Relating Neural Text Degeneration to Exposure Bias

Anonymous ACL submission

Abstract

This work focuses on relating two mysteries in neural-based text generation: exposure bias, and text degeneration. Exposure bias, despite the long time since the it was proposed and the numerous studies for its remedy, to our knowledge, its impact on text generation has not yet been verified. Text degeneration, is a problem the widely-used pre-trained language model GPT-2 was recently found to suffer from (Holtzman et al., 2020). Motivated by the unknown causation of the text degeneration, in this paper we attempt to relate these two mysteries. Specifically, we first qualitatively and quantitatively identify mistakes made before text degeneration occurs. Then we investigate the significance of the mistakes by inspecting the hidden states in GPT-2. Our results show that text degeneration is likely to be caused by exposure bias. We also study the self-reinforcing mechanism of text degeneration, explaining why the mistakes amplify. In sum, our findings and experiments provide a more concrete foundation for further investigation on exposure bias and text degeneration problems.

1 Introduction

One mythology in neural text generation is exposure bias (Bengio et al., 2015; Pomerleau, 1989; Thrun, 1995). In the context of text generation, exposure bias refers to mistakes made by the model at the beginning of text generation, which may amplify, and lead the model to a state unseen in training time, and may thus cause misbehavior in the following generation. Phenomena related to exposure bias were first observed in (Pomerleau, 1989) in the self-driving vehicles field. After that, exposure bias was mainly discussed in the context of imitation learning (Thrun, 1995; Ross and Bagnell, 2010; Ross et al., 2011). In 2015, it was introduced in the context of neural text generation (Bengio et al., 2015). However, its impact on text generation is questionable from both the empirical and theoretical perspectives. Empirically, despite the number of studies for its remedy (Bengio et al., 2015; Huszár, 2015; Ranzato et al., 2016; Lamb et al., 2016; Yu et al., 2017; Wiseman and Rush, 2016; Schmidt, 2019; Zhang et al., 2019a), phenomena about exposure bias have not yet been explicitly identified. On the other hand, theories attained in the context of imitation learning are not necessarily applicable for the above text generation tasks. For example, (Ross and Bagnell, 2010) achieves a $O(T^2)$ trend of cost with respect to number of step $T$ in an episode. It implies that the cost grows quadratically when $T$ is large. However, most of the above empirical tasks, such as machine translation and image captioning, do not generate very long text. It is thus not clear how impactful exposure bias is in those text generation tasks.

A younger mystery is the recently found enigma of text degeneration (Holtzman et al., 2020). It refers to the phenomenon in which bland or strange repetitive texts may be generated when the likelihood is the objective of generation, for example, when some commonly used strategies, such as greedy decoding and beam-search decoding, are used. Especially, the prior work (Holtzman et al., 2020) observed such problems in GPT-2 (Radford et al.), a pre-trained language model that has been shown useful in many NLP tasks (Radford, 2018; Zhang et al., 2019b; Petroni et al., 2019; Talmor et al., 2019; See et al., 2019). Despite many attempts proposed to address this issue (Holtzman et al., 2020; Welleck et al., 2020; Li et al., 2019), its root cause remains unknown.

Motivated by the unknown issues, we wonder whether text degeneration can be connected to the well-known exposure bias. If text degeneration is the misbehavior caused by exposure bias, it actually provides us a perfect opportunity to identify the existence of exposure bias. One of misbehavior of text degeneration is the occurrence of repeti-
GraphiQL is an interesting technology originating at Facebook. It is a query language that allows you to query a database and then query the database for the results. The query language is called GraphQL. It is a query language...

We first saw Anki Overdrive, the company’s follow-up to the original game, in the early 2000s. It was a game that was a bit of a hit that was a bit of a hit that was a bit of a hit that was a bit of a hit that was a bit of a hit that was a bit of a hit. It indicates that the significance of mistakes made at the early state. According to the indications we discover, we conclude that exposure bias is likely similar to the states generated by encoding real text. It is a phenomenon that a model tends to repeat a span of text during generation (an example is shown in Table 1). This phenomenon is salient enough to be detected automatically, and occur when greedy decoding strategy is used with high probability. The easiness of spotting can help the identification of exposure bias. Therefore, this work aims at looking for the indications of exposure bias when repetitive loops are generated by the greedy decoding strategy.

