TAIL FREE SAMPLING FOR OPEN-ENDED NEURAL TEXT GENERATION

A PREPRINT

Trenton B. Bricken[*]
Duke University
trenton.bricken@duke.edu

AUTHOR LIST NOT FINALIZED.

August 21, 2019

ABSTRACT

With the increasing ability for neural networks to model natural language accurately, there is likely to be a growing number of applications for open-ended neural generation tasks. Yet, recent efforts in open-ended generation have, and continue to, produce questions as to why likelihood-maximization methods such as greedy search produce degenerate outputs. This issue, and the natural replaceability of words, motivates the use of stochastic, sampling based approaches. However, sampling in a way that generates both high quality and diverse outputs remains a non-trivial problem.

In this paper, we give theoretical and empirical evidence for why existing approaches to sampling are not robust enough across generation contexts and problem domains. We propose a new sampling algorithm, Tail Free sampling, which uses the second derivative of the model's probability distribution over its tokens to ensure that only those above a certain probability threshold are sampled from. We show through computational [and human (NEED TO DO)] evaluation that this results in generations that are on average of a higher quality and less degenerate that samples from other methods, namely Top-K and Nucleus sampling. This is because samples more often come from only the subset of tokens that are replaceable in a given context.

1 Introduction

Only with the advent of deep learning has generating long, novel natural language sequences across a wide swathe of domains become desirable. These domains include: image captioning (Hossain et al. [2019]), story generation (Fan et al. [2018]), speech synthesis (van den Oord et al. [2016]), and even novel antibody design (Liu et al. [2019]).

Tasks requiring sequence generation exist on a spectrum between close and open-ended tasks. These differ in the degree to which there is a semantic relationship between the input provided and generation produced, with a particularly important factor being how predictable the length of the output is, provided the input (Murray and Chiang [2018]). For example, more close-ended generation tasks include image captioning and machine translation. More open-ended tasks include story writing and the generation of novel protein sequences.

The Problem of Maximization The failure of current likelihood-maximizing strategies such as greedy and beam search in the domain of open-ended generation by producing short, highly repetitive outputs, has led to many questions, investigations, and possible explanations (Holtzman et al. [2019]; Hashimoto et al. [2019]; Radford et al. [2019]; Murray and Chiang [2018]; Welleck et al. [2019]). These possible explanations have included: (i) in order to maximize information density, surprising, less predictable words are used with a high frequency to avoid speaking the predictable (Holtzman et al. [2019]; Grice [1975]); (ii) models over-fitting to the training data (Hashimoto et al. [2019]); (iii)

[*]CAN ADD EXTRA THINGS IN HERE

likelihood-maximization choosing avenues down the tree of possible generations that are dead-ends with the resulting lowest entropy (Murray and Chiang [2018]); (iv) the cross-entropy loss lacking a term to penalize high probabilities assigned to previously generated tokens (Welleck et al. [2019]). THEY DONT ALL EXPLICITLY PUT FORWARD A HYPOTHESIS BUT IT CAN BE INFERRED FROM OTHER PARTS OF THE PAPERS, IS IT OK TO THEN CALL THESE EXPLANATIONS?.

The Promise of Sampling It is likely, and in some cases already shown, that these ideas contribute to the degeneracy of likelihood-maximization strategies with Welleck et al. [2019]'s new cost function looking particularly promising. However, one solution that addresses a number of issues with creating diverse, high quality generations is sampling. In the domain of producing natural human language, there are often a large number of options at both the word and sentence level that convey similar meaning[2] this motivates sampling from a set of replaceable words, rather than choosing only the best. In addition to being theoretically sound, sampling methods are one of the most promising approaches to ensuring that generations are diverse. These produce diversity in two ways: (i) generated outputs are independent from, or only weakly correlated to, other outputs because of their stochastic nature[3]. This is a problem with likelihood-maximization strategies that many have tried to address (Ippolito et al. [2019]; Vijayakumar et al. [2016]; Kulikov et al. [2018]); (ii) sampling produces diversity in the sense of deviating from the highest probability regions of the model. Hashimoto et al. [2019] acknowledges the trade-off between generating high quality sequences that are from high probability regions but may be a consequence of over-fitting, and producing novel sequences that represent more distant regions of probability space that can lose their quality as a result. This problem can be seen as an example of the ubiquitous explore exploit trade-off.

Tail Free Sampling While sampling is advantageous, we show that existing sampling methods fail to find the subset of replaceable words (or tokens) at a given point in a generation. As a result they either include too few words, resulting in a loss of diversity, or too many, resulting in a loss of quality. This is because the size of the subset of replaceable words can change dramatically given the context. We investigate existing methods and show them to not be sufficiently dynamic. Rather than considering only the Top-K words at any point (Fan et al. [2018]) or using a fixed cumulative distribution function threshold (Holtzman et al. [2019]), we use the second derivative of the distribution over tokens to find the "tail" of the distribution where the tokens plateau in probability. Any token beyond the tail on this plateau we interpret as the model to no longer considering to be replaceable and should be pruned. This results in a method that is more robust to both the particular generation context and problem domain and is shown to improve performance through an in depth analysis of where the tail is located and the amount token diversity created. As the ultimate task of any language generation task is to produce text that humans rate to be of a high quality, existing sampling strategies are also systematically compared between each other and tail free sampling using human evaluators.

2 Background and Related Work

It has been estimated that the "average 20-year-old native speaker of American English knows 42,000 lemmas"[4] (Brysbaert et al. [2016]). While we have large vocabularies, our neural architecture seems to very efficiently prune the large majority of words in our vocabulary from our conscious consideration in a given context so that only a few, replaceable words, are considered. This is not to say that all replaceable tokens are as optimal, only that they are reasonable substitutes. Neural networks trained for language tasks, similarly, have a vocabulary of tokens that they can use to generate outputs from. However, modern implementations (Fan et al. [2018]; Radford et al. [2019]; Welleck et al. [2019]; Yang et al. [2019]) do not prune any tokens before generation, instead assigning a probability to every token in the vocabulary. [WHY DONT THEY? WHAT ABOUT HIERARCHICAL SOFTMAX??]. This failure to prune most of the vocabulary means that many words which are not replaceable remain to be considered for generation, and, if chosen, can significantly reduce the quality of the output. As shown in Equation 1, the probability of sampling at least one bad token and derailing a model across a sequence generation is exponential (see also Figure 1). A single bad token can ruin a generative output. However, it is particularly damaging for an auto-regressive model because it can derail the generation of every later token conditioned upon this bad one.

$$P(\text{Derailed}) = 1 - P(\text{Good Token Sampled})^{\text{Num Sampling Steps}} \tag{1}$$

[2]Interestingly, this is also true of biology in the case of protein sequences and codons (Lagerkvist [1978]). In the protein case, each of the 20 amino acids used to make our natural proteins are unique but they cluster in terms of their chemical properties making a subset replaceable with each other (Henikoff and Henikoff [1992]).

