MLE-Guided Parameter Search for Task Loss Minimization in Neural Sequence Modeling

Sean Welleck,[*] **Kyunghyun Cho**
New York University

Abstract

Neural autoregressive sequence models are used to generate sequences in a variety of natural language processing (NLP) tasks, where they are evaluated according to sequence-level task losses. These models are typically trained with maximum likelihood estimation, which ignores the task loss, yet empirically performs well as a surrogate objective. Typical approaches to directly optimizing the task loss such as policy gradient and minimum risk training are based around sampling in the sequence space to obtain candidate update directions that are scored based on the loss of a single sequence. In this paper, we develop an alternative method based on random search in the parameter space that leverages access to the maximum likelihood gradient. We propose maximum likelihood guided parameter search (MGS), which samples from a distribution over update directions that is a mixture of random search around the current parameters and around the maximum likelihood gradient, with each direction weighted by its improvement in the task loss. MGS shifts sampling to the parameter space, and scores candidates using losses that are pooled from multiple sequences. Our experiments show that MGS is capable of optimizing sequence-level losses, with substantial reductions in repetition and non-termination in sequence completion, and similar improvements to those of minimum risk training in machine translation.

1 Introduction

Neural autoregressive sequence models are used in a variety of natural language processing (NLP) tasks, such as machine translation (Bahdanau, Cho, and Bengio 2015), summarization (Rush, Chopra, and Weston 2015), dialogue modeling (Vinyals, Quoc, and Le 2015), and text completion (Sutskever, Martens, and Hinton 2011; Radford et al. 2018; Holtzman et al. 2019; Welleck et al. 2020b). In these tasks, a decoding algorithm is used to produce sequences that are evaluated according to a sequence (or corpus) level task loss such as BLEU (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005), or alternative n-gram based metrics.

The conventional training approach, maximum likelihood, optimizes a token-level surrogate to the 0-1 loss, and only leverages sequences drawn from the ground-truth distribution. The resulting mismatch between the training and evaluation loss functions, and the discrepancy between the sequence distributions used for training and the distribution encountered at evaluation time has prompted alternative sequence-level training algorithms (e.g. (Daumé, Langford, and Marcu 2009; Ranzato et al. 2016; Shen et al. 2016)). Nevertheless, maximizing the likelihood has empirically performed well as a surrogate to minimizing the task loss, achieving strong performance on the aforementioned tasks. In this paper, we develop a sequence-level training procedure that addresses the downsides of maximum likelihood by leveraging its strengths as a surrogate objective.

It is challenging to optimize a task loss, as the loss is typically non-differentiable with respect to the model parameters, and optimization is done over a high-dimensional parameter space. Typical approaches to this problem in natural language processing are based around the policy gradient estimator (Williams 1992), such as Shen et al. (2016); Ranzato et al. (2016); Bahdanau et al. (2017); Yu et al. (2017). This estimator is used to optimize an arbitrary task loss by introducing stochasticity via autoregressive sampling in the action space, which is a critical downside in NLP, where the action space (vocabulary) is large and the sequence-level reward is sparse. The estimator's variance grows with the sequence length, necessitating a parameterized baseline or a heuristic sampling schedule in practice, while requiring initialization from a pretrained model. Recently, the effectiveness of these methods in NLP has been called into question (Caccia et al. 2020; Choshen et al. 2020).

An alternative class of methods optimize a black-box function without requiring gradient information. Of these, estimation-of-distribution algorithms, including the cross-entropy method (Rubinstein 1999), evolutionary strategies (Rechenberg 1978), and their variants (Hansen and Ostermeier 2001; Salimans et al. 2017), operate by maintaining a search distribution from which a set of random perturbations are sampled. The function value (i.e. task loss) at each of the perturbed points is used to update the search distribution. Because stochasticity is introduced in the parameter space rather than the action space, rewards associated with each search candidate are pooled over multiple examples, 'densifying' the sparse reward. This parameter-space exploration (Rückstieß et al. 2010) is attractive for NLP since the same decoding algorithm can be used for training and evaluation, and the variance is independent of the action space size or the se-

[*]Correspondence to: wellecks@nyu.edu.

quence length. However, a key challenge for these black-box methods is handling high-dimensional search spaces. This typically restricts their use to training networks that are small compared to those used in neural sequence modeling (Mania, Guy, and Recht 2018) or requires massive parallelization (Salimans et al. 2017), and their use with large-scale natural language processing models has been under-explored.

In this paper, we leverage the fact that in many sequence modeling tasks encountered in natural language processing, a surrogate update direction is available in the form of the maximum likelihood gradient. We hypothesize that incorporating this surrogate information into a random-search method can substantially alleviate issues stemming from the large search space. We frame learning as sampling from a distribution over parameter update directions that is proportional to the improvement in task loss. Since this distribution is only accessible for evaluation up to a normalizing constant, we propose to use self-normalized importance sampling for obtaining update directions.

The key idea behind our method is to form a proposal distribution that is a mixture of random search around the current parameters and around the maximum-likelihood update direction. Our experiments show that the resulting procedure, called *maximum-likelihood guided parameter search* (MGS), is effective for minimizing sequence-level losses in natural language generation and machine translation, offering an alternative to policy gradient and minimum risk methods.

