TLDR: Token Loss Dynamic Reweighting for Reducing Repetitive Utterance Generation

Shaojie Jiang[1] **Thomas Wolf**[2] **Christof Monz**[1] **Maarten de Rijke**[1]
[1]University of Amsterdam, Amsterdam, The Netherlands
[2]Hugging Face, Brooklyn, New York, United States
{s.jiang, c.monz, m.derijke}@uva.nl, thomas@huggingface.co

Abstract

Natural Language Generation (NLG) models are prone to generating repetitive utterances. In this work, we study the repetition problem for encoder-decoder models, using both recurrent neural network (RNN) and transformer architectures. To this end, we consider the chit-chat task, where the problem is more prominent than in other tasks that need encoder-decoder architectures. We first study the influence of model architectures. By using pre-attention and highway connections for RNNs, we manage to achieve lower repetition rates. However, this method does not generalize to other models such as transformers. We hypothesize that the deeper reason is that in the training corpora, there are *hard tokens* that are more difficult for a generative model to learn than others and, once learning has finished, hard tokens are still under-learned, so that repetitive generations are more likely to happen. Based on this hypothesis, we propose *token loss dynamic reweighting* (TLDR) that applies differentiable weights to individual token losses. By using higher weights for hard tokens and lower weights for easy tokens, NLG models are able to learn individual tokens at different paces. Experiments on chit-chat benchmark datasets show that TLDR is more effective in repetition reduction for both RNN and transformer architectures than baselines using different weighting functions.

1 Introduction

It has been widely reported that Natural Language Generation (NLG) models are prone to generating repetitive utterances, which is a cross-task and cross-architecture problem. E.g., the repetition problem has been observed in question answering (QA) (Fan et al., 2019), language modeling (LM) (Holtzman et al., 2019; Keskar et al., 2019; Welleck et al., 2019), abstractive summarization (Suzuki and Nagata, 2017; Nallapati et al., 2016), machine translation (MT) (Tu et al., 2016; Mi et al., 2016; Tu et al., 2017), and image captioning (Cornia et al., 2018), etc. The problem arises with most popular neural architectures, including sequence-to-sequence (Seq2Seq) modeling using recurrent neural networks (RNNs) (Sutskever et al., 2014) and transformers (Vaswani et al., 2017). Previous approaches to the repetition problem have mostly been developed for decoder-only models (Holtzman et al., 2019; Keskar et al., 2019; Welleck et al., 2019). Whenever it is possible to use them, encoder-decoder models usually outperform decoder-only models (Raffel et al., 2019); because of this, we study the repetition problem for encoder-decoder models in this work. We base our experiments on the chit-chat task,[1] where the repetition problem is more prominent than other tasks that need encoder-decoder architectures. Figure 1 is an illustration of a chit-chat scenario. The repetition problem on which we focus in this paper is *not* to be confused with the problem where responses of different turns share a great deal of similarity, which is sometimes also referred to as repetitive responses (Li et al., 2016b).

User Message:	Hi, Jim. How are you? I haven't seen you for a while.
System Response:	i've been out of town. i've been out of town. i've been out of town and got some old ideas on i've been out of town.

Figure 1: An illustration of the repetitive response problem.

We first surmise that the repetition problem is caused by model architectures. By studying existing approaches to repetition reduction involving changing model architectures, we identify that a certain model design for RNNs can help reduce repetition. More specifically, by using pre-attention (§3) together with RNNs instead of the conven-

[1]Throughout this paper, we use 'chit-chat,' 'open-domain dialogue,' and 'response generation' to refer to the same task interchangeably, and use 'chatbot' to refer to the model trained for this task.

tional usage of post-attention, the model generates less repetitive utterances. The usage of highway connection (Srivastava et al., 2015) after pre-attention can further reduce repetition. Though it is effective for RNNs, this architectural design unfortunately does not generalize to transformers. Therefore, we believe that there are deeper reasons to the repetition problem.

