Straight to the Gradient: Learning to Use Novel Tokens for Neural Text Generation

Xiang Lin [1] **Simeng Han** [1] **Shafiq Joty** [1 2]

Abstract

Advanced large-scale neural language models have led to significant success in many language generation tasks. However, the most commonly used training objective, Maximum Likelihood Estimation (MLE), has been shown problematic, where the trained model prefers using dull and repetitive phrases. In this work, we introduce ScaleGrad, a modification straight to the gradient of the loss function, to remedy the degeneration issue of the standard MLE objective. By directly maneuvering the gradient information, ScaleGrad makes the model learn to use novel tokens. Empirical results show the effectiveness of our method not only in open-ended generation, but also in directed generation tasks. With the simplicity in architecture, our method can serve as a general training objective that is applicable to most of the neural text generation tasks.

1. Introduction

Text generation has been one of the most important research problems in natural language processing (NLP). Thanks to the advances in neural architectures, models are now capable of generating texts that are of better quality than before (Brown et al., 2020). However, despite the countless efforts that have been made to improve neural architectures, models trained with the standard Maximum Likelihood Estimation (MLE) objective are known to prefer generating dull and highly repetitive texts. For instance, in *open-ended generation* tasks, such as story continuation or open dialogue generation, it has been observed that even with large pre-trained models like GPT-2 (Radford et al., 2019), high frequency tokens largely dominate the generation (Welleck et al., 2020; Holtzman et al., 2020). Similar observation has been reported in *directed generation* tasks such as summarization (See et al., 2017), image captioning (Melas-Kyriazi et al., 2018; Wang & Chan, 2019) and machine translation (Tu et al., 2016; Stahlberg & Byrne, 2019).

The methods proposed to solve the degeneration issues with neural text generation can be primarily categorized into two groups: (*i*) *training* based methods, which include incorporating auxiliary losses (See et al., 2017; Welleck et al., 2020; Li et al., 2020) and coverage vector (See et al., 2017; Tu et al., 2016); (*ii*) *decoding* based methods, such as stochastic beam search (Kool et al., 2019), top-k sampling (Fan et al., 2018), nucleus or top-p sampling (Holtzman et al., 2020), and inverse probability weighting (Zhang et al., 2021).

Though decoding based methods, in particular nucleus and top-k sampling, perform well in practice in open-ended generation tasks, significantly reducing the degeneration problem, they do not address the fundamental modeling issue that the token-level probabilities produced by the neural model are problematic (Welleck et al., 2020). In addition, our experiments demonstrate that sampling methods also fail to generate high-quality texts in directed generation tasks such as abstractive text summarization.

In this work, based on the known observation that the text generation models trained with MLE objective tend to generate repetitive tokens or phrases, we introduce a novel method called ScaleGrad for neural text generation training, by directly maneuvering the gradients to make the model learn to use novel tokens during training. Our method lies in the training based group, which aims to address the **fundamental modeling** problem, that is, the token-level distribution predicted by the generation model.

In a concurrent work, Wang et al. (2020) introduce a temperature scaling approach called Contextual Temperature to improve general language modeling. In this approach, the temperature value in the $\mathrm{softmax}$ function is parameterized by a neural network that is jointly trained with the main model. Though the objective of their work is not explicitly related to text degeneration, their analysis shows temperature scaling essentially changes the gradient updates that each token receives during training, which further motivates our work.

We conduct experiments with different neural architectures

[1]Nanyang Technological University, Singapore [2]Salesforce Research Asia, Singapore. Correspondence to: Xiang Lin <linx0057@e.ntu.edu.sg>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

including LSTM (Hochreiter & Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) across different tasks in opened-ended and directed text generation. Through extensive analysis we demonstrate that ScaleGrad consistently improves the generation quality according to both human evaluation and automatic metrics. Compared to other training based methods, ScaleGrad is architecturally simpler and easier to fit into current neural models (§3.2), while possessing a wide applicability to different text generation tasks (§4.2 and §5.2). The source code is available at https://github.com/shawnlimn/ScaleGrad.

2. Background

2.1. Neural text generation

The NLP tasks involving text generation can be broadly categorized into two types: *directed generation* and *open-ended generation* (Holtzman et al., 2020). In the former case, the output text can be seen as a constrained transformation of the input. Examples include text summarization, machine translation, and image captioning. In the latter case, the input context only provides a certain degree of constraints such that the model is allowed to generate the following texts with a considerable degree of freedom. Story/text continuation and dialogue generation fall in this category.

Neural models frame text generation tasks as some form of conditional language modeling, which is typically trained to maximize the log likelihood (equivalently, minimize the negative log likelihood) of the training data. The *Maximum Likelihood Estimation* or MLE objective for an input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ can be expressed as follows.

$$\mathcal{L}_{\text{MLE}} = -\sum_{t=1}^{T} \log \mathcal{P}_\theta(y_t|y_{<t}, \boldsymbol{x}) \tag{1}$$

where θ denotes model parameters, T is the length of the output sequence $\boldsymbol{y}$, and $\boldsymbol{x}$ is the task-specific input condition, *e.g.,* source document in summarization, image in image captioning, conversation history in dialogue generation and $\emptyset$ in text continuation. *Teacher Forcing* (Williams & Zipser, 1989), where current step's target token is passed as the next input to the decoder rather than the predicted token, is usually used to train the models for faster convergence.

Degeneration Degeneration has been a key problem in neural text generation models for open-ended tasks, where the model generates texts that are repetitive, overly generic (dull), incoherent and gibberish. It can happen at different levels of granularity – token, phrase, sentence and paragraph. The problem has not been mitigated even with large-scale pre-trained models like GPT-2 Large (Radford et al., 2019; Holtzman et al., 2020). Degeneration has also been observed in directed generation tasks even though the output in these tasks is confined by the input. For instance, in text summarization, most of the advanced models such as BertSum (Liu & Lapata, 2019), BART (Lewis et al., 2020) and ProphetNet (Qi et al., 2020) make use of tri-gram blocking (Paulus et al., 2018) within beam search to remove duplicate trigrams during decoding, which improves the generation quality in terms of the automatic metric. This implies that even with involvement of large-scale pre-trained models, degeneration still exists. Similar issues have been reported in machine translation (Koehn & Knowles, 2017; Stahlberg & Byrne, 2019), image-description generation (Melas-Kyriazi et al., 2018; Wang & Chan, 2019) and next utterance generation in conversations (Jiang et al., 2020).

