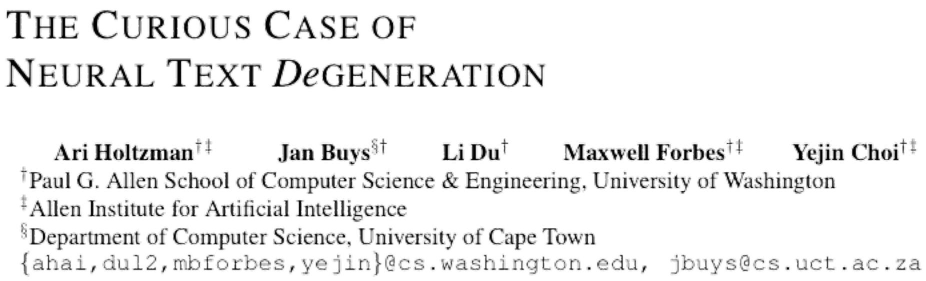
Nucleus Sampling for Open-ended and Directed Text Generation

Mathilde Kretz, Alexandre Ngau, Thomas Boudras[1]

[1]PSL Research University - Large Language Models Paper-based Course Project

January 2024



The Curious Case of
Neural Text *De*Generation

Ari Holtzman[†‡] Jan Buys[§†] Li Du[†] Maxwell Forbes[†‡] Yejin Choi[†‡]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
[§]Department of Computer Science, University of Cape Town
{ahai,dul2,mbforbes,yejin}@cs.washington.edu, jbuys@cs.uct.ac.za

Introduction

This project is aimed at reproducing the method presented in an article related to the Large Language Models field of research, and to explore extensions and adaptations.

The following project report focuses on the nucleus sampling method detailed in the ”The Curious Case of Neural Text *De*Generation” article by Holtzman et al. [1]. Our first step involves replicating the results outlined in the article, specifically those pertaining to text generation. Following this, our exploration extends to the integration of the nucleus sampling method within the domain of text translation.

1 Nucleus Sampling for Text Generation : The Motivation behind the Paper [1]

The article ”The Curious Case of Neural Text Degeneration” [1] addresses the challenge of determining the optimal decoding strategy for text generation from neural language models. Indeed, despite advancements in neural language modeling, the article highlights the counter-intuitive phenomenon that some decoding methods such as beam search, which maximizes token likelihood during generation, often results in degenerate outputs : bland, incoherent, or repetitive text. This phenomenon can be clarified by noting that the repeated generation of the most probable output occurs due to a particular score linked to the specific context.

On the contrary, purely stochastic decoding strategies such as top-k sampling, which samples from the top-k most probable outputs, tends to generate incoherent text, due to the unreliable tail.

To grasp this idea better, let's picture two scenarios in text generation. In the first one, just a few words are way more likely than the rest (like ”hot” after ”I ate the pizza after it was still”). This is called a ”peaked distribution.” Now, in the second situation, lots of words have an equal chance (such as ”knew,” ”thought,” etc., after ”She said: 'I never'”). This is termed a ”flat distribution.” Now, if we use top-k sampling, it works well when the distribution is flat because it picks from roughly equally probable words. However, in a peaked distribution, it ends up choosing unlikely words, known as the ”unreliable tail,” leading to confusing results. It's worth noting that using a small k is like maximizing likelihood, causing the problem mentioned earlier. Keep this example in mind to understand the solution proposed by the paper.

The authors introduce a new way of decoding, known as Nucleus Sampling. This method seeks to make the generated text better by preventing a decline in quality. It does this by cutting off the unreliable tail of the probability distribution and selecting from the dynamic nucleus of tokens that hold most of the probability mass. Going back to the earlier example, instead of holding onto the k most likely tokens, nucleus sampling retains the most likely tokens that, when combined, have a probability of p of happening. This technique is recognized as the most successful decoding strategy, resulting in high-quality and varied long-form text.