To the best of our knowledge, this paper is the first work that attempts to relate text degeneration to exposure bias. Based on the intuition of exposure bias, we conclude two necessities of its occurrence: 1) mistakes are made at the early phase of text generation, and 2) the mistake is significant to the model. We look for the two necessities in the behavior of GPT-2. We focus on GPT-2, because it is the only publicly available language model trained on massive amount of data at the time this work is done, and it is widely used by the community.

We then inspect the two necessities with experiments. For the first necessity, we inspect the text generated in the early phase qualitatively and quantitatively. Our results show evidence of mistakes made at the early phase. For the second necessity, we inspect the hidden states of the generation model at each decoding time step. These states reflect the way the model processes the generated text it conditions on. We find that before repetitive loops occur, the states deviate to an area less similar to the states generated by encoding real text. It indicates that the significance of mistakes made at the early state. According to the indications we discover, we conclude that exposure bias is likely to cause repetitive loops.

Finally, we investigate how the mistakes made in the early stage are amplified. We discover the self-reinforcing mechanism of text degeneration. Combined the results from the aforementioned experiments, it provide a possible outline how a model is trapped in repetitive loops. Our findings should be helpful for future studies on exposure bias and remedies for text degeneration.

2 Related Work

2.1 Exposure Bias in Imitation Learning

Imitation learning aims at imitating an expert policy $\pi^*$ by learning from trajectories generated by the expert, namely finding the policy

$$\hat{\pi} = \arg \min_{\pi} E_{s \sim d_{\pi^*}} I[\pi(s) = \pi^*(s)],$$

where $d_{\pi^*}$ is the distribution of states visited by the expert policy $\pi^*$. It has succeed in many applications (Pomerleau, 1989; Schaal, 1999; Muller et al., 2006; Ratliff et al., 2006). However, it was mentioned in (Pomerleau, 1989) that when a model makes a mistake and thus encounters a state that the expert rarely encounters, it may fail to recover from the mistake. It was the first time the concept of exposure bias was mentioned. Similar issue was also considered in (Thrun, 1995; Daume et al., 2009). Ross and Bagnell proved that the cost in a trajectory grows at the rate $O(T^2)$ instead of $O(T)$ if mistakes are made with a non-zero probability. It can be seen as a theoretical analysis of exposure bias. Nevertheless, in the context of text generation, the total number of steps in a trajectory is finite and is usually not large. Therefore, it is still not clear how meaningful this growth rate of cost is for text generation tasks. In Ross and Bagnell; Ross et al., theoretically-grounded algorithms are proposed. However, they require the access of expert policy to annotate the trajectories generated by the learnt agent. It is generally not feasible in text generation tasks.

2.2 Exposure Bias in Text Generation

Then the concept of exposure bias is introduced in the context of text generation by (Bengio et al., 2015; Ranzato et al., 2016). Since then, there have been many methods proposed to tackle this problem (Bengio et al., 2015; Huszár, 2015; Ranzato et al., 2016; Lamb et al., 2016; Yu et al., 2017; Wiseman and Rush, 2016; Schmidt, 2019; Zhang et al., 2019a; Wang and Sennrich, 2020). They proposed their remedies based on the assumption...
that exposure bias is causing problems, and their approaches were justified by the improvement of performance when they are adopted. However, to our knowledge, He et al. (2019) is the only study attempting to verify the impact of exposure bias, where they proposed metrics for estimating the impact of exposure bias in models. Different from the prior work, this paper focuses on directly checking whether a specific phenomenon is resulted from exposure bias.

2.3 Neural Text Degeneration

The term neural text degeneration was first defined recently in Holtzman et al. (2020), which focused on GPT-2. Similar phenomenon was also observed in LSTM language models (Strobelt et al., 2018). Regarding to its causation, Welleck et al. (2020) summarized three possible reasons about repetitive loops generated by GPT-2: i) The Transformer architecture of GPT-2 prefers repeating. ii) Repeating is an intrinsic property of human language. iii) The model is unable to model real language usage due to the fixed training corpora. However, none of them have been proven theoretically or verified empirically.

Before this work, this phenomenon has not been linked to exposure bias, and thus remedies different from those for exposure bias are proposed. Holtzman et al. (2020) proposed sampling from the language model with nucleus sampling. Welleck et al. (2020) proposed to train neural language models with an unlikelihood as a regularization. Li et al. (2019) further applied unlikelihood training on dialogue tasks. Since in this work we discover the link between exposure bias and text degeneration, new approaches that specifically tackle exposure bias may be found effective for text degeneration in the future.