[3]This is particularly the case for auto-regressive models where a different token selected early on alters every the conditional probability and token sampled later.

[4]A lemma is defined as an: "uninflected word from which all inflected words are derived" and also excludes proper nouns eg. the names of locations (Brysbaert et al. [2016]).

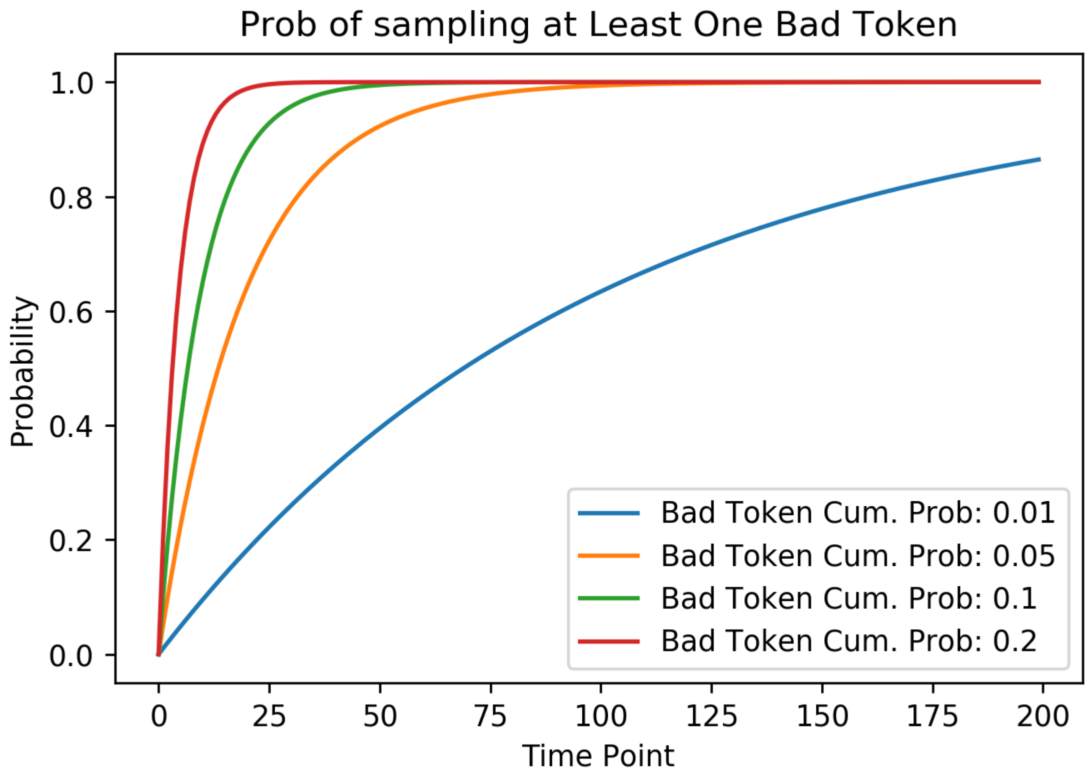


Figure 1: Each line represents a different cumulative probability that a bad token is sampled (the part of the distribution that should be pruned). This approach of sampling from the whole distribution is referred to as Ancestral sampling [NEED TO ADD A CITIATION FOR THIS].

The response to this problem has been either: (i) use likelihood-maximization; (ii) use sampling methods that prune the vocabulary. As has already been outlined in the introduction, likelihood-maximization for open-ended generation results in highly repetitive, degenerate output while sampling can address this problem.

Current Sampling Pruning Methods Top-K Sampling prunes the vocabulary by taking the highest K probability words from the distribution, normalizing, and then sampling from them (Fan et al. [2018]). This can be represented by maximizing the equation: $\sum_{x \in V^{(k)}} p_\theta(x|x_{<t})$ where x is a vector of all tokens in the vocabulary, p_θ is the probability distribution over the tokens in the vocabulary, parameterized by θ, $x_{<t}$ is the context given for the current generation step, V is the vocabulary and $V^{(k)}$ is the subset of size k of tokens.

However, a problem with this approach, realized by Holtzman et al. [2019], is that in different contexts there will be fewer or more tokens that are replaceable. This is true for human language, and should likewise be reflected in the relative probabilities the model assigns to different tokens. This motivated Nucleus sampling, which dynamically changes the pruning location by using the CDF of the token probability distribution. This is given by the equation: $\sum_{x \in V^{(p)}} p_\theta(x|x_{<t}) \geq p$ where p is the probability $p \in [0, 1]$ and $V^{(p)}$ is the smallest subset of V.

Another final approach that doesn't involve explicitly pruning words but tries to achieve the same effect is by using temperature. This is applied as the model outputs (often called logits) are converted into probabilities using the softmax equation $p(x) = \frac{\exp(x/t)}{\sum_{x \in V}(\exp{(x/t)})}$ where t is the temperature. Temperature serves to increase or decrease the relative probabilities of tokens. However, the temperature walks a fine line between being low enough that non-replaceable tokens are given such low probabilities they no longer threaten to derail the model and maintaining high enough diversity to not become approximately greedy. There are two reasons that temperature sampling is not further investigated in this paper: (i) it is not mutually exclusive to the other pruning methods and could be applied either before, after, or before and after a method that prunes; (ii) applying a temperature alters the relative probability of the tokens, changing the location of the tail and any signal for where the model considers tokens replaceable. We propose that only if the

