Implicit Unlikelihood Training: Improving Neural Text Generation with Reinforcement Learning

Anonymous COLING submission

Abstract

Holtzman et al. (2019) showed that likelihood training and maximization-based decoding result in dull and repetitive generated texts even when using powerful language models. Adding a loss function for regularization was shown to improve text generation output by helping avoid unwanted properties, such as contradiction or repetition (Li et al., 2020). In this work, we propose fine-tuning a language model by using policy gradient reinforcement learning, directly optimizing for better generation. We apply this approach to minimizing repetition in generated text, and show that, when combined with unlikelihood training (Welleck et al., 2020), our method further reduces repetition without impacting the language model quality. We also evaluate other methods for improving generation at training and decoding time, and compare them using various metrics aimed at control for better text generation output.

1 Introduction

Language models have become a subject of close attention in the Natural Language Processing field over the past few years. They are widely used not only for unsupervised pre-training, but also for text generation, such as what is implemented in dialogue systems (Roller et al., 2020). While there are ongoing efforts to develop non-autoregressive models for language modeling, most current state-of-the-art approaches use the autoregressive method of generating text (i.e., word by word). Holtzman et al. (2019) showed that even powerful trained models with a high likelihood value for test data can output repetitive results. Schmidt (2019) argues that the reason for that is train-test discrepancy and lack of generalization when running standard maximum likelihood estimation (MLE) training.

Unwanted repetition can be remedied at decoding and training time. Decoding methods focus on sampling techniques that generate less repetitive or incoherent samples, while other methods aim to improve model training to minimize the effects of degeneration. An effective method for reducing language model degeneration is unlikelihood training (Welleck et al., 2020), where a regularization term forces the model to reduce the probability of generating a token that has already occurred in a sequence. Li et al. (2020) further explored this idea and showed that adding a loss function for regularization *to avoid undesirable sequences* improves text generation not only by reducing repetition, but also by decreasing contradiction. Roller et al. (2020) reported that adding unlikelihood training also improves the humanness of generated text.

In this paper, we propose **Implicit Unlikelihood Training**, a method for regularizing output by fine-tuning a language model with policy gradient reinforcement learning to improve generation results. We apply this method for a repetition objective, and show that combining Implicit Unlikelihood Training with minimizing unlikelihood loss results in reduced repetition and perplexity. We also evaluate alternative approaches to improving generated texts in terms of repetitiveness, and compare these methods using a wide variety of metrics.

2 Related Work

2.1 Decoding Strategies

Holtzman et al. (2019) observed that maximization-based decoding methods, such as top-k sampling

(Fan et al., 2018), beam search and its variations, can all lead to degeneration. They addressed this problem by using top-p (nucleus) sampling, proposing sampling from the top portion of the probability mass. Paulus et al. (2017) reported that ground-truth sentences for summarization tasks almost never contain the same trigram twice, and proposed the beam-blocking approach, where the decoder is forced to never output the same trigram more than once during testing. Penalized sampling (Keskar et al., 2019) works by discounting the scores of previously generated tokens. Martins et al. (2020) proposed preventing unlikely words from receiving any probability mass by using entmax sampling.

2.2 Training Strategies

Jiang et al. (2020) suggested that some tokens can be more difficult for a model to learn than others. These tokens are still under-learned after training, making their repetition more likely to happen. This issue is addressed by token loss dynamic reweighting (TLDR), which applies differentiable weights to individual token losses. Repetition can also be improved at training time by adding unlikelihood loss (Welleck et al., 2020; Li et al., 2020) to regular likelihood loss. Unlikelihood training is aimed at decreasing the probability of previously generated tokens, and it was shown that it can outperform beam blocking and top-p sampling.

Coverage mechanisms (Tu et al., 2016; See et al., 2017) can also be used to reduce repetition. Adding pre-attention and highway connections was shown to decrease repetition for RNNs (Jiang et al., 2020), while the architecture tweaks required for Transformers (Vaswani et al., 2017) are still an open question.