3 Background and Notations

To better elaborate the investigation of the above problems, background knowledge and notations are briefly introduced here.

3.1 Real and Fake Passages

Considering that this paper focuses on analyzing the issues in text generation, we first define real passages as natural language and fake passage as the generated language for following study.

Real Passages and Real Distribution Real passages and real distribution are related to training data. Given $Y$ denoting the training set, a real passage $y \in Y$ is a sequence of tokens $\{y_1, y_2, \cdots, y_T\}$, and real distribution $P_Y$ is the distribution passages $y \in Y$ are drawn from, and it can be factorized as

$$P_Y(y) = P_Y(y_1) \prod_{t=2}^T P_Y(y_t | y_1, y_2, \cdots, y_{t-1}).$$

(2)

Fake Passages and Fake Distribution A fake passage $\hat{y}$ is a sequence of tokens $\{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T\}$ generated by a model. We denote the set of generated passages as $\hat{Y}$, where each $\hat{y}$ is generated based on the conditional probability, $P_M(\hat{y}_t | \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1})$, predicted by an auto-regressive language model $M$ such as GPT-2. We define fake distribution $P_{\hat{Y}}$ as the distribution of $\{\hat{y} \in \hat{Y}\}$ detailed below. Note that $P_{\hat{Y}}$ could be different from $P_M$, depending on the decoding strategy used. A decoding strategy is how a token $\hat{y}_t$ is chosen based on the conditional probability $P_M(\hat{y}_t | \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1})$. Decoding strategies we considered includes:

- Greedy: At the time step $t$, $\hat{y}_t$ that maximizes the conditional probability is chosen, namely $\arg \max_{\hat{y}_t} P_M(\hat{y}_t | \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1})$.
  It aim at maximizing $P_M(\hat{y}_t)$, and text degeneration is found pervasively when this strategy is used.
- Sampling-based: This category of strategies includes directly sampling, sampling from the top-k candidates at each step (Fan et al., 2018), nucleus sampling (Holtzman et al., 2020). Details are included in the appendix.

3.2 States of GPT-2

GPT-2 is a pre-trained language model constituted with $L$ layers of Transformer blocks (Vaswani et al., 2017). Considering that exposure bias is described as a general problem of neural text generation models, we pick GPT-2 as an example model for the study. When the tokens $\{y_t\}_{t=1,2,\cdots,T}$, which we refer to as the conditioned passage, are fed in, we denote the states outputted by each layers as

$$[h_{1,1}^y, h_{1,2}^y, \cdots, h_{1,T}^y] = \text{transformer}_1(\text{embedding}([y_1, \cdots, y_T])), \quad (3)$$

$$[h_{l,1}^y, h_{l,2}^y, \cdots, h_{l,T}^y] = \text{transformer}_l([h_{l-1,1}^y, \cdots, h_{l-1,T}^y]), \quad \forall l = 1, 2, \cdots, L. \quad (4)$$
It predicts the conditional probability as

$$P(y_T \mid \{y_t\}_{t=1}^{T-1}) = \text{softmax}(\text{MLP}(h_{L,T-1}^{(y)})). \quad (5)$$

We refer to real states as the states outputted when $y \sim Y$ is fed in, and fake states as the states when $\hat{y} \sim \hat{Y}$ is fed in. States of a token $y_t$ refer to the set of states $\{h_{l,T}^{(y)}\}_{l=1,2,\cdots,L}$.

### 3.3 Exposure Bias

In the literature, exposure bias was conceptually proposed (Bengio et al., 2015), which is described as the discrepancy between the way the model is used during training and the way during inference. When training, at the time step $t$, the model objective is to maximize the probability of the correct token $y_t$ conditioning on the real past tokens $y_1, y_2, \cdots, y_{t-1}$. However, during inference, $\hat{y}_t$ is predicted conditioning on the generated past tokens $\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}$. Therefore, mistakes in the early stage may lead the model to a state unseen in training time, and errors may consequently amplify quickly.