2 Nucleus Sampling for Open-ended Text Generation : Reproducing the Original Paper's Method [1]

2.1 Setup and Issues Encountered

2.1.1 Open-ended Text Generation

We chose to recreate the method outlined in the paper for generating open-ended text, which is the primary focus of [1]. Our decision involved generating text only from a given context, also known as a prompt. To achieve this, we had to work with a collection of texts that would fill the prompts and provide a starting point for the model to generate. The dataset we used is designed for the gpt-2 model, the pre-trained model applied in our case, and is accessible through OpenAI's repository.

To enable proper text generation, we first downloaded the complete dataset. Following that, we tokenized the prompts, filtered them to keep only those with a maximum length and that constituted full sentences. The generation can then begin.

2.1.2 Greedy Search

The simplest method for generating text is called greedy search. In this approach, for every new token produced by the model, we choose the one with the highest likelihood of coming after the preceding sequence.

2.1.3 Beam Search

Beam searching adds a bit more complexity. It begins by setting up a cluster with the initial prompt repeated w times. Then, the w most likely sequences are generated and assigned one by one to lines in the cluster. Starting from the second iteration, for each of the w alternatives, another set of w alternatives is generated. Among these $w \times w$ possibilities, the w alternatives or beams with the highest cumulative probabilities are retained. The cumulative probability of a sequence is calculated by adding up the probabilities associated with each token in the sequence. This iterative process continues until the sequence reaches the desired size or the final word is reached. In the end, the final sequence is selected by choosing the one with the highest cumulative probability.

This method is entirely deterministic, following a fixed process at each iteration and selecting alternatives based on their cumulative probabilities.

2.1.4 Top-k Sampling

In top-k sampling, we pick the k tokens that are most likely to come after the preceding sequence when generating a new token. Subsequently, we separate the probabilities linked to these k tokens and form a new distribution. The final step involves randomly selecting a token from this new distribution, and that token is then used to generate the next part of the text.

Ultimately, it's worth noting that the greedy search method aligns with the top-k method when $k = 1$. This is because, in both cases, we opt for the single most likely token during each stage of generation.

2.1.5 Nucleus Sampling

In nucleus sampling, when generating each token, we pick tokens until the total probabilities of these tokens reach a specified threshold value, denoted as p. Once these chosen tokens are identified, we isolate them, create a new distribution using their associated probabilities, and then randomly select a token from this fresh distribution. This selected token is subsequently utilized in the process of text generation, the decoding method thus being stochastic.

2.2 Results

2.2.1 A Quality Metric: Perplexity

Perplexity is a crucial metric in evaluating the quality of language models and decoding strategies, particularly in the context of text generation. It is a measure of how well a probability distribution or a language model predicts a sample. In the context of this project, perplexity is computed based on the negative log-likelihood (NLL) of the token probabilities stored during the generation process, using a token-by-token perplexity computation.

Given a sequence of tokens, the perplexity is calculated as the exponential of the average negative log-likelihood per token. Formally, if N represents the total number of tokens in the generated text and $p(n_i)$ denotes the probability assigned to the i-th token by the language model, the perplexity is computed as:

$$Perplixity = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(n_i)\right)$$

Lower perplexity values indicate that the language model assigns higher probabilities to the observed tokens, thus proving a sequence more likely to be generated by the model. However, in the context of nucleus sampling, which aims to improve the quality of generated text by truncating the unreliable tail of the probability distribution, a lower perplexity does not necessarily mean a better generation, in the sense of a more diverse and unique way, as humans would do. This is why not only the perplexity of generated texts has to be computed, but it has to be compared to a *golden human standard*, that is to say a perplexity value for human quality.

2.2.2 Results : Generating Text

Once the code is running smoothly, we need to create two batches of contexts to test the generated results and decode them with the database outlined in paragraph 2.1.1. We've chosen a batch size of 8 and a generation length of 50 tokens.