2.2. Combating neural text degeneration

Out of the methods proposed to tackle neural text degeneration, top-k sampling (Fan et al., 2018) and nucleus sampling (Holtzman et al., 2020) stand out as representatives of decoding based methods and unlikelihood training (Welleck et al., 2020) as a representative training based method. During each decoding step, nucleus and top-k sampling use different functions to filter the candidate tokens, thus reformalizing the probability distribution. Then they sample the next token from the new distribution instead of maximizing the actual likelihood. Randomness brought by these sampling methods reduces duplicate tokens in the output. However, decoding strategy solely does not solve the underlying modeling problem with MLE training, as pointed out by Welleck et al. (2020). Our analysis in §5.2 also reveals that sampling methods fail to generate high-quality texts in directed generation tasks.

To address the issue with MLE, Welleck et al. (2020) propose the neural unlikelihood (UL) training method. During training, at each decoding step t, UL adds an auxiliary loss to the original cross entropy loss as follows.

$$\mathcal{L}_{\text{UL}}^t = \underbrace{-\log \mathcal{P}_\theta(y_t|y_{<t})}_{\text{MLE}} \underbrace{-\alpha \sum_{c \in \mathbb{C}^t} \log(1 - \mathcal{P}_\theta(c|y_{<t}))}_{\text{UL}} \tag{2}$$

where α is a hyper-parameter and $\mathbb{C}^t$ is the set of *negative tokens* at decoding step t, which is constructed by previous context tokens that are not the current token, $\mathbb{C}^t = \{y^1, \ldots, y^{t-1}\} \setminus y^t$. The auxiliary UL loss decreases the total loss based on the "unlikely" probabilities of negative tokens, thus implicitly reducing the probability assigned to the repetitive tokens. UL training targets at improving the underlying modeling problem, which accords with our goal. Therefore, we mainly compare our method with UL training.[1] We discuss how our method is different from UL training from the gradient perspective in §3.3.

[1] Welleck et al. (2020) also propose a sequence-level UL. Since our work focuses on token-level modeling, we compare with their token-level UL training in this work.

3. Methodology: learning to use novel tokens

Training a text generation model with MLE objective treats each token in the gold (ground truth) sequence equally. It has been shown that with this approach, the model exhibits the tendency to generate repetitive tokens/phrases during inference (Welleck et al., 2020; Holtzman et al., 2020). To mitigate this degeneration problem, we argue that the model should focus more on *learning to use novel tokens*, rather than treating all the tokens in a sequence equally. Our main idea is to maintain a dynamic list of novel tokens at each decoding step during training and encourage the model to learn to use tokens from this list for generation.

Formally, let $\boldsymbol{y} = (y^1, \ldots, y^t, \ldots, y^T)$ be the ground-truth (target) token sequence that the model is learning to generate in an auto-regressive manner, one token at a time. At time step t, we define the token $\tilde{y}_i^t$ in the vocabulary $\mathbb{V}$ as a **novel token**, if $\tilde{y}_i^t$ has not been generated before, *i.e.,* $\tilde{y}_i^t \notin \{y^1, \ldots, y^{t-1}\}$. By the definition, we have a dynamic set of novel tokens $\mathbb{S}_{\text{novel}}^t \subseteq \mathbb{V}$ at each decoding step t in training, which shrinks over time as new tokens are observed in the ground-truth sequence (see Appendix B for an illustration). Note that the set of *non-novel* tokens at each step (*i.e.,* $\mathbb{V} \setminus \mathbb{S}_{\text{novel}}^t$) is equivalent to the set of negative tokens $\mathbb{C}^t$ in UL (Eq. 2) except that it may contain the current target token y^t, if it was observed before. To encourage the model to focus on learning to use novel tokens, we propose an architecturally-simple yet effective method that can fit into most of the auto-regressive generation models. Our method, requiring no carefully-designed components, is derived directly from the gradient analysis of the loss function.

3.1. Gradient analysis for MLE training

Let us first consider the gradient analysis of the model trained with MLE. Let $\boldsymbol{o}^t \in \mathbb{R}^{|\mathbb{V}|}$ denote the pre-softmax scores (*i.e.,* logits) over the vocabulary at time step t, where o_i^t is the score for the token with index i. Similarly, let $p_k^t = [\text{softmax}(\boldsymbol{o}^t)]_k$ represent the probability of the ground truth token with index k in the vocabulary. The partial derivative of the MLE objective (Eq. 1) at time step t with respect to the logit o_i^t can be shown as (omitting t and 'MLE' subscript for simplicity):

$$\nabla_{o_i}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial p_k} \cdot \frac{\partial p_k}{\partial o_i} = p_i - \mathbb{1}(i = k) \tag{3}$$

where $p_i = [\text{softmax}(\boldsymbol{o})]_i$ and $\mathbb{1}(\cdot)$ is the Indicator function (derivation is given in Appendix A). Specifically, the gradient of the loss *w.r.t.* the ground truth token logit o_k is $(p_k - 1)$ and for any other token logit o_i is p_i. As the gradient-based optimization proceeds, the gradient converges to ϵ, a number that is close enough to 0. Another interpretation is that the gradient of the loss is supposed to be close to 0 around a (local) minimum. Therefore, to reach the minimum, or to make the gradient close to 0, the model would try to increase the probability of ground truth token p_k and reduce the probability of non-ground truth token p_i in the MLE training.

From Eq. 3, it is clear that the gradient that every token $o_i \in \mathbb{V}$ receives is directly related to its generation probability p_i. Therefore, we hypothesize that directly manipulating the generation probabilities of tokens, thereby controlling their gradients, can help us achieve our goal, which is to train the model so that it is encouraged to use novel tokens.