#### 3.4 Repetitive Loops

Let the time step at which a passage $\hat{y}$ starts to repeat be $\rho$, and the length of the repeated part be $\lambda$. Then a passage $\hat{y}$, where a repetitive loop occurs, is of the form

$$\hat{y} = \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\rho-1}, \hat{y}_\rho, \cdots, \hat{y}_{\rho+\lambda}, \hat{y}_\rho, \cdots, \hat{y}_{\rho+\lambda}, \cdots \quad (6)$$

We refer to the repeated part $\hat{y}_\rho, \cdots, \hat{y}_{\rho+\lambda}$ as a looping sequence.

### 4 Relating Text Degeneration to Exposure Bias

More explicitly, based on the description of bias in Bengio et al. (2015), we summarize the necessities as follow: If some misbehavior, such as repetitive loop, starts at time $t$ is the result of exposure bias, then the two conditions must be satisfied:

#### 1. Mistakes are made in the early phase: In the context of text generation, qualitatively, it means the unnatural sequence is generated before time step $t$. Quantitatively, it means that $P_{\hat{y}}(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1})$, the likelihood that the previous generated text is real, is low.

#### 2. Mistakes are significant to the model: The mistakes must be significant enough to lead the model to a state unseen in training time. Specifically, here we analyze the hidden states of GPT-2. We posit that, if some misbehavior is due to exposure bias, then the mistakes in the early stage should be significant enough to cause the model to generate an unseen state.

In this section, we investigate whether the above conditions are satisfied when text degeneration occurs.

#### 4.1 Experimental Setting

As in Holtzman et al. (2020), we focus on the pre-trained language model GPT-2. GPT-2 is trained on the WebText dataset. We use the training, validation and testing subsets of WebText released by OpenAI.

When generating passages, first 50 tokens from passages $y \in Y$ are given as the condition. Therefore, for different conditions $y$, even if the decoding strategy is deterministic, the generated passages $\hat{y}$ could be different. We empirically observe that repetitive loops tend to occur later when the number of conditioned tokens is greater. We choose to condition on 50 tokens, so the sequences before repetitive loops are lengthy enough for analysis while the computation power required is affordable.

#### 4.2 Qualitative Inspection on Generated Tokens Prior to Repetitive Loops

We inspect the first condition about exposure bias by subjectively examining the passages generated before a repetitive loop occurs. For each passage $\hat{y}$ generated by conditioning on $\{y\}_{t=1,2,\cdots,50}$, we compare the pair $\hat{y}_{t=51,\cdots,\rho-1}$ (generated) and $y_{t=51,\cdots,\rho-1}$ (real), where $\rho$ is the time step where the repeating sentence first appears. We want to check if the model does make mistakes during $t = 51, \cdots, \rho - 1$. We manually examine 50 randomly sampled pairs. We observe that the generated passages are often less informative, less relevant or coherent to $\{y\}_{t=1}^{50}$. As a result, without knowing which passage in the pair is real, we can still correctly identify the generated ones for 78% of them. We also inspect the

---

2We use the implementation from Hugging Face (https://huggingface.co/transformers/index.html).

3https://github.com/openai/gpt-2-output-dataset
sequence pairs from time 0 to $\rho + \lambda - 1$, the time step after which the model starts to repeat. In that case, our correctness is even higher, up to 92%. It shows that a portion of passages generated in the early stage are perceivably dissimilar to real language. Namely $P_Y(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\rho-1})$ and $P_Y(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\rho+\lambda-1})$ is low from human judgement. It is an indication that the mistakes are indeed generated before the repeating loop occurs. Hence, the first condition about exposure bias is satisfied.

4.3 Quantitative Inspection on Generated Tokens Prior to Repetitive Loops

We further inspect the first condition of exposure bias quantitatively and objectively. We want to estimate $P_Y(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\rho-1})$, the likelihood $\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{\rho-1}$ is real. However, the true $P_Y$ is not tractable. Using an auto-regressive models to estimate the likelihood is not feasible either, since they may give higher probability to passages that is also generated by auto-regressive models and thus favor GPT-2. To this end, we choose to use a pre-trained masked language model RoBERTa-Large (Liu et al., 2019). It is trained non-autoregressively, so it does not favor auto-regressively generated passages. Therefore it should be a good proxy estimating the realness of the passages generated by GPT-2.

Specifically, to estimate the likelihood of tokens in a passage, real passages and fake passages with repetitive loops are fed in RoBERTa with 15% randomly selected tokens masked. Log likelihood of recovering the masked tokens is calculated. To anneal the randomness due to the selection of masked tokens, this process is repeated 10 times for each passage. Finally, the likelihood for each time step is averaged and shown in Figure 1. The results show that starting from the time step where the conditioned passages end (dashed line), the likelihood of the generated passages are lower than real passages, indicating that the generated passages are indeed unreal.