2 Maximum-Likelihood Guided Parameter Search

Sequence generation. Sequence generation is the problem of mapping an input X to an output $Y = (y_1, \dots, y_{|Y|})$. In our setting of neural sequence generation, this mapping is a deterministic decoding algorithm $\mathcal{F}(\theta, X)$, which uses an autoregressive model $p_\theta(Y|X) = \prod_{t=1}^{|Y|} p_\theta(y_t|y_{<t}, X)$ to produce an output $\hat{Y}$ given an input X. This includes greedy and beam search decoding, and stochastic decoding algorithms with a noise input, $\mathcal{F}(\theta, X, \epsilon)$. The goal of sequence generation is to find a model whose generations have minimal task loss on a set $\mathcal{D} = \{(X, Y)\}$ of input-output pairs,

$$C(\theta, \mathcal{D}) = \sum_{X,Y \in \mathcal{D}} c(\mathcal{F}(\theta, X), Y), \quad (1)$$

where we assume $c(\hat{Y}, Y) \in \mathbb{R}$ is an arbitrary sequence-level loss (e.g. sentence-BLEU). The most widely used approach to training such a model is minimizing the negative log-likelihood given a training set, which ignores the task loss : $\mathcal{L}_{\text{MLE}}(\theta; \mathcal{D}) = -\sum_{X,Y \in \mathcal{D}} \sum_{t=1}^{|Y|} \log p_\theta(y_t|y_{<t}, X)$.

Method. To directly optimize (1), we iteratively update the parameters θ in the direction of maximal improvement in the task loss. Each update corresponds to the expected update under a distribution that weights each direction according to its improvement,

$$\Delta_* = \mathbb{E}_{\Delta \sim p_*(\Delta|\theta;\alpha)}\left[\Delta\right], \quad (2)$$

where,

$$p_*(\Delta|\theta;\alpha) \propto \tilde{p}_*(\Delta|\theta;\alpha) = \exp\left(\alpha(C(\theta) - C(\theta + \Delta))\right),$$

and $\alpha \in \mathbb{R}_{>0}$ is a temperature parameter. When $\alpha \to 0$, the distribution becomes uniform, and when $\alpha \to \infty$ it concentrates on the direction(s) of maximal task loss improvement. Since p_* is only known up to a normalizing constant and is defined over a high-dimensional parameter space, it is impractical to approximate the update direction Δ_* with samples from p_*. Instead, we use self-normalized importance sampling with a proposal distribution $q(\Delta|\theta)$:

$$\Delta_* = \mathop{\mathbb{E}}_{\Delta \sim q(\Delta|\theta)}\left[\frac{p_*(\Delta|\theta;\alpha)}{q(\Delta|\theta)}\Delta\right] \quad (3)$$

$$\approx \sum_{k=1}^{K} \frac{w(\Delta_k)}{\sum_{k=1}^{K} w(\Delta_k)} \Delta_k = \Delta_{\text{MGS}}, \quad (4)$$

where $\Delta_k \sim q(\Delta|\theta)$, each $w(\Delta_k)$ is $\frac{\exp(\alpha(C(\theta) - C(\theta+\Delta_k)))}{\tilde{q}(\Delta_k|\theta)}$, and $q \propto \tilde{q}$. This update direction equals Δ_* in the limit: $\mathbb{P}\left(\lim_{K\to\infty} \Delta_{\text{MGS}} = \Delta_*\right) = 1$. [1]

The sample complexity of such a random-search method is known to depend on the dimensionality of the sample space (Sener and Koltun 2020), thus it is crucial to choose a good proposal distribution. Our contribution is a proposal distribution for use in sequence generation, where we have access to the maximum likelihood gradient $\nabla_\theta \mathcal{L}_{\text{MLE}}$. Specifically, we propose a mixture of two Gaussians, whose components are centered at the origin and at the maximum-likelihood gradient, respectively:

$$q_{\text{MGS}}(\Delta|\theta) = \mathcal{N}(\Delta|0, I\sigma^2) \cdot \pi + \mathcal{N}(\Delta|\nabla_\theta \mathcal{L}_{\text{MLE}}, I\sigma^2) \cdot (1 - \pi), \quad (5)$$

where $\pi \in [0, 1]$ is a mixture parameter that we set to 0.5 in practice. Given a batch of examples, we compute the gradient of the maximum likelihood loss, sample candidate directions from the proposal distribution (5), then evaluate the task loss of each candidate and form the update direction (3). Algorithm 1 summarizes the procedure, called **maximum-likelihood guided parameter search** (MGS).

3 Other Task Loss Minimization Methods

Comparison with policy gradient. Policy gradient (PG) methods such as REINFORCE (Williams 1992) consist of the objective and gradient estimator:

$$C_{\text{PG}}(\theta) = \mathop{\mathbb{E}}_{(X,Y)\sim\mathcal{D}} \mathbb{E}_{\hat{Y}\sim p_\theta(\cdot|X)}\left[c(\hat{Y}, Y)\right], \quad (6)$$

$$\nabla_\theta^{\text{PG}} = \mathbb{E}_{\hat{Y}\sim p_\theta(\cdot|X)}\left[c(\hat{Y}, Y)\nabla_\theta \log p_\theta(\hat{Y}|X)\right]. \quad (7)$$

The policy gradient objective contains an expectation over the output distribution $p_\theta(\cdot|X)$, unlike the objective optimized by MGS (Equation 1). In particular, computing the PG objective involves decoding with ancestral sampling, while the objective (1) uses an arbitrary decoding algorithm. Naturally, approximating the policy gradient also uses ancestral sampling

[1] See the Appendix for a review of self-normalized importance sampling.

Algorithm 1: MLE-guided parameter search (MGS).

```
Given: Batch ${X_i, Y_i}_{i=1}^B$, model $p_\theta$, decoding
        algorithm $\mathcal{F}$, task-loss $c(\hat{Y}, Y)$.
Hyperparams: Number of candidates $K$,
        temperature $\alpha$, noise level $\sigma^2$.
Output: Update direction $\Delta_{\text{MGS}}$.
$\{\hat{Y}_i\} = \mathcal{F}(\theta, \{X_i\})$    // decode
$C(\theta) = \frac{1}{B}\sum_{i=1}^B c(\hat{Y}_i, Y_i)$    // eqn. 1
$\nabla_\theta \mathcal{L}_{\text{MLE}} = \texttt{backprop}(\mathcal{L}_{\text{MLE}}(\theta; \{X_i, Y_i\}))$
for $k \in 1, \ldots, K$ do
    $\Delta_k \sim q_{\text{MGS}}(\cdot|\theta, \nabla_\theta \mathcal{L}_{\text{MLE}}, \sigma^2)$    // eqn. 5
    $\{\hat{Y}_i\} = \mathcal{F}(\theta + \Delta_k, \{X_i\})$    // decode
    $C(\theta + \Delta_k) = \frac{1}{B}\sum_{i=1}^B c(\hat{Y}_i, Y_i)$    // eqn. 1
    $w(\Delta_k) = \frac{\exp(\alpha(C(\theta) - C(\theta + \Delta_k)))}{q_{\text{MGS}}(\Delta_k|\theta)}$
$\Delta_{\text{MGS}} = \sum_{i=1}^K \frac{w(\Delta_k)}{\sum_{k'} w(\Delta_{k'})}\Delta_k$    // eqn. 3
```

