NEURAL TEXT ~~DE~~GENERATION WITH UNLIKELIHOOD TRAINING

Sean Welleck[1,2*] **Ilia Kulikov**[1,2*] **Stephen Roller**[2] **Emily Dinan**[2]

Kyunghyun Cho[1,2,3] **& Jason Weston**[1,2]
[1]New York University, [2]Facebook AI Research, [3]CIFAR Azrieli Global Scholar

ABSTRACT

Neural text generation is a key tool in natural language applications, but it is well known there are major problems at its core. In particular, standard likelihood training and decoding leads to dull and repetitive responses (Holtzman et al., 2019). While some post-hoc fixes have been proposed, in particular top-k and nucleus sampling, they do not address the fact that the token-level probabilities predicted by the model itself are poor. In this paper we show that the likelihood objective itself is at fault, resulting in a model that assigns too much probability to sequences that contain repeats and frequent words unlike the human training distribution. We propose a new objective, *unlikelihood training*, which forces unlikely generations to be assigned lower probability by the model. We show that both token and sequence level unlikelihood training give less repetitive, less dull text while maintaining perplexity, giving far superior generations using standard greedy or beam search. Our approach provides a strong alternative to traditional training.

1 INTRODUCTION

Neural text generation is a vital tool in a wide range of natural language applications. However, the standard approach – training a sequence to sequence model, e.g. Transformer (Vaswani et al., 2017), to maximize log-likelihood and approximately decoding the most likely sequence from it – is known to be fundamentally flawed. The generated text in open-ended applications such as language modeling or dialogue has been observed to be dull, using high frequency tokens too often and interesting content words too rarely (Holtzman et al., 2019; Dinan et al., 2019). Moreover, the models repeat themselves at the token, phrase and sentence level to the point of monotony. Comparison between simple statistics collected from a training set of human-generated utterances and model-generated responses shows the discrepancy between them. This does not appear to be rectified by simply training on more data (Radford et al., 2019). The current fix is to modify the decoding strategy using either more sophisticated beam search variants or sampling strategies. However, these can be considered band-aid solutions rather than getting to the root of the problem, as the model's underlying predicted token probabilities are clearly not correct.

Several reasons for why exactly neural text is degenerate have been posited, with the cause currently unknown. Possible candidates include the problem being (i) a by-product of the model architecture choices, e.g. the Transformer attention architecture preferring repeats (Holtzman et al., 2019; Vig, 2018), (ii) an intrinsic property of human language (Holtzman et al., 2019) rather than a modeling deficiency, or that (iii) a training objective relying on fixed corpora cannot take into account the real goal of using the language (Choi, 2018). Our work shows that, while the above may be factors, a primary factor is actually the use of the likelihood objective itself, as we demonstrate that this issue is alleviated if we replace the likelihood objective with our proposal.

While low perplexity in the limit should lead to predicting the correct next target word, there are two major flaws of the likelihood objective: (i) it pays relatively little attention to the argmax or the top of the ranked list of next token probabilities, instead optimizing the likelihood of the entire distribution; (ii) it is not focused on optimizing sequence generation, only on producing the next

[*]Equal contribution; the ordering was decided by a coin flip.

token. The first issue means that greedy or beam search decoding, which rely on the top of the list to generate, are not optimized – there is a discrepancy between maximizing the log-probability of a ground-truth token and ensuring the rank of the ground-truth token to be one. The second issue means poor token choices early on in generation make the results even worse, leading to a vicious circle – any imperfection of next token prediction leads to error accumulation in sequence generation, and likelihood training does not have a way to address this.

In this work, we introduce *unlikelihood training*, an approach that addresses the two aforementioned issues. It works by combining two types of updates: a likelihood update on the true target tokens so they are assigned high probability, and an *unlikelihood* update on tokens that are otherwise assigned too high a probability. As we can collect these *unlikely* token candidates from either token level generation or sequence level generation, we can train our model at the sequence level as well. Both token and sequence level unlikelihood training are shown to improve metrics that measure dullness and repetition of the model, while maintaining the performance in other metrics such as perplexity or token accuracy compared to the baseline. The final generations have vastly improved quality compared to likelihood training, as shown in human evaluations.

2 Related Work

The case of neural degeneration has been observed in recent papers, where it appears that the more open-ended the task, the more degeneration occurs. In dialogue, it has been shown that there is a frequency distribution shift from the human distribution, where generative models are more likely to output frequent words, and use rare words significantly less than humans. For example, this was observed across all generative models submitted to the ConvAI2 NeurIPS 2018 competition (Dinan et al., 2019). For language modeling, the work of Holtzman et al. (2019), from which this paper takes its name, highlighted both problems with the frequency distribution and the level of repetition of the model during generations compared to human utterances. This is not remedied by simply increasing the amount of the training data, e.g. with GPT-2 models (Radford et al., 2019) which still display the same problem.

There are several methods that have been proposed to rectify these issues, the primary ones being to change the final generation stage from greedy/beam search to alternative methods. In particular, researchers have studied improved beam search variants and sampling variants.

Several forms of diverse beam search have been explored (Li et al., 2016; Vijayakumar et al., 2018; Kulikov et al., 2018; Holtzman et al., 2018) which can to some degree decrease the level of repetition of a model by selecting candidates unlike previously chosen ones, due to their variety. Separately, hard or soft beam blocking has been investigated (Klein et al., 2017), whereby previously generated n-grams should not be generated again. This approach is often used in dialogue generation, fixing some obvious token or phrase level repetitions.

The second major approach is that of sampling from the model at generation time. Top k-sampling (Fan et al., 2018) and nucleus sampling (Holtzman et al., 2019) are two methods that sample sequences based on a function of the predicted next token probability distribution given by the model. Both approaches vastly improve the repetition issue, as the randomized aspect of the model means it is less likely to repeat itself, even if highly scored paths under the model (represented by beam search candidates) contain repetitions. However, as the underlying model is unchanged, the model often prefers semantically similar phrasing, depending on the temperature parameter of the sampling (Holtzman et al., 2019). Further, as beam search variants are the preferred method in less open-ended tasks such as machine translation, this solution is less relevant there. Ideally we would like a model that can work with both beam and sampling decoding methods.