Below are some results with the prompt: "Out on one of the busiest Colorado highways north of Denver in 2009, there were 57 bicyclists found dead, and eight of those were cyclists, according to victim recovery agencies." The generation outcomes are displayed in Table 1.

It's essential to mention that the calculated perplexity applies to the two batches for which the generation was carried out. This should be compared with the perplexity of the *human golden reference* proposed by the paper (12.38) to ensure it falls within a reasonable range, neither too high nor too low.

In order to obtain less data-sensitive perplexity results, we decided to generate 100 tokens on 100 contexts (i.e. 12 matches) from the small-117M dataset, for different parameterizations of the techniques mentioned above. The results are shown in Table 2.

3 Nucleus Sampling for Directed Text Generation : Neural Machine Translation

3.1 Approach and Overview

In the context of Neural Machine Translation (NMT) – a directed text generation task – we wanted to adapt nucleus sampling to enhance the generation of contextually relevant translated sentences by allowing more diversity in the vocabulary used. The relevance of this approach lies in its ability to introduce controlled randomness, providing a more nuanced exploration of potential translations by dynamically adjusting the nucleus size. Contrary to the traditional decoding method – beam search – we can balance between deterministic and stochastic translation strategies, allowing for a richer exploration of the translation space, which beam search refrains the model from doing, being itself a deterministic decoding approach.

Parameters	Results	Perplexity
$k = 20$	The Colorado Department of Transportation said the fatalities involved the following incidents: - A bicyclist died after sustaining a head injury that struck him or her head on the road near Lakeland on Tuesday. A pedestrian was also killed following a traffic accident,	9.602
$k = 640$	Another cyclist, who never spent a day in jail, ended up living in a van behind Morrison Hill as a consolation prize in 2008, through last year. The judge said it went too far to call that irony. Justice Bjergsen's injunction against	29.857
$k = 40, t = 0.7$	A bicyclist was found dead in the backseat of a car on the same highway, near the Denver International Airport. Authorities say the body was not identified. The crash happened in the southbound lanes of Interstate 84 near the intersection of U	4.00
greedy	The Colorado Department of Transportation said it was investigating the deaths. The Colorado Department of Transportation said it was investigating the deaths. The Colorado Department of Transportation said it was investigating the deaths. The Colorado Department of Transportation said it was	2.73
$p = 0.5$	The two deaths were reported in Colorado Springs, where the Boulder County Sheriff's Office said a 19-year-old bicyclist was killed by a man in a stolen vehicle on Saturday.In Colorado Springs, the Sheriff's Office said two bicycl	5.33
$p = 0.95$	The scenario is ”nice to think of when you're growing up,” said Santa Fe Police Chief Jeff Garibaldi. Garcaldi said he and the department have heard that there have been a handful of crashes and a string of accidents involving	34.173
$p = 0.7$	The two deaths of bicyclists occurred at Route 24 and 52, near the University of Colorado Boulder campus.Those incidents occurred on Feb. 2, 2009, and May 6, 2009, respectively. Those incidents occurred in both years.	11.13
bs, $w = 16$	According to the Colorado Department of Public Safety, more than 1,000 bicyclists were killed in Colorado in 2009, and more than 1,000 bicyclists were killed in Colorado in 2010, according to the Colorado Department of Public Safety	2.16
original text	Surveillance video that showed injuries to 11 people was released today. Three women who died of forgotten injuries showed up drooling, bruising and constipation. Fifteen people, including the major victim, had poor treatment there – nine had no hearings or exams and the others had eaten nothing but grass and sweat.Officials originally had set up a memorial fire inside the home, but after it had become public early morning Tuesday night, they changed it to a cyanide fire that officials say was ”in response to inclement weather conditions. ”After taking the defective fire into custody, investigative officers made it clear that the blaze was simply a Good Samaritan speed bomb, not simply a traffic accident.	