instead of the algorithm used at inference time (e.g. greedy or beam search). To contrast this with maximum-likelihood guided parameter search, we formalize the sampling and examine the per-sequence gradient.

Ancestral sampling decodes a sequence by sampling autoregressively from the model's per-step categorical distributions. Given noise $\epsilon \sim \mathcal{U}(0, 1)$, ancestral sampling, which consists of repeated categorical sampling $\hat{y}_t \sim p_\theta(\cdot|\hat{y}_{<t}, X)$, can be written as a deterministic function $\hat{Y} = \mathcal{F}_{\text{anc}}(\theta, X, \epsilon)$. The policy gradient estimator is an expectation over the noise used to produce the categorical samples,

$$\nabla_\theta^{\text{PG}} = \mathbb{E}_\epsilon \left[c\left(\mathcal{F}_{\text{anc}}(\theta, X, \epsilon), Y\right) \nabla_\theta \log p_\theta(\mathcal{F}_{\text{anc}}(\theta, X, \epsilon))\right].$$

Maximum-likelihood guided parameter search uses any arbitrary decoding algorithm, e.g. $\hat{Y} = \mathcal{F}_{\text{greedy}}(\theta, X)$, which can be chosen to be the same algorithm used at evaluation time. The MGS estimator is an expectation over noise in the *parameter space*, $\nabla_\theta^{\text{MGS}} =$

$$\mathbb{E}_{\epsilon \sim q}\left[\hat{w}(\epsilon)\exp\left(\alpha(c(\mathcal{F}(\theta, X), Y) - c(\mathcal{F}(\theta + \epsilon, X), Y)\right)\epsilon\right],$$

where we consider a single example and rewrite the MGS update (3) in order to illustrate how the use of noise and the decoding algorithm differ from policy gradient. See the Appendix for the derivation. In short, policy gradient uses each parameter θ to sample multiple sequences for each input, while MGS samples multiple parameters, and uses each to decode a single sequence per input.

Comparison with minimum risk training. Minimum risk training (MRT) (Shen et al. 2016) approximates the policy gradient objective (6) as,

$$C_{\text{MRT}}(\theta) = \underset{(X,Y)\sim\mathcal{D}}{\mathbb{E}} \mathbb{E}_{\hat{Y}\sim q_\theta(\cdot|X,S)}\left[c(\hat{Y}, Y)\right], \quad (8)$$

$$q_\theta(Y|X, S) = \begin{cases} \frac{p_\theta(Y|X)^\alpha}{Z_\theta(X,S)}, & \text{if } Y \in S, \\ 0, & \text{otherwise,} \end{cases} \quad (9)$$

where $S = \{\hat{Y}_1, \ldots, \hat{Y}_k\}$ is a set of candidate output sequences, and $Z_\theta(X, S) = \sum_{Y\in S} p_\theta(Y|X)^\alpha$. There are no importance weights, and q_θ is not a valid proposal, unlike q_{MGS}. The gradient is,[2]

$$\nabla_\theta C_{\text{MRT}} = \alpha\left[\mathbb{E}_{q_\theta}\left[c(\hat{Y}, Y)\nabla_\theta \log p_\theta(\hat{Y}|X)\right] - \mathbb{E}_{q_\theta}\left[c(\hat{Y}, Y)\right]\mathbb{E}_{q_\theta}\left[\nabla_\theta \log p_\theta(\hat{Y}|X)\right]\right], \quad (10)$$

where $\mathbb{E}_{q_\theta}$ denotes $\mathbb{E}_{\hat{Y}\sim q_\theta(\cdot|X,S)}$. The MRT gradient consists of the policy gradient, minus a term that includes the score function and the expected loss. Minimum risk training can incorporate the maximum likelihood gradient by including the ground truth sequence Y^* as a candidate,

$$\nabla_\theta C_{\text{MRT}} = \alpha[(w(Y^*) - \bar{w}(Y^*))\nabla_\theta \log p_\theta(Y^*|X) + \sum_{\hat{Y}\in S\setminus Y^*}\left(w(\hat{Y}) - \bar{w}(\hat{Y})\right)\nabla_\theta \log p_\theta(\hat{Y}|X)]$$

where $w(Y') = c(Y', Y)q_\theta(Y'|X, S)$, and $\bar{w}(Y') = \mathbb{E}_{Y''\sim q_\theta}[c(Y'', Y)]\, q_\theta(Y'|X, S)$. Unlike MGS, the other candidate directions in MRT are not related to the maximum-likelihood gradient. Instead, the candidates are determined by action-space sampling, similar to policy gradient.