Some other lines of work of note are the use of generative control (Kikuchi et al., 2016; Fan et al., 2017; Peng et al., 2018; See et al., 2019) and the retrieve-and-refine setting (Weston et al., 2018; Guu et al., 2018). The former has been used effectively to control specificity or sequence length in summarization, story and dialogue generation systems, which can result in the use of rarer words in generation (See et al., 2019). The latter employs retrieval methods to provide human generated sentences – which follow the desired distribution – as conditioning for final generation, which improves the final generation statistics (Weston et al., 2018).

3 NEURAL TEXT GENERATION

Assume there is a probability distribution $p_*(\mathbf{x})$ over variable-length sequences $\mathbf{x} \in \mathcal{V} \times \cdots \times \mathcal{V}$ where $\mathcal{V}$ is a finite set, and $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ is a sequence composed of tokens $x_t \in \mathcal{V}$.

3.1 LANGUAGE MODELING

Our goal is to find a model $p_\theta(\mathbf{x})$ which resembles $p_*(\mathbf{x})$, meaning that samples $\hat{x} \sim p_\theta$ are similar to samples from p_*, and $p_\theta(\mathbf{x}) \approx p_*(\mathbf{x})$ for all $\mathbf{x}$. When p_* is a distribution over text sequences, we call the problem of finding p_θ **language modeling**, and when additionally $p_\theta(\mathbf{x})$ is parameterized by a neural network, we call p_θ a **neural language model**. In this paper we assume that p_θ is autoregressive and monotonic, taking the form

$$p_\theta(\mathbf{x}) = \prod_{t=1}^{|\mathbf{x}|} p_\theta(x_t | x_{<t}). \tag{1}$$

The *de facto* approach to training such a model is to find parameters θ that maximize the log-likelihood of a finite set of samples $\mathcal{D}$ from p_* by minimizing:

$$\mathcal{L}_{\text{MLE}}(p_\theta, \mathcal{D}) = -\sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \log p_\theta(x_t^{(i)} | x_{<t}^{(i)}). \tag{2}$$

In §4 we demonstrate that this training criterion can yield degenerate neural language models, then propose a method in Section 5 to fix the degeneration.

3.2 SEQUENCE COMPLETION

A closely related problem consists of sampling a sub-sequence, or **prefix**, $\mathbf{x}_{1:k} \sim p_*$, then using p_θ to conditionally decode a **continuation**, $\hat{\mathbf{x}}_{k+1:N} \sim p_\theta(\cdot | \mathbf{x}_{1:k})$. We now want the resulting **completion** $(x_1, \ldots, x_k, \hat{x}_{k+1}, \ldots, \hat{x}_N)$ to resemble a sample from p_*.

We adopt sequence completion as a setting under which we study the behavior of neural text generators, due to its generality. For instance, sequence completion encompasses conditional story generation (Fan et al., 2018), contextual text completion (Radford et al., 2019), language modeling as the special case of *k=0*, and dialogue modeling (Zhang et al., 2018) where $\mathbf{x}_{1:k}$ is a dialogue history and a completion is the dialogue history plus a next utterance.

3.3 DECODING

Given p_θ and a prefix $\mathbf{x}_{1:k}$, finding the optimal continuation is not tractable, so in practice approximate decoding strategies are used to generate continuations. We categorize the strategies as either deterministic or stochastic.

Deterministic Decoding Two widely used deterministic decoding approaches are greedy search and beam search. The former can be seen as a special case of the latter.

Greedy search performs a single pass, selecting the highest probability token at each time step: $x_t = \arg\max p_\theta(x_t | x_{<t})$. Greedy search is efficient compared to more sophisticated deterministic approaches, but often leads to a sub-optimal completion since it does not take into account future consequences of token selection.

Beam search maintains a fixed-size set of partially-decoded sequences, called hypotheses. At each time step, beam search forms new hypotheses by appending each token in the vocabulary to each existing hypothesis, scoring the resulting sequences (e.g. using model probabilities), then selecting the highest scoring sequences. Compared to greedy search, beam search explores a larger subset of possible outputs at the expense of efficiency.

As we demonstrate in Section 4, these deterministic decoding strategies, which depend highly on underlying model probabilities, expose issues with conventionally trained neural language models.

Prefix	*Lyrically the song has excerpts of different languages including French , Spanish ,*
GPT2-greedy	German , Italian , Portuguese , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , ...
GPT2-top3	German , Italian , Portuguese , Spanish , Portuguese , Portuguese, Italian and Spanish. It is also available in English, French and Spanish. In addition to its lyrics, the album features a number of original songs, ...
GPT2-nucleus-0.3	German , Italian , Portuguese , Spanish , Portuguese , Italian , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , ...
Prefix	*The first pair of pereiopods is armed with a large , asymm*
GPT2-greedy	etrical, and long-range laser cannon. The second pair is armed with a large , asymmetrical, and long-range laser cannon. The third pair is armed with a large , asymmetrical, and long-range laser cannon. The fourth pair is armed with a large ...
GPT2-top3	etrical, and highly mobile head, and the second pair has a small , asymmetrical, and highly mobile head. The second pair has a large and highly mobile head, and the third pair is armed with a large and highly mobile head. The first pair ...
GPT2-nucleus-0.3	etrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed ...

Table 1: Top: Degenerate repetition in completions from a state-of-the-art large-scale language model (Radford et al., 2019). The examples contain single-word repetitions, phrase-level repetitions, and structural repetitions where some tokens within a repeating phrase vary. Recently proposed stochastic samplers (top-k, nucleus) exhibit degeneration based on hyper-parameter settings.