Inspired by the well-studied class imbalance problem (Lin et al., 2017), we hypothesize that in training corpora there are *hard* tokens and *easy* tokens; if the training stops when hard tokens are still under-learned, repetitive generations are more likely to happen. We empirically validate this hypothesis by using the Focal Loss (FL) method proposed by Lin et al. (2017). Based on this finding, we propose *token loss dynamic reweighting* (TLDR), which is inspired by FL (§4) and which assigns differentiable weights to training losses according to the training difficulty of tokens. By dynamically using higher weights for hard tokens and lower weights for easy tokens, the model is able to learn tokens at different paces. Experiments on benchmark chit-chat datasets show that TLDR is more effective in reducing repetitions for RNN and transformer architectures than the baselines.

The main contributions of this work include:

- We discover that RNNs with pre-attention and highway connections can help reduce repetition.
- We hypothesize that the deeper reason for repetitions is due to hard tokens, and empirically validate this by using FL.
- We propose a more effective token loss dynamic reweighting (TLDR) method for reducing repetitive generations that is architecture insensitive.
- We share our source code at https://github.com/ShaojieJiang/tldr.

2 Related Work

2.1 Repetition reduction

There are generally three types of approach to the repetition problem, namely architecture-based methods, auxiliary loss, and sampling-based methods.

Architecture-based methods try to address the repetition problem by exerting inductive biases on model architectures, such as introducing coverage vectors to attention mechanisms (Tu et al., 2016; Mi et al., 2016). However, these solutions are based on the desirable property of source-target correspondence of neural machine translation (NMT) tasks, which rarely holds for other tasks like open-domain dialogue generation and abstractive summarization. Instead, we observe that by using pre-attention (§3), RNN models can generate less repetitive responses. Adding a highway connection to attention layers can further reduce repetition. However, it is not yet clear which modifications to transformers can help reduce repetitive generations, which we surmise to be mainly due to their lack of temporal inner states.

Auxiliary loss adds an unlikelihood item of repeated tokens to the conventional cross-entropy (CE) loss (Welleck et al., 2019; Li et al., 2019) to discourage repetitive generations. However, besides the need for extra data types (Li et al., 2019) or extra processing on model generations (Welleck et al., 2019), it is also worth noting that the repetition problem seldom exists in the training data (§6), and therefore, the models are *not* trained to be repetitive in the first place. Our experiments show that the repetition problem is mainly due to the fact that trained models have different levels of proficiency toward easy and hard tokens. Based on this, we propose to weight the token losses w.r.t. their training difficulties to address the repetition problem. Though, normally, hard tokens already result in larger gradients, apparently this is not enough to learn them well as their effects may be dissolved by large amount of easy tokens.

Sampling based approaches discourage repetition at the sampling stage, e.g., by creating an n-gram blacklist (Paulus et al., 2018). Some more sophisticated methods include discounting the scores of previously generated tokens (Keskar et al., 2019), top-k random sampling (Fan et al., 2018), and top-p sampling with a dynamic nucleus (Holtzman et al., 2019). These methods can only be used at inference time to remedy the repetition problem. In this work, we tackle the problem at a deeper level and show that by dynamically reweighting token losses at training time, the problem can be largely alleviated.

2.2 Class imbalance

Class imbalance is a common cause of poor performance of trained models. To mitigate the low efficiency of heuristic sampling approaches to balancing classes, Lin et al. (2017) try to balance the CE loss by weighting easy and hard examples differently for visual object detection. Similarly, FACE (Jiang et al., 2019) tries to balance the CE loss for frequent and rare tokens through heuristics such as token frequency for improving generation diversity. In our case, however, the frequency of a token does not necessarily reflect its training difficulty, as the difficulty is also influenced by the context of this token.

We hypothesize that the repetition problem is also caused by different training difficulties of tokens. This hypothesis is in line with (Fadaee and Monz, 2018), where the authors improve the learning of hard tokens by using back-translation and by sampling examples with difficult words in similar contexts. Considering that at the token level in the chit-chat task, balancing easy and hard tokens through sampling is infeasible, loss reweighting such as Focal Loss (Lin et al., 2017) is a preferable solution. Focal Loss down-weights[2] both easy and hard examples, albeit at different ratios, which is likely to slow down training. Instead, we propose a different weighting function that simultaneously up-weights hard tokens and down-weights easy tokens.

3 RNN with Pre-Attention and Highway Connection

In this section, we introduce some architectural modifications that alleviate the repetition problem.