Pooled task losses. PG and MRT both sample in the action space (i.e. vocabulary), while the proposed MGS samples in the parameter space. This difference affects the *amount of supervision* that is used to weight each candidate update direction. To see this, consider a minibatch $\{X_n, Y_n\}_{n=1}^N$. The policy gradient estimator with K samples per batch element is,

$$\nabla_\theta^{\text{PG}} = \frac{1}{NK}\sum_{n,k} c(\hat{Y}_n^{(k)}, Y_n)\nabla_\theta \log p_\theta(\hat{Y}_n^{(k)}|X_n), \quad (11)$$

where $\hat{Y}_n^{(k)}$ is a sampled sequence. Policy gradient uses a *single* sequence loss to weight each candidate update direction. A similar inspection reveals that MRT shares this property. On the other hand, MLE-guided parameter search,

$$\nabla_\theta^{\text{MGS}} = \sum_k \left[\hat{w}(\Delta_k)\exp\left(\alpha(C(\theta) - C(\theta + \Delta_k))\right)\Delta_k\right],$$

weights each candidate direction using a loss $C(\cdot)$ computed over the entire minibatch (see Equation 1). This has the effect of 'densifying' the sparse loss by pooling the losses from multiple examples.

4 Related Work

Sequence-level training for NLP. Sequence-level training methods based on policy gradient have been applied to several NLP tasks (Liu et al. 2017; Ziegler et al. 2019). Related methods use policy gradient with generative adversarial networks (GAN) (Yu et al. 2017). Policy gradient methods often face training instability and sensitivity to hyper-parameters (Henderson et al. 2018), and GAN methods under-perform maximum likelihood (Caccia et al. 2020).

[2]See the Appendix for the derivation.

	LM ↓	Edit ↓	Nonterm ↓	Repetition ↓	Avg. len.	Perplexity↓
MLE	157.6 (13.5)	.945 (.008)	.344 (.063)	.530 (.062)	228.1 (33.3)	21.3 (0.2)
MGS-LM	64.9 (2.09)	.937 (.002)	.012 (.003)	.046 (.009)	22.8 (2.2)	22.0 (0.1)
MRT-LM (+MLE 0.1)	57.4 (.967)	.948 (.002)	.013 (.004)	.023 (.005)	16.9 (2.3)	25.8 (1.7)
PG-LM (+MLE 0.1)	48.4 (.523)	.967 (.004)	.000 (.000)	.002 (.002)	3.8 (1.0)	30.7 (7.3)
MGS-edit	78.2 (1.38)	.925 (.003)	.037 (.008)	.098 (.007)	44.0 (2.2)	21.6 (0.1)
MRT-edit (+MLE 0.3)	138.7 (11.1)	.929 (.011)	.227 (.094)	.472 (.066)	178.4 (43.1)	23.2 (1.0)
PG-edit (+MLE 0.1)	103.0 (4.05)	.904 (.001)	.051 (.016)	.246 (.027)	68.5 (8.2)	24.5 (0.8)
Human	–	–	.000	.011	107.9	–

Table 1: Text completion results (GPT-2, Wikitext-103 test set), reported as `mean (stdev)` using 5 random seeds. Policy gradient (PG) and minimum risk training (MRT) are stochastically mixed with MLE and reported as (+MLE α), with the α values selected based on the task loss. Results here are with greedy decoding; see the Appendix for ancestral sampling.

Reward augmented maximum-likelihood (RAML) (Norouzi et al. 2016) maximizes the likelihood of sequences that are sampled proportional to their rewards, which in practice relies on a sampling method designed for a specific task loss. Our method weights parameter, rather than sequence, samples proportional to their rewards. Minimum risk training originated in statistical machine translation (Och 2003) and was applied to end-to-end neural machine translation (Shen et al. 2016; Edunov et al. 2018). Other approaches train a greedy decoder given a learned model (Gu, Cho, and Li 2017), which is a different setting than ours.

A separate family of methods, including globally normalized models, (Sountsov and Sarawagi 2016), energy-based models (Deng et al. 2020), unlikelihood training (Welleck et al. 2020b; Li et al. 2020), and beam search optimization (Wiseman and Rush 2016), incorporate sequence-level scores without reference to an external reward function.

Drawbacks of MLE in NLP. Several studies investigate drawbacks of maximum likelihood training, including label bias (Lafferty, McCallum, and Pereira 2001), exposure bias (Daumé, Langford, and Marcu 2009; Ross, Gordon, and Bagnell 2011; Bengio et al. 2015), and loss mismatch (Lee et al. 2020). Neural machine translation models trained with maximum likelihood have been shown to exhibit decreased performance with increased beam size (Koehn and Knowles 2017; Ott et al. 2018) and a bias towards short sequences (Sountsov and Sarawagi 2016; Stahlberg and Byrne 2019), which have been attributed to label bias due to local normalization (Murray and Chiang 2018).

In open-ended text generation, MLE-trained models have been observed to produce non-terminating sequences (Welleck et al. 2020a), degenerate repetition (Holtzman et al. 2019; Welleck et al. 2020b), and a mismatched unigram distribution (Li et al. 2020). These motivate our investigation of an alternative training procedure.

Black-box optimization. Our approach is motivated by black-box optimization methods, specifically those based on random search (Matyas 1965; Rechenberg 1978). Several methods augment random search with auxiliary information (Lehman et al. 2018; Pourchot and Sigaud 2019). Related to our method are learned manifold random search (Sener and Koltun 2020) which requires an inner optimization to learn parameters of a search manifold, and guided evolutionary strategies (Maheswaranathan et al. 2019) which uses surrogate directions to modify the search distribution's covariance; their method requires QR decomposition and was evaluated on synthetic and unrolled optimization tasks with smaller networks than those we consider.

5 Experiments

5.1 Text Completion with GPT-2

First, we evaluate MGS on a text completion task, which has previously been used to evaluate the effectiveness of sequence models (e.g. Sutskever, Martens, and Hinton (2011); Radford et al. (2018); Holtzman et al. (2019); Welleck et al. (2020b)). The task consists of decoding a continuation $\hat{Y} = \mathcal{F}(\theta, X)$ given a prefix $X = (x_1, \ldots, x_k)$.