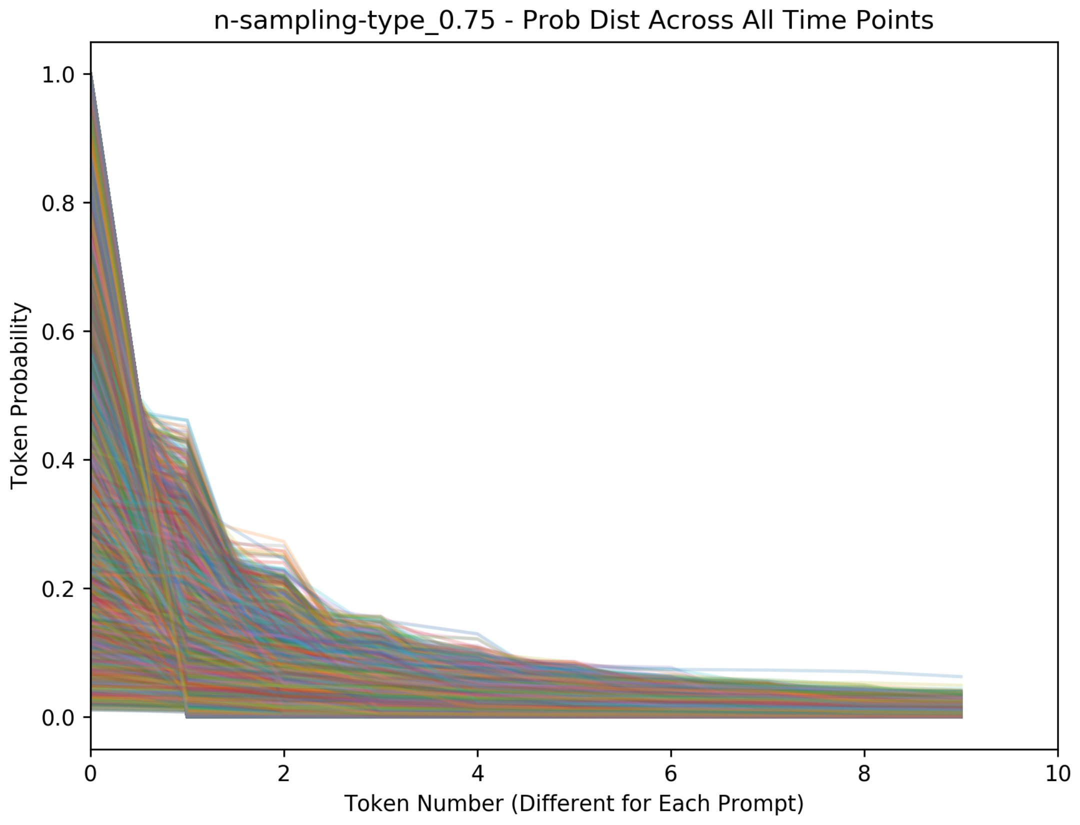


Figure 2: Sorted Probabilities for different tokens across 13,800 different generation points showing the high variation in these distributions given different contexts. This makes it it difficult for Nucleus sampling to find the distribution "tail" to prune using the CDF.

model used to produce the probability distributions is assigning probabilities across its tokens in a biased way should temperature be applied in addition to one of the pruning methods to correct this.

3 Tail Free Sampling

Tail Free Sampling (TFS) is motivated by trying to dynamically find where to prune the distribution depending on the context of how many tokens are replaceable. However, TFS deviates from Nucleus sampling by using the second derivative of the probability distribution rather than its CDF to more accurately find the "tail" of the distribution where the tokens are no longer replaceable. Specifically, the point at which the probabilities assigned to tokens plateaus and becomes almost uniform, indicating that the model does not predict these tokens to be replaceable for the context. This application of the second derivative leads to a more robust and dynamic discovery of the model's tail as will be shown in various ways in Section 4 after explaining the algorithm more formally.

We argue that aside from being more dynamic in finding the tail of the distribution, TFS is also more dynamic across models and problem domains with different vocabularies and contexts. The problem domain in which a model is generating will alter the average number of replaceable words that exist at any given context. The size of a model's vocabulary will inevitably alter its CDF but should not change the absolute number of words that are replaceable. Both of these facts make Nucleus sampling and Top-K necessary to tune across models and domains to try and approximate the subset of replaceable words, while TFS with a given z hyper-parameter will continue to find the same tail location for any distribution.

TFS Algorithm The TFS algorithm is shown in formally and also described step by step here. TFS first converts logits output by a model into probabilities using the softmax function before sorting them in descending order. It then calculates the first and second derivatives, as the tokens are discreet this can be found with subtraction. The magnitude of each second derivative is then taken and normalized so that they sum to 1. Finally, a threshold z is used to determine what part of the cumulative distribution of the second derivative weights to define the "tail" of the distribution to be at. This is a hyper-parameter that can be tuned. However, as we show later, there is lower variance in the effects of this hyper-parameter than for Nucleus sampling and empirically values ranging from 90% - 95% work well at reliably finding the last drop in the probability distribution before it plateaus. After pruning the tail, the remaining token probabilities are re-normalized and sampled from.

Algorithm 1: Tail Free Sampling Algorithm, all operations are vectorized

```
Input: Sequence of Logits, $L = (l_1, ..., l_N)$ & TFS parameter, $z$
Output: Location of the Tail, $t$
$L \leftarrow \text{sort}(L)$, where $l_i > l_2 > \ldots > l_N$ (sorting the probabilities in descending order)
$L \leftarrow \exp((l_i)_{i=1}^{N}) / \sum_{l' \in L} \exp(l')$ (applying the softmax to convert logits into a probability distribution)
$D^1 \leftarrow (l_i)_{i=2}^{N} - (l_i)_{i=1}^{N-1}$ (calculating the first derivative)
$D^2 \leftarrow (D_i^1)_{i=2}^{N} - (D_i^1)_{i=1}^{N-1}$ (calculating the second derivative)
$D^2 \leftarrow |(D_i^2)_{i=1}^{N}|$ (calculating the absolute values of the second derivative)
$W \leftarrow (D_i^2)_{i=1}^{N} / \sum_{d' \in D^2} d'$ (normalizing the second derivative)
$W^{(z)} \leftarrow \sum_{w' \in W^{(z)}} w' \geq z$, such that $W^{(z)} \subset W$ is the smallest subset (finding the smallest subset of second
  derivative magnitudes that surpass the threshold)
$t \leftarrow |W^{(z)}|$ (getting the cardinality of the subset)
return $t$
```