3.1 Notation and response generation

We write $x = (x_1, x_2, \ldots, x_{|x|})$, $y = (y_1, y_2, \ldots, y_{|y|})$ for message and response utterances, respectively. Given x, the objective of a chatbot model parameterized by θ is to assign a higher conditional probability $p(y|x;\theta)$ to the ground truth response y than to other responses, which usually follows a sequential decomposition:

$$p(y|x;\theta) = \prod_{t=1}^{|y|} p(y_t|y_{<t}, x;\theta). \quad (1)$$

Here, $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$, and $p(y_t|y_{<t}, x;\theta)$ represents the model-predicted probability for token y_t.

For the sake of clarity, we further denote $p_t = p(y_t|y_{<t}, x;\theta)$. Training the above chatbot model is usually achieved by minimizing the CE loss:

$$\text{CE}(p_t) = -\log(p_t). \quad (2)$$

For estimating p_t using an RNN model, we largely follow the notation in (Bahdanau et al., 2015) by denoting the decoder output and attention output at decoding step t as s_t and c_t, respectively. Readers are referred to (Bahdanau et al., 2015) for more details.

3.2 Pre-attention

Inspired by Input-feeding (IF) attention (Luong et al., 2015), we find that pre-attention performs better w.r.t. repetition rates. Figure 2 is an illustration of the difference between pre-attention and IF attention.

Formally, the input of RNN using IF attention at each decoding step t, i.e., y_{t-1}, is modified:

$$\tilde{y}'_{t-1} = [y_{t-1}; \tilde{s}_{t-1}], \quad (3)$$

with $[\cdot\,;\cdot]$ denoting vector concatenation. Besides, at step t the decoder output s_t is also transformed with attention output c_t:

$$\tilde{s}_t = g'(c_t, s_t). \quad (4)$$

The $g'(\cdot)$ function used in IF attention is a vector concatenation followed by a linear projection, to resize $\tilde{s}_t$ to the dimension of s_t.

We find that only transforming the input y_{t-1} is more helpful for reducing repetition, which leads us to pre-attention:

$$\tilde{y}_{t-1} = g(c_{t-1}, y_{t-1}), \quad (5)$$

with $g(\cdot)$ being a transformation function, and $\tilde{y}_{t-1}$ has the same dimensionality as y_{t-1}. The $g(\cdot)$ function can take the form of $g'(\cdot)$, but in this paper, we observe that by using a highway connection for $g(\cdot)$, the repetition rate can be further reduced. The highway connection is formulated as follows:

$$\begin{aligned}\tilde{y}_{t-1} &= \textit{Highway}(c_{t-1}, y_{t-1}) \\ &= z \circ c_{t-1} + (1 - z) \circ y_{t-1}, \end{aligned} \quad (6)$$

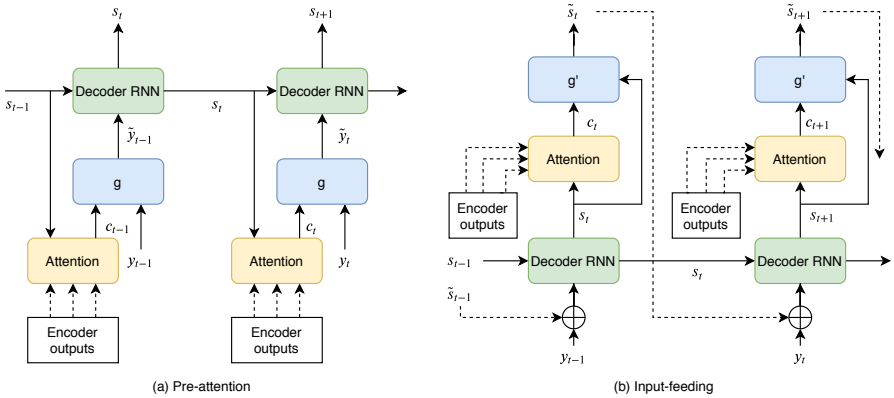
where $\circ$ represents element-wise product; z is a learnable gating vector, through which the model learns how to effectively combine attentional information c_{t-1} and input information y_{t-1}. Readers are referred to (Srivastava et al., 2015) for details of the highway connection.

