Semi-Offline Reinforcement Learning for Optimized Text Generation

Anonymous Authors[1]

Abstract

In reinforcement learning (RL), there are two majors settings for interacting with the environment: online and offline. Online methods explore the environment with expensive time cost, and offline methods efficiently obtain reward signals by sacrificing the exploration capability. We propose semi-offline RL, a novel paradigm that can smoothly transit from the offline setting to the online setting, balances the exploration capability and training cost, and provides a theoretical foundation for comparing different RL settings. Based on the semi-offline MDP formulation, we present the RL setting that is optimal in terms of optimization cost, asymptotic error, and overfitting error bound. Extensive experiments show that our semi-offline RL approach is effective in various text generation tasks and datasets, and yields comparable or usually better performance compared with the state-of-the-art methods.

1. Introduction

Pre-trained language models have achieved great success in improving the quality of text generation (Devlin et al., 2019; Liu et al., 2019). Recent research shows that a key for further improving pre-trained language models is reinforcement learning (RL), which provides a principled solution for directly optimizing the final objective such as ROUGE (Lin & Hovy, 2003), factual correctness (Goodrich et al., 2019), and satisfaction of user needs (Ouyang et al.). Compared with traditional optimization objectives, such as the maximum likelihood estimation (MLE), reinforcement learning solves problems such as exposure bias and the mismatch between MLE and the final optimization objective, leading to great success in further refining large pretrained models. Recent large pretrained models that incorporate reinforcement learning, e.g., InstructGPT (Ouyang et al.) and ChatGPT[1], have demonstrated superior performance in aligning with user needs compared to the original GPT-3.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

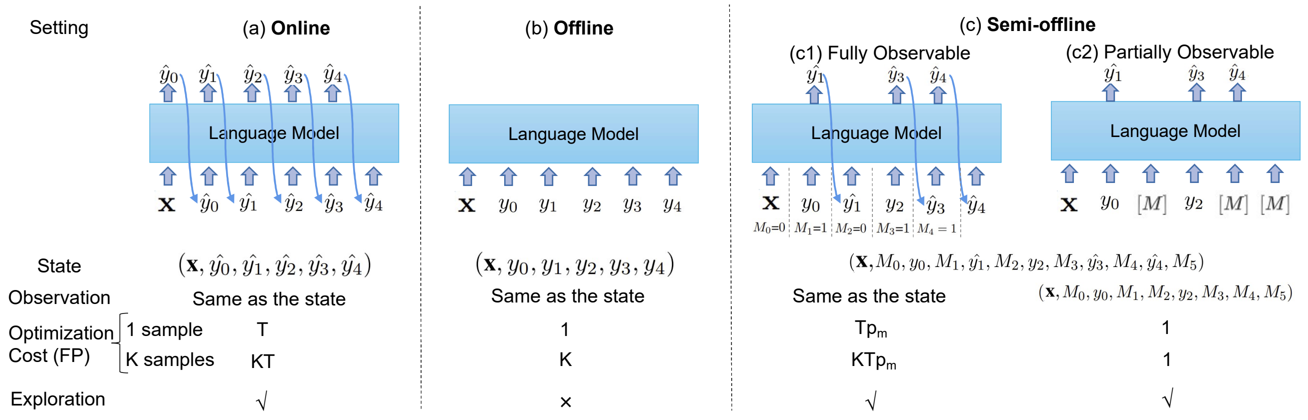
In reinforcement learning, there are two major settings for interacting with the environment: online and offline.