2.2.1 Unlikelihood Training

Unlikelihood Training involves adding Unlikelihood Loss to lower the probability $p_\theta(c_i|x_{<t})$ of negative candidates $\mathcal{C}^t = \{c_1, c_2, ..., c_n\}$ at each timestamp:

$$\mathcal{L}_{\mathrm{UL}}^t \left(p_\theta \left(\cdot | x_{<t} \right), \mathcal{C}^t \right) = - \sum_{c \in \mathcal{C}^t} \log \left(1 - p_\theta \left(c | x_{<t} \right) \right). \tag{1}$$

We can construct the negative candidate set as $\mathcal{C}^t = \{x_{t-1}, x_{t_2}, ..., x_1\} \backslash \{x_t\}$ to improve generation results through reducing repetition. Welleck et al. (2020) also proposed using Sequence-Level Unlikelihood Loss on sampled continuations, where a sequence $(x_{t+1}, x_{t+2}, ..., x_{t+N})$ from a prefix $(x_1, x_2, ..., x_t)$ is sampled first, and then the loss defined in Eq. 1 for each x_{t+i} (where $1 \le i \le N$) with negative examples $\mathcal{C}^{t+i}$ equal to $\{x_{t+i}\}$ if x_{t+i} is a part of a repeating n-gram at a position before $t + i$ is minimized (see **Appendix A. Algorithm Details** for details on Sequence-Level Unlikelihood Loss). Fine-tuning a language model is then performed by equally alternating between sequence-level unlikelihood and likelihood updates.

2.3 Evaluation Metrics

There are many metrics available for evaluating the performance (diversity or non-repetition) of language models. These metrics include the number of unique next-token predictions (**uniq**), **uniq-seq**, **rep/l**, **wrep/l** (Welleck et al., 2020), ϵ-**perplexity**, **sparsemax score** (sp), Jensen-Shannon divergence (**JSD**) (Martins et al., 2020), and **DIMEN** (Jiang et al., 2020). The metrics we used are described in detail in **Appendix C. Evaluation Metrics**.

In this paper, we mainly focused on reducing sequence repetition (**seq_rep_n**) (Welleck et al., 2020), which is a portion of duplicate n-grams in a generated sequence. Improving generation results by minimizing repetition should not significantly affect the **perplexity** of the language model.

3 Implicit Unlikelihood Training

Li et al. (2020) showed that Unlikelihood Training can be employed as a general framework for reducing the likelihood of undesirable text generation results through training on negative examples. However, we argue that, in some cases, it could be difficult to construct negative samples for specific types of

method	ppl↓	uniq↑ top-k, k=1	seq_rep_4↓				
			top-k, k=1	top-k, k=3	top-k, k=8	top-p, p=0.3	top-p, p=0.9
small GPT-2 model, 0.5 update rate							
PG, c=3	19.409±.195	11259±52	.032±.014	.024±.010	.017±.006	.029±.012	.007±.001
PG + UT, c=3	19.344±.034	11279±53	.010±.002	.008±.001	.008±.000	.012±.002	.007±.001
i-UT, c=3	19.182±.032	11308±73	.009±.002	.008±.001	.008±.001	.012±.002	.007±.001
i-UT, c=9	19.302±.051	11297±36	.006±.001	.006±.001	.007±.002	.009±.002	.006±.001
i-UT, c=15	**19.170±.123**	**11432±34**	.007±.005	.007±.002	**.007±.001**	.010±.003	.007±.001
i-UT, c=30	19.504±.065	11427±66	**.005±.000**	**.005±.001**	**.007±.001**	**.005±.001**	**.006±.001**
UT	19.442±.085	11210±47	.056±.033	.011±.002	.008±.001	.014±.003	**.006±.001**
UT w/ warmup	19.347±.065	11295±4	.055±.016	.010±.001	.008±.001	.014±.002	**.006±.001**
small GPT-2 model, 0.25 update rate							
PG, c=3	18.516±.086	11492±32	.060±.011	.035±.005	.022±.003	.043±.006	.009±.001
PG, c=9	18.638±.022	11496±21	.025±.011	.022±.008	.018±.005	.025±.008	.009±.001
PG, c=15	18.696±.062	11533±3	.022±.002	.019±.001	.016±.000	.021±.001	.008±.001
PG, c=30	19.026±.099	11487±36	.031±.021	.027±.017	.020±.010	.027±.016	.008±.001
i-UT, c=3	18.504±.059	11519±21	.020±.002	.014±.001	.012±.001	.017±.002	**.007±.001**
i-UT, c=9	**18.487±.022**	11552±51	.011±.003	.011±.003	.011±.002	.016±.003	**.007±.000**
i-UT, c=15	18.590±.065	11504±24	**.008±.001**	**.008±.001**	**.008±.001**	.011±.002	**.007±.001**
i-UT, c=30	18.733±.082	**11558±27**	**.007±.001**	**.008±.001**	.009±.001	**.010±.001**	.008±.000
UT	18.764±.164	11377±55	.055±.013	.011±.002	.009±.001	.016±.003	**.007±.001**
medium GPT-2 model, 0.5 update rate							
i-UT, c=3	**13.620±.015**	12437±29	.013±.002	.010±.002	.010±.001	.011±.002	.008±.001
i-UT, c=15	13.669±.018	12355±3	**.011±.003**	.010±.002	.011±.002	.012±.001	.009±.000
i-UT, c=30	13.785±.004	**13319±898**	**.011±.003**	.009±.003	.011±.001	**.010±.004**	**.008±.000**
UT	13.710±.009	12386±29	.024±.003	**.008±.001**	**.009±.001**	.013±.001	.008±.001