Table 1: Results of generating 50 tokens for various settings. Repetitions, Incoherences.

To perform the adaptation of nucleus sampling to text translation, we chose to use the Helsinki-NLP/opus-mt-en-fr [1] model. This model is an english-to-french NMT model developed by the Language Technology Research Group at the University of Helsinki for the OPUS-MT project. This model is crafted for efficiency on regular desktops, and works well even for languages with limited resources, which gives to pre-trained models on small datasets the ability to output high quality translations, and thus be quite small. This is why it was chosen.

3.2 Setup and Issues Encountered

The implementation begins by loading the Helsinki-NLP/opus-mt-en-fr model from the Hugging Face Transformers Library, as well as its corresponding tokenizer. Then, random sentence pairs are loaded from the English-French News-Commentary v16 parallel corpus [2] dataset, and translations are generated using both nucleus sampling and

[1] https://huggingface.co/Helsinki-NLP/opus-mt-en-fr

[2] https://opus.nlpl.eu/News-Commentary-v16.php

Parameters	Perplexity results
$w = 16$	1.669
`greedy`	1.849
$k = 20$	7.40
$k = 40, t = 0.7$	3.889
$k = 80$	**12.8358**
$k = 250$	18.539
$k = 640$	26.377
$p = 0.5$	4.359
$p = 0.7$	8.054
$p = 0.8$	**12.178**
$p = 0.95$	27.617

Table 2: Results of generating 100 tokens for various settings.
*The **bold** perplexities are those closest to the golden human standard.*

beam search as decoding methods.

Evaluating the performance of the decoding methods is subsequently done by calculating the perplexity of the generated translations. However, in the context of machine translation, this introduces some challenges. Indeed, because the input text is in English and the output text in French, there is a discrepancy in the token probabilities between the input and the output. To tackle this issue, we opted to compute perplexity for the entire token list and comparing with respect to a ground truth translation, which we extracted from the News-Commentary v16 parallel corpus.

It is crucial to highlight a key distinction in our perplexity calculation approach compared to the previous approach in paragraph 2. While the latter method involved computing perplexity token-by-token by storing the token probabilities during the generation, and comparing its value to the one of a human golden standard, we opted for a holistic approach, considering the entire generated sequence in relation to its associated ground truth, comparing it to the value of the ground truth's perplexity. This method aligns with our adaptation of nucleus sampling, allowing us to evaluate translations in a more comprehensive manner. However, it also introduces a challenge in precisely assessing the quality of translations without human evaluation. The absence of direct grounding in human assessment necessitates careful consideration of the trade-offs between automated evaluations (such as perplexity) and human judgment to capture both the intricacies of language translation and the user's expectations.

Indeed, it is possible to find a translation that has a worse absolute difference perplexity – denoted ADP – between its perplexity and the ground truth's, while still being a better translation. Such an example can be found in Table 3, where it is obvious that the nucleus sampling translation is better (because closer to the target translation) than the beam search translation, the former only missing a comma. This is why there is a need for human assessment to complement automated evaluations.

Text Type	Text	Perplexity
Input English Sentence	And yet halting the payments did not stabilize German politics.	
Target French Translation	Et pourtant, l'arrêt des paiements n'a pas stabilisé la politique allemande.	1677.87
Beam Search ($b = 4$) Translation	Et pourtant, arrêter les paiements n'a pas stabilisé la politique allemande.	2063.07 ($ADP = 385.20$)
Nucleus Sampling ($p = 0.7$) Translation	Et pourtant l'arrêt des paiements n'a pas stabilisé la politique allemande.	3763.16 ($ADP = 2085.29$)

Table 3: Example of a worse absolute difference perplexity and a better translation

3.3 Results : Translating Text

As an attempt to introduce human judgment in the evaluation of the translation results, we decided to compute the BLEU [2] (BiLingual Evaluation Understudy) score, which is a metric (ranging from 0 to 1) used to measure the quality of machine-generated translations by comparing them to one or more reference translations. In our case, we computed the BLEU score on roughly a thousand sentences of the parallel corpus dataset.