3.2. Our method: ScaleGrad

To encourage the model to learn to use novel tokens for generation, we can control the gradient to force the model to either increase the probability of novel tokens or decrease the probability for non-novel tokens. Based on this basic idea, we propose an effective training method keeping it in the simplest form. Specifically, at each decoding step of training, we re-normalize the softmax output (the probability distribution over the vocabulary) in a way such that the model is informed of the current set of novel tokens and encouraged to use them. Assuming that $\boldsymbol{p}^t$ is the softmax output at step t and $\mathbb{S}_{\text{novel}}^t$ is the corresponding set of novel tokens at that step, we re-compute the probability distribution as follows (again omitting t for notational simplicity):

$$\tilde{p}_i = \begin{cases} \dfrac{\gamma \cdot p_i}{\sum_{j=1}^{|\mathbb{S}_{\text{novel}}|} \gamma \cdot p_j + \sum_{j=1}^{|\mathbb{V}'|} p_j}, & \text{if } i \in \mathbb{S}_{\text{novel}} \\ \dfrac{p_i}{\sum_{j=1}^{|\mathbb{S}_{\text{novel}}|} \gamma \cdot p_j + \sum_{j=1}^{|\mathbb{V}'|} p_j}, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbb{V}' = \mathbb{V} \setminus \mathbb{S}_{\text{novel}}^t$ is the non-novel tokens set at step t and $\gamma \in (0, 1)$ is the only hyper-parameter in our method that controls to what degree we want to encourage the model to focus on novel tokens; a smaller value of γ incurs more aggressive push for using novel tokens.

The effect of the above change is that we directly re-scale the generation probability (after re-normalization) of the tokens. For $i \in \mathbb{S}_{\text{novel}}$, the effective probability becomes $\tilde{p}_i = \lambda_i \cdot p_i$ with $\lambda_i \in (0, 1)$, and for $i \notin \mathbb{S}_{\text{novel}}$, the effective probability becomes $\tilde{p}_i = \alpha_i \cdot p_i$ with $\alpha_i > 1$.[2] Since $\lambda_i \cdot p_i$ and $\alpha_i \cdot p_i$ are new re-normalized probabilities, they both are naturally bounded in $[0, 1]$. Consequently, assuming that the ground truth token is indexed at k, the modified loss function at step t for our proposed method becomes:

$$\mathcal{L}_{\text{SG}} = -\sum_{i=1}^{|\mathbb{V}|} \mathbb{1}(i = k) \Big[\mathbb{1}(i \in \mathbb{S}_{\text{novel}}) \log(\lambda_i \cdot p_i) + \mathbb{1}(i \notin \mathbb{S}_{\text{novel}}) \log(\alpha_i \cdot p_i) \Big] \tag{5}$$

[2] Note that λ_i and α_i are functions of p_i rather than constant numbers. *E.g.,* $\lambda_i = \gamma / (\sum_{j=1}^{|\mathbb{S}_{\text{novel}}|} \gamma \cdot p_j + \sum_{j=1}^{|\mathbb{V}'|} p_j)$.

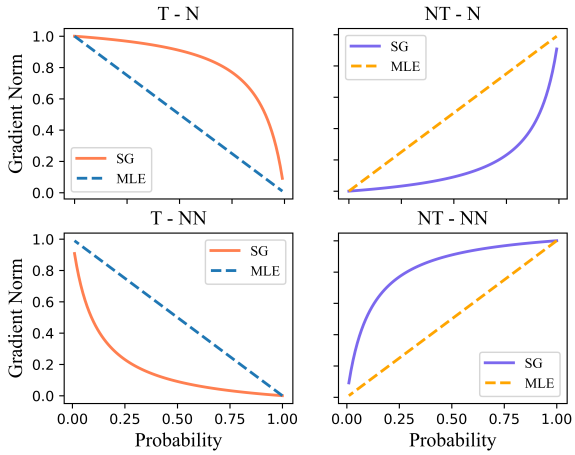


Figure 1: Illustration of the gradient norm for ScaleGrad and MLE. T-N denotes the **T**arget (ground truth) - **N**ovel token, T-NN denotes the **T**arget - **N**on-**N**ovel token, NT-N denotes the **N**on-**T**arget - **N**ovel token and NT-NN denotes the **N**on-**T**arget - **N**on-**N**ovel token.

The gradient for each token has been changed to[3]:

$$\nabla_{o_i}\mathcal{L} = \tilde{p}_i - \mathbb{1}(i=k) = \begin{cases} \lambda_i \cdot p_i - 1, & \text{if } i=k \text{ and } i \in \mathbb{S}_{\text{novel}} \\ \alpha_i \cdot p_i - 1, & \text{if } i=k \text{ and } i \notin \mathbb{S}_{\text{novel}} \\ \lambda_i \cdot p_i, & \text{if } i \neq k \text{ and } i \in \mathbb{S}_{\text{novel}} \\ \alpha_i \cdot p_i, & \text{if } i \neq k \text{ and } i \notin \mathbb{S}_{\text{novel}} \end{cases} \quad (6)$$

We now discuss why this re-scaling encourages the model to use novel tokens. As mentioned, during training with each gradient update the model tries to decrease the gradient norm to 0 to reach a local minimum. First, for a ground truth token (*i.e.,* $i=k$), if it is also a novel token, the gradient norm $|\lambda_i \cdot p_i - 1|$ is pushed away from 0 so that the model has to learn to increase the probability p_i further to reduce the gradient norm; if it is not a novel token, $|\alpha_i \cdot p_i - 1|$ is pushed slightly closer to 0, which still makes the model learn to predict the ground truth but with a relatively lower strength. For non-ground truth tokens (*i.e.,* $i \neq k$), when it is not a novel token, $|\alpha_i \cdot p_i|$ increases the gradient norm so that the model learns to assign much lower probability p_i to reduce it. Similarly, when the token is novel but not a ground truth token, the resulting gradient norm $|\lambda_i \cdot p_i|$ becomes smaller, for which the model only moderately learns to decrease the probability p_i to reduce the norm further.