3.3 Discussion

The attention mechanism does not have internal states to keep track of attentions previously paid, therefore when using post-attention, similar attentions may be paid repeatedly, increasing the chance of repetition. On the other hand, when using pre-attention as part of the input to RNNs, it enables RNN states to keep track of previous attention status. Furthermore, by using highway connections, the model learns to more effectively combine context and step-wise input than simple concatenation and linear projection.

However, noticing that the transformer model (Vaswani et al., 2017) already has an analogous architecture as our pre-attention, we tried to apply highway connections to the transformer attention

[2] Models trained without weighting can be thought of as using a uniform weight of 1.



(a) Pre-attention (b) Input-feeding

Figure 2: Pre-attention vs IF attention. $\oplus$ denotes vector concatenation.

outputs, but have not yet been successful with reducing repetitive generations in this manner. We suspect that this is due to the lack of temporal inner states in transformers, and more work needs to be done to mitigate this drawback in order to reduce repetition. Meanwhile, we believe there are more fundamental reasons for the repetition problem.

4 Token Loss Dynamic Reweighting

Inspired by Lin et al. (2017), we hypothesize that the difficulty of some training examples is the main reason for repetition. Based on this hypothesis, example reweighting methods such as FL (Lin et al., 2017) can be used to alleviate the repetition problem. In this section, we first adapt FL to our chit-chat task, and later we present a more effective weighting method than FL.

4.1 Example-level loss reweighting

Using the notation introduced in §3.1 and writing p^y for $p(y|x;\theta)$, a direct application of FL to the response generation task can be formulated as follows:

$$\text{FL}(p^y) = w \cdot L(y). \tag{7}$$

Here, $w = (1-p^y)^\gamma$ is the weighting coefficient, which is scaled by an exponential focusing parameter $\gamma \geq 0$, with $(1-p^y)$ reflecting the training difficulty of an example, that is, a pair (x, y). $L(y)$ denotes the loss of predicting y that is usually calculated as an average over the CE losses of all tokens that make up y:

$$L(y) = \frac{1}{|y|}\sum_{t=1}^{|y|}\text{CE}(p_t). \tag{8}$$

Since $p_t \in [0, 1]$, in practice p^y is usually extremely small according to the sequential decomposition Eq. (1), turning almost all training examples into hard examples. Therefore, we propose to approximate the training difficulty of (x, y) w.r.t. the average of p_t as follows:

$$\tilde{p}^y = 1 - \frac{1}{|y|}\sum_{t=1}^{|y|}p_t, \tag{9}$$

which is more suitable for our task.

However, there is another unwanted feature of FL. Although w in Eq. (7) is higher for hard examples than for easy ones, to make the model *focus* on examples that are hard to train, which is essentially what the method is named after, this method is likely to slow down the training as w is always smaller than 1. To address this, we propose to use a different weighting function:

$$\text{cosw}(\tilde{p}^y) = \cos(\tilde{p}^y * \pi) + 1, \tag{10}$$

which projects all probabilities to the weight domain of $[0, 2]$. With $\tilde{p}^y \leq 0.5$ indicating a hard example, which results in $\text{cosw}(\tilde{p}^y) > 1$, we up-weight the loss of hard examples, and otherwise we down-weight easy examples with $\tilde{p}^y > 0.5$. To distinguish our method (using (10)) from the adapted FL, we refer to it method as *loss dynamic reweighting* (LDR):

$$\text{LDR}(p^y) = \text{cosw}(\tilde{p}^y) \cdot L(y). \tag{11}$$

4.2 Token-level loss reweighting

Noticing from Eq. (2) that in our task, CE losses are calculated for each token, we can alternatively infer the difficulty at the token-level and apply FL or LDR methods thereafter. To achieve this, we use p_t instead of p^y for Eq. (7), resulting in *token-level Focal Loss* (TFL):

$$\text{TFL}(p_t) = w_t \cdot \text{CE}(p_t), \tag{12}$$

where $w_t = (1-p_t)^\gamma$. Using p_t instead of $\tilde{p}^y$ and p^y for Eq. (11) results in *token loss dynamic*