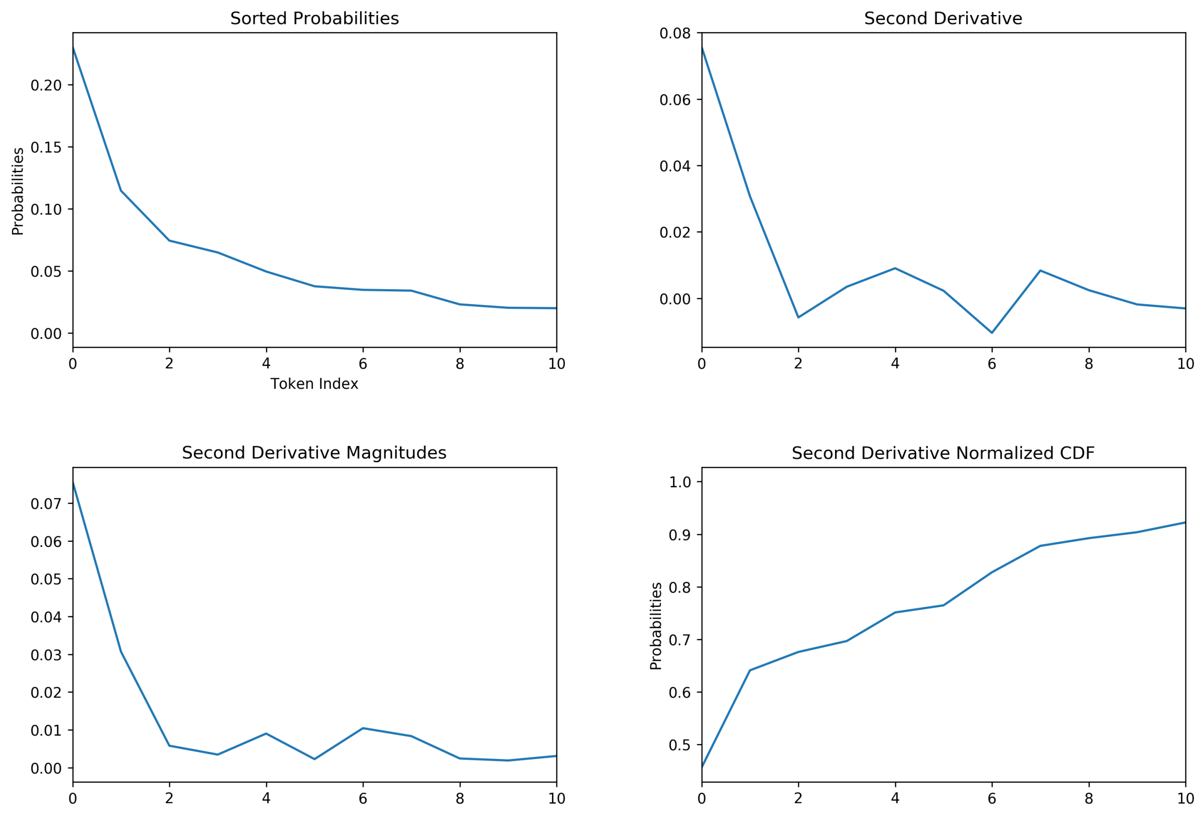


Figure 3: Steps in the Tail Free Sampling (TFS) algorithm using example logits generated from GPT-2.

4 Experiments

NEED TO SAY IF I AM USING GROVER! (Depends if I use it to validate).

Model Architecture In order to validate the TFS algorithm, we used GPT-2's largest publicly available pretrained model with 345 million parameters (Radford et al. [2019]) [5]. No additional training of the model was performed.

Dataset Our generation tasks used the same Writing Prompts database (Fan et al. [2018]) as Nucleus sampling, which was scraped from Reddit's /r/WritingPrompts channel. Posts consist of a user writing a creative story prompt such as "[WP] A man finally discovers his superpower... well into his 80's." which other users then write short stories in response to. The dataset is presented as a pair of a prompt and human written creative story response.

Text Generation In order to compare the different sampling strategies, 100 prompts were randomly selected from the writing prompts test set. Using GPT-2's encoding, 100 tokens from the story prompt and start of the human completion combined were given as the starting point for GPT-2 which was then allowed to continue the story for another 150 tokens[6]. First, in order for none of the sampling strategies to influence the generation and probability distributions, the human written context was given to the model at each time point. In other words, the sampling strategies were analyzed without actually using them to generate completions from the prompt. GPT-2 has a vocabulary size of 50,527 and at all 150 time points for all 100 prompts, every logit was stored for analysis by the different sampling methods to see how each would prune the distribution. Eight of the prompts randomly selected did not have 150 tokens in the human completion of their stories and were removed so that each completion would be the same length, leaving 92 generations

[5]Only the largest model of the Radford et al. [2019] paper with 1.5 billion parameters is officially called GPT-2, however for convenience the GPT 345M model will be referred to as GPT-2 for the rest of the paper.

[6]These prompt and generation lengths are similar to those in other open-endeed generation papers (Fan et al. [2018]; Zellers et al. [2019])