In this task, neural language models such as GPT-2 (Radford et al. 2018) exhibit degenerate repetition (Holtzman et al. 2019) and non-termination with greedy decoding; (Welleck et al. 2020a) conjectured that the lack of a decoding algorithm in maximum-likelihood training is the cause of the latter. We evaluate whether MGS, which uses a decoding algorithm during training, can alleviate these issues.[3]

Experimental setup. We use the Wikitext-103 dataset (Merity et al. 2016), a large-scale collection of Wikipedia articles containing over 100 million words that has been used for language modeling (Baevski and Auli 2019) and text completion (Welleck et al. 2020b). We model individual sequences by splitting the corpus according to its newline boundaries, then splitting each sequence into a context X and continuation Y, resulting in a dataset of (X, Y) pairs. Each continuation ends in a special $\langle\text{eos}\rangle$ token. We use a context size of $k = 10$ tokens, discarding sequences that are length k or shorter. The resulting dataset consists of 874,556 training, 1,896 validation, and 2,162 test pairs.

We use GPT-2 117M (Radford et al. 2018), a transformer (Vaswani et al. 2017) language model with a byte-level BPE vocabulary of 50k tokens, pre-trained with maximum likelihood on WebText, a dataset of scraped web pages (see

[3]Code available at https://github.com/wellecks/mgs.

Prefix	*The British organized an expedition in early 1776 for*
MLE	the defense of the French colonies in the Caribbean. The expedition was led by Captain William Henry St. Clair, who had been appointed to command the expedition. The expedition was led by Captain William Henry St. Clair, who had been appointed to command the expedition in 1776. The expedition $\rightarrow \infty$
PG-LM	the Continental Army. $\langle$eos$\rangle$
PG-edit	the defence of the French colony of New France. The expedition was to be led by Captain Samuel Frederick Gray, who was to become the first commander of the expedition. $\langle$eos$\rangle$
MGS-LM	the defense of the Ohio River. The expedition was led by Colonel John C. St. Clair, who had been appointed to command the expedition. $\langle$eos$\rangle$
MGS-edit	the defense of the French colonies. The expedition was led by Lieutenant Colonel John Henry, who was promoted to lieutenant colonel in 1776. $\langle$eos$\rangle$
Prefix	*On The Joshua Tree Tour, " Where the Streets*
MLE	Have No Name " was performed by the band in a medley with " The Way You Make Me Feel ", " The Way You Make Me Feel ", " The Way You Make Me Feel ", and " The Way You Make Me Feel ". The medley was included on the DVD release of the tour. $\langle$eos$\rangle$
PG-LM	Have No Name " $\langle$eos$\rangle$
PG-edit	Have No Name " was performed by the band. The performance was recorded in the studio in Los Angeles, California, and was recorded in the same room as the band's previous two singles, " The Way " and " The Way ". $\langle$eos$\rangle$
MGS-LM	Have No Name " was performed by the band. $\langle$eos$\rangle$
MGS-edit	Have No Name " was performed by the band in a medley with" The Day the World Gets 'Round ". $\langle$eos$\rangle$

Table 2: Example greedy continuations (GPT-2, Wikitext-103 validation set).

Radford et al. (2018) for details). We fine-tune the pretrained GPT-2 model using MLE and select the model state with the lowest validation perplexity. We then continue with MGS beginning at the selected model state. We use 4 candidates, and compute training task loss with a max decoding length of 1.3 times the ground-truth length. Models are evaluated with a max decoding length of 500 tokens. See the Appendix for more details.

For the MRT and PG baselines we finetune using 4 samples. For policy gradient we used an exponential moving average baseline. Each method is stochastically mixed with MLE according to a hyper-parameter $\alpha \in [0, 1]$: given a training batch, we draw $z \sim \text{Bernoulli}(\alpha)$ and use MLE when z is zero. We performed a grid search using $\alpha \in \{0.1, 0.3, 0.5\}$, selecting α based on the validation task loss that the model is optimizing. In the Appendix we also report results for MRT and PG without stochastically mixing MLE and an ablation of the choice of MRT candidates.

Our main results are reported with greedy decoding; refer to the Appendix for results with ancestral sampling.

Task losses. We experiment with two sequence-level task losses. We define a language modeling (**LM**) loss which scores each sequence with a fixed language model:

$$c_{\text{LM}}(\hat{Y}) = -\log p_{\text{score}}(\hat{Y}). \quad (12)$$

Intuitively, minimizing this loss adjusts the MLE model to work well with greedy decoding. We use the fine-tuned GPT-2 model as p_{score}, which is the starting point of MGS training. As a task loss that incorporates the ground-truth sequence, we use **edit** distance $c_{\text{edit}}(\hat{Y}, Y)$, normalized by $|Y|$.

Metrics. Motivated by prior work which showed that MLE-trained LMs produce repetitive, non-terminating text with greedy decoding, we measure the portion of duplicate n-grams (we use $n = 4$) (Welleck et al. 2020b) and the proportion of non-terminating continuations (Welleck et al. 2020a):

$$\textbf{repetition}(\hat{Y}) = 1 - |\text{unique n-grams}| / |\text{n-grams}|,$$
$$\textbf{nonterm}(\hat{Y}) = \mathbb{I}\left[\langle\text{eos}\rangle \notin \hat{Y}\right].$$

We also report the task loss, average length of the generated continuations, and the perplexity.

Effect on sequence-level task loss. Table 1 shows the task losses and metrics for the baseline fine-tuned model (MLE) and each model trained with MGS to optimize the indicated task loss (MGS-*loss*). The baseline has the highest task losses, and a high degree of non-termination (.387) and repetition (.538). MGS-LM substantially reduces the LM task loss (59.1), along with non-termination (.012) and repetition (.035).

MGS-edit achieves lower edit distance (.928) than MGS-LM, while also substantially reducing LM task loss, non-termination, and repetition. Both MGS variants result in short sequences, especially MGS-LM, which is expected due to the bias towards short sequences in MLE-trained LMs (Stahlberg and Byrne 2019).

Table 2 shows representative continuations (see the Appendix for more). The first example shows how MGS can fix non-termination, and the second shows how MGS reduces repetition in a terminating sequence.