Table 1: Repetition on small and medium GPT-2 models. Validation data was used as sampling prefixes for evaluating the metrics.

Unlikelihood Loss[1]. To address this issue, we propose extending Unlikelihood Training with policy gradient reinforcement learning, which does not need explicitly created negative samples.

We chose to test this approach for repetition as the most widely considered property of neural text degeneration. To directly minimize repetition (see Equation 2) for sequence x, we define the reward as $R = 1 - \text{seq\_rep\_n}(\mathbf{x})$ with $n = 4$. We alternated between maximizing the reward R, optimizing the likelihood of training data, and Sequence-Level Unlikelihood Loss (see **Appendix A. Algorithm Details** for details on the process of Implicit Unlikelihood Training and policy gradient update).

4 Experiment Details

4.1 Setup

We fine-tuned small and medium GPT-2 models (175M and 345M parameters, respectively) (Radford et al., 2019) on the WikiText-103 dataset (Merity et al., 2016).

Our experiments consisted of alternating between three types of updates: Maximizing Likelihood (MLE), minimizing Sequence-Level Unlikelihood Loss (UL), and minimizing repetition with policy gradient reinforcement learning (PG). The first approach is a plain **MLE** update, for which we do not use any specific methods for reducing repetition in samples. We also experimented with Unlikelihood Training (**UT**), which involves alternating between MLE and UL updates.

In policy gradient experiments, we trained models in three different scenarios: a plain **PG** for which we alternated MLE and PG updates; a combined **PG + UT** approach, where we alternated between maximizing the likelihood and minimizing the sum of policy gradient and unlikelihood losses; and finally,

---

[1]This can include reducing the toxicity or bias level of generated sequences by using a score from an external classifier.

sampling	method	uniq↑	rep↓	wrep↓	$\epsilon-$ppl/ppl↓	JSD↓	sp↑
softmax $\epsilon = 0$	MLE	19932	.373	**.174**	13.830	.382	.680
	i-UT, c=3	**20493**	.372	**.174**	**13.161**	.379	**.683**
	UT	20419	.371	**.174**	13.266	.380	.682
greedy $\epsilon = 2 \times 10^{-5}$	MLE	12639	.489	.230	539.930	.358	.483
	i-UT, c=3	12859	.488	.228	511.478	**.355**	.488
	UT	12826	.488	.228	517.316	.356	.487
top-k, k=10 $\epsilon = 8 \times 10^{-6}$	MLE	14326	.436	.220	50.003	.358	.668
	i-UT, c=3	14731	.435	.219	46.885	**.355**	.672
	UT	14648	.435	.219	47.248	.356	.671
top-p, p=0.9 $\epsilon = 2 \times 10^{-6}$	MLE	17589	.395	.186	19.689	.371	.678
	i-UT, c=3	18116	.394	.185	18.662	.368	.682
	UT	17964	.393	.186	18.782	.369	.681
α-entmax ($\alpha = 1.2$) $\epsilon = 1 \times 10^{-6}$	α-entmax loss ($\alpha = 1.2$)	19942	**.370**	.176	15.124	.389	.680