Stochastic Decoding An alternative is to sample from a model-dependent distribution at each step, $x_t \sim q(x_t|x_{<t}, p_\theta)$. In order to prevent sampling low probability tokens, a typical approach is to restrict sampling to a subset of the vocabulary $U \subset \mathcal{V}$:

$$q(x_t|x_{<t}, p_\theta) = \begin{cases} p_\theta(x_t|x_{<t})/Z & x_t \in U \\ 0 & \text{otherwise,} \end{cases}$$

where $Z = \sum_{x \in U} p_\theta(x|x_{<t})$. The *top-k* sampler restricts sampling to the k most-probable tokens; i.e. U is the size k subset of $\mathcal{V}$ which maximizes $\sum_{x \in U} p_\theta(x|x_{<t})$ (Fan et al., 2018). The *nucleus* sampler instead restricts sampling to the smallest set of tokens with total mass above a threshold $p \in [0, 1]$; i.e. U is the smallest subset with $\sum_{x \in U} p_\theta(x|x_{<t}) >= p$ (Holtzman et al., 2019).

A training criterion should ideally yield text generation models that are capable of using **both** deterministic and stochastic decoding, since the desired decoding method can vary by problem domain.

4 Neural Text Degeneration

In this section we discuss two degenerate properties that frequently occur in conventional neural text generators trained with the maximum likelihood objective (Equation 2).

Repetition First, neural text continuations exhibit **sequence-level** repetition, especially with deterministic decoding. The problem is easily seen by observing samples in Table 1, which shows completions from the state-of-the-art GPT-2 language model (Radford et al., 2019). Greedy decoding as well as top-k and nucleus sampling exhibit degenerate repetition (with a certain hyper-parameter setting), although greedy decoding shows the worst degradation. Using a Transformer language model trained with maximum likelihood (6.1), we find that the average percentage of repeated n-grams in model continuations with greedy decoding (44%) far exceeds that of humans (0.5%), computed over prefixes drawn from a validation corpus.

Unlike previous work which only focused on degenerate sequence-level repeats (Holtzman et al., 2019), we additionally observe that neural text generators exhibit substantially more repetition in **next-token** prediction compared to human text:

$$\Pr\left(\hat{x}_{k+1} = \arg\max p_\theta(x|\mathbf{x}_{1:k}) \in \mathbf{x}_{1:k}\right) > \Pr\left(x_{k+1} \in \mathbf{x}_{1:k}\right). \quad (3)$$

For instance, the Transformer language model (6.1) predicted next-tokens that appeared in the preceding 128 words 63% of the time, versus 49% in ground-truth text. This is especially concerning since the maximum-likelihood objective focuses on optimizing next-token conditional distributions.

Token Distribution Mismatch Second, both greedy continuations and next-token predictions from conventional neural text generators have different token distributions from human text. As demonstrated by Holtzman et al. (2019), such models with greedy or beam search tend to produce high frequency tokens too often and low frequency tokens too rarely, where frequency is defined by the human token distribution. With the Transformer language model (6.1), the set of next-token greedy predictions on a held-out validation set had roughly 40% fewer unique tokens than the ground-truth tokens (11.5k vs. 18.9k). Such behavior has been linked to generations being judged as dull by humans because rare words add specificity which can engage human readers (Weston et al., 2018; See et al., 2019).

5 THE UNLIKELIHOOD TRAINING OBJECTIVE

We now describe *unlikelihood training* for neural language models, then in Section 6 demonstrate empirically that the objectives substantially improve neural text degeneration (4).

5.1 UNLIKELIHOOD LOSS

The key idea behind the unlikelihood loss is decreasing the model's probability of certain tokens, called *negative candidates*. Given a sequence $(x_1, \ldots, x_T)$ and a set of negative candidate tokens $\mathcal{C}^t = \{c_1, \ldots, c_m\}$, where each $c_j \in \mathcal{V}$, we define the unlikelihood loss for step t as:

$$\mathcal{L}^t_{\text{UL}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t})). \tag{4}$$

The loss decreases as $p_\theta(c|x_{<t})$ decreases. Next, we define token-level and sequence-level training objectives, and discuss procedures for selecting negative candidates in each case.

5.2 TOKEN LEVEL OBJECTIVE

Given a sequence $(x_1, \ldots, x_T)$, the token-level objective applies the unlikelihood loss to a set of negative candidates at each time-step of maximum likelihood training:

$$\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\alpha \cdot \underbrace{\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}. \tag{5}$$

We propose a candidate set which uses the previous context tokens:

$$\mathcal{C}^t_{\text{prev-context}} = \{x_1, \ldots, x_{t-1}\} \setminus \{x_t\}. \tag{6}$$

Intuitively, the unlikelihood loss with this candidate set makes (i) incorrect repeating tokens less likely, as the previous context contains potential repeats, and (ii) frequent tokens less likely, as these tokens appear often in the previous context. This candidate set is also efficient to compute and requires no additional supervision.

Gradient analysis We assume $p_\theta(x_t|x_{<t}) = \text{softmax}(a)$ and consider the gradient of (5) with respect to the softmax input $a \in \mathbb{R}^{\mathcal{V}}$. With a single negative candidate, the (negative) gradient is:

$$\nabla \mathcal{L}_a = x^* - m \odot p, \tag{7}$$

$$m_i = \begin{cases} (1 - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}) & \text{if } i \neq i_{\text{neg}} \\ (1 + \alpha) & \text{if } i = i_{\text{neg}}, \end{cases} \tag{8}$$

where $x^* \in \{0, 1\}^{\mathcal{V}}$ is a one-hot ground-truth vector, $m \in \mathbb{R}^{\mathcal{V}}$, $p = p_\theta(\cdot|x_{<t})$, and p_{neg} is the probability of the negative candidate at index i_{neg}. See Appendix A for the derivation and a note about how the result generalizes to multiple candidates.