Online RL (Fig. 1(a)). The language model in this setting generates word token $\hat{y}_t$ by sampling from its output probability distribution, and obtains the reward signal about the samples to learn how well they fulfill the final objective. (Paulus et al., 2018; Li et al., 2019; Schulman et al., 2017; Le et al., 2022) The online setting allows the language model to **fully explore** the environment: the model can interact with the environment to see the reward of different samples and hence obtains a comprehensive understanding about the final objective, which is crucial for finding the optimal generation. While it is theoretically guaranteed that the optimal generation can be found when the number of samples approaches infinity, empirically, it is **time-expensive** to obtain even only a few samples from large pretrained language models. In particular, optimizing with K samples requires KT forward propagations (FPs) through the language models, where T is the maximum number of tokens in the generated text. This cost is quite large and impractical in some real-world scenarios (Wang et al., 2021) considering the complexity of large language models.

Offline RL (Fig. 1(b)). This setting eliminates the need for generating text during the training process by utilizing a static dataset for learning. Example static data $y_1, \cdots, y_T$ includes demonstrations (Pang & He, 2020; Jaques et al., 2019; Serban et al., 2017) (or ground-truth labels) for an input $\mathbf{x}$ and text pre-generated with beam search (Liu et al., 2022). By avoiding generating text in an auto-regressive manner during training, offline methods **mitigate the expensive optimization costs** associated with online methods and reduces the cost from KT forward propagations to K. However, offline methods **cannot explore** the environment to find the optimal generation: language models are only given the reward signals for specific static data, which prevents them from better understanding the final objective and converging to a better solution.

The above analysis shows that different RL settings have

[1] https://openai.com/blog/chatgpt/


Setting
(a) Online
(b) Offline
(c) Semi-offline
(c1) Fully Observable
(c2) Partially Observable
Language Model
$\mathbf{x}$ $\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
$\mathbf{x}$ y_0 y_1 y_2 y_3 y_4
$\mathbf{x}$ y_0 $\hat{y}_1$ y_2 $\hat{y}_3$ $\hat{y}_4$
$M_0{=}0$ $M_1{=}1$ $M_2{=}0$ $M_3{=}1$ $M_4=1$
$\mathbf{x}$ y_0 $[M]$ y_2 $[M]$ $[M]$

Setting		(a) Online	(b) Offline	(c1) Fully Observable	(c2) Partially Observable
State		$(\mathbf{x}, \hat{y}_0, \hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4)$	$(\mathbf{x}, y_0, y_1, y_2, y_3, y_4)$	$(\mathbf{x}, M_0, y_0, M_1, \hat{y}_1, M_2, y_2, M_3, \hat{y}_3, M_4, \hat{y}_4, M_5)$	
Observation		Same as the state	Same as the state	Same as the state	$(\mathbf{x}, M_0, y_0, M_1, M_2, y_2, M_3, M_4, M_5)$
Optimization Cost (FP)	1 sample	T	1	$\mathrm{Tp_m}$	1
	K samples	KT	K	$\mathrm{KTp_m}$	1
Exploration		√	×	√	√

Figure 1. The comparison of different RL settings: (a) online methods explore the environment with a large optimization cost; (b) offline methods efficiently obtain reward signals by sacrificing the capability for exploration; (c) our proposed semi-offline setting allows exploration with minimum optimization cost. Here, $M_t = 1$ (or $M_t = 0$) means that the token at time t is sampled based on the language model (or comes from static data), $0 \le p_m \le 1$ is the probability for M_t to be 1, FP denotes the number of forward propagations through the language model, and T is the maximum number of word tokens in an output.

entirely different exploration capabilities and optimization costs. A fundamental research question is: **can we refine the RL setting so that effective exploration could be achieved with minimum optimization cost**? In this paper, we answer this question by making three contributions.