Table 2: Repetition on the medium GPT-2 model with a 0.5 update rate. Regular perplexity is reported for softmax sampling. Test data is used as sampling prefixes for evaluating the metrics.

the proposed Implicit Unlikelihood Training (**i-UT**), which consisted of alternating between MLE, UL and PG updates. We used 0.25 and 0.5 alternating update rates (r in **Appendix A. Algorithm Details)** for the small GPT-2 model, and r equal to 0.5 for the medium GPT-2 model.

Full optimization details are provided in **Appendix B. Optimization Details**.

4.2 Evaluation

We used top-k and top-p samplings with different k and p to evaluate sequence repetition[2] (seq_rep_4) for the described approaches. For these experiments, we used validation data to evaluate perplexity and to generate the sampling prefixes for evaluating uniq and seq_rep_4 metrics. The number of unique tokens (uniq) was evaluated using greedy sampling. We also evaluated the proposed method with rep, wrep, and JSD metrics using different sampling methods on test data, and compared it with other related approaches (MLE, UT, entmax). We repeated each experiment 5 times and reported the mean and standard deviation values of the measurements.

More experiments and their results are described in **Appendix D. Experiments**.

5   Results

We showed that Implicit Unlikelihood Training is a competitive approach that outperforms other methods in sequence repetition when fine-tuning small and medium GPT-2 models (see Table 1) on most variants of top-k and top-p sampling, while maintaining the lowest perplexity and the highest count of unique tokens generated. This approach also achieved better results than training with entmax loss and other related approaches, using a different range of sampling methods (see Table 2), with the only exception being the rep metric, where entmax performed similar to i-UT.

Samples of generated outputs are provided in **Appendix F. Sample Generation Output**.

We also described approaches we tried and found ineffective in **Appendix E. Negative Results**.

6   Future Work

The described and evaluated reinforcement learning framework makes it possible to optimize text generation for any objective. In future work, we intend to test the approach not only for repetition, but also for various other metrics, such as the toxicity level or bias of generated text.

---

[2]Note that top-k with $k = 1$ is greedy sampling by definition.

References

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*, page 889–898.

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

Shaojie Jiang, Thomas Wolf, Christof Monz, and Maarten de Rijke. 2020. Tldr: Token loss dynamic reweighting for reducing repetitive utterance generation. In *arXiv preprint arXiv:2003.11963*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, , and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. In *arXiv preprint arXiv:1909.05858*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2020. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In *ACL*.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2020. Sparse text generation. In *arXiv preprint arXiv:2004.02644*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *arXiv preprint arXiv:1609.07843*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *arXiv preprint arXiv:1705.04304*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI Blog*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau, and Jason Weston. 2020. Recipes for building an open-domain chatbot. In *arXiv preprint arXiv:2004.13637*.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *EMNLP Workshop on Neural Generation and Translation (WNGT)*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. In *arXiv preprint arXiv:1506.02438*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, page 1073–1083.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Association for Computational Linguistics*.

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.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

A   Algorithm Details

Algorithm 1: Implicit Unlikelihood Training:
alternating between MLE, UT and PG updates

Input: update rate r, total number of updates N
for $i = 1\ to\ N$ **do**
    sample $x \sim \mathrm{U}[0, 1]$
    if $x < r$ **then**
        sample $y \sim \mathrm{U}[0, 1]$
        if $y < 0.5$ **then**
          | do a policy gradient update
        else
          | do a sequence-level unlikelihood update
        end
    else
      | do a MLE update
    end
end