PG & MRT comparison. The MRT-LM and PG-LM methods result in a lower LM loss than MLE and MGS-LM. However, the perplexity is higher than that of MGS-LM (25.8 and 30.7 vs. 22.0), with a larger standard deviation (1.7 and 7.3 vs. 0.1). Policy gradient finds a solution with very short sequences (average length 3.8). For edit distance, MRT-edit underperforms MGS-edit on average (.929 vs .925), with higher nontermination, repetition, and perplexity. PG-edit achieves the best edit distance, though with higher repetition (.246 vs. .098) and perplexity (24.5 vs. 21.6) than MGS.

Prefix	*The manga was licensed for English language release by Del*	Task-Loss
$\mathcal{N}_{\textbf{MLE}}$	Rey in the United States, and was released in the United Kingdom in the United States in the first volume of the series, and in the United States in the second, and third, volumes of the series, in the United States in the first and second volumes of the first and second volumes of the second ...	137.8
$\mathcal{N}_0$	Rey Manga in the United States. $\langle$eos$\rangle$	51.2

Table 3: Example sequences decoded from sampled candidates, showing the component that the candidate was sampled from.

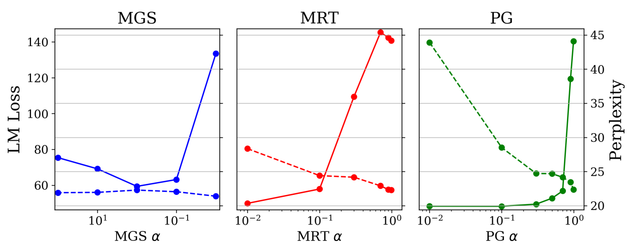

Figure 1: Task loss (solid) and perplexity (dashed) as α varies.

We also report results with ancestral sampling in the Appendix. We observe similar trends - MGS performs comparably to MRT but with better perplexity, and PG finds a degenerate short-sequence solution under the LM loss.

In summary, all three methods improve the task loss, and MGS does so while having a favorable balance across the other metrics (e.g. perplexity, repetition). We find that α trades perplexity for task loss minimization in PG and MRT, while MGS finds solutions that are much more stable in terms of perplexity, as shown in Figure 3. Our conclusion is that MGS is an attractive alternative to mixing minimum risk training and policy gradient with maximum likelihood training for the problem of text generation.

MGS candidate analysis. First, we perform an ablation of the proposal distribution q_{MGS}, which is a mixture of two components. We compare against only using the zero-mean (q_{zero}) or MLE-mean (q_{MLE}) components as proposals, and find that the training loss only decreases when both components in the q_{MGS} mixture are included. The task loss on the validation set (see Appendix) is analogous.

Next, we inspect how the pooled task loss varies between the sampled candidates. The standard deviation in candidate weights $w(\Delta_k)$ during training fall within 0.35-0.45, implying that each proposal samples candidates with varied task losses. As a qualitative example, we sample two candidates from q_{MGS} at the end of training, decode a batch of sequences with each candidate, and in Table 3 show an example sequence and the pooled loss. The MLE candidate's sequence is non-terminating, while the zero candidate decodes a shorter sequence and has a lower pooled loss.

We investigate which candidates contribute to the update direction over the course of training by showing the total weight of MLE-component candidates in Figure 2 ($\alpha = 1.0$). The MLE candidates are highly weighted at the beginning of training, only contributing occasionally thereafter. Finally, we analyze the effect of the α hyper-parameter, which controls the entropy of the candidate weights. As α decreases, the candidate weights are smoothed towards uniform, which allocates more weight to the MLE candidates, as seen in Figure 2. Performance decreases when the weights are either too uniform or too peaked, as seen in Figure 3.

5.2 Machine Translation

Experimental setup. We experiment on the IWSLT '14 German to English task (Cettolo et al. 2014) using a standard experimental setup from the fairseq (Ott et al. 2019) repository which we detail in the Appendix. We train the MLE baseline and a MGS models with the same hyper-parameters. We use 4 candidates and a grid search over noise ($\{0.01, 0.1, 1.0\}$) and α ($\{1.0, 10.0, 100.0\}$). The noise is scaled by $\frac{1}{|\theta|}\|\nabla_\theta \mathcal{L}_{\text{MLE}}\|_1$.

For fine-tuning, we use a batch size of 16k tokens, and accumulate gradients for 4 iterations. We select $\alpha = 100.0$ and noise 1.0 for all MGS fine-tuning based on a grid search with MGS-SBLEU. For training from scratch, we select α 1.0 and noise 1.0. All models are selected by validation BLEU using beam search with width 5.

Results. Results for the baseline, MGS fine-tuned models, and models trained from scratch with MGS are in Table 4, along with prior work that fine-tuned with minimum risk training in Table 5.

The fine-tuned MGS-SBLEU model improves BLEU over the baseline MLE model (+0.32 test) at a comparable level to the improvement from fine-tuning with MRT (+0.24 and +0.50 test), with MGS-METEOR showing a similar gain. All of the fine-tuned MGS models improve the sequence-level task losses that are computed with greedy decoding (SBLEU, METEOR, EDIT), with each model achieving the best score on its associated task loss. MGS-EDIT shows the largest difference, underperforming on BLEU yet outperforming the baseline by a full point on EDIT.

The MGS model trained from scratch outperforms the baseline MLE model on BLEU, though by a smaller margin than the fine-tuned models. We observed the validation BLEU over time for MGS and the baseline, indicating that they arrive at their performance levels via different paths. Figure 4 shows the proportion of MLE candidates that had the highest weight out of the four candidates sampled from the mixture (q_{MGS}), and Table 3 shows an example sequence decoded from a candidate sampled from each component.

Candidates sampled from the zero-component tend to locally improve the task loss more than those from the MLE component. However, we find that at the end of training, roughly 46% of the weight comes from the MLE candidates.

