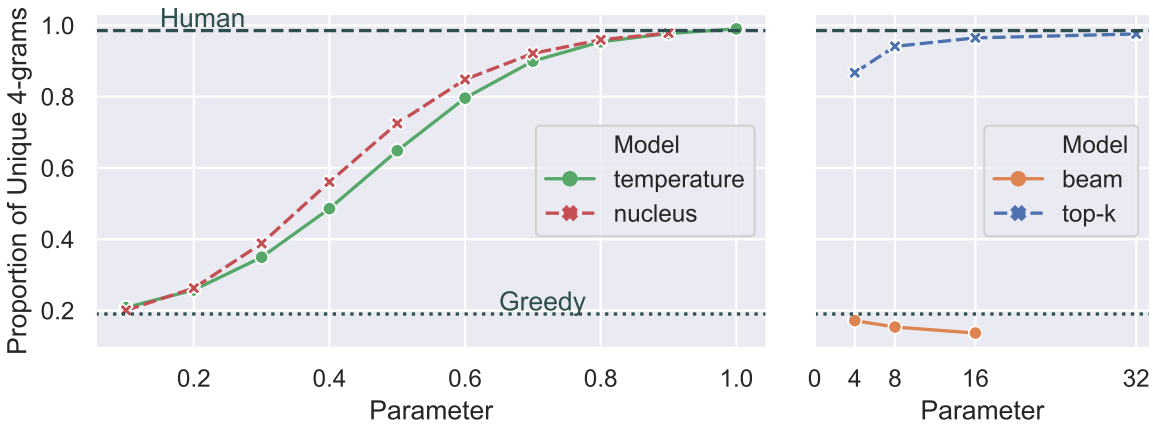

Figure 9: Unique 4-Grams across decoding methods. Gold and Greedy have no hyperparameters, whereas for BeamSearch the parameter is the beam width b, for Sampling the it is the temperature t, for Top-k Sampling it the number of candidates k, and for Nucleus Sampling the parameter is the cumulative threshold p.

corpus coverage by the model under the given decoding method for an increase its hyperparameter value. In contrast, no range of values of k displays this well-calibrated behavior.

We hypothesize that within the range of well-calibrated distributions, the distribution threshold value does not cause such a large of a trade-off between fluency and diversity as in other models (such as sampling with temperature and top-k sampling). One of the reasons this trade-off occurs in the candidate-based thresholding of top-k sampling is because there are frequently too many or too few reasonable options. This is because the decoding model of top-k sampling operates at the wrong layer of abstraction, reasoning about individual candidates, instead of clusters of likelihood. Under Nucleus Sampling the number of candidates considered rises and falls dynamically, corresponding to the changes in the model's confidence region over the vocabulary.

6.2 Comparison to Other Methods

Having given the intuition of why Nucleus Sampling works, we examine how it compares to the other decoding methods that we've explored. In terms of diversity, Figure 8 shows that Nucleus sampling is the closest to the human distribution; though higher temperature sampling is closer, it also makes sampling from the tail and devolving into nonsense highly likely.

In terms of repetition, on the other hand, Figure 9 reveals that Nucleus sampling and top-k are the clear winners, with temperature lagging a bit behind and likelihood maximization methods suffering greatly. It is interesting to note that the larger the beam size in beam search the *worse* this problem gets, as expected from our previous analysis about repetition loops in §3.1.

6.3 Qualitative Analysis

The most striking qualitative observation from Figure 8, which is quite representative of the greater evaluation set, is the fact that the two likelihood maximizing methods, Greedy and Beam Search, both get stuck in repetition loops. Of the stochastic decoding schemes, Sampling is clearly the hardest to understand; it appears to be a disconnected set of clauses, each grammatical and meaningless in its turn. The generation produced by Nucleus Sampling isn't perfect, with the phrasing of "What was the guy doing here?" reading quite strange, since "the guy" is too generic for the only apparent intended referent. Yet, Top-k Sampling's generation arguably has the highest cognitive load (also generally true) as it introduces terminology as with "special trip" and "that kid".