4.4 Significance of Mistakes Prior to Repetitive Loops

We then check how significant the mistakes are to the GPT-2 model. Though the previous sections have shown the existence of the mistakes in the early stage. However, to cause misbehavior, the mistakes must be significant enough to cause GPT-2 to behave differently. Therefore, we check how differently GPT-2 processes the generated text compared to the real ones.

4.4.1 Measuring the Significance of Mistakes

To measure the significance of mistakes, we inspect the hidden states of GPT-2 when generating passages. For each layer $l > 1$ and time step $t$, the fake state $h^\langle\hat{y}\rangle_{l,t}$ is the result of applying the transformer function $l - 1$ times over the input sequence $\{\hat{y}_{l-1}, \tau\}_{\tau=1...t-1}$, which is the generated passage. Therefore, if a fake state $h^\langle\hat{y}\rangle_{l,t}$ is significantly dissimilar to any real states, then it implies that the generated passage $\{\hat{y}_{l-1}, \tau\}_{\tau=1...t-1}$ contains mistakes that are significant to the model, and that the mistakes do cause the model to an unseen state. Thus, the similarity between the fake states and the real state indicates how significant the mistakes in the passage are.

Specifically, we measure how many real state is similar to a fake state. It is done by calculating the number of real states in the neighbor of the fake state. A lower number of real neighborhoods suggests that the fake state is more unseen, and thus implies higher significance of the mistakes.

Formally, given a hidden state $h^\langle\hat{y}\rangle_{l,t}$ at the time step $t$ in the layer $l$, we calculate the number of real neighbors

$$N\left(h^\langle\hat{y}\rangle_{l,t}\right) = \left| \left\{ h \in H_{l,t} \mid \left\| h^\langle\hat{y}\rangle_{l,t} - h \right\|_2 < r \right\} \right|$$

(7)

where $r$ is the predefined radius, and $H_{l,t}$ is the support set of real states to be considered.

The constitution of $H_{l,t}$ depends on the layer $l$ and the time step $t$ of the hidden state $h^\langle\hat{y}\rangle_{l,t}$ to be considered. $h^\langle\hat{y}\rangle_{l,t}$ is compared with only the real state of the same layer. We also limit the set $H_{l,t}$ to the state of the tokens with time step differ to $t$ by less than $\delta^4$, because we found the distribution of the states is time-step-dependent. It is shown by

$^4$We use $\delta = 5$. 

![Figure 1: The average log likelihood (y-axis) predicted by RoBERTa at each time step (x-axis).](image-url)
of fake states’ real neighbor in \( h_{\text{sup}} \). We also use \( y_{\text{cond}} \) calculate \( N(h_{\text{cond}}) \). It is referred to as "real" in Figure 3 and 4.

We consider two approaches to construct \( Y_{\text{sup}} \) and \( Y_{\text{cond}} \):

**compare-seen** The training split is used as \( Y_{\text{sup}} \). They are *seen* when training. Real passages in the validation split and the testing split are used as \( Y_{\text{cond}} \).

**compare-unseen** The union of the validation split and the testing split is split into two disjoint subsets by ratio 9:1. They are used as \( Y_{\text{sup}} \) and \( Y_{\text{cond}} \) respectively. \( Y_{\text{sup}} \) is *unseen* when training.

### 4.4.3 Sanity Check

We experiment with a set of *shuffled* states as a sanity check of our approach. It verifies whether the number of neighbor is indicative to measure the significance of mistakes. The *shuffled* set is constructed by first shuffling the real passages in the \( Y_{\text{cond}} \), and is then encoded with GPT-2. The shuffled passages have the same 1-gram distribution as real natural language, but has low likelihood to be real. We expect them to have low numbers of real neighbors.

The results show that the number of real neighbors is a good indication of mistakes for middle layers from layer 5 to layer 9 when \( r = 1024 \) for both the compare-seen and compare-unseen settings. The average number of neighbors for different time steps at the seventh layer is plot in Figure 3. We include the results of other layers in the appendix. The figure shows that the number of neighbor of the *shuffled* states are consistently low for all time steps. It implies that the number of neighbor is indicative for detecting unreal passages. However, it is less indicative when \( R \) is small. We posit that it is due to the high sparsity of the states due to their high dimensionality \(^5\).