We highlight a few properties of the unlikelihood gradient (7). First, it differs from the (negative) likelihood (2) gradient, $(x^* - p)$, due to the term m which varies based on the hyper-parameter α and the model's negative candidate probability, p_{neg}. At the ground-truth token index i^*, the unlikelihood gradient is positive, hence increasing the ground-truth token's probability with a magnitude that grows with p_{neg}. Conversely, at the negative candidate index i_{neg} the gradient is negative. At all other token indices $i \notin \{i^*, i_{\text{neg}}\}$, the gradient moves from negative to positive as p_{neg} increases. For instance, with $\alpha = 1.0$ the gradient increases the probability of each token x_i when the model assigns high probability to the negative candidate ($p_{\text{neg}} > 0.5$), otherwise decreasing it.

5.3 Sequence Level Objective

While the token-level objective (subsection 5.2) efficiently augments maximum likelihood training with token-level penalties, it is limited to prefixes drawn from the training distribution. The resulting *distribution mismatch* between training sequences and generated sequences is a known issue with maximum-likelihood training, motivating objectives that operate on model-generated sequences (Daumé et al., 2009; Ross et al., 2011; Ranzato et al., 2015; Yu et al., 2016).

We thus propose a sequence-level objective which applies the unlikelihood loss to decoded continuations. That is, given a prefix $(x_1, \ldots, x_k) \sim p_*$, we decode a continuation $(x_{k+1}, \ldots, x_{k+N}) \sim p_\theta(\cdot|x_1, \ldots, x_k)$, construct per-step negative candidates $(\mathcal{C}^{k+1}, \ldots, \mathcal{C}^{k+N})$, and define each per-step sequence-level loss as:

$$\mathcal{L}^t_{\text{ULS}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t})), \tag{9}$$

for $t \in \{k+1, \ldots, k+N\}$.

Intuitively, the negative candidates can identify problematic tokens for the loss to penalize. We choose to penalize repeating n-grams in the continuation:

$$\mathcal{C}^t_{\text{repeat-n}} = \{x_t\} \text{ if } (x_{t-i}, \ldots, x_t, \ldots, x_{t+j}) \in x_{<t-i} \text{ for any } (j-i) = n, i \leq n \leq j, \tag{10}$$

which says that the token x_t is the (single) negative candidate for step t if it is part of a repeating n-gram.

In our experiments we apply this sequence loss in two ways: (i) using it to fine-tune a standard MLE baseline; and (ii) using it to fine-tune an unlikelihood model trained at the token level, $\mathcal{L}_{\text{UL-token}}$. We refer to the former as $\mathcal{L}_{\text{UL-seq}}$ and the latter as $\mathcal{L}_{\text{UL-token+seq}}$. In both cases, fine-tuning is done by equally mixing sequence-level unlikelihood updates (9) and the token-level loss from which it was initially trained (either likelihood updates (2) or token-level unlikelihood updates (5)).

Efficiency Any objective that requires explicitly decoding a sequence is constrained by sample efficiency when decoding is slow. If sample efficiency is high, we can tolerate slow decoding, since we do not need to decode too many times. If sample efficiency is low, the total decoding time is too large for practical use. In our experiments we show that when used for fine-tuning, the sequence-level unlikelihood objective can substantially reduce degeneration in **under 1,500 updates**, rendering it practical for modern large-scale neural models, even with high decoding costs.

6 Experiments

In our experiments, we use the proposed unlikelihood objectives to train large-scale neural language models, which are then used as text generators in a completion task. We analyze the models' degree of degeneration in terms of repetition and token-level mismatch, and compare the models against a standard maximum likelihood baseline. Our main findings are that (i) the proposed unlikelihood objectives substantially reduce degeneration according to all metrics, while maintaining language modeling capability as measured by perplexity and single-token prediction accuracy; and (ii) human evaluation indicates the proposed methods' completions are superior to the maximum likelihood baseline when both methods use standard beam search.

6.1 EXPERIMENTAL SETUP

We follow a standard language modeling setup from Baevski and Auli (2019), which we detail along with our sequence completion protocol below.

Model Architecture Recent large-scale language models are based on the Transformer architecture, a multi-layer feed-forward network with self-attention (Vaswani et al., 2017). Thus we adopt a 16-layer Transformer architecture, with 8 attention heads, an embedding dimension of 1024, and a fully-connected dimension of 4096 ; the architecture is based on (Baevski and Auli, 2019) but with standard embedding and softmax layers. We emphasize that our proposed method is architecture agnostic; we choose this one as a representative of recent large-scale language models, e.g. (Baevski and Auli, 2019; Radford et al., 2019).

Dataset We use the Wikitext-103 dataset (Merity et al., 2016), a large-scale collection of Wikipedia articles containing over 100 million words and 260 thousand unique tokens. As a document-level dataset, Wikitext-103 is an open-source representative of recent datasets used for large-scale language modeling (Baevski and Auli, 2019; Radford et al., 2019). We follow standard pre-processing and perform experiments at the word level as in Baevski and Auli (2019).

Training As in Baevski and Auli (2019); Radford et al. (2019), we train on fixed-length contiguous sequences, in our case of length 1,536, which was selected based on GPU memory constraints. For the token-level losses ($\mathcal{L}_{\text{MLE}}$, $\mathcal{L}_{\text{UL-token}}$), we train each model on 8 GPUs for a maximum of 150k updates, evaluating on the validation set and saving the model state every 10k updates. For the experiments below, we select the saved model state with the best validation perplexity.

Sequence-level fine-tuning begins with the model state selected based on the validation perplexity. Models are fine-tuned for 1,500 total updates. With probability 0.5 an update uses $\mathcal{L}_{\text{ULS}}$ and otherwise uses the token-level loss with which the model was trained. For a $\mathcal{L}_{\text{ULS}}$ update, we split each training sequence and greedily decode continuations (details below). The experiments use a prefix length $k = 50$ and continuation length $N = 100$ for fine-tuning.