To give more insights, Figure 1 plots the gradient norm for a toy example with two tokens, one of which is a novel token (*i.e.,* $|\mathbb{V}|=2$, $|\mathbb{S}_{\text{novel}}|=1$). The dash lines represent the gradient information for MLE training, *i.e.,* $|p_i - \mathbb{1}(i=k)|$. We can see how ScaleGrad scales the gradient dynamically for different types of tokens. For instance, for a target token belonging to the novel token set (T-N), its gradient norm has been scaled to a larger value compared to MLE, rendering that the model needs to learn to assign even higher probability to the target token to decrease the gradient. Similarly, for a non-target token that is not a novel token (NT-NN), the scaled gradient makes the model assign even lower probability to the token in order to decrease the gradient. Moreover, the monotonic relation between the probability and the gradient norm guarantees that the model still learns to predict target tokens and reject non-target tokens, but in more dynamic degrees of strength.

[3]Derivation is given in Appendix A.

3.3. Comparison with unlikelihood training

We now analyze UL from the perspective of its gradients and compare with ours. The gradient of the UL loss (Eq. 2) with a single negative token (*i.e.,* $|\mathbb{C}^t|=1$) is:

$$\nabla_{o_i}\mathcal{L} = m_i \cdot p_i - \mathbb{1}(i=k) = \begin{cases} (1-\alpha\frac{p_{\text{neg}}}{1-p_{\text{neg}}})p_i - 1, & \text{if } i=k \\ (1-\alpha\frac{p_{\text{neg}}}{1-p_{\text{neg}}})p_i, & \text{if } i \neq k \text{ and } i \neq i_{\text{neg}} \\ (1+\alpha)p_i, & \text{if } i \neq k \text{ and } i = i_{\text{neg}} \end{cases} \quad (7)$$

where $p_i = [\text{softmax}(\boldsymbol{o})]_i$, p_{neg} is the probability of the negative-candidate token with index i_{neg}, and $\mathbb{1}(i=k)$ is the indicator function with k being the index of the ground truth or target token (see the original paper for derivation).

We know that as the gradient-based optimization progresses, the gradient norm decreases and converges to near 0 (§3.1-§3.2). To generate a ground truth token, the model must learn to assign the highest probability to it. In other words, the probability assigned by the model to a ground truth token (*i.e.,* p_k) should always increase as the training progresses, or equivalently the norm $|\nabla_{o_k}\mathcal{L}|$ should decrease (monotonic relation). If this is not the case, the model may not learn to predict the ground truth tokens correctly, which in turn hurts the generation quality.

Based on the gradients (Eq. 7), we can identify one case where UL may provide such undesired property. Since the ground truth is always by definition a non-negative token in UL (*i.e.,* $i=k \neq i_{\text{neg}}$), the gradient norm from Eq. 7 is $|\nabla_{o_k}\mathcal{L}| = |\mu_k \cdot p_k - 1|$ where $\mu_k = (1-\alpha\frac{p_{\text{neg}}}{1-p_{\text{neg}}})$. We see that when $p_{\text{neg}} > \frac{1}{\alpha+1}$ (*e.g.,* when $\alpha=1$ and $p_{\text{neg}} > 0.5$), μ_k becomes negative, having the gradient norm $|\nabla_{o_k}\mathcal{L}| = |-|\mu_k| \cdot p_k - 1| = |\mu_k| \cdot p_k + 1$. In this case, the training procedure will decrease p_k to reduce $|\nabla_{o_k}\mathcal{L}|$, which contradicts with the optimization principle. Thus, UL may become less effective in such special cases (subject to the choice of the value of α and p_{neg}). Appendix C further clarifies this issue using the same notation as the original paper (Welleck et al., 2020). In contrast, the gradient analysis

Table 1: Results for open-ended generation tasks on the **Wikitext-103 testset**. **ppl**, **uniq** and **Rep/l** are computed at BPE-level and the rest are at word-level. The "↑" denotes higher value for better performance and "↓" is the opposite. Number marked with * are estimated based on the testset. The results are averaged over 3 runs with different random seeds. Full results with standard deviation are reported in Appendix F.1.

	Language Modeling					Auto Completion			
Models	**ppl** ↓	**uniq** ↑	**Rep/16** ↓	**Rep/32** ↓	**Rep/128** ↓	**Rep-1** ↓	**Rep-2** ↓	**Rep-3** ↓	**uniq-w** ↑
MLE	**13.241**	12.54k	0.234	0.380	0.619	0.661	0.500	0.424	16.83k
UL ($\alpha = 1.0$)	16.062	13.18k	0.212	0.341	0.558	0.559	0.363	0.291	19.11k
SG ($\gamma = 0.2$)	14.203	**13.61k**	**0.197**	**0.317**	**0.522**	**0.443**	**0.215**	**0.143**	**22.25k**
Human	-	18.27k	0.177	0.285	0.480	0.382*	0.096*	0.037*	27.55k*

in Eq. 6 shows that ScaleGrad does not have such properties in learning to predict ground truth tokens.

4. Experiments

We showcase the performance of ScaleGrad in both open-ended and directed generation tasks. To verify the effectiveness of our approach, for all the experiments below, we use exactly the same hyper-parameters (except for method-specific ones) and setup as the corresponding baseline unless stated otherwise. All the experimental details, such as model hyper-parameters, training and dataset settings regarding the reproducibility can be found in Appendix G. For qualitative assessments, we show examples of generated texts in Table 6 and more in Appendix K. For both open-ended and directed generation tasks, in order to model different regularization strengths imposed by the methods, we choose $\alpha \in \{0.5, 1.0, 1.5\}$ for unlikelihood training and $\gamma \in \{0.2, 0.5, 0.8\}$ for ScaleGrad.[4] The final models are chosen based on their performance on the corresponding development sets.