### 4.4.4 Results

Figure 3 also plots the number of real neighbors \((h)\) for states generated with greedy strategy and the sampling-based strategies \((\hat{h})\). For the greedy strategy, the number of neighbor declines rapidly as time steps increases. Note that we observe that repetitive loops occur in about 93% of the sequences. It shows that GPT-2 indeed fails to recover from mistakes, and the mistakes are amplified through time. It is aligned with the description of

---

\(^5\) Each state \( \in \mathbb{R}^{1536} \)
exposure bias. On the other hand, compared with real sequences (the control group), the number only decreases slightly when sampling-based strategies are used. In contrast to the case of greedy decoding, repetitive loops are rarely observed when those sampling-based methods are used (< 1% for all of the strategies). It implies that if GPT-2 has misbehavior when using those strategies, the misbehavior is less likely to be related to exposure bias.

We further inspect the number of neighbors of the fake state prior to the time step \( \rho + \lambda \), when a repetitive loop starts. We want to know, whether the model does make significant mistakes before \( \rho + \lambda \). It is not shown in Figure 3, as it only shows the significance of mistakes in the late stage. To this end, we plot the number of neighbors again in Figure 4. Different from Figure 3, in Figure 4, the \( x \)-axis is the time step \( \text{relative to } \rho + \lambda \), so the significance of mistakes before repetitive loops can be manifested. In particular, we compare the number of real neighbors around the real states and the fake states. Formally, for each fake passage \( \hat{y} \) conditioning on \( y_{1,2,\ldots,50} \), we compare the number of neighbors around the state of \( n_{l,t}(\hat{y}) \), and the state of the real passage following the condition \( n_{l,t}(y) \). Here we set the \( y \)-axis of Figure 4 to be the difference \( (n_{l,t}(\hat{y}) - n_{l,t}(y)) \).

Surprisingly, in Figure 4, the compare-seen and compare-unseen settings show different trends. At the beginning, the number of neighbors decreases relatively slowly in both of the two settings. At around of the place \( x = -10 \), the number in both of them drop to less than zero. It indicates that at this time step, some significant mistakes are made. However, the number in the compare-seen setting dramatically grows while the number continues decreasing in the compare-unseen setting. The low number of neighbor in the compare-unseen indicates the low realness of the generated passages. The high number of neighbor in the seen-setting indicates that the model encodes those unreal passages to space close to the states of training data. It may imply that, at this moment, the model fails to generalize, so it incorrectly encodes the unreal passages as seen ones. Finally, the mistakes are amplified. Consequently, the number in both of the settings drops to less than zero. In sum, Figure 4 shows the significance of mistakes made before it starts repeating a looping sequence. Therefore, the second necessity of exposure bias is satisfied.

Table 2: Similarity between the conditioned passage and the generated passage of the same length.

<table>
<thead>
<tr>
<th>Conditioned Sentences</th>
<th>Similarity (mean/std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looping sequences</td>
<td>0.7327 / 0.3226</td>
</tr>
<tr>
<td>First sentences</td>
<td>0.2157 / 0.1911</td>
</tr>
<tr>
<td>Last sentences</td>
<td>0.1837 / 0.1848</td>
</tr>
</tbody>
</table>

5 Mechanisms after a Repetitive Loop Starts

While the above experiments show the indications of exposure bias, in this section we further investigate how the early stage mistakes cause the model to degenerate. Figure 4 indicates some mistakes are made prior to time step \( \rho + \lambda \). Thus, in this
section, we investigate the characteristics of the sequence generated prior to \( \rho + \lambda \), the looping sequence \( \hat{y}_\rho \cdots \hat{y}_{\rho+\lambda} \) (as defined in Section 3.4).

### 5.1 The Looping Sequence is Loop-Inducing

We investigate how the looping sequences are loop-inducing by using them as condition when generating text. We construct a looping sequence set that is constituted with all looping sequences generated when conditioning on the first 50 tokens of real sequences. In a generated sequence \( \hat{y} \), since \( \hat{y}_\rho \) may not be a start point of a grammatical sentence, we use the sequence \( \hat{y}_{\rho+\delta+1} \cdots \hat{y}_{\rho+\lambda} \hat{y}_\rho \cdots \hat{y}_{\rho+\delta} \), where \( \delta \) is chosen based on the punctuation in it \(^6\). As control groups, we also construct two real sequences sets, first sentence set and last sentence set. They consist of the first sentence and the last sentence of the articles in WikiText validation split and testing split.