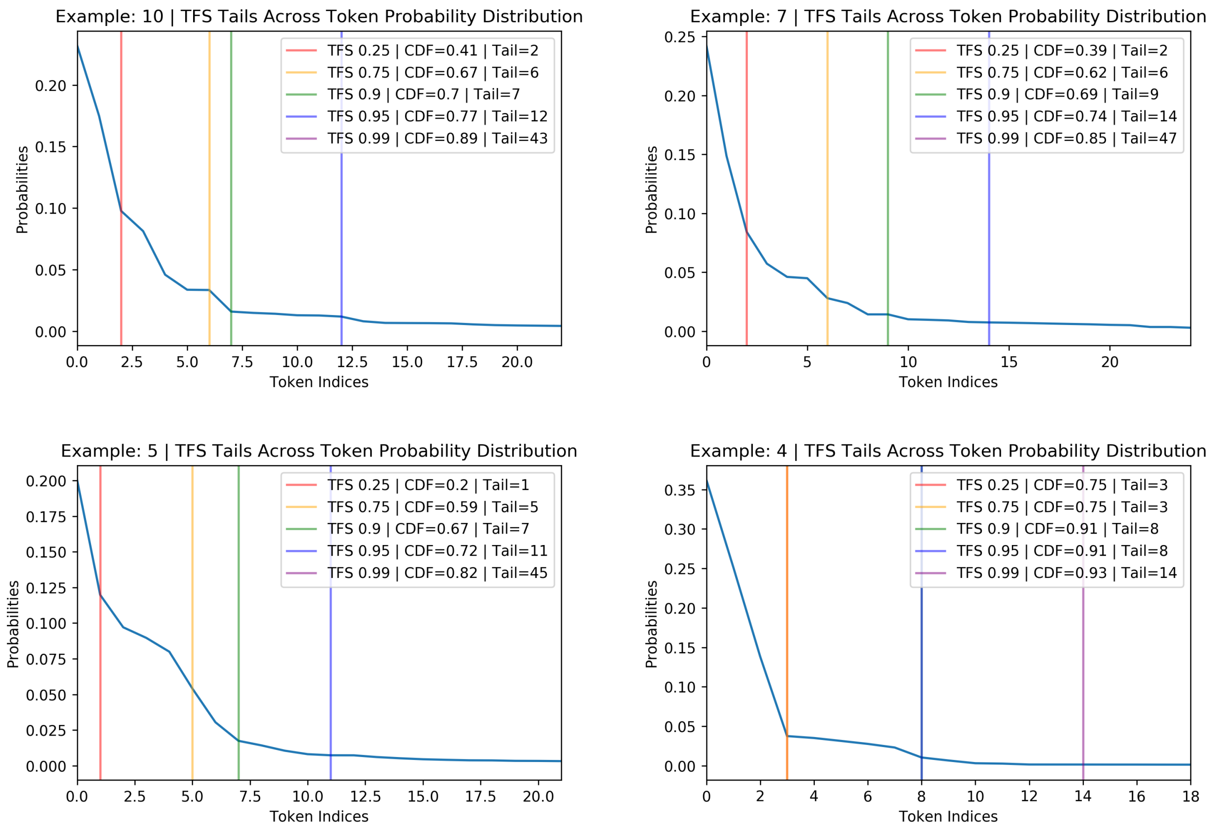


Figure 4: Four randomly chosen token probability distributions out of the 13,800 possible options demonstrating how different z thresholds change where the tail location is identified. The $z = 0.99$ threshold does not always appear in the plot and the other thresholds sometimes overlap such as in the bottom right plot.

for further analysis. This resulted in 13,800 different sampling steps. For later assessments by human evaluators and of token diversity, the same 92 prompts were used but each sampling strategy actually generated completions.

4.1 Evaluation Metrics

SAY IF I AM USING GROVER! ESP FOR THE HUMAN GENERATED EVALUATIONS.

The evaluation metrics used are briefly described here and more in depth in the Results (Section 5). Each evaluation used the Top-K, Nucleus, and TFS sampling strategies for a number of different hyper-parameter values, in particular ones that seemed to be the best baselines against TFS. These baselines were discovered both from our analysis and existing literature. The evaluations used are: (i) the locations of the "tail" for each sampling method. A comparison of different tails and the CDFs of the probability distribution that they retain; (ii) the token diversity of each sampling strategy is calculated and compared to that of the human text; (iii) [MAY ADD THIS the model perplexity and probability assigned to its highest probability token are assessed]; (iv) the computational time that each method requires was recorded; (v) text completions were generated and evaluated in head to head comparisons by crowdsourced human evaluators.

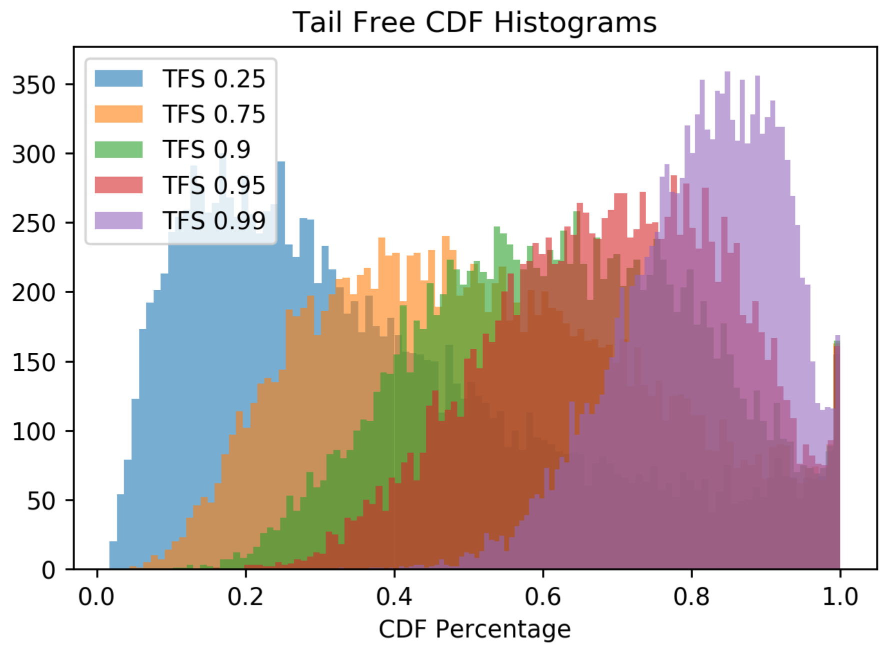
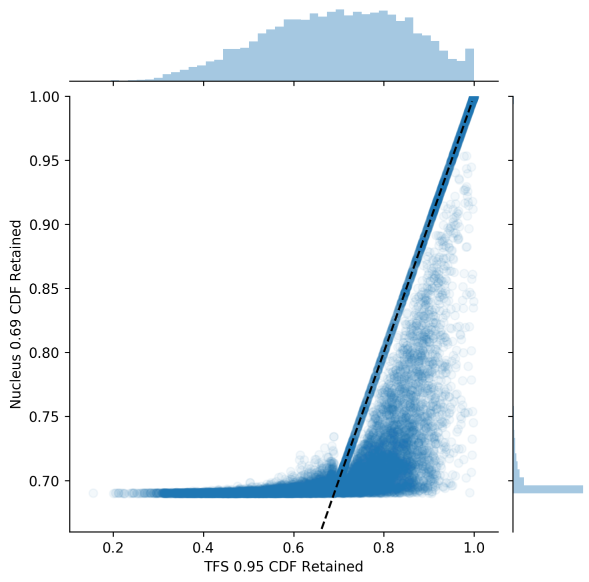


Figure 5: TFS is used to find the tail of the distribution and the CDF that will remain after pruning. Histograms of 13,800 results are plotted for the different thresholds. None of the TFS z thresholds has a constant CDF threshold in its tail found.