First, we define the design space of different RL settings by proposing semi-offline RL, which bridges the gap between online and offline RL and provides a theoretical foundation to compare different RL settings. As shown in Fig. 1(c), semi-offline RL composes a sample by mixing tokens generated by the language model and tokens from the static dataset with a probability $p_m \in [0, 1]$. Different values of p_m correspond to different MDPs and allows a smooth transition from the offline to online setting. In particular, at the fully observable scenario in which the model input (observation) is equal to the environment state (Fig. 1(c1)), the semi-offline setting becomes offline when $p_m = 0$, and becomes online when $p_m = 1$. When $p_m \in (0, 1)$, we could optimize the reward with an intermediate optimization cost KTp_m while keeping the capability for exploring the environment. Compared with the offline setting, semi-offline methods only utilize the static data as initial points for exploration, thereby allowing the model to identify better improvement directions. Compared with the online setting, semi-offline methods may find the optimal improvement directions with a fewer number of samples by more quickly estimating token-wise rewards (Sec. 3.3, Proposition 3).

Second, based on the semi-offline MDP formulation, we present the RL setting that is optimal in terms of the following desirable properties:

DP1. Minimum optimization cost: the policy can be optimized by using only 1 FP per input.

DP2. Minimum asymptotic bias: the estimated error when the number of instances is unlimited is minimum among all possible RL settings that satisfy DP1.

DP3. Minimum overfitting error bound: the chosen RL setting has the lowest bound of error (François-Lavet et al., 2019) when data is limited, among all settings that satisfy DP1 and DP2.

We prove that the optimal RL setting in terms of DP1−DP3 could be easily implemented by mixing static data and the mask token, as shown in Fig. 1(c2). This masked language model (MLM) setting fits naturally into existing pretrained language models, can explore K samples with only 1 FP, and effectively find improvement directions by using static data points as a starting point.

Third, we validate the effectiveness of our semi-offline RL approach in various text generation tasks and datasets, and show that it yields comparable or usually better performance compared to state-of-the-art methods.

2. Background

2.1. Preliminares about Reinforcement Learning

In text generation, we often use a human-annotated corpus as ground truth for performing supervised learning. Reinforcement learning (RL) provides an additional way of learning in which the agent can optimize its behavior by interacting with the environment. An agent-environment interaction can be described by the following process: 1) the environment tells the agent the current **state**, 2) the agent outputs an **action** given the state through a function called **policy**, and 3) after the agent acts, the environment shifts to a new state and the environment gives a **reward** based on the agent's action and the updated state. The goal of the agent is to learn a policy that yields the maximum cumulative reward. We use a pretrained language model as the policy.

In RL, the environment is typically formulated as a **Markov Decision Process (MDP)**. An MDP is defined as a tuple $\mathbf{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space, respectively. The reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$, maps a state-action pair to a scalar value, and the transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{R}$, describes the probability of transitioning from one state-action pair to another. Given an MDP, various methods of RL can be applied in the search space of the environment to learn a policy that maximizes the cumulative reward.

2.2. Reinforcement Learning for Text Generation

The MDP for text generation is usually defined as follows:

State $s_t \in \mathcal{S}$ consists of the input sequence $\mathbf{x}$ and the part of output text that has already been derived: $s_0 = \mathbf{x}$ and $s_t = (\mathbf{x}, y_0^s, \cdots, y_{t-1}^s)$. Here, $y_t^s \in \mathcal{V}$ is the token at time t, and $\mathcal{V}$ denotes the vocabulary.

Action $a_t = y_t^s \in \mathcal{A}$ is one of the $|\mathcal{V}|$ tokens.

Transition: $\mathcal{T}(s_{t+1}|s_t, a_t)$ transits s_t to s_{t+1} at time step t: $s_{t+1} = s_t \cup y_t^s$. This deterministic transition appends the next token to the previous state.

Reward: $R(s_t) = f(\mathbf{x}, y_0^s, \cdots, y_{t-1}^s)$ quantifies how good the derived sentence $y_0^s, \cdots, y_{t-1}^s$ is according to the final objective like the BLEU score or user satisfaction. In text generation, we often consider terminal reward, which means that the reward is computed after the whole text is generated, in other words $R(s_t) \neq 0$ only when $t = T$.

Both online and offline RL methods can be depicted by using this MDP.