Completions We evaluate a model on sequence completion by using the model to decode continuations of prefixes derived from the validation (or test) set. Specifically, the validation (or test) set is first partitioned into sequences of 1,536 tokens, as in training. Then we split each sequence into a batch of prefixes of length k (discarding extra tokens), and decode a continuation of length N for each prefix. The experiments below use $k = 50$ and $N = 100$ for evaluation. We evaluate model continuations using both deterministic and stochastic inference methods. For the former we use greedy search and beam search with beam size 10, and for the latter we use top-k sampling with $k \in \{3, 50\}$ and nucleus sampling with $p \in \{0.3, 0.9\}$.

Token-Level Models We train a baseline language model ($\mathcal{L}_{\text{MLE}}$) with maximum-likelihood (Equation 2), and a model which uses the unlikelihood token-level objective ($\mathcal{L}_{\text{UL-token}}$).

Sequence-Level Models Using the unlikelihood sequence objective (subsection 5.3) we fine-tune the baseline MLE model ($\mathcal{L}_{\text{UL-seq}}$) or our best-performing token-level unlikelihood model ($\mathcal{L}_{\text{UL-token+seq}}$).

6.2 EVALUATION METRICS

We evaluate each model's degree of token-level and sequence-level degeneration in terms of repetition, token distribution, and language modeling quality using the following metrics.

Repetition As a token-level metric for repetition, we use the fraction of next-token (top-1) predictions that occur in the previous ℓ tokens (**rep/ℓ**). That is, given a validation set $\mathcal{D}$ of length-T sequences,

$$\frac{1}{|\mathcal{D}|T} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{t=1}^{T} \mathbb{I}\left[\arg\max p_\theta(x|\mathbf{x}_{<t}) \in \mathbf{x}_{t-\ell-1:t-1}\right]. \tag{11}$$

		seq-rep-4
Prefix	*... Lyrically the song has excerpts of different languages including French , Spanish*	
$\mathcal{L}_{\text{MLE}}$	, Italian , Spanish , Italian , Spanish , Italian , Spanish , Spanish , Portuguese	0.744
$\mathcal{L}_{\text{UL-token+seq}}$	, and German . In the first verse , the protagonist sings about being a " girl who 's been in love with someone else " , while the second verse describes the relationship between the protagonist and her lover . In the third verse , the protagonist sings about	0.063
Prefix	*... with timely advice from General Lee , adopted a strong defensive position that was virtually*	
$\mathcal{L}_{\text{MLE}}$	impregnable . Lee 's forces were well prepared for the night , and the battle was a disaster . Lee 's forces were well prepared for the night , and the battle was a disaster . Lee 's forces were well prepared for the night , and the battle was	0.617
$\mathcal{L}_{\text{UL-token+seq}}$	impregnable . The Americans were also able to use the boats to bombard the fort from the rear , and the guns fired at the British ships in the harbor . The British bombardment began at 9 : 30 am , when the first shots were fired on the	0.000
Prefix	*... starboard engines and was going to crash . " We 're going in ,"*	
$\mathcal{L}_{\text{MLE}}$	he said . " We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to	0.787
$\mathcal{L}_{\text{UL-token+seq}}$	Hood said . " I 'm going to make sure we 're going to get back to the water . " The order to abandon ship was given by Admiral Beatty , who ordered the remaining two battlecruisers to turn away . At 18 : 25 , Hood turned his	0.000
Prefix	*... career - high 27 points on 8 - for - 11 shooting with three rebounds*	
$\mathcal{L}_{\text{MLE}}$	and two assists . On January 3 , 2012 , he was named to the 2012 13 All - Atlantic 10 first team . On February 3 , 2012 , he was named to the Atlantic 10 first team . On February 5 , 2012 , he was named	0.277
$\mathcal{L}_{\text{UL-token+seq}}$	and a career - high 7 assists against the Minnesota Timberwolves . On February 3 , 2012 , he was named to the 2012 All - NBA First Team . On March 7 , 2012 , he was named one of five finalists for the Naismith Award , which is	0.064

Table 2: Example greedy completions, showing the last 15 tokens of a 50 token prefix, and 50-token continuations. The completions show representative examples of the MLE model's degenerate single token repetition (top), phrase-level repetition (middle two), and 'structural' repetition (bottom), as well as the proposed method's ability to fix these degenerate behaviors.

A predicted token is called a "single-token repeat" when $\mathbb{I}[\cdot]$ is 1. Some of these single-token repeats also occur in the human-generated sequences, and we thus report a variant which only counts single-token repeats that are additionally not equal to the ground-truth next-token (**wrep/**ℓ).

We use the portion of duplicate n-grams (**seq-rep-n**) in a generated sequence to measure sequence-level repetition. That is, for a continuation $\mathbf{x}_{k+1:k+N}$ we compute,

$$1.0 - \frac{|\text{unique n-grams}(\mathbf{x}_{k+1:k+N})|}{|\text{n-grams}|}, \tag{12}$$

and average over continuations. **seq-rep-n** is zero when the continuation has no repeating n-grams, and increases towards 1.0 as the model repeats. We compute **seq-rep-n** on the continuation rather than the full completion since we are interested in measuring degenerate repeats in the continuation.

Token Distribution We quantify a model's predicted token distribution using the number of unique tokens. As a token-level metric (**uniq**), we use the number of unique next-token predictions on the validation set, i.e. $|\{\arg\max p(x_t|x_{<t}) \mid x_{<t} \in \mathcal{D}_{\text{valid}}\}|$. As a sequence-level metric (**uniq-seq**) we use the number of unique tokens in continuations of prefixes from the validation set (subsection 6.1).

Language Modeling Quality To quantify a model's language modeling quality we use the standard perplexity metric (**ppl**), and next-token greedy prediction accuracy (**acc**).