Algorithm 2: Policy Gradient Update

Input: LM θ, m prefixes $\mathcal{D}_m = \{(x_1^{(j)}, \ldots, x_k^{(j)}),\ j = \overline{1..m}\}$, continuation length T
Output: loss $\mathcal{L}(\theta, \mathcal{D}_m)$
for $j = 1\ to\ m$ **do**
    for $t = k + 1\ to\ k + T$ **do**
        Get $p_\theta(\cdot \mid x_{<t}^{(j)})$
        $x_t^{(j)} = \arg\max\limits_{x \in \mathcal{V}} p_\theta(\cdot \mid x_{<t}^{(j)})$
    end
end
for $j = 1\ to\ m$ **do**
    | $R_j = 1 - \mathrm{seq\_rep\_n}(\mathbf{x}^{(j)})$
end
$b(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}) = \sum\limits_{j=1}^{m} R_j$
for $j = 1\ to\ m$ **do**
    | $\Psi_j = R_j - b(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}) \cdot \frac{1}{m}$
end
$\mathcal{L}(\theta, \mathcal{D}_m) = -\frac{1}{m} \sum\limits_{j=1}^{m} \Psi_j \cdot \frac{1}{T} \sum\limits_{t=k+1}^{k+T} \log p_\theta(x_t^{(j)} \mid x_{<t}^{(j)})$

B   Optimization Details

For likelihood update, we evaluated the likelihood of a sequence of tokens with lengths equal to 300. For both UL and PG updates, we formed prefixes using a sequence of 300 tokens to form 6 sequences with lengths equal to 50. We then used these prefixes to sample sequences with a maximum length of 100 tokens.

For optimization, we used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 6.25×10^{-5}. Similar to (Welleck et al., 2020) and (Martins et al., 2020), we did no warmup steps for UT and $\alpha-$entmax training. For i-UT, we did 500 linear warm-up steps . After warm-up steps, we linearly decayed the learning rate to zero.

In all our experiments, we fine-tuned language models for 5000 total updates.

Once training was complete, we selected a checkpoint with the least validation perplexity obtained during training. This is the last checkpoint in most of our experiments, which means that general log-likelihood loss converges.

As shown in Algorithm 1, we equally alternated between UL and PG updates. We also found that reducing unlikelihood update rate to 0.25 may also be effective, taking twice less time (see Table 1). The

Algorithm 3: Sequence-Level Unlikelihood Update

Input: LM θ, batch of prefixes $\mathcal{D}_m = \{(x_1^{(j)}, \ldots, x_k^{(j)}), \ j = \overline{1..m}\}$, continuation length T
Output: loss $\mathcal{L}(\theta, \mathcal{D}_m)$
for $j = 1$ *to* m **do**
    for $t = k + 1$ *to* $k + T$ **do**
        Get $p_\theta(\cdot \mid x_{<t}^{(j)})$
        $x_t^{(j)} = \arg\max\limits_{x \in \mathcal{V}} p_\theta(\cdot \mid x_{<t}^{(j)})$
    end
end
for $j = 1$ *to* m **do**
    for $t = k + 1$ *to* $k + T$ **do**
        if $(x_{t-i}, \ldots, x_t, \ldots, x_{t+h}) \in x_{<t-i}^{(j)}$ *for any* $(h - i) = n, i \leq n \leq h$ **then**
            $\mathcal{C}_{\text{repeat-n}}^t(\mathbf{x}^{(j)}) = \{x_t\}$
        else
            $\mathcal{C}_{\text{repeat-n}}^t(\mathbf{x}^{(j)}) = \emptyset$
        end
    end
end

$$\mathcal{L}(\theta, \mathcal{D}_m) = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{T} \sum_{t=k+1}^{k+T} \mathcal{L}_{UL}^t \left(p_\theta(\cdot \mid x_{<t}^{(j)}), \mathcal{C}_{\text{repeat-n}}^t \right)$$

---

parameters ϵ for $\epsilon - ppl$ and α for $\alpha - \text{entmax}$ training were taken from (Martins et al., 2020) (except ϵ for $\alpha - \text{entmax}$, which we set to 1×10^{-6}).

We conducted coefficient search on our policy gradient loss with $c = \{3, 9, 15, 30\}$ for the small GPT-2 model, and $c = \{3, 15, 30\}$ for medium GPT-2 model. We chose the best models based on the results on the validation set, and also reported the metrics on the test set.

C   Evaluation Metrics

Perplexity is the metric used to evaluate language model quality. It is defined as $ppl(x) = p(x_1, x_2, ..., x_t)^{-\frac{1}{t}}$, where $x_1, x_2, ..., x_t$ is the sequence of tokens from test data. The lower the perplexity, the better the language model.