4.1. Open-ended generation

Setup We consider language modeling and text auto-completion, where we compare the performance of the model trained with ScaleGrad against the models trained with MLE and unlikelihood (UL) training introduced lately to mitigate degeneration in open-ended tasks. We follow the same setup as (Welleck et al., 2020). Specifically, we fine-tune the pre-trained GPT-2 (Radford et al., 2019) on Wikitext-103 (Merity et al., 2017) with a maximum sequence length of 300 tokens. Each model is trained for a maximum of 35k iterations and evaluated based on the perplexity on the validation set after every 1k iterations. We report language modeling results on the testset for each model selected according to the perplexity on the validation set. The same saved models are also used for text auto-completion, where 50 BPE (Sennrich et al., 2016) tokens (from testset) are given as prefix and the models are to generate the continuation of 100 next tokens. To evaluate the modeling capability exclusively, following Welleck et al. (2020), we apply greedy decoding in all our experiments in this section. Later, in §5.1, we analyze how our method performs with different decoding methods.

In language modeling, we measure the generation quality by the standard perplexity (ppl), and Rep/l and 'uniq' measures of token-level distribution as (Welleck et al., 2020). Rep/l measures the number of times that a token from the previous l tokens is repeated, when generating the following token; in our case, $l \in \{16, 32, 128\}$. The 'uniq' is defined as the number of unique next-token predictions on a test/validation set. For auto-completion, we report the repetition ratios of n-gram words (Rep-n) as well as the number of unique words (uniq-w) that are used during generation on the testset.

Results We present our main results on the *testset* in Table 1. The results with different hyper-parameters for both methods on the *validation* set are reported in Appendix F.1 and in §5.3. From Table 1, we notice that in language modeling, the model trained with ScaleGrad (SG) yields a token distribution that is much closer to human, while maintaining a lower perplexity. Compared to the UL baseline, SG achieves 1%, 2%, 4% lower repetitions in Rep/16, Rep/32 and Rep/128, respectively, while having 11% lower perplexity. It also uses more **uniq**ue tokens compared to others (*e.g.,* 3% more compared to UL training). Overall, our method significantly improves the token-level distribution and keeps a high generation quality. In auto-completion, from the quantitative perspective, SG produces texts with much fewer repetitive n-grams compared to MLE and UL. It uses nearly 5.5k more unique words compared to the MLE baseline, which agrees with the purpose of making the model learn to use novel tokens in training.

Human evaluation We have conducted a user study to verify the quality of generated texts. The study is conducted

[4] $\alpha = 1.0$ is recommended by Welleck et al. (2020), which can be seen as applying unlikelihood loss with a moderate strength. We use $\alpha = 0.5$ and 1.5 to evaluate for weak and strong strengths.

Table 2: Results for open-ended generations on **PTB** testset. **ppl**, **uniq** and **Rep/l** are computed at BPE-level and the rest are at word-level. The "↑" denotes higher value for better performance and "↓" is the opposite. Numbers marked with * are estimated based on the testset.

	Language Modeling					Auto Completion			
Models	**ppl** ↓	**uniq** ↑	**Rep/16** ↓	**Rep/32** ↓	**Rep/128** ↓	**Rep-1** ↓	**Rep-2** ↓	**Rep-3** ↓	**uniq-w** ↑
PTB									
MLE	**33.952**	5.60k	0.157	0.292	0.530	0.652	0.493	0.424	6.46k
UL ($\alpha = 1.0$)	41.232	5.96k	0.139	0.260	0.476	0.533	0.333	0.259	7.60k
SG ($\gamma = 0.2$)	40.731	**6.15k**	**0.126**	**0.231**	**0.426**	**0.417**	**0.198**	**0.131**	**8.42k**
Human	-	8.84k	0.118	0.222	0.421	0.362*	0.089*	0.033*	11.32k*

Table 3: Human evaluation results for auto-completion. **% Agr.** is the percentage agreement and **AC1** denotes Gwet's AC1/gamma coefficient. Winners are marked in **bold**.

	Win Rate	**% Agr.**	**AC1**
SG vs MLE	84.0%	84.0%	0.78
SG vs UL	70.5%	79.0%	0.64

for two pairs of systems (SG vs. UL, SG vs. MLE). For each pair, we randomly choose the same 100 prefixes for the systems to produce their own continuations and ask two native speakers of English to judge which text is the better continuation of the given prefix in terms of their *relevance* to the prefix, *grammaticality* and *readability*. More details about the study can be found in Appendix D.

From the results in Table 3, we can observe that the texts produced by the models trained with ScaleGrad (SG) are preferred by the human users in most of the cases, *i.e.,* 84.0% and 70.5%, respectively. We also compute the percentage agreement and chance-correlated Gwet's AC1/gamma coefficient (Gwet, 2008) as inter-user agreement to verify the reliability of the study (details in Appendix D). We see that the agreements are substantial in both measures.

Generalizability To further verify the generalizability (*i.e.,* different datasets and domains) of our method, apart from WikiText-103 (Merity et al., 2017), we evaluate the models on two other datasets: Penn TreeBank or PTB (Marcus et al., 1993) and IMDB (Maas et al., 2011). In particular, after fine-tuning GPT-2 with different training strategies (MLE, SG and UL) on WikiText-103 training data, we test the language modeling and auto-completion performance on the other two datasets. For PTB, we use the standard testset. As for IMDB, we randomly sample 500 movie reviews from the dataset.

In Table 2, we show the experimental results on the PTB testset, from which we can see that SG consistently improves over the MLE baseline in degeneration while possessing an acceptable increase in perplexity, and it outperforms UL consistently. Additionally, we present the results on IMDB movies review in Table 12 in Appendix F.2, where we observe similar performance trending as in the experiment on PTB testset. From the two experiments, we can draw the conclusion that our method, SG, is capable of generalizing well to different datasets and domains. Examples of generated text for auto completion task can be found in Appendix K.

4.2. Directed generation

For directed generation, we consider two tasks: image paragraph captioning and text summarization.

Table 4: Results for image paragraph captioning on the Visual Genome testset.