7 Related Work

One of the most prominent recent research directions in open-ended text generation has been using generative adversarial networks (GANs; Yu et al., 2017a; Xu et al., 2018). A number of metrics (based on BLEU and cross entropy) have been proposed to quantify the diversity and quality of open-ended generations (Caccia et al., 2018; Zhu et al., 2018; Cfka et al., 2018). However, these

🗣 **Context**: On days when he woke early, the president liked to have coffee in the oval office. There was something about watching the sky grow lighter and lighter as you sat your pajama'd behind in the most powerful chair in the free world, sipping marine strength coffee, that you just couldn't achieve anywhere else.

💰 **Greedy**: He'd been in the oval office for over a year now, and he'd never been so happy to be there. He'd been in the oval office for almost a year, and he'd never been so happy to be there.

🚦 **BeamSearch** (k=4): The President was a man of his word. He was a man of his word. He was a man of his word. He was a man of his word. He was a man of his word.

🥄 **Sampling** (t=1.0): You couldn't be sure if that's what you were really doing, and If you decided to take the day off. The president wanted you to take the day off, but he wanted to maintain a curfew and use his influence wisely.

🇰 **Top-k Sampling** (k=8): "I'm sorry, sir." "No, it's okay, son," the president said. "I understand." "You're going to have to make a special trip down there to get that kid. He has no idea where he's going."

⚛ **Nucleus Sampling** (p=0.9): But that wasn't what drew the president's attention. He'd been seated for maybe a minute when he noticed the other man. What was the guy doing here?

🗣 **Gold**: He was therefore disagreeably surprised to find a man in an understated grey suit sitting in that selfsame chair sipping tea. The president turned around and went looking for his chief of staff.

Figure 10: Example generations from all discussed decoding strategies, hyperparameters were chosen by experts. All generations for all hyperparameters will be made publicly available.

evaluations were usually performed for sentence generation, while we focused on generating larger coherent text passages. Recent work has shown that when both quality and diversity is considered, GAN-generated text is substantially worse than language model generations (Caccia et al., 2018; Tevet et al., 2018; Semeniuta et al., 2018).

Another line of research has focused on generating diverse text. Vijayakumar et al. (2018) proposed diverse beam search - a method for encouraging diversity in beam search-based generation which supports incorporating a task-specific diversity scoring function. Kulikov et al. (2018) proposed iterative beam search, applied to dialog modeling, which also imposes hard constraints which forces beam hypotheses to be sufficiently different from each other. Li et al. (2016b) also proposed a method to discourage beam hypothesis with shared prefixes.

8 Conclusion

We have shown that likelihood maximizing decoding causes repetition and overly generic language usage, while sampling methods risk sampling from the low-confidence tail of a model's predicted distribution. Further, we propose Nucleus Sampling as a solution that captures the "region of confidence" effect. In future work, we wish to dynamically characterize this region of confidence as well as using more complex decoding techniques to search the graph of confident generations for text that meets a learned criteria.

Acknowledgments

This research was supported in part by NSF (IIS-1524371), DARPA CwC through ARO (W911NF-15-1-0543), Samsung AI Research, and gifts by Google, and Facebook.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the 2015 International Conference on Learning Representations*.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. In *Critiquing and Correcting Trends in Machine Learning: NeurIPS 2018 Workshop*.

Kyunghyun Cho, Bart van Merrienboer, 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.

Ondej Cfka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. 2018. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *CoRR*, abs/1804.07972.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Language modeling with longer-term dependency. *arXiv preprint arXiv: 1901.02860*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

H Paul Grice. 1975. Logic and conversation. In P Cole and J L Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 41–58. Academic Press.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the Association for Computational Linguistics*.

Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sėbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Unpublished manuscript.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Unpublished manuscript.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. 2018. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Guy Tevet, Gavriel Habib, Vered Shwartz, and Jonathan Berant. 2018. Evaluating text gans as language models. *CoRR*, abs/1810.12686.

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*, pages 5998–6008.

Jesse Vig. 2018. Deconstructing bert: Distilling 6 patterns from 100 million parameters. *Medium*.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, Brussels, Belgium.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017a. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017b. The neural noisy channel. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *SIGIR*.