Model	search	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
$\mathcal{L}_{\text{MLE}}$	greedy	.442	10.2k	**24.52**	**.401**	.619	.345	11.5k
	beam	.507	9.2k					
$\mathcal{L}_{\text{UL-token}}$	greedy	**.267**	**12.0k**	25.68	.397	**.568**	**.304**	**12.3k**
	beam	**.330**	**11.0k**					
$\mathcal{L}_{\text{UL-seq}}$	greedy	.134	11.7k	**23.95**	**.408**	.606	.331	12.4k
	beam	.015	16.1k					
$\mathcal{L}_{\text{UL-token+seq}}$	greedy	**.051**	**14.6k**	25.37	.401	**.553**	**.288**	**13.3k**
	beam	**.012**	**16.9k**					
Human	-	.005	18.9k	-	-	.479	-	18.9k

Table 3: Results for token-level objectives (upper) and sequence-level fine-tuning (lower) according to sequence-level (left) and token-level (right) metrics using the **validation subset of wikitext-103**. The best metrics achieved by both token-level and sequence-level models using both greedy and beam search are shown in bold. rep and wrep use $\ell = 128$; relative rankings hold for other ℓ.

6.3 RESULTS

Token-level and sequence-level results using the validation set are shown in Table 3.

Baseline The baseline model trained with maximum likelihood ($\mathcal{L}_{\text{MLE}}$) achieved 24.52 validation perplexity (25.71 test), comparable to a current state-of-the-art system (Baevski and Auli, 2019) (23.87 valid, 24.92 test). However, the baseline's sequence-level repeats (**seq-rep-4** .442) and single-token repeats (**rep** .619) far exceed those in human text (.005, .479 respectively). The baseline continuations contain far fewer unique tokens than human text (**uniq-seq** 10.2k vs 18.9k).

Token-Level Objective First, focusing on the token-level objective ($\mathcal{L}_{\text{UL-token}}$), we see that compared to the baseline ($\mathcal{L}_{\text{MLE}}$), the proposed unlikelihood objective reduced (improved) next-token wrong repetition (**wrep** .304 vs. .345) while increasing the number of unique next-tokens (**uniq** 12.3k vs. 11.5k). Perplexity and accuracy were kept essentially constant.

Importantly, the token-level unlikelihood objective yielded substantial improvements in *sequence-level generations*. With greedy search, unlikelihood training improved the 4-gram repetition in continuations by 40% (**seq-rep-4** .267 vs. .442) while generating roughly 15% more unique tokens than the baseline (**uniq** 12k vs. 10.2k). With beam search, unlikelihood training showed similar improvements over the baseline.

Sequence-Level Objective The sequence level fine-tuning ($\mathcal{L}_{\text{UL-token+seq}}$) yielded significant improvements in continuation quality, with a **97% reduction** in 4-gram repetitions (**seq-rep-4** .012 vs. .442) from the baseline level (greedy $\mathcal{L}_{\text{MLE}}$), and roughly **65% more** unique tokens (**uniq** 16.9k vs. 10.2k) with beam search.

Compared to the token-level unlikelihood model ($\mathcal{L}_{\text{UL-token}}$) which was the starting point of fine-tuning, the fine-tuned model's repetition substantially improved (**seq-rep-4** .051 vs. .267), unique tokens increased (**uniq** 14.6k vs. 12k), and token-level metrics such as perplexity improved (**ppl** 25.37 vs. 25.68), despite using *only 1,500 updates*.

The fine-tuned maximum-likelihood model ($\mathcal{L}_{\text{UL-seq}}$) showed similar improvements over the baseline. This demonstrates the usefulness of the proposed sequence-level fine-tuning as a cheap, effective way to improve existing, pretrained language models. However, the combination of the token-level unlikelihood training with sequence-level fine-tuning yielded the best performing model in terms of repetition and token distribution.

Finally, after sequence-level fine-tuning, beam search performed better than greedy search. This was not the case for models only trained with token-level objectives. We do not currently have an explanation for this difference and leave it to be investigated further in the future.

Model	search	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
$\mathcal{L}_{\text{MLE}}$	greedy	.453	10.4k	**25.701**	**.394**	.629	.355	11.7k
	beam	.528	9.4k					
$\mathcal{L}_{\text{UL-token}}$	greedy	**.276**	**12.5k**	27.020	.390	**.575**	**.309**	**12.6k**
	beam	**.336**	**11.6k**					
$\mathcal{L}_{\text{UL-seq}}$	greedy	.144	12.1k	**25.112**	**.401**	.613	.338	12.7k
	beam	.014	17.5k					
$\mathcal{L}_{\text{UL-token+seq}}$	greedy	**.059**	**15.2k**	26.839	.394	**.559**	**.293**	**13.6k**
	beam	**.012**	**18.1k**					

Table 4: Results for token-level objectives (upper) and sequence-level fine-tuning (lower) according to sequence-level (left) and token-level (right) metrics using the **test subset of Wikitext-103**.

Model 1		Model 2	Win rate
$\mathcal{L}_{\text{MLE}}$ baseline		$\mathcal{L}_{\text{UL-token}}$	62%
$\mathcal{L}_{\text{MLE}}$ baseline		$\mathcal{L}_{\text{UL-seq}}$	*70%
$\mathcal{L}_{\text{MLE}}$ baseline	*beaten by*	$\mathcal{L}_{\text{UL-token+seq}}$	*84%
$\mathcal{L}_{\text{UL-token}}$		$\mathcal{L}_{\text{UL-token+seq}}$	*72%
$\mathcal{L}_{\text{UL-seq}}$		$\mathcal{L}_{\text{UL-token+seq}}$	58%
$\mathcal{L}_{\text{MLE}}$ baseline		Reference	*74%
$\mathcal{L}_{\text{UL-token}}$	*beaten by*	Reference	*68%
$\mathcal{L}_{\text{UL-seq}}$		Reference	56%
$\mathcal{L}_{\text{UL-token+seq}}$		Reference	52%

Table 5: **Human evaluation results**. Human evaluators preferred generations from our models over the baseline model, and $\mathcal{L}_{\text{UL-token+seq}}$ outperformed our other variants. The sequence-tuned models approach human-level performance. Comparisons marked with * are statistically significant (one-sided binomial test, $p < .05$).