Models	**CIDEr**
MLE w/o 3-block	10.51
UL w/o 3-block (α=0.5)	14.65
SG w/o 3-block (γ=0.5)	**19.42**
MLE w/ 3-block	22.77
UL w/ 3-block (α=0.5)	22.25
SG w/ 3-block (γ=0.5)	**24.62**

4.2.1. Image paragraph captioning

Setup We use the captioning model proposed by Melas-Kyriazi et al. (2018) as the baseline, which comprises a CNN encoder that is pre-trained for object detection and a 1-layer LSTM decoder. The models are trained and evaluated on the paragraph captioning dataset, Visual Genome (Krause et al., 2017). We train the model with SG and compare it to the ones trained with MLE and UL. The performance is measured by CIDEr (Vedantam et al., 2015), which computes TF-IDF weighted n-gram overlaps between the model generated captions and the reference captions. We follow Melas-Kyriazi et al. (2018) to apply greedy inference since beam search did not yield any further gain.

Results Table 4 shows the CIDEr scores for different training methods on Visual Genome testset with and without tri-gram blocking (Paulus et al., 2018) during inference. Without tri-gram blocking, MLE produces texts that are full of repetitive phrases (see Appendix K for examples), which leads to a low CIDEr score. When UL or SG is incorporated, the performance has been notably improved from 10.51 to 14.65 and 19.42, respectively. When tri-gram blocking is applied, our method is still capable of yielding 1.85 point improvement. This is because SG further improves the token-level degeneration on top of tri-gram blocking. In contrast, the model trained with UL has a slightly worse CIDEr score compared to the MLE baseline. We analyze n-gram level degeneration further in §5.2.

Table 5: Experimental results for text summarization on CNN/DM and NYT50 testsets. **R-1**, **R-2** and **R-L** stand for F1-based ROUGE-1, ROUGE-2 and ROUGE-L. **WMD-1** denotes 1-gram MoverScore.

Models	R-1	R-2	R-L	WMD-1
CNN/DM				
BertSum w/ MLE	41.87	19.42	38.93	19.89
BertSum w/ UL ($\alpha = 0.5$)	42.03	19.36	39.09	20.21
BertSum w/ SG ($\gamma = 0.8$)	**42.19**	**19.53**	**39.25**	**20.23**
NYT50				
BertSum w/ MLE	48.73	31.00	45.23	28.73
BertSum w/ UL ($\alpha = 0.5$)	48.54	30.73	44.99	28.50
BertSum w/ SG ($\gamma = 0.8$)	**49.29**	**31.30**	**45.78**	**29.14**

4.2.2. Abstractive Text Summarization

Setup We use the abstractive summarization model BertSum (Liu & Lapata, 2019) as our baseline, which adopts a Transformer architecture to take advantage of pre-trained BERT (Devlin et al., 2019) as the encoder. At the first stage, the encoder is trained with an extractive summarization objective (binary classification for sentence selection). At the second stage, it initializes the decoder randomly and (re)trains the entire encoder-decoder model with an abstractive (or generative) objective. For our experiments, we take the encoder that was trained at the first stage and train the entire (abstractive) model with different training methods (MLE, UL and SG) using the default training setup on two benchmark datasets: CNN/DM (Hermann et al., 2015; Nallapati et al., 2016) and NYT50 (Durrett et al., 2016). During inference, length normalization (Wu et al., 2016), tri-gram blocking and beam search (beam size = 5) are used as in (Liu & Lapata, 2019).

We evaluate performance of the models with the standard F1-based ROUGE (Lin, 2004) scores (R-1, R-2, R-L) and a model-based evaluation MoverScore (Zhao et al., 2019), which computes the Word Mover Distance (WMD) between the reference summary and generated summary based on the representations from BERT. We report 1-gram MoverScore (WMD-1), which has been proven to have higher correlation with human than other metrics (Zhao et al., 2019).

Results From Table 5, we notice that on CNN/DM, the model trained with SG outperforms the models trained with MLE and UL when measured by ROUGE. In WMD-1, UL yields similar performance as ours. Both SG and UL further improve over the MLE baseline. The improvements imply that token-level degeneration may still exist even when tri-gram blocking is applied. On NYT-50, UL underperforms MLE, while our method improves in all measures. In §3.3, we discussed a possible reason behind UL's underperformance from a gradient perspective.

5. Analysis of ScaleGrad

After comparing with UL and MLE on both directed and open-ended generation tasks, we now analyze ScaleGrad from different perspectives to gain more insights.

5.1. Open-ended generation

Compatibility with decoding strategies One advantage of SG training is that it is compatible with decoding-based methods. One can choose different decoding strategies based on the specific needs. Table 7 provides the results of different decoding strategies used along with SG training for text auto-completion (results for other variations and baselines are in Appendix H). We observe that beam search, even with larger beam size, is not effective in mitigating the degeneration issue, which accords with the observation in (Holtzman et al., 2020). As expected, stochastic decoding, top-k and nucleus (top-p) sampling, help to further reduce repetition. This sets good examples of combining training and decoding strategies for the task in hand.

Auto completion with different decoding lengths From a practice point of view, we analyze how SG performs in text generation with varied decoding lengths. In Figure 2, we show the **Rep-1** of generated text from the auto completion task with the constraints in different decoding (continuation) lengths. We see that compared to MLE counterpart, SG yields consistent improvements on **Rep-1**, or token-level degeneration, regardless the different decoding lengths, which further verifies the effectiveness and generalizability of our method.

5.2. Directed generation

Comparison with stochastic decoding Although top-p and top-k sampling have been proven successful in open-ended generation, they have not been tested in directed generation tasks. In order to see if they could lead to the same improvements as SG training, we conduct additional exper-

Table 6: An example of generated continuations for the same prefix.

Prefix: You's patriotism, and Mei's reflections on the quotidian are a few examples. More broadly, Du Fu's work in transforming the lushi from mere word play into "a vehicle for serious poetic utterance"
MLE: is a major contribution to the genre. The first of the three major works of the genre, was written in the early 1920s, and was the first of the three to be published in English. It was published in the United States in the United States, and was translated into French in the 1930s. The work was the first of the three
UL: is a major influence on the modern Chinese novel. The modern Chinese novel is often characterized as being "literary in nature", but it is not limited to literary genres. The modern Chinese novel has been described as "a literary genre that has been influenced by the Western literary tradition". The modern Chinese novel has been described as "a literary genre that has been influenced by the Western literary tradition"
SG: is a powerful example of how the Chinese literary tradition can be used to explore themes of nationalism and self determination. In addition to his own poetry, Du Fu wrote several works of prose. The first was a collection of short stories called "The Story of the Three Sisters", which was published in 1885. It was followed by a series of novels, including the first two volumes of the Song dynasty epic poem "The Three Sisters"

Table 7: Results of different decoding strategies with ScaleGrad training for auto-completion on WikiText-103 testset.