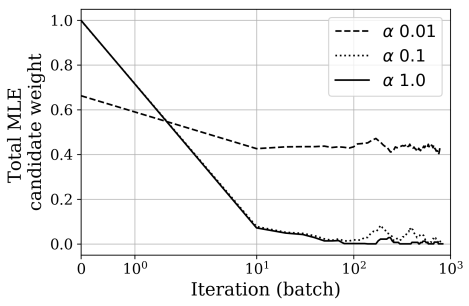

Figure 2: Weight of candidates from the MLE component.

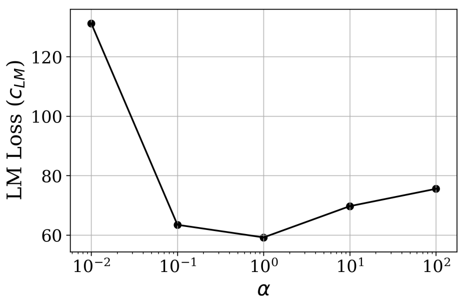

Figure 3: Validation sequence loss as α varies (MGS-LM).

	Valid				Test			
	BLEU↑	SBLEU↑	MET.↑	EDIT↓	BLEU↑	SBLEU↑	MET.↑	EDIT↓
MLE	36.00	36.22	63.82	47.88	34.71	35.67	62.19	50.74
MGS-SBLEU	36.22	**36.58**	64.08	47.25	**35.03**	35.89	62.2	50.23
MGS-METEOR	**36.26**	36.51	**64.13**	47.35	34.98	**35.97**	**62.49**	50.29
MGS-EDIT	35.73	36.42	63.73	**46.83**	34.73	35.95	62.04	**49.45**
MGS-SBLEU (train)	36.19	36.13	63.65	48.40	34.80	35.32	61.95	51.38

Table 4: Machine translation results (IWSLT '14 De→En). BLEU is computed with beam search (width 5). SBLEU, METEOR, and EDIT are computed with greedy decoding to match the training conditions.

	Valid	**Test**
W & S (2020) (MLE)	-	34.70
W & S (2020) (MRT)	-	35.20
Ed. (2018) (MLE)	33.11	32.21
Ed. (2018) (MRT)	33.55	32.45

Table 5: IWSLT '14 De→En with minimum risk (BLEU).

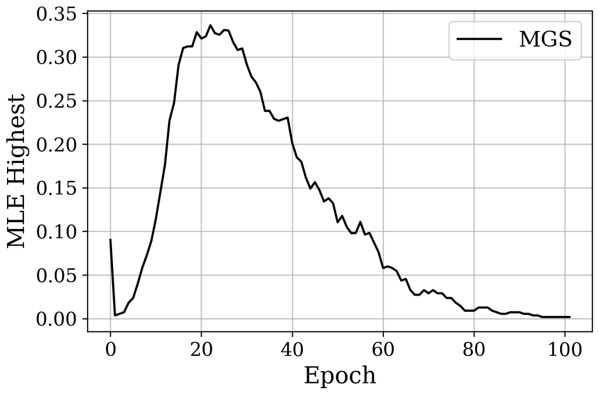

Figure 4: Proportion of highest-weight MLE candidates.

We attribute this to the variations in weight between the candidates, which are smaller than those in the text completion task, with a standard deviation ranging from .005 to .025 over the course of training.

The task losses used in MT are highly concentrated on matching a reference translation and are similar to the 0-1 loss to which the log loss (MLE) is a proxy. We suspect that it is more difficult to find candidates that improve substantially over MLE, resulting in smaller improvements than in text completion.

6 Conclusion

We propose maximum-likelihood guided parameter search (MGS), a training method for optimizing an arbitrary sequence-level task loss. MGS samples update directions and weights them according to their improvement in task loss. Key to our method is a proposal distribution which either performs random search around the current parameter or around the maximum-likelihood gradient.

MGS substantially reduced non-termination and repetition in a text completion task, and outperformed maximum likelihood on machine translation, with fine-tuning and when trained from scratch. MGS incorporates the maximum-likelihood gradient into its objective, which led to solutions that were more stable with respect to perplexity than those found by policy and minimum risk training, which required MLE as an auxiliary loss in practice. The results suggest that MGS is a promising alternative to minimum risk and policy gradient, and improving upon its simple, yet effective, form of exploration is a fruitful direction for future research.

References

Baevski, A.; and Auli, M. 2019. Adaptive Input Representations for Neural Language Modeling. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=ByxZX20qFQ.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Courville, A.;

Pineau, R. L. J.; and Bengio, Y. 2017. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. URL http://arxiv.org/abs/1409.0473.

Banerjee, S.; and Lavie, A. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. ISSN 10495258.

Caccia, M.; Caccia, L.; Fedus, W.; Larochelle Google Brain, H.; Mila, M.; Research, F. A.; and Canada CIFAR Chair, M. A. 2020. Language GANs Falling Short. In *International Conference on Learning Representations (ICLR)*.

Cettolo, M.; Niehues, J.; Stüker, S.; Bentivogli, L.; and Federico, M. 2014. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *Proceedings of the 11th International Workshop on Spoken Language Translation*.

Choshen, L.; Fox, L.; Aizenbud, Z.; and Abend, O. 2020. On the Weaknesses of Reinforcement Learning for Neural Machine Translation. In *International Conference on Learning Representations (ICLR)*.

Daumé, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine Learning* ISSN 08856125. doi:10.1007/s10994-009-5106-x.

Deng, Y.; Bakhtin, A.; Ott, M.; Szlam, A.; and Ranzato, M. 2020. Residual Energy-Based Models for Text Generation. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=B1l4SgHKDH.

Edunov, S.; Ott, M.; Auli, M.; Grangier, D.; and Ranzato, M. 2018. Classical Structured Prediction Losses for Sequence to Sequence Learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 355–364. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-1033. URL https://www.aclweb.org/anthology/N18-1033.