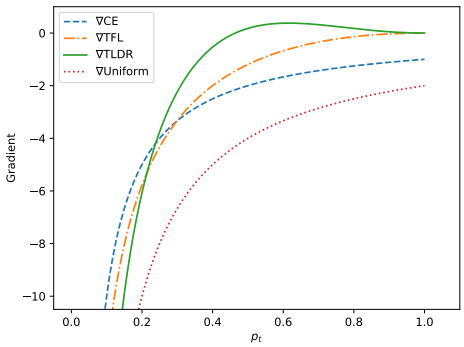


Figure 3: Gradient curves of each loss function. For TFL, we use $\gamma = 2$ as recommended by (Lin et al., 2017).

reweighting (TLDR):

$$\text{TLDR}(p_t) = \text{cosw}(p_t) \cdot \text{CE}(p_t). \quad (13)$$

Although treating each token as an example is somewhat counterintuitive, it has been shown in (Jiang et al., 2019) that such a token-level formulation can have a more direct impact on the generation of each token. We confirm in §6 that TLDR results in less repetition than LDR.

4.3 Gradient analysis

We now compare the gradients of our loss function TLDR to those of CE and TFL w.r.t. p_t. The gradient function of CE is:

$$\frac{\partial \text{CE}(p_t)}{\partial p_t} = -\frac{1}{p_t}. \quad (14)$$

The gradient function of TFL is:

$$\frac{\partial \text{TFL}(p_t)}{\partial p_t} = \gamma(1-p_t)^{\gamma-1} \cdot \log(p_t) - \frac{(1-p_t)^{\gamma}}{p_t}. \quad (15)$$

Finally, the gradient function of TLDR is:

$$\frac{\partial \text{TLDR}(p_t)}{\partial p_t} = -\pi \sin(p_t \cdot \pi) \log(p_t) - \frac{\cos(p_t \cdot \pi) + 1}{p_t}. \quad (16)$$

To see the differences between the three loss functions more clearly, we visualize their curves in Figure 3. Compared to CE and TFL, the gradients for easy tokens derived by TLDR are suppressed and gradients for hard tokens are amplified.

5 Experimental Setup

For our experiments, we use 3 multi-turn chit-chat datasets: Dailydialog (Li et al., 2017), Cornell Movie Dialogs (Danescu-Niculescu-Mizil and Lee, 2011) and Ubuntu Dialogue (Lowe et al., 2015), in increasing order of sizes. As a quantitative measure of system generated responses, we use BLEU-4 scores (Papineni et al., 2002). For repetition measurement, unlike existing work utilizing repeated n-gram counts (Mi et al., 2016) or n-gram diversity (Li et al., 2016a; Welleck et al., 2019) that reflect only one order of n-gram repetition (usually 4-gram), we propose to use a DIversity Metric based on N-grams (DIMEN), which is inspired by BLEU by weighted-averaging the n-gram diversities at multiple granularities, thus being more comprehensive. To introduce DIMEN, we first introduce the definition of n-gram diversity proposed by Li et al. (2016a):

$$\text{distinct}(text, k) = \frac{|\{k\text{-gram}(text)\}|}{\max(|1\text{-gram}(text)| - k, 1)}. \quad (17)$$

Here, k-gram$(text)$ means getting the n-gram list of $text$ for $n = k$; $\{\cdot\}$ maps a list to a set, and $|\cdot|$ denotes the cardinality of a set/list, respectively. For $text$ with less than k unigrams, we clip the denominator to be 1 such that distinct$(text, k) = 1$. With n-gram diversity defined, the DIMEN score of a given $text$ is calculated as

$$\text{DIMEN}(text, n) = \sum_{k=1}^{n} \alpha_k \cdot \text{distinct}(text, k), \quad (18)$$

where n is the highest order of n-grams and α_k is the weight coefficient corresponding to k-gram with $\sum_1^n \alpha_k = 1$.

The $text$ argument can be either a response utterance, in which case we refer to the score as u-DIMEN, or a list of response utterances for a whole validation set, where we refer to as l-DIMEN. u-DIMEN and l-DIMEN scores are compensating. With DIMEN$(\cdot) \in [0, 1]$, a high u-DIMEN score represents a non-repetitive utterance, and a high l-DIMEN score represents a diverse list of utterances.