To measure how those sequences are loop-inducing, we calculate the similarity between \( x \) and \( \hat{y} \), where \( x \) is the sequence used as condition, and \( \hat{y} \) is the generated passage. Specifically, we measure ROUGE-L (Lin, 2004)\(^7\) between \( x \) and the first \( \text{length}(x) \) tokens of \( \hat{y} \). A higher score implies higher similarity, and thus more loop-inducing. Results shown in Table 2 indicate that looping sequences are indeed more loop-inducing.

### 5.2 Any Repeating Sequence is Loop-Inducing

We further discover that any sequence that is repeated is loop-inducing, regardless of contexts. We create the conditioned sequence by concatenating \( c \) with \( x \) repeated from 1 to 3 times, where \( c \) is the first 5 sentences from a random article of WebText, and \( x \) is either from the looping sequence set or the real sets. Measurement same as in Section 5.1 is applied on \( x \) and the generated passages. The results are shown in Table 3, and it shows that even when the conditioned sequence is real, it is more loop-inducing if it is repeated more times.

### 5.3 The Self-Reinforcing Mechanism of Text Degeneration

In sum, in this section, we discover the self-reinforcing mechanism of text degeneration. First, Section 5.1 a looping sequence is loop-inducing. Thus, after a looping sequence is generated, it is likely to be repeated. Second, Section 5.2 shows that when a sequence is repeated, then GPT-2 would be more likely to continue repeating it. Therefore, it shows how GPT-2 fails to recover from the mistake.

### 6 Conclusion

In conclusion, we provide a deeper insight into the relation between exposure bias and text degeneration. We qualitatively and quantitatively show that mistakes are indeed made in the early stage of generation. Especially, some significant mistakes are made prior to \( \rho + \lambda \), the time step when the model starts repeating. We then show why the model fails to recover from the mistakes. The looping sequence, which is the sequence generated prior to \( \rho + \lambda \), is loop-inducing. Also, repeated sequences are also loop-inducing. That is how the model fails to recover from the mistakes, and how the mistakes amplify.

Our contributions are four-fold: 1) We explicitly formulate the necessities of exposure bias. 2) For each necessity, we design the associated experiments for validation. 3) By the experiments, we relate text generation to exposure bias. 4) Finally, we provide a possible explanation how a model fails to recover from the mistake. Our formulation and the conducted experiments build a solid foundation for future study on exposure bias.

\[^6\] For example, if the looping sequence is "an apple. It is", we use "It is an apple."

\[^7\] We use the implementation in [https://github.com/google-research/google-research/tree/master/rouge](https://github.com/google-research/google-research/tree/master/rouge).

<table>
<thead>
<tr>
<th>Repeat #</th>
<th>Looping Seq.</th>
<th>First Sent.</th>
<th>Last Sent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4514 / 0.3682</td>
<td>0.1483 / 0.1545</td>
<td>0.1306 / 0.1435</td>
</tr>
<tr>
<td>2</td>
<td>0.6805 / 0.4228</td>
<td>0.3307 / 0.3730</td>
<td>0.3365 / 0.3770</td>
</tr>
<tr>
<td>3</td>
<td>0.8876 / 0.2819</td>
<td>0.4921 / 0.4347</td>
<td>0.5780 / 0.4309</td>
</tr>
</tbody>
</table>

Table 3: The ROUGE-L (mean/std) between the sentences in the generated repetitive loops and \( x \), when GPT-2 conditions on the pattern \( c, x, \cdots, x \).
References


Alec Radford. 2018. Improving language understanding by generative pre-training.


A Sample-based Decoding Strategies

- Sampling: \( \hat{y}_t \) is directly sampled from the conditional probability \( P_M(\hat{y}_t \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) \).