Approaches	**Rep-1**	**Rep-2**	**Rep-3**	**uniq-w**
SG+Greedy Search	0.443	0.215	0.143	22.25k
SG+Beam Search ($b = 6$)	0.453	0.250	0.171	8.32k
SG+Top-p ($p = 0.3$)	0.356	0.107	0.049	30.48k
SG+Top-k ($k = 40$)	0.254	0.039	0.012	39.50k

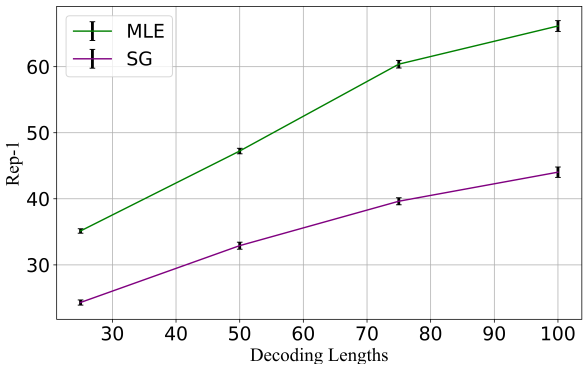


Figure 2: **Rep-1** in auto completion with different decoding lengths. All the numbers are computed based on the results from 3 runs with different random seeds.

iments with the BertSum summmarization model, whose underlying language model is more mature due to the involvement of BERT, compared to the image paragraph captioning model. For the interested readers, we also provide the results of stochastic decoding on image paragraph captioning in Appendix I.

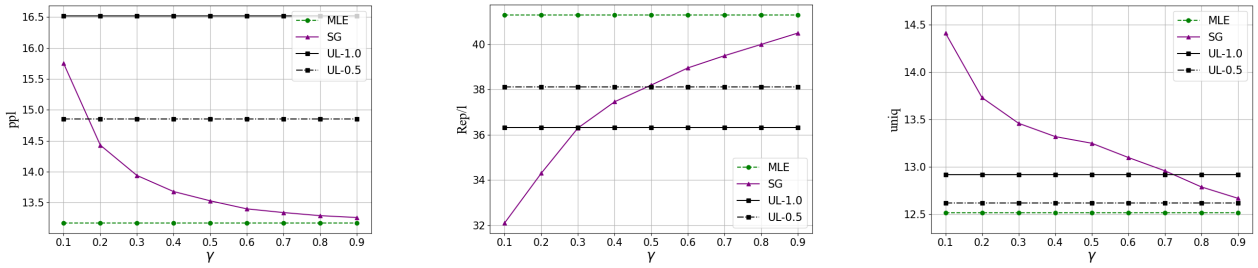
Table 8 shows the performance for stochastic decoding in BertSum trained with MLE. Since ROUGE-1 measures the exact 1-gram overlaps between reference and generated summaries, it may not be sufficient to evaluate the performance of stochastic decoding methods, which may generate more diverse output while conveying the same meaning. Therefore, we also report the MoverScore that is capable of considering the semantic similarity rather than just n-gram overlaps. Both the ROUGE and MoverScore in Table 8 lead to the conclusion that stochastic decoding methods significantly lower the performance compared to the standard beam search. This implies that they may not be a good fit for directed generation tasks. In contrast, SG possesses a wider applicability in mitigating degeneration issues as shown earlier in Table 5.

Table 8: Summarization results (F1-based ROUGE-1 and MoverScore) for stochastic decoding on NYT50 testset.

Models	**ROUGE-1**	**WMD-1**
Top-p (p=0.3)	45.44	24.61
Top-p (p=0.9)	42.33	21.67
Top-k (k=40)	41.23	20.70
Top-k (k=100)	40.86	20.38
Baseline	48.73	28.73

N-gram degeneration To investigate further how SG minimizes degeneration and helps to improve the performance in automatic measures, we compute the n-gram repetition ratios of the outputs from the image captioning model (Melas-Kyriazi et al., 2018) and report the numbers in Table 9.[5] Compared to human, the MLE baseline has significantly higher repetitions, thus having the lowest CIDEr score (Table 4). With SG, the model yields a much better repetition ratio, which explains the notable performance boost in CIDEr. Tri-gram blocking resolves the issue of 3- or higher n-gram degeneration in a hard-coded way, improving CIDEr significantly. However, the token and 2-gram repetitions still remain high and improvable in MLE with tri-gram blocking. When both tri-gram blocking and SG

[5]Since Melas-Kyriazi et al. (2018) used a soft tri-gram blocking, some of the duplicate tri-grams still remain.



(a) Perplexity (b) Rep/l (c) # of unique tokens

Figure 3: Hyper-parameter (γ) sensitivity in the language modeling task on **Wikitext-103 development** set (best viewed in color). Rep/l is computed as the average of Rep/16, Rep/32 and Rep/128. UL-1.0 and UL-0.5 represent unlikelihood training with α=1.0 and 0.5, respectively. $\alpha = 1.5$ is not included as it incurs significantly higher perplexity compared to others. Individual Rep/l results can be found in Appendix J.

are applied, the generated texts have the lowest and most human-like repetitions.

Table 9: Degeneration analysis for image paragraph captioning with/without tri-gram blocking. Numbers in bold are closest to human.