To further measure the repetition performance on a validation set, we can consider averaging the u-DIMEN scores of each utterance. However, since the majority of system generated utterances are non-repetitive, the averaged u-DIMEN score can be very close to 1 and thus indistinguishable from model to model. Instead, we first bin the

u-DIMEN scores into m buckets to create a histogram vector $hist^{\text{DI}} \in \mathbb{Z}^m$, and then calculate the weighted L2-norm of the count of each bucket to emphasize the overall repetition performance of a model. Formally,

$$\text{WL2}(hist^{\text{DI}}) = \sqrt{\sum_{i=1}^{m} \beta_i \cdot (hist_i^{\text{DI}})^2}, \quad (19)$$

where $\beta \in \mathbb{R}^m$ is a weighting vector. When we assign higher weights to bins with lower u-DIMEN scores, the WL2 score emphasizes highly repetitive utterances.

With all the datasets and metrics introduced, we seek to answer the following research questions through our experiments:

(RQ1) Are pre-attention and highway connection helpful for reducing repetition of RNN Seq2Seq?

(RQ2) Is our hypothesis on hard tokens correct?

(RQ3) How effectively can LDR and TLDR reduce repetition for both RNN and transformer models?

(RQ4) Is the improvement of LDR and TLDR merely due to up-weighting?

To answer RQ1, we use an RNN Seq2Seq with post-attention as the baseline. To answer RQ2, we obtain results of using TFL on both RNN and transformer models. For answering RQ3, we compare the results of LDR and TLDR to those of baseline models trained without weighting and with TFL weighting. For RQ4, we also add a baseline using uniform weights, with $w = 2$ for all training examples.

Below we introduce the model parameters. Unless stated otherwise, we use the same settings for different models on each dataset whenever possible. For RNN Seq2Seq, we use 2-layer LSTMs (Hochreiter and Schmidhuber, 1997) with a hidden size of 512 and separate embedding matrices with an embedding size of 200 for both the encoder and the decoder. We use the 'general' attention variant from (Luong et al., 2015). For transformer Seq2Seq, we use 6-layer transformer blocks with 8 attention heads (Vaswani et al., 2017) and an embedding size of 256 with separate matrices for both the encoder and the decoder. We use 800 as the feed-forward layer size with 'relu' activation. We train both RNN and transformer Seq2Seq models using Adam optimizer (Kingma and Ba, 2014) with fixed learning rate of 0.001 and $\beta_1 = 0.9, \beta_2 = 0.999$. The gradients of RNNs are clipped with L2-norm (Pascanu et al., 2013), no larger than 5 during training, while for transformers we clip to 1. For all the intermediate outputs of both models, we use dropout rate of 0.1 (Srivastava et al., 2014).

We train the models on the Dailydialog and Movie Dialogs datasets for 100 epochs and check the performances on the validation sets every 0.5 epochs. On the Ubuntu Dialogue dataset, we train the models for 30 epochs with a validation interval of 0.1 epochs. We save checkpoints after every validation and select the best checkpoint according to the lowest repetition rate. Since the repetition rate can sometimes be very low, this checkpoint-saving strategy may fail occasionally, in which case we use the last checkpoint during training. For all three datasets, we tokenize the utterances using the NLTK (Loper and Bird, 2002) tokenizer and keep the most frequent 30,000 tokens according to their training sets, respectively. We use up to 3 turns of history as the input message, and truncate the message to 128 tokens and response to 32 tokens. A batch size of 256 is used.

For FL, we use a focusing factor of $\gamma = 2$ as recommended in (Lin et al., 2017). For the hyper-parameters introduced in this work, we use $n = 4$ for calculating the DIMEN scores, with $\alpha = [0.25, 0.25, 0.25, 0.25]$. We group the u-DIMEN scores into $m = 10$ bins, with the weighting vector $\beta = [0.9, 0.8, \ldots, 0.0]$.

6 Results

The results of all models on the three datasets are shown in Table 1. Since the validation repetition (WL2 scores) of models can be very low in the early stage of training when the l-DIMEN score is still very low, we mainly use l-DIMEN to make sure that the results are shown for checkpoints at similar stages.