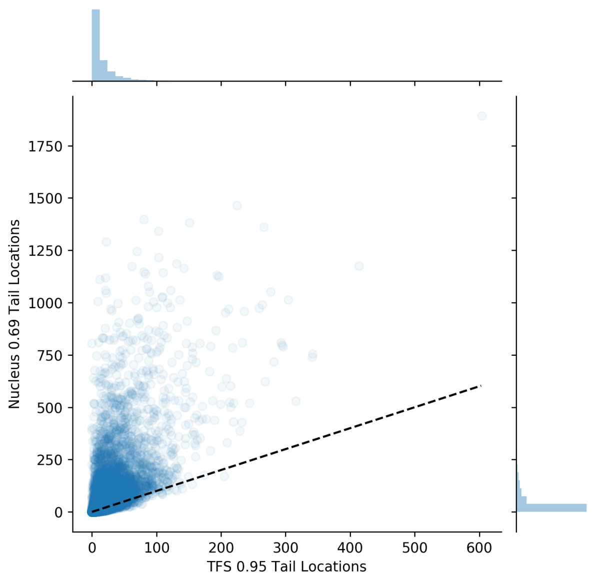
5 Results

Tail Locations Five different TFS hyper-parameters for z, determining the CDF of the second derivative cutoff, were evaluated. In Figure 4, we show how these different z values change where the tail is identified along randomly chosen token probability distributions created by the human text completion. We decide that $z = 0.95$ is a good value for relibably finding the tail and use it for our further investigations. While Nucleus sampling prunes the token distribution such that the CDF remaining is as close as possible to satisfying its threshold parameter p, TFS pruning produces a high variance CDF as shown by Figure 5. The TFS where $z = 0.95$ yields an average CDF threshold of 0.69 which motivated the use of Nucleus sampling with $p = 0.69$ as the most similar baseline.

We first investigate the differences in the tail location and CDF retained for the TFS $z = 0.95$ and Nucleus $p = 0.69$, shown in Figure 6. Overall, $\sim 15\%$ of the 13,800 sampling steps agree (this is shown as the black dotted line and can be seen most clearly in Figure 6(a). These agreements are most often where the first few tokens have disproportionately large probabilities, causing Nucleus to exceed its CDF threshold and creating a strong second derivative signal for TFS. In $\sim 48\%$ of cases TFS has a tighter tail and CDF location with the remaining $\sim 38\%$ Nucleus being tighter.



(a) The Nucleus CDF deviates from its set 0.69 point when the smallest set of tokens surpasses the cutoff.



(b) TFS having a tighter tail for the same token probability distributions is shown clearly here.

Figure 6: The CDF retained (Left) and Tail Location (Right) for TFS where $z = 0.95$ against Nucleus where $p = 0.69$. A black dotted line is added where $x = y \sim 15\%$ of the 13,800 points exist along it. This is where the token probability distribution is very tight with the first few or single token taking almost all of the probability, causing the TFS and Nucleus methods to agree. $\sim 48\%$ of points are to the left of $x = y$ and $\sim 38\%$ to the right. Therefore, a majority of the time the TFS prunes more tightly than Nucleus leaving less of the CDF and a tighter tail location.

While TFS deviates from Nucleus sampling in its tail location, this is neither good nor bad without looking at plots of exactly where each method draws its cutoff. This resulted in an investigation into cases of maximum deviation between the methods. The most extreme deviations exist in the bottom left and portions of the CDF plot Figure 6(a) where Nucleus behaves as it should with a CDF around 0.69 but TFS finds a tail leads to a very small or large CDF. The top ten maximum deviations were located and four were randomly selected to be displayed in Figure 7 with the remainder in the Appendix.

Deviations where the TFS tail is tighter than for Nucleus are particularly damaging to the generation. This is because many low probability tokens the model does not consider replaceable fail to be pruned and can potentially derail the model. For example, the bottom left subplot in Figure 7 TFS identifies the tail at position 16 with a CDF of 0.21. Nucleus identifies the tail at 637 with a CDF of 0.69. Upon pruning, Nucleus keeps an additional 621 tokens which will represent 70% of the pruned, re-normalized probability mass that is sampled from ($\frac{0.69-0.21}{0.69} = 0.7$) .

This analysis of the maximum deviation between Nucleus and TFS was also helpful in ruling out another sampling method that was initially considered a simpler alternative to TFS in the form of a flat probability threshold. For example, only keeping and sampling from tokens that were above a 2% probability. The bottom row of Figure 7 where both of the tails are identified to be below the 2% mark shows that such an approach would not be dynamic enough to identify the tail across different probability distributions.

Token Diversity One validation method used by Holtzman et al. [2019] was to calculate the diversity of vocabulary used across all generations made by different sampling approaches in comparison to the diversity seen in the human text completions. As can be seen in Figure 8 greedy likelihood-maximization (which is analogous to Top-K where $k = 1$) has only a few words that are generated across the prompts which are repeated many times contributing to larger parts of the CDF. Meanwhile, a looser Top-K, TFS and Nucleus sampling with the investigated parameters closely model the human language distribution across all of the prompts.

HERE IS WHERE GENERATING MORE SAMPLES MIGHT MAKE A DIFFERENCE. [ALSO WITH THE PERPLEXITY CI'S]