To visualize how these improvements in metrics translate to generation quality, Table 2 shows example greedy completions, characterizing the baseline's degeneration and $\mathcal{L}_{\text{UL-token+seq}}$'s improved behavior. Later in Section 6.4, we quantitatively evaluate these differences with human evaluation.

Performance on the test split While the preceding results were on the validation set, in Table 4 we confirm that similar trends hold on the test set; the proposed unlikelihood objectives result in substantial repetition and token-distribution improvements compared to the baseline.

Sampling-based decoding Although we have focused on deterministic decoding so far, in this experiment we confirm that a model trained with the proposed unlikelihood objective can still be used with stochastic decoders. Appendix Table 6 shows metrics for completions generated with top-k sampling ($k \in \{3, 50\}$) and nucleus sampling ($p \in \{0.3, 0.9\}$) (Holtzman et al., 2019), respectively. Models trained with the proposed unlikelihood objectives maintain performance with the maximum likelihood model, but with improvements in repetition.

6.4 Human Evaluation

We perform human evaluation to judge the overall quality of the generations of the proposed models compared to each other, the baseline, and the human reference. The model evaluation is performed in a pairwise manner, in which each crowd worker is presented with a prompt and shown continuations from two different models. Subjects were asked which continuation they found more natural, and instructed to disregard factual errors and focus on the quality of writing. They were also asked to enter a reason for their choice, which we logged. See Appendix B for a screenshot of the user interface. All prompts were generated from the Wikitext-103 test set, and we collected 50 annotations per pairwise model comparison, where each comparison used a unique prefix. All models used beam search (beam size 10) for generation. We report the win rates for each pairwise comparison.

The results of the human evaluation are presented in Table 5. We find that both the token-level and the two sequence-level models are preferred over the baseline, and that the token+sequence model ($\mathcal{L}_{\text{UL-token}}$) outperforms other variants. We also observe that, in agreement with automatic metrics, the win rates improve after adding the sequence level objective. Crowd workers frequently expressed their preference for our models by describing the baseline as "spammy", or that the baseline "just repeats itself over and over".

We also compare our models against the human-authored reference gold continuations from the Wikitext-103 test set. While the reference continuation always outperforms the model's prediction, the human win rates decrease (i.e. become closer to chance) as we add the sequence level objective; ultimately, our $\mathcal{L}_{\text{UL-token+seq}}$model obtains a 48% win rate against the reference responses.

7 CONCLUSION

We described *unlikelihood training*, an approach to training neural language models. We observed that state-of-the-art models trained to maximize likelihood demonstrate neural text degeneration, which we characterized and quantified in terms of repetition and token distribution mismatch. Our results show that the likelihood objective is not constrained enough in the sense that two models with the same perplexity can exhibit wildly different generation performance. We empirically showed that unlikelihood training - both at the token and sequence levels - substantially reduced degeneration according to automatic metrics and human evaluation with deterministic decoding. Our approach is capable of using a wide variety of decoding methods, provides a strong alternative to traditional training, and is a promising method for other sequence generation tasks.

REFERENCES

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

Yejin Choi. 2018. The missing representation in neural (language) models. *3rd Workshop on Representation Learning for NLP (RepL4NLP)*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*.

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

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649. Association for Computational Linguistics.

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

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

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

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

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

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota. 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.

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

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

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yingrui Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *ArXiv*, abs/1609.05473.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

A GRADIENT

Notation Let x_t^* be the true next-token (index $i^* \in \mathcal{V}$) at step t, and let x_{neg} be a negative candidate (index i_{neg}). Let $p = p(x_t|x_{<t}) \in \mathbb{R}^{\mathcal{V}}$ be the output of softmax(a) where $a \in \mathbb{R}^{\mathcal{V}}$.

Denote the probability of an element $i \in \{1, \ldots, V\}$ as $p_i = p(x_t^i|x_{<t})$, and let p_*, p_{neg}, and $\tilde{p}_i$ be probabilities of the true next-token, negative-candidate token, and any other token with $i \notin \{i^*, i\}$.

A.1 DERIVATION

The (negative) token-level loss with a single candidate is,

$$\mathcal{L}_t = \log p(x_t^*|x_{<t}) + \alpha \cdot \log(1 - p(x_{\text{neg}}|x_{<t})), \tag{13}$$

and its gradient with respect to a logit a_i is:

$$\frac{\partial \mathcal{L}}{\partial p_i}\frac{\partial p_i}{\partial a_i} = (\mathbb{I}[i = i^*] - p_i) - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}} (\mathbb{I}[i = i_{\text{neg}}] - p_i)\,. \tag{14}$$

We consider the gradient when i is the true next-token, a negative-candidate, and any other token.

True Next-Token ($i = i^*$)

$$\frac{\partial \mathcal{L}}{\partial p_*}\frac{\partial p_*}{\partial a_{i^*}} = (1 - p_*) - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(0 - p_*) \tag{15}$$

$$= 1 - p_*(1 - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}). \tag{16}$$

Negative Candidate ($i = i_{\text{neg}}$)

$$\frac{\partial \mathcal{L}}{\partial p_{\text{neg}}}\frac{\partial p_{\text{neg}}}{\partial a_{\text{neg}}} = (0 - p_{\text{neg}}) - \alpha^* \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(1 - p_{\text{neg}}) \tag{17}$$

$$= -p_{\text{neg}}(1 + \alpha^*). \tag{18}$$

Other Token ($i \notin \{i^*, i_{\text{neg}}\}$)

$$\frac{\partial \mathcal{L}}{\partial \tilde{p}_i}\frac{\partial \tilde{p}_i}{\partial a_i} = (0 - \tilde{p}_i) - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}(0 - \tilde{p}_i) \tag{19}$$

$$= -\tilde{p}_i(1 - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}). \tag{20}$$