Welleck et al. (2020) used a portion of duplicate n-grams in a generated sequence to measure **sequence repetition**:

$$\text{seq\_rep\_n}(\mathbf{x}) = 1 - \frac{\#\text{uniq\_ngram}(\mathbf{x})}{\#\text{total\_ngram}(\mathbf{x})}. \tag{2}$$

Higher repetition values mean that a language model tends to produce more repetitive output, which might appear less natural. Note that $0 \leq \text{seq\_rep\_n}(\mathbf{x}) < 1$.

Same as (Welleck et al., 2020), we controlled for the number of unique next-token predictions (**uniq**), as it was shown that generated texts are less diverse than those written by a human. We also used the number of unique tokens in continuations of validation or test prefixes (**uniq-seq**) as a measure of token distribution in generated text. **rep/l** is the fraction of next-token (top-1) predictions that occur in the previous l tokens. **wrep/l** is a variant of rep/l which only counts single token repetitions that are not equal to the ground truth next-token. We use 16,32,128,512 as l and average the results to compute rep and wrep.

Martins et al. (2020) introduced ϵ-**perplexity** for computing perplexity of sparse distributions. The perplexity is smoothed by adding a small value of ϵ to all terms, followed by renormalization. They also introduced **sparsemax score** (sp) and **Jensen-Shannon divergence** (JSD) for evaluating quality and sparsity of probability distributions. For deterministic models, sparsemax score becomes word accuracy, and is bounded by 0 and 1. With JSD, the distance between the sparse or truncated distribution and the one-hot encoded ground truth distribution can be measured. It is used as a metric for language models using different decoding strategies. Unlike perplexity, JSD is bounded by $\log 2$.

Jiang et al. (2020) evaluate methods with a diversity metric based on n-grams (**DIMEN**). A high DIMEN score means that a set of generated sequences is diverse.

D Experiments

We evaluated DIMEN and uniq-seq for UT and i-UT methods, applied to small and medium GPT-2 models using different sampling methods for DIMEN, and greedy sampling for evaluation of uniq-seq. In this experiment, we observed that Implicit Unlikelihood Training performed better or equal to Unlikelihood Training with different sampling methods measured by the DIMEN metric, having a significantly better value of uniq-seq (see Table 7).

We also evaluated sequence repetition with beam-search sampling for MLE, UT, and i-UT methods for both small and medium GPT-2 models, using validation data to form sampling prefixes. When sampling with beam search, we found that Implicit Unlikelihood Training produced better results than Unlikelihood Training (see Table 3).

For greedy sampling with small GPT-2 model, we evaluated sequence repetition, wrep, uniq, and perplexity. We used test data to evaluate the perplexity, and to form sampling prefixes for other methods. We observed that MLE, UT, and i-UT methods had similar performance in terms of repetition using greedy sampling, while i-UT still had the best number of unique tokens (see Table 4).

Finally, we evaluated the TLDR method using both sequence repetition and DIMEN metrics (see Tables 5, 6). In our experimental setup, TLDR performed on par with MLE approach.

GPT-2:	seq_rep_4↓		
	small		medium
seq rate:	.5	.25	.5
MLE	.67±.01	.67±.01	.64±.01
i-UT	**.03±.04**	**.08±.03**	**.08±.02**
UT	.08±.03	.14±.02	.13±.02

Table 3: Repetition with Beam Search. Validation data is used as sampling prefixes for evaluating the metrics. We reported the results of i-UT model with the value of c for which we had the best validation perplexity.

method	ppl↓	rep↓	wrep↓	uniq↑
MLE	17.94±.03	.504±.001	.252±.001	11790±92
i-UT, c=15	18.54±.13	.504±.001	.254±.0	11847±32
UT	18.76±.09	.503±.001	.253±.001	11597±52

Table 4: Repetition on small GPT-2 model, UT and i-UT 0.5 update rate with greedy sampling. Test data is used as sampling prefixes for evaluating the metrics.

method	ppl↓	uniq↑ top-k, k=1	seq_rep_4↓				
			top-k, k=1	top-k, k=3	top-k, k=8	top-p, p=0.3	top-p, p=0.9
MLE	18.611±.088	11361±37	.550±.035	.131±.004	.054±.001	.233±.005	.013±.001
TLDR	18.713±.059	11322±42	.514±.012	.118±.004	.047±.002	.241±.005	.011±.001