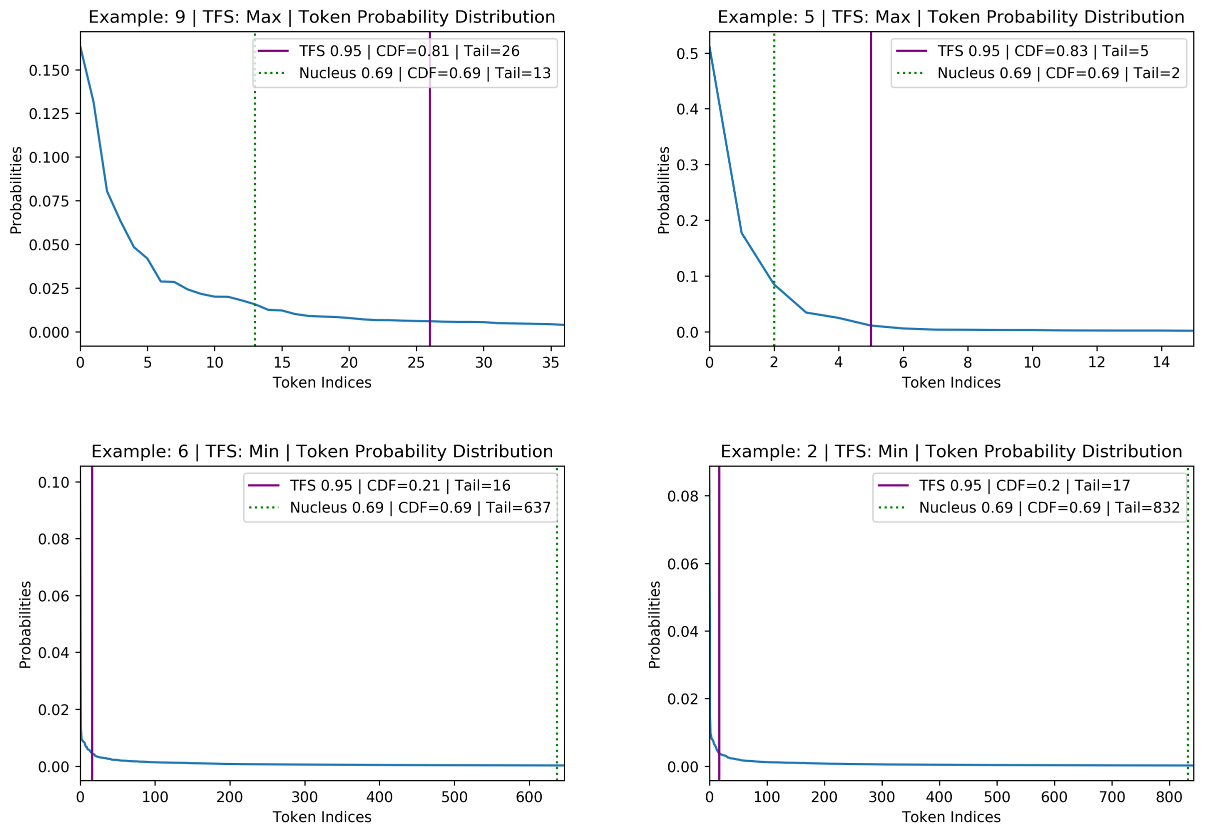


Figure 7: Four randomly selected examples of maximum disagreement between TFS and Nucleus sampling. The top row is where the TFS had the larger CDF than Nucleus (Figure 6(a) bottom right of the scatter plot). The bottom row is where the TFS had the smaller CDF than Nucleus (Figure 6(a) bottom left of the scatter plot). These examples show that Nucleus sampling's use of a fixed CDF value fails to generalize across the possible token probability distributions in finding the tail of the distribution.

[IGNORE FOR NOW - SEEING IF I CAN GET MORE STATITSICAL SIGNIFICANCE FOR THESE: **Model Perplexity** Another method that Zellers et al. [2019] uses to evaluate sampling quality is the perplexity of model generations. Perplexity is noisy and the probabilities are all similar. Don't seem to add much.

Need to see w/o CIs. and then with them. Need to get more examples? Need to run for longer period of time?? Run for a longer period of time and see if stabilizes over time? DO THIS FOR ONE (TFS 0.95) AND SEE WHAT HAPPENS. WILL PROBABLY WANT TO COMPUTE PERP ETC ON THE FLY? DO THIS BUT ADD IT TO Appenix FOR NOW?

RELATE GLTR PAPER TO ENTROPY.]

Efficiency A potential limitation of TFS is that it is a more complex algorithm than Nucleus or Top-K sampling. In particular, the mathematical operations of both of these other sampling approaches are a subset of the TFS algorithm's operations. To evaluate differences in speed, the sampling strategies across all parameters were run across all 100 prompts with 150 tokens generated for each in batches of 25. This was replicated five times using a different random seed each time so that different prompts and tokens were selected resulting in 75,000 tokens generated. The start and end of the batch generations of the 150 tokens was timed and compared for each method, shown in Figure 9. At $\alpha = 0.05$ there is no statistically significant difference between the sampling strategies except for Top K sampling. This is likely to be because Top K sampling was already implemented in the GPT-2 source code by Radford et al. [2019] and is vectorized. None of the other implementations are currently vectorized and use a for loop to iterate through the batch. [I WILL RUN AGAIN WITH ALL EITHER VECTORIZED OR FOR LOOP] While TFS is a more complex algorithm,

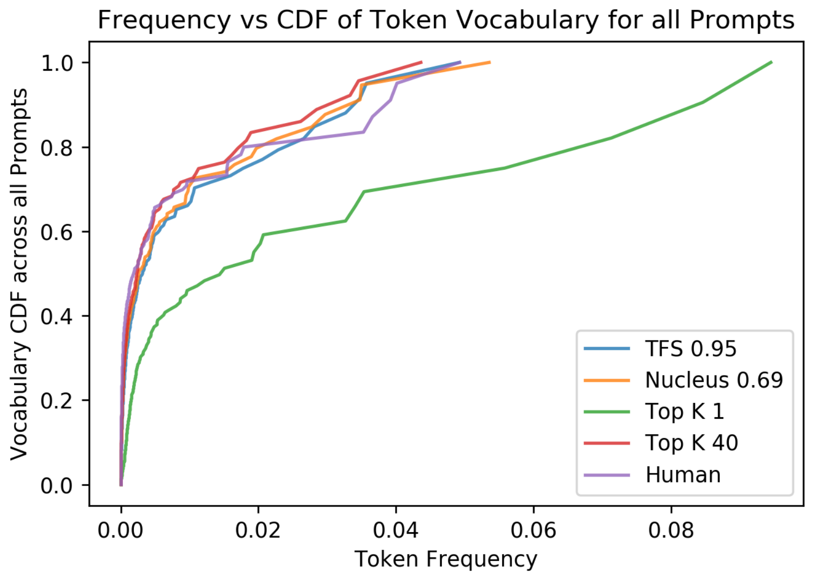


Figure 8: Distributions of the token frequency and its contribution to the CDF. The distributions for other sampling hyper-parameters can be found in the Appendix

needing to compute the second derivative, take its absolute value, and normalize it, these operations are shown to be highly efficient on a GPU [7].

ENSURE THAT FLAT ISNT IN THE PLOT.