As a result :

- Beam-Search-generated translations with $b = 4$ produced a BLEU score of 0.2866
- Nucleus-Sampling-generated translations with $p = 0.7$ produced a BLEU score of 0.2312

These findings indicate that there is minimal variation in translation quality between using Beam Search and Nucleus Sampling for the model, both falling within the lower end of the average range. Consequently, it appears that neither decoding technique significantly influences the quality of translations, as evidenced by the similar perplexities calculated for each. The choice between the two ultimately comes down to personal preferences.

4 Limitations and Refinements

The method outlined in the paper is quite straightforward, and its superiority in theory is apparent. However, upon examining their results more closely, it becomes evident that top-k sampling is also capable of yielding excellent outcomes, comparable to those obtained by nucleus sampling. An analysis of the variance of this perplexity (since we're calculating the mean of it) could account for the superiority of nucleus sampling in the different situations we presented in the introduction: a greater variance (which top-k would have compared to top-p) would be due to poor generation in the case of flattened or peaked distributions. Surprisingly, the authors did not present these types of results in their findings.

We encountered another significant issue during the analysis of results, particularly concerning the quality measures. While perplexity does offer a fair representation of the coherence of the output with the context, it doesn't capture the repetitiveness effectively. To address this, we require a reference, perhaps from the input dataset or a human-defined golden standard. However, the paper lacks detailed information on this aspect.

Furthermore, as mentioned in paragraph 3.2, perplexity is a measure that, in some instances, lacks meaningful interpretation, making it difficult to understand. This is why there is a need for human assessment to complete automatic assessments (such as perplexity). However, the metrics involving human evaluations used in the paper were either too computationally intensive, or too time costly. A satisfying first approach in this direction would nonetheless be to compute the BLEU score more thoroughly and on more parallel corpus datasets than in paragraph 3.3.

Conclusion

In both open-ended and directed text generation contexts, Nucleus Sampling yielded results that appeared satisfactory based on the instances we reviewed. While Nucleus Sampling demonstrated superior effectiveness in open-ended tasks overall, it was observed that top-k sampling, when appropriately fine-tuned, could achieve comparable outcomes. This similarity extends to directed generation tasks, where the decision between beam search and nucleus sampling for decoding appears to hinge on personal preferences. However, there is a necessity to develop a more robust method for evaluating the efficacy of this approach in comparison to existing methods. While metrics such as perplexity provide valuable insights, incorporating human-centric assessments and establishing standardized benchmarks for repetitiveness can provide a more nuanced understanding of the method's quality.

Contributions

We would like to acknowledge the contribution of the codebase used in this project, which was largely inspired by the implementation available at `https://github.com/ari-holtzman/degen`. The original code is associated with the work presented in the paper titled **THE CURIOUS CASE OF NEURAL TEXT *De*GENERATION** by Holtzman et al. [1].

The code used to conduct experiments in relation to nucleus sampling in machine translation – related to the project's GitHub subdirectory `nmt_nucleus_sampling` – on the contrary, is original code.

It should be noted that the code used to download the datasets provided by OpenAI for our context is also provided by OpenAI and is taken from the associated GitHub repository.

Chat-GPT played a key role in comprehending the code and its intricacies, as well as rectifying any bugs or errors that surfaced during the rewriting process. To implement specific ideas, Chat-GPT, along with official documentation, proved valuable for code creation – helping in identifying the right functions and ensuring the coherence of certain code sections.

Certain sections of the code underwent a complete rewrite, particularly the decoding functions. The concept of expanding Nucleus Sampling for directed text generation with translation originated as our own idea.

References

[1] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2020.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (P. Isabelle, E. Charniak, and D. Lin, eds.), (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.