Table 5: Repetition on small GPT-2 model for TLDR and MLE approaches. Validation data is used as sampling prefixes for evaluating the metrics.

method	uniq-seq↑ top-k, k=1	DIMEN↑				
		top-k, k=1	top-k, k=3	top-k, k=8	top-p, p=0.3	top-p, p=0.9
MLE	8074.75±182.398	.365±.023	.685±.003	.778±.001	.599±.004	.869±.001
TLDR	8122.0±112.018	.390±.008	.692±.004	.781±.002	.590±.004	.867±.001

Table 6: Diversity on small GPT-2 model for TLDR and MLE approaches. Validation data is used as sampling prefixes for evaluating the metrics.

method	uniq-seq↑ top-k, k=1	DIMEN↑				
		top-k, k=1	top-k, k=3	top-k, k=8	top-p, p=0.3	top-p, p=0.9
small GPT-2 model, 0.5 update rate						
i-UT, c=3	**11340**±**703**	.855±.008	.859±.005	.859±.003	.834±.005	.880±.003
i-UT, c=9	10547±533	**.881**±**.014**	.876±.003	.870±.004	.852±.0	**.881**±**.003**
i-UT, c=15	10621±489	.863±.023	.867±.013	.864±.006	.855±.014	**.881**±**.002**
i-UT, c=30	10771±265	**.881**±**.009**	**.880**±**.008**	**.871**±**.007**	**.880**±**.012**	.880±.003
UT	9651±436	.785±.044	.847±.012	.856±.005	.831±.011	.880±.002
small GPT-2 model, 0.25 update rate						
i-UT, c=3	10885±375	.817±.006	.834±.003	.845±.002	.819±.003	.878±.002
i-UT, c=9	11208±884	.852±.014	.849±.01	.851±.007	.828±.01	.879±.001
i-UT, c=15	**11966**±**1019**	.848±.012	.854±.006	.856±.004	.837±.006	.879±.001
i-UT, c=30	10418±762	**.874**±**.012**	**.871**±**.01**	**.863**±**.007**	**.863**±**.017**	.878±.002
UT	9696±441	.778±.017	.843±.007	.854±.003	.827±.008	**.880**±**.002**
medium GPT-2 model, 0.5 update rate						
i-UT, c=3	**12900**±**303**	.843±.006	.854±.005	.857±.004	.845±.006	**.877**±**.002**
i-UT, c=15	13849±0	.869±.003	.861±.002	.857±.003	.854±.0	.875±.001
i-UT, c=30	11748±340	**.880**±**.004**	**.871**±**.01**	**.861**±**.006**	**.867**±**.009**	**.877**±**.002**
UT	12649±262	.837±.006	.869±.004	.860±.003	.842±.005	.875±.001

Table 7: Diversity on small and medium GPT-2 models. Validation data is used as sampling prefixes for evaluating the metrics.

E Negative Results

Our results showed that all sampling methods, other than greedy sampling, led to worse convergence of the seq_rep_4 metric.

We experimented with using the Proximal Policy Optimization algorithm (Schulman et al., 2017) for PG update, but faced unstable validation perplexity behavior during training, and did not obtain any comparable results.

Another unsuccessful direction of our experiments was substituting the estimation of the reward calculated on the full sequence with the reward put on each token separately. We tried using two variants of the binary reward function: does the current n-gram appear first time in the text, and does the current n-gram appear in the following part of the text. We experimented with advantage estimation by using a value function estimator, and without it by using pure rewards. In the former case, we adjusted different values of λ and γ for the Generalized Advantage Estimation algorithm (Schulman et al., 2015), and in the latter, we used a general discounted future reward. We observed that the approach of estimating a single reward for a whole sequence and subtracting a baseline value to reduce the variance of the gradient estimation performed best.