Combining the three cases above, we get:

$$\nabla \mathcal{L}_a = x^* - m \odot p, \tag{21}$$

where $x^* \in \{0, 1\}^V$ is 1 at index i^* and 0 otherwise, and $m \in \mathbb{R}^V$ is:

$$m_i = \begin{cases} (1 - \alpha \frac{p_{\text{neg}}}{1 - p_{\text{neg}}}) & i \neq i_{\text{neg}} \\ (1 + \alpha) & i = i_{\text{neg}}. \end{cases} \quad \square \tag{22}$$

Multiple Candidates In general the objective considers multiple candidates (see section 5):

$$\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = \underbrace{-\alpha \cdot \sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} \underbrace{- \log p_\theta(x_t|x_{<t})}_{\text{likelihood}}. \tag{23}$$

We regroup the token-level objective to be a weighted sum of per-candidate objectives:

$$-\mathcal{L}^t_{\text{UL-token}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = \frac{1}{|\mathcal{C}^t|} \sum_{c \in \mathcal{C}^t} \left(\log p_\theta(x_t|x_{<t}) + \alpha_c \cdot \log(1 - p_\theta(c|x_{<t})) \right) \tag{24}$$

where $\alpha_c = \alpha \cdot |\mathcal{C}^t|$.

Now the gradient can be generalized to multiple candidates, in which case the gradient takes the same form as Eqn. 22, but with α_c in place of α.

Search	Model	seq-rep-4	uniq-seq	ppl	acc	rep	wrep	uniq
top-k-3	$\mathcal{L}_{\text{MLE}}$	.0991	14.7k	25.70	.350	.597	.355	12.6k
	$\mathcal{L}_{\text{UL-token}}$	.0491	16.4k	27.02	.344	.539	.306	13.6k
	$\mathcal{L}_{\text{UL-seq}}$	.0068	17.9k	25.11	.353	.581	.341	13.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0087	15.2k	26.84	.347	.524	.292	14.6k
top-k-50	$\mathcal{L}_{\text{MLE}}$	.0165	21.9k	25.70	.302	.511	.303	16.1k
	$\mathcal{L}_{\text{UL-token}}$	.006	23.5k	27.02	.286	.440	.247	17.8k
	$\mathcal{L}_{\text{UL-seq}}$	.0005	25.7k	25.11	.291	.497	.291	17.3k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0009	23.7k	26.84	.289	.430	.238	18.8k
top-p-0.3	$\mathcal{L}_{\text{MLE}}$	.2773	13.6k	25.70	.264	.339	.154	12.6k
	$\mathcal{L}_{\text{UL-token}}$	.1005	16.5k	27.02	.247	.290	.121	13.9k
	$\mathcal{L}_{\text{UL-seq}}$	.0033	20.8k	25.11	.266	.327	.145	13.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0041	19.1k	26.84	.250	.284	.116	14.9k
top-p-0.9	$\mathcal{L}_{\text{MLE}}$	.0154	26.9k	25.70	.288	.462	.263	18.6k
	$\mathcal{L}_{\text{UL-token}}$	.004	30.2k	27.02	.266	.381	.202	22.3k
	$\mathcal{L}_{\text{UL-seq}}$	.0003	34.7k	25.11	.290	.450	.254	19.6k
	$\mathcal{L}_{\text{UL-token+seq}}$	.0007	32.4k	26.84	.269	.376	.198	22.7k
Human	-	.006	19.8k	-	-	.487	-	19.8k

Table 6: Stochastic decoding results according to sequence-level (left) and token-level (right) metrics using the test subset of Wikitext-103. In genera,l sampling methods yield worse next token prediction than deterministic approaches (0.302 vs. 0.394 acc for top-k-50 vs. greedy MLE); compare with Table 4. As the choice of sampling hyperparameter gets closer to greedy (i.e. lower values of k and p) next token accuracy improves, eventually approaching the greedy MLE results. The unlikelihood objective trained sampling models have similar next token accuracy (acc) to their likelihood trained counterparts, but exhibit fewer repetitions. For lower values of p and k the improvements of unlikelihood training are larger, e.g. 0.2773 reduced to 0.0041 for 4-gram sequence repetitions (seq-rep-4) using top-p-0.3. At higher levels of p and k for all methods the continuations contain more unique tokens than that of humans, meaning those values may be too high.

B Human Evaluation User Interface

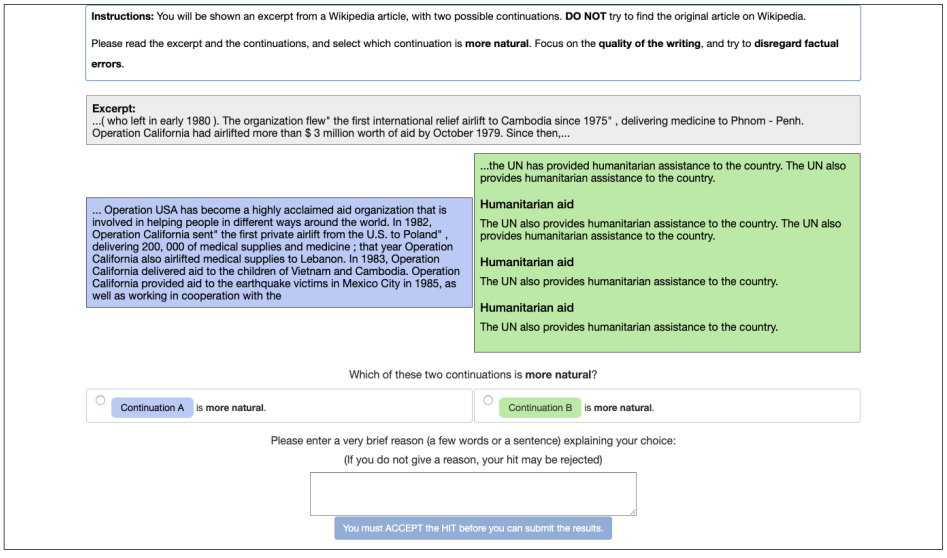

Figure 1: Screen shot of the user interface used in the human evaluation.