Table 1 is divided into three main blocks row-wise. The first block ranging from rows (a)–(c) is for answering RQ1. The second (rows (d)–(g)) and third (rows (h)–(l)) blocks are for answering RQ2 and RQ3, with RNNs and transformer models, respectively. For reference, we also include the *Human* performance calculated using the ground truth of each dataset in the bottom row of Table 1.

Answer to RQ1: Row (a) of Table 1 are the results for RNN models with post-attention. From row (b), we can see that pre-attention is effective in reducing repetition on all three datasets. On top of pre-attention, row (c) shows that highway connections are helpful with repetition reduction on two datasets, namely Dailydialog and Movie Dialogs. BLEU scores show that the quality of the utterances generated by our models (b) and (c) are

		Dailydialog			Movie Dialogs			Ubuntu Dialogue		
		WL2↓	BLEU↑	l-DIMEN↑	WL2↓	BLEU↑	l-DIMEN↑	WL2↓	BLEU↑	l-DIMEN↑
(a)	RNN	8.04	**19.63**	0.35	36.71	0.35	0.25	208.7	0.08	**0.10**
(b)	RNN w/ pre-attn	5.70	18.17	**0.36**	**28.15**	**0.41**	**0.27**	173.0	**0.09**	0.08
(c)	(b) w/ highway	**5.50**	19.60	**0.36**	29.00	0.38	0.25	**157.1**	0.07	0.08
(d)	(b) w/ LDR	5.20	11.73	0.36	26.90	0.10	0.22	230.4	0.06	0.06
(e)	(b) w/ TFL	4.67	11.48	0.33	**19.11**	0.38	0.26	153.3	0.06	**0.08**
(f)	(b) w/ uniform	5.32	**18.47**	**0.37**	31.96	**0.43**	**0.27**	132.0	**0.08**	**0.08**
(g)	(b) w/ TLDR	**2.71**	17.61	0.36	22.45	0.39	0.26	**101.4**	0.05	**0.08**
(h)	transformer	7.72	20.38	0.35	42.50	**0.57**	**0.30**	79.84	0.05	0.11
(i)	(h) w/ LDR	7.92	19.19	**0.36**	42.40	0.28	0.29	255.0	0.05	0.12
(j)	(h) w/ TFL	7.33	20.69	0.35	38.52	0.51	0.29	103.2	0.05	**0.15**
(k)	(h) w/ uniform	**6.42**	**20.74**	**0.36**	39.13	0.56	0.29	150.6	**0.07**	**0.15**
(l)	(h) w/ TLDR	6.67	20.13	0.35	**28.61**	0.53	0.29	**75.11**	0.05	0.13
	Human	4.78	–	0.43	40.24	–	0.42	32.01	–	0.42

Table 1: Results for all our methods and the baselines. 'w/ pre-attn' and 'w/ uniform' are short for 'with pre-attention' and 'with weight of 2 uniformly'. Best scores in each column within each row-wise block are highlighted in **bold face**. ↑/↓ indicate higher/lower is better. All results are on the test set.

on a par with those generated by the baseline (a). These results indicate that proper architectural modifications can indeed reduce repetition, and more work needs to be done to find such modifications for transformers.

Answer to RQ2: From rows (e) and (j), we can see that in most cases TFL helps to reduce repetition, compared to rows (b) and (h), respectively. This empirically shows that our hypothesis on hard tokens is correct. However, due to monotonically down-weighting all training examples, the improvement brought by TFL is limited.

Answer to RQ3: Next, we proceed to our proposed weighting methods. Giving that the LDR and TLDR methods are about model learning, thus should be architecture insensitive, we use the RNN variant (b) for the following experiments, as (c) needs more computation due to the utilization of a gating (6). From the second and third blocks of Table 1, we can see that both RNN (row (g)) and transformer (row (l)) models trained using TLDR are consistently better than their corresponding baselines without weighting or with TFL weighting. Though TLDR occasionally performs worse than TFL, e.g., row (g) vs row (e) on the Movie Dialogs dataset, we note that this is probably due to sample bias of the test sets. We plot the result curves for the Movie Dialogs validation set at each validation point in Figure 4, where we can see that TLDR is no worse, if not consistently better, than the baselines. These comparisons indicate that dynamically weighting tokens according to their difficulty is an effective solution to reducing repetition.