F Sample Generation Output

		seq_rep_4
prefix:	Air Vice Marshal Frank Headlam, CB, CBE (15 July 1914 – 23 December 1976) was a senior commander in the Royal Australian Air Force (RAAF). Born and educated in Tasmania, he joined the RAAF in	
	medium GPT-2 model, 0.5 update rate, top-k, k=1	
MLE	1914, and was promoted to the rank of CB. He was promoted to the rank of CBE in 1916, and was promoted to the rank of CB. He was promoted to the rank of CBE in 1917, and was promoted to the rank of CB. He was promoted to the rank of CBE in 1918, and was promoted to the rank of CB. He was promoted to the rank of CBE in 1919, and was promoted to the rank of CB. He was promoted to	.6535
UT	1914 and was promoted to the rank of lieutenant in 1916. He was promoted to commandant in the Australian Air Force (R @-@ Special Operations Command (C-MEDF) in spare parts for related acronym changed to <unk>) Special Operations Command (Special Commander in Command by extension) being a technical specialist in communications equipment and equipment related to intelligence gathering and analysis, and related related related jargon, and related related related equipment related to logistics and logistics management. Headlam	.0288
i-UT	1914 and served in the Australian Army Air Corps (AAF) from 1915 to 1918. Headlam was promoted to the rank of lieutenant colonel in 1918 and promoted to the rank and title of commander in the Pacific theatre in the Pacific theatre in (R & R) <unk> in order to be eligible for the Medal of Honor for gallantry in action in the Battle of the Gulf, but the award was not awarded until the end of the war. Headley was awarded the Distinguished Service	.0189
	medium GPT-2 model, 0.5 update rate, top-k, k=8	
MLE	1917 at the age of twenty @-@ seven, and served at both the Western and Eastern Airports, and the Western Reserve Air Training Center. He was awarded a Distinguished Service Order in 1917, and a Military Medal for his service in World War I, and a Distinguished Service Order for his services during the Great Depression. In 1920 he was appointed as Chief of the Staff of the RAAF and served as the RAAF's Chief of Staff. In 1921 he was appointed a	.0211
UT	1939 as a private with the rank of Captain, serving on board Australian carriers, such as the RAN @-@ class battleships HMAS Melbourne and HMAS Sydney. In the early years of the war the Australian navy's air force, with the exception of the RAN, was under the command and control of Rear @-@ Admiral Robert W. Campbell, the commander of the RAAF in Australia. During the Second World War, Headlam became a member of the RAN	.0105
i-UT	1917 as a private and served in the Second World War. Headlam became the second man in the RAAF to be awarded the Victoria Cross in December 1941 for valour and gallantry in a landmine @-@ exploding raid on a German convoy in the Gulf of Mexico. Headlam became Commander of the Royal Australian Air Force in July 1943 after serving as a Lieutenant in the RAAF from 1917 to 1919 and then as Commander of the RAAF in the First World War and	.0104

	seq_rep_4
prefix:	Air Vice Marshal Frank Headlam, CB, CBE (15 July 1914 – 23 December 1976) was a senior commander in the Royal Australian Air Force (RAAF). Born and educated in Tasmania, he joined the RAAF in

medium GPT-2 model, 0.5 update rate, top-p, k=0.9	
MLE	1940 as a gunnery instructor and officer at Hobson Air Station, Melbourne. On 3 January 1940 he led a small squad that formed part of the Caraboo Air Force (the first to be taught at a Melbourne naval base) that did air training and drills at the Haro Air Force Base. Headlam was later commanding the 200th (First United States Army Air Service Group, Unit A) and Unit B (the second to be taught at a Melbourne naval base). In the	.0526
UT	December 1914 at the age of 15 years and 177 days. Headlam served with the RAAF and the Italian Free Fire Division before leaving in the middle of the War in October 1917 to become a field marshal of the Melbourne Transport Service. In 1919 he was nominated for a Distinguished Service Order (DSO) and he retired as a Field Marshal in December 1918. After the War he served in the RAAF as a field marshal until leaving to become an air force field marshal in Headlam	.0104
i-UT	October 1914 as a radio operator and served with distinction until his discharge from the RAAF in December. In February 1915, Headlam was appointed to the role of President of the RAC, and a policy advisory officer to the PM. He was promoted to the rank of Chief of the Air Staff, and underlined that he was a major and not to have the responsibility of determining the combatant classes (like the Royal Australian Flying Corps). When the RAC was created in	.0