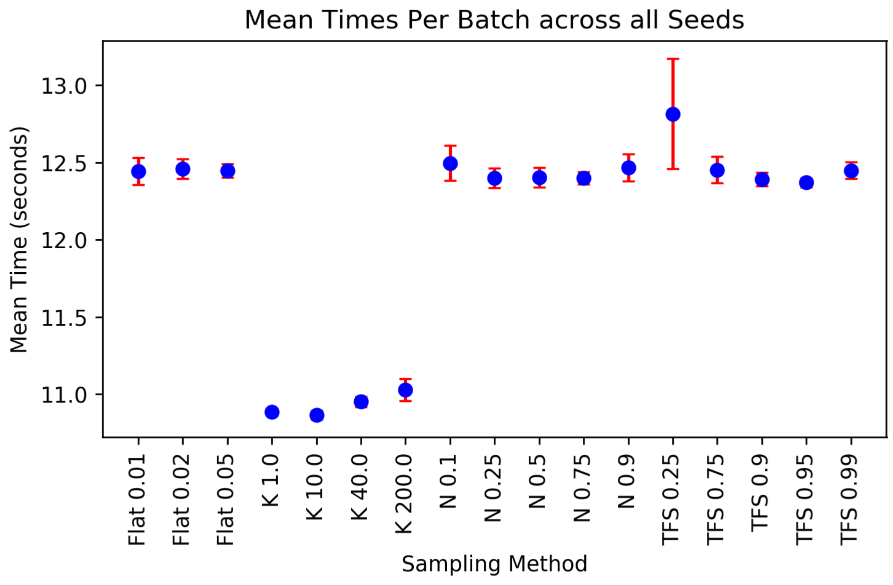


Figure 9: Average times taken to produce a batch of 25, 150 token long generations. 95% Confidence Intervals in red

Human Evaluation The terminal goal of any language generation task is to produce text that humans rate to be of a high quality while also maintaining diversity. Holtzman et al. [2019] failed to perform any human evaluation of Nucleus sampling against Top-K sampling. To the author's knowledge, these existing methods are compared with each other and also TFS using human evaluators for the first time. The evaluators were crowdsourced from Amazon Mechanical

[7]A single NVIDIA T4 GPU was used for all of the research.

Turk [Do I need to cite this?] and were tasked with determining which of two text generations for a given prompt was better in quality. ADD IN ALL OTHER DETAILS INCLUDING IF IT WAS FROM GROVER OR NOT.

6 Conclusion

In this paper we present Tail Free sampling, a new method to sample from neural networks for open-ended generation tasks. Our method is motivated by finding the subset of a vocabulary that is replaceable in a given context and does so by finding where the model, which hopefully matches reality, plateaus in the probability assigned to its tokens. We show that existing approaches fail to achieve the same objective and as a consequence do not perform [ONLY IF ACTUALLY TESTED AND TRUE] as well in human evaluations of text generation. In addition, this approach is computationally efficient and can generalize across problem domains and model vocabulary sizes where existing methods cannot not.

Acknowledgements

DOUBLE CHECK AUTHORS ARENT IN ACKNOWLEDGEMENTS!? CHECK IF AUTHOR SHOULD BE PLURAL OR SINGULAR. The authors would like to thank David Banks, Miles Turpin, Nathan Rollins, Debora Marks, Elle Deich, Gwern Branwen without whose commentary, feedback, and inspiration this work would not have been possible. Funding for this project was provided indirectly through Daniel & Kathryn Bricken, the Robertson Scholars Leadership Program, and the Google Cloud Platform's free trial credits.

References

Marc Brysbaert, Michaël Stevens, Paweł Mandera, and Emmanuel Keuleers. How many words do we know? practical estimates of vocabulary size dependent on word definition, the degree of language input and the participant's age. *Frontiers in Psychology*, 7:1116, 2016. ISSN 1664-1078. doi: 10.3389/fpsyg.2016.01116. URL `https://www.frontiersin.org/article/10.3389/fpsyg.2016.01116`.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical Neural Story Generation. *arXiv e-prints*, art. arXiv:1805.04833, May 2018.

H. P. Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York, 1975. URL `http://www.ucl.ac.uk/ls/studypacks/Grice-Logic.pdf`.

Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation. *CoRR*, abs/1904.02792, 2019. URL `http://arxiv.org/abs/1904.02792`.

S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, Nov 1992. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC50453/`.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration. *arXiv e-prints*, art. arXiv:1904.09751, Apr 2019.

MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv.*, 51(6):118:1–118:36, February 2019. ISSN 0360-0300. doi: 10.1145/3295748. URL `http://doi.acm.org/10.1145/3295748`.

Daphne Ippolito, Reno Kriz, Maria Kustikova, João Sedoc, and Chris Callison-Burch. Comparison of Diverse Decoding Methods from Conditional Language Models. *arXiv e-prints*, art. arXiv:1906.06362, Jun 2019.

Ilya Kulikov, Alexander H. Miller, Kyunghyun Cho, and Jason Weston. Importance of a Search Strategy in Neural Dialogue Modelling. *arXiv e-prints*, art. arXiv:1811.00907, Nov 2018.

U Lagerkvist. "two out of three": an alternative method for codon reading. *Proceedings of the National Academy of Sciences*, 75(4):1759–1762, 1978. ISSN 0027-8424. doi: 10.1073/pnas.75.4.1759. URL `https://www.pnas.org/content/75/4/1759`.

Ge Liu, Haoyang Zeng, Jonas Mueller, Brandon Carter, Ziheng Wang, Jonas Schilz, Geraldine Horny, Michael E. Birnbaum, Stefan Ewert, and David K. Gifford. Antibody complementarity determining region design using high-capacity machine learning. *bioRxiv*, 2019. doi: 10.1101/682880. URL `https://www.biorxiv.org/content/early/2019/07/18/682880`.

Kenton Murray and David Chiang. Correcting Length Bias in Neural Machine Translation. *arXiv e-prints*, art. arXiv:1808.10006, Aug 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *arXiv e-prints*, 2019.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv e-prints*, art. arXiv:1609.03499, Sep 2016.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424, 2016. URL `http://arxiv.org/abs/1610.02424`.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural Text Generation with Unlikelihood Training. *arXiv e-prints*, art. arXiv:1908.04319, Aug 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv e-prints*, art. arXiv:1906.08237, Jun 2019.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending Against Neural Fake News. *arXiv e-prints*, art. arXiv:1905.12616, May 2019.

Appendix

CITE NUMPY, PANDAS, SCIPY, MATPLOTLIB, SEABORN, TENSORFLOW, ANACONDA/JUPYTERNOTEBOOKS(?), AND THANK ALL THE OPEN SOURCE CONTRIBUTORS.

Also need to add the other examples of the tails found and the max disagreements. Add other vocab max deviation plots too!