Gu, J.; Cho, K.; and Li, V. O. 2017. Trainable greedy decoding for neural machine translation. In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*. ISBN 9781945626838. doi:10.18653/v1/d17-1210.

Hansen, N.; and Ostermeier, A. 2001. Completely derandomized self-adaptation in evolution strategies. doi:10.1162/106365601750190398.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. ISBN 9781577358008.

Holtzman, A.; Buys, J.; Forbes, M.; and Choi, Y. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* .

Koehn, P.; and Knowles, R. 2017. Six Challenges for Neural Machine Translation. doi:10.18653/v1/w17-3204.

Lafferty, J.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning* ISSN 1750-2799. doi:10.1038/nprot.2006.61.

Lee, J.; Tran, D.; Firat, O.; and Cho, K. 2020. On the Discrepancy between Density Estimation and Sequence Generation. *arXiv preprint arXiv:2002.07233* .

Lehman, J.; Chen, J.; Clune, J.; and Stanley, K. O. 2018. Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, 117–124. New York, NY, USA: Association for Computing Machinery. ISBN 9781450356183. doi:10.1145/3205455.3205473. URL https://doi.org/10.1145/3205455.3205473.

Li, M.; Roller, S.; Kulikov, I.; Welleck, S.; Boureau, Y.-L.; Cho, K.; and Weston, J. 2020. Don't Say That! Making Inconsistent Dialogue Unlikely with Unlikelihood Training.

Liu, S.; Zhu, Z.; Ye, N.; Guadarrama, S.; and Murphy, K. 2017. Improved Image Captioning via Policy Gradient optimization of SPIDEr. In *Proceedings of the IEEE International Conference on Computer Vision*. ISBN 9781538610329. ISSN 15505499. doi:10.1109/ICCV.2017.100.

Maheswaranathan, N.; Metz, L.; Tucker, G.; Choi, D.; and Sohl-Dickstein, J. 2019. Guided evolutionary strategies: augmenting random search with surrogate gradients. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 4264–4273. Long Beach, California, USA: PMLR. URL http://proceedings.mlr.press/v97/maheswaranathan19a.html.

Mania, H.; Guy, A.; and Recht, B. 2018. Simple random search provides a competitive approach to reinforcement learning. Technical report. URL https://github.com/modestyachts/ARS.

Matyas, J. 1965. Random Optimization. *Automat. i Telemekh* .

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models. *ArXiv* abs/1609.07843.

Murray, K.; and Chiang, D. 2018. Correcting Length Bias in Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, 212–223. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/W18-6322. URL https://www.aclweb.org/anthology/W18-6322.

Norouzi, M.; Bengio, S.; Chen, Z.; Jaitly, N.; Schuster, M.; Wu, Y.; and Schuurmans, D. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems*. ISSN 10495258.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. doi:10.3115/1075096.1075117.

Ott, M.; Auli, M.; Grangier, D.; and Ranzato, M. 2018. Analyzing uncertainty in neural machine translation. In *35th International Conference on Machine Learning, ICML 2018*. ISBN 9781510867963.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-j. 2002. BLEU : a Method for Automatic Evaluation of Machine Translation. *Computational Linguistics* .

Pourchot; and Sigaud. 2019. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=BkeU5j0ctQ.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2018. Language Models are Unsupervised Multitask Learners. In *OpenAI*. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Rechenberg, I. 1978. Evolutionsstrategien. doi:10.1007/978-3-642-81283-5_8.

Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Journal of Machine Learning Research*. ISSN 15324435.

Rubinstein, R. 1999. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability* ISSN 1387-5841. doi:10.1023/A:1010091220143.

Rückstieß, T.; Sehnke, F.; Schaul, T.; Wierstra, D.; Sun, Y.; and Schmidhuber, J. 2010. Exploring Parameter Space in Reinforcement Learning. *Paladyn, Journal of Behavioral Robotics* ISSN 2081-4836. doi:10.2478/s13230-010-0002-4.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for sentence summarization. In *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*. ISBN 9781941643327.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. In *arXiv preprint*.

Sener, O.; and Koltun, V. 2020. Learning to Guide Random Search. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=B1gHokBKwS.

Shen, S.; Cheng, Y.; He, Z.; He, W.; Wu, H.; Sun, M.; and Liu, Y. 2016. Minimum risk training for neural machine translation. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*. ISBN 9781510827585. doi:10.18653/v1/p16-1159.

Sountsov, P.; and Sarawagi, S. 2016. Length bias in encoder decoder models and a case for global conditioning. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*. ISBN 9781945626258. doi:10.18653/v1/d16-1158.

Stahlberg, F.; and Byrne, B. 2019. On NMT Search Errors and Model Errors: Cat Got Your Tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3354–3360. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1331. URL https://www.aclweb.org/anthology/D19-1331.

Sutskever, I.; Martens, J.; and Hinton, G. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*. ISBN 9781450306195.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. ISSN 10495258.

Vinyals, O.; Quoc, G.; and Le, V. 2015. A Neural Conversational Model. In *ICML Deep Learning Workshop*.

Wang, C.; and Sennrich, R. 2020. On Exposure Bias, Hallucination and Domain Shift in Neural Machine Translation URL http://arxiv.org/abs/2005.03642.

Welleck, S.; Kulikov, I.; Kim, J.; Pang, R. Y.; and Cho, K. 2020a. Consistency of a Recurrent Language Model With Respect to Incomplete Decoding. *arXiv preprint arXiv:2002.02492* .

Welleck, S.; Kulikov, I.; Roller, S.; Dinan, E.; Cho, K.; and Weston, J. 2020b. Neural Text Generation With Unlikelihood Training. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=SJeYe0NtvH.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* ISSN 0885-6125. doi:10.1007/bf00992696.

Wiseman, S.; and Rush, A. M. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*. ISBN 9781945626258. doi:10.18653/v1/d16-1137.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-Tuning Language Models from Human Preferences URL http://arxiv.org/abs/1909.08593.