The results in rows (d) and (i) show that, on the smaller datasets Dailydialog and Movie Dialogs, LDR has a limited effect on reducing repetition.

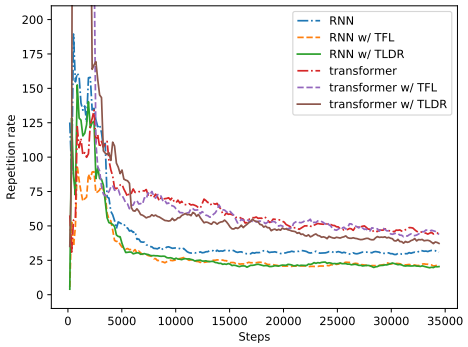


Figure 4: Repetition curves for Movie Dialogs validation set during training. TLDR can reduce the repetition rate earlier than baseline. Better viewed in color.

However, on the large Ubuntu Dialogue dataset, LDR hurts the repetition performance. Besides, LDR consistently results in lower BLEU scores, which implies that more work needs to be done to estimate the training difficulty at the example level.

Answer to RQ4: To understand whether the improvement of TLDR is brought by up-weighting, let us compare the performance of models trained using uniform weight $w = 2$ at rows (f) and (k) to rows (g) and (l), respectively. We can see that w.r.t. repetition, TLDR wins the majority of the times, which supports our claim that TLDR has more effect than simply up-weighting, as can also be seen from the gradient visualization in Figure 3. Multiplying a constant factor ($w = 2$ in our experiments) to the loss function also multiplies the gradients with the same factor, while TLDR changes the gradient function in a more complex way so that the gradients resulting from hard examples are amplified while those resulting from easy examples are suppressed, which is important for reducing repe-

tition. It is also worth noting that using a uniform weight of 2 is effectively doubling the base learning rate, which seems to be helpful with improving BLEU and l-DIMEN.

7 Conclusion and Discussion

We have studied the repetition problem of encoder-decoder architectures, using both RNN and transformer models. We have discovered that by using pre-attention instead of post-attention for RNNs, the repetition problem can be alleviated. Together with highway connections, repetitive generations can be further reduced.

We have hypothesized that the repetition problem is caused by hard tokens and find empirical support for this claim using FL. We then propose a more effective weighting function than FL, namely TLDR. With a differentiable cosine weighting function, TLDR amplifies the gradients resulting from hard tokens while it suppresses those from easy tokens. Through experiments we show that TLDR outperforms strong baselines like TFL and uniform weighting.

Our hard token hypothesis and the TLDR weighting function both operate mainly on target side tokens, while hard source side tokens can also have a detrimental effect on the decoder generations. Future work can be done by applying TLDR to source side language representation learning, possibly by training the encoder independently on a language representation task. It might also be worth exploring the effect on decoder-only models on tasks like LM.

We also observe that in our experiments, before the repetition rate converges during training, the repetition rate on the Ubuntu Dialogues validation set fluctuates with a regular pattern, which suggests that there can be certain training examples that can harm the repetition rate. If this is true, then the work by Sharchilev et al. (2018) can be used to target such harmful examples and hence reduce repetition. We leave this for future work.

Acknowledgments

This research was supported by the China Scholarship Council, Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), and the Innovation Center for Artificial Intelligence (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. 2018. Paying more attention to saliency: Image captioning with saliency and context attention. *TOMM*, 14(2):48.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd workshop on cognitive modeling and computational linguistics*, pages 76–87.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In *EMNLP*, pages 436–446.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Shaojie Jiang, Pengjie Ren, Christof Monz, and Maarten de Rijke. 2019. Improving neural response diversity with frequency-aware cross-entropy loss. In *The Web Conference*, pages 2879–2885.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv preprint arXiv:1911.03860*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*, pages 2980–2988.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70.

Ryan Lowe, Nissan Pow, Iulian Vlad Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Boris Sharchilev, Yury Ustinovsky, Pavel Serdyukov, and Maarten de Rijke. 2018. Finding influential training samples for gradient boosted decision trees. In *ICML*, pages 4584–4592.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NeurIPS*, pages 2377–2385.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 291–297.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *TACL*, 5:87–99.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 6000–6010.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.