Models	**Rep-1**	**Rep-2**	**Rep-3**
MLE	0.723	0.587	0.530
SG	0.500	0.270	0.195
MLE w/ 3-block	0.575	0.271	0.094
SG w/ 3-block	**0.440**	**0.146**	**0.037**
Human	0.421	0.123	0.042

5.3. Hyper-parameter sensitivity

Towards better usage and understanding of ScaleGrad, we show how the key metrics in language modeling change with the hyper-parameter γ in Figure 3.[6] As discussed, a smaller value of γ incurs a stronger push to use novel tokens, giving higher perplexity and more unique tokens. Having a low perplexity and a low repetition ratio could be seen as a trade-off between general generation quality and diversity. However, we observe that when UL achieves similar performance in Rep/l with SG, *i.e.,* when $\gamma = 0.5$, $\alpha = 0.5$ and $\gamma = 0.3$, $\alpha = 1.0$ (fig. 3b), it exhibits much higher perplexity compared to SG with a difference of 1.35 and 2.58, respectively (fig. 3a). Similarly, when both methods have similar performance on perplexity, *i.e.,* when $\gamma = 0.2$ and $\alpha = 0.5$ (fig. 3a), SG yields 3.82% lower in Rep/l (fig. 3b) and uses 1.11k more unique tokens (fig. 3c). In summary, SG is able to reduce the degeneration without detracting much from the generation quality.

In general, γ in ScaleGrad can be chosen based on the performance of the baseline model. If the baseline produces many repetitive tokens/phrases (*e.g.,* image paragraph captioning experiments), a smaller value of γ should be used. Conversely, in tasks with less degeneration (*e.g.,* summarization experiments), a larger γ can be used to further improve the unigram and bigram level degeneration without affecting the perplexity much.

6. Conclusion

We have introduced a novel training method, called ScaleGrad, directly modifying the gradient of the standard MLE objective to remedy the text degeneration issues. The improvement verified by both automatic metrics and human evaluation against the baselines in extensive experiments across different tasks in open-ended and directed generation and different architectures (*i.e.,* LSTM and Transformer) demonstrate the effectiveness and generalizability of our method. Further analysis shows that ScaleGrad yields token distributions that are much closer to human-written texts compared to the baselines. Our method brings a good alternative to current training strategies for language generation.

In future, we plan to extend the idea in two directions. First, we would like to repurpose the definition of the set of the tokens that ScaleGrad operates on (*i.e.,* the novel token set) to enable the model to realize other objectives, *e.g.,* Zhao et al. (2021) has successfully adapted ScaleGrad to prevent early endpointing for online automatic speech recognition. Second, we would like to investigate a mechanism to dynamically adjust the hyper-parameter γ in the decoding steps such that the model could learn with different degrees of strength depending on the context.

[6] Note that for our main results in §4, we only search hyper-parameters from 3 chosen values. More numbers of γ in Figure 3 is intended to show the hyper-parameter sensitivity of ScaleGrad. One should not regard this as unfair comparison where different numbers of hyper-parameter are explored for different methods.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Durrett, G., Berg-Kirkpatrick, T., and Klein, D. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1998–2008, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1188. URL https://www.aclweb.org/anthology/P16-1188.

Fan, A., Lewis, M., and Dauphin, Y. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/P18-1082.

Gwet, K. L. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48, 2008.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1693–1701. Curran Associates, Inc., 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Jiang, S., Wolf, T., Monz, C., and de Rijke, M. Tldr: Token loss dynamic reweighting for reducing repetitive utterance generation. *arXiv*, 2020. URL https://arxiv.org/abs/2003.11963.

Koehn, P. and Knowles, R. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 28–39, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3204. URL https://www.aclweb.org/anthology/W17-3204.

Kool, W., Van Hoof, H., and Welling, M. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3499–3508, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/kool19a.html.

Krause, J., Johnson, J., Krishna, R., and Fei-Fei, L. A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Patterm Recognition (CVPR)*, 2017.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

Li, M., Roller, S., Kulikov, I., Welleck, S., Boureau, Y.-L., Cho, K., and Weston, J. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4715–4728, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.

acl-main.428. URL https://www.aclweb.org/anthology/2020.acl-main.428.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, pp. 10, 2004. URL http://research.microsoft.com/~cyl/download/papers/WAS2004.pdf.

Liu, Y. and Lapata, M. Text summarization with pretrained encoders. 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)*, pp. 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387. URL https://www.aclweb.org/anthology/D19-1387.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

Melas-Kyriazi, L., Rush, A., and Han, G. Training for diversity in image paragraph captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 757–761, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1084. URL https://www.aclweb.org/anthology/D18-1084.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *ICLR*, 2017. URL https://openreview.net/pdf?id=Byj72udxe.

Nallapati, R., Zhou, B., dos Santos, C., GuÌ‡lçehre, Ç., and Xiang, B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https://www.aclweb.org/anthology/K16-1028.

Paulus, R., Xiong, C., and Socher, R. A deep reinforced model for abstractive summarization. In *ICLR*, 2018. URL https://openreview.net/pdf?id=HkAClQgA-.

Qi, W., Yan, Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. ProphetNet: Predicting future n-gram for sequence-to-SequencePretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2401–2410, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.217. URL https://www.aclweb.org/anthology/2020.findings-emnlp.217.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *Open-AI Blog*, 2019.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Stahlberg, F. and Byrne, B. 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)*, pp. 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331. URL https://www.aclweb.org/anthology/D19-1331.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1008. URL https://www.aclweb.org/anthology/P16-1008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio,

S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

Wang, P.-H., Hsieh, S.-I., Chang, S.-C., Chen, Y.-T., Pan, J.-Y., Wei, W., and Juan, D.-C. Contextual temperature for language modeling. *arXiv*, 2020. URL https://arxiv.org/abs/2012.13575.

Wang, Q. and Chan, A. B. Describing like humans: On diversity in image captioning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4190–4198, 2019.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Zhang, X., Sun, M., Liu, J., and Li, X. Improving diversity of neural text generation via inverse probability weighting. *arXiv*, 2021. URL https://arxiv.org/abs/2103.07649.

Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. 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)*, pp. 563–578, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1053. URL https://www.aclweb.org/anthology/D19-1053.

Zhao, Y., Ni, C., Leung, C.-C., Joty, S., Chng, E. S., and Ma, B. Preventing early endpointing for online automatic speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6813–6817, 2021. doi: 10.1109/ICASSP39728.2021.9413613.