- Top-\( k \) sampling (Fan et al., 2018): At the time step \( t \), \( \hat{y}_t \) is sampled from the conditional probability:

\[
P_Y(\hat{y}_t \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) \propto \begin{cases} 
P_M(\hat{y}_t \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) & \text{if } \hat{y}_t \in \text{top-}\( k \),} \\ 0 & \text{otherwise.} \end{cases}
\] (9)

- Nucleus sampling (Holtzman et al., 2020): At the time step \( t \), \( \hat{y}_t \) is sampled from the conditional probability

\[
P_Y(\hat{y}_t \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) \propto \begin{cases} 
P_M(\hat{y}_t \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) & \text{if } \hat{y}_t \in V^{(p)},} \\ 0 & \text{otherwise.} \end{cases}
\] (10)

and for a predefined \( p \in (0, 1] \), \( V^{(p)} \) is the minimal set that satisfies

\[
\sum_{v \in V^{(p)}} P_M(v \mid \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{t-1}) \geq p \quad \text{(11)}
\]

B Dataset

We use the subsets of WebText released by OpenAI (https://github.com/openai/gpt-2-output-dataset). It is an English dataset. There are 25000, 5000, 5000 passages in the train, validation, testing splits respectively. For experiments in Section 4.3 and Section 4.4), we only the passages with more than 512 tokens are used. After passages with less than 512 tokens are removed, there are 5269 passages in the union of the validation split and the testing split.

C Detail of Experiments

C.1 Quantitative Inspection of Generated Tokens Priors to Repetitive Loops (Section 4.3)

Implementation of RoBERTa from Python package transformers 2.8.0 by Hugging Face (https://huggingface.co/transformers/) is used.
C.2 Significance of Mistakes Prior to Repetitive Loops (Section 4.4)

We use Faiss (Johnson et al., 2017) to calculate the number of neighbor vectors within a radius. For Figure 3, the number of neighbors is calculated for 20 time steps. For Figure 4, the number of neighbors is calculated at time steps \{-32, -16, -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 16, 32, 64, 128\} relative to $\rho + \lambda$.

**Seen-setting:** 2500 passage are sampled from the WebText training split. Each line in Figure 4 is the average over 500 passages generated by each decoding strategy. The result in Figure 4 is the average over 1000 passages.

**Unseen-setting:** We first combine the validation split and the testing split as the set of all real unseen text $\bar{Y}$. Then we split it into 10 equal-sized subsets $\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_{10}$. We repeat the following process 3 times:

- From \{\$\bar{Y}_1, \bar{Y}_2, \ldots, \bar{Y}_{10}\}, a subset $Y_{real}$ is selected, and the rest $\bar{Y} \setminus Y_{real}$ is used as the support set $Y_{support}$.
- Real states are collected by encoding passages in $Y_{support}$ with GPT-2. When we are calculating the number of neighbors, only these real states are counted.
- Fake passages are generated by conditioning on the first 50 tokens for passages in $Y_{real}$ using the decoding strategies.
- The number of neighbors is calculated for each decoding strategies.

Finally, the result is averaged to plot Figure 3 and Figure 4.

C.3 Automatic Detection of Looping Sequence

Given a passage $x_1, x_2, \ldots, x_T$, we first search for the length of a repetitive loop by comparing $x_{T-\lambda+1}, \ldots, x_T$ and $x_{T-2\lambda+1}, \ldots, x_{T-\lambda}$ for $\lambda = 4, 5, \ldots, \lfloor T/2 \rfloor, 1, 2, 3$. If there exists some $\lambda$ such that $x_{T-\lambda+1}, \ldots, x_T = x_{T-2\lambda+1}, \ldots, x_{T-\lambda}$, then we search $\rho$ as the first place such that $x_{\rho+i\lambda}, \ldots, x_{\rho+(i+1)\lambda-1} = x_{T-2\lambda+1+\delta}, \ldots, x_{T-\lambda+\delta}$ for some $\delta \in [0, \lambda - 1]$ and all $i$ such that $\rho + i\lambda < T$.

D Computing Infrastructure

Each of our experiments were run on a workstation with 187 GiB RAM. A workstation is equipped with either two Intel Xeon 5218 CPUs or two Intel Xeon 4110 CPUs. Every experiment can be run with 1 Nvidia GTX 2080Ti GPU.
Figure 5: Number of neighbors for compare-seen setting. The figures are the number of layer 1, 3, 5, 7, 9, 11, from left to right, top to bottom. The x-axis is the time step of the tokens. The y-axis is the number of real neighbors with the radius.

Figure 6: Number of neighbors for compare-unseen setting. The figures are the number of layer 1, 3, 5, 7, 9, 11, from left to right, top to bottom. The x-axis is the time step of the tokens. The y-axis is the number of real neighbors with the radius.