In the **online** setting, each action is obtained by sampling from the probability distribution (Fig. 1(a)), i.e., $a_t = y_t^s = \hat{y}_t$, where $\hat{y}_t \sim p(\hat{y}_t|s_t; \theta)$, where θ is the parameter for the language model. Accordingly, the reward is computed by considering state $s_t = (\mathbf{x}, \hat{y}_0, \cdots, \hat{y}_{t-1})$. Online RL methods have a large search space, allowing them to search for the optimal solution across the entire space. However, this can make them difficult to optimize and result in high variance in reward signals. To address this, methods such as actor-critic (Konda & Tsitsiklis, 1999; Bahdanau et al., 2016), self-critic (Zhang et al., 2019; Paulus et al., 2018; Li et al., 2019), and PPO (Schulman et al., 2017; Ouyang et al.) have been developed. However, these methods still explore the environment with a large optimization cost due to the auto-regressive generation of output text.

In the **offline** setting, each action is derived by using the token in the static data (Fig. 1(b)), i.e., $a_t = y_t^s = y_t$, where y_t is the t-th token in the static data. Accordingly, the reward is computed by considering state $s_t = (\mathbf{x}, y_0, \cdots, y_{t-1})$. Offline methods are widely used in dialogue systems to reduce the number of interactions with people in real-time Serban et al. (2017); Jaques et al. (2019). Recently, GOLD (Pang & He, 2020) posits that acquisition of useful data through exploration can be challenging. Therefore, it directly employs the ground-truth. BRIO (Liu et al., 2022), on the other hand, harnesses a contrastive loss and generates multiple candidates for each instance as the static dataset. While offline methods efficiently obtain reward signals by leveraging the static dataset by sacrificing the exploration capability.

3. Semi-Offline MDP

3.1. Formulation of Semi-Offline MDP

In order to lay a theoretical foundation for comparing different RL settings, we define the design space of different RL settings by proposing semi-offline RL, which can smoothly transit from offline methods to online methods by using different values of hyperparamter p_m. More specifically, semi-offline RL composes a sample by mixing tokens generated by the language model and tokens from the static dataset with a probability $p_m \in [0, 1]$, as shown in Fig. 1(c). The formal definition is as follows.

Definition 1 (MDP of semi-offline RL)**.** In semi-offline RL:

State $s_t = (\mathbf{x}, M_0, y_0^s, \cdots, M_{t-1}, y_{t-1}^s, M_t)$ consists of the input sequence $\mathbf{x}$, the part of output text that has already been derived $y_0^s, \cdots, y_{t-1}^s$, as well as the binary values $M_0, \cdots, M_t$, each $M_t \in \{0, 1\}$ denotes whether the next token y_t^s will be determined according to the agent's generation ($M_t = 1$) or the static dataset ($M_t = 0$).

Action a_t is the output token of agent at time t. If $M_t = 1$, the agent will output one of the $|\mathcal{V}|$ tokens by sampling from the probability distribution of the language model: $a_t = \hat{y}_t$. If $M_t = 0$, The agent will give a $NULL$ token.

Transition: $\mathcal{T}(s_{t+1}|s_t, a_t)$ transits s_t to s_{t+1} at time t with

$$s_{t+1} = s_t \cup y_t^s \cup M_{t+1}, \tag{1}$$

$$y_t^s = \begin{cases} \hat{y}_t, & \text{if } M_t = 1 \\ y_t, & \text{if } M_t = 0 \end{cases} \tag{2}$$

$$M_{t+1} \sim Bernoulli(p_m) \tag{3}$$

where $\hat{y}_t$ is a token generated by the language model, y_t is a token from the dataset (e.g., the t-th token in the ground-truth), and M_{t+1} is sampled from a Bernoulli distribution parameterized with p_m, which means that M_{t+1} takes the value of 1 with a probability p_m takes the value 0 with a probability $1 - p_m$.

Reward: $R(s_t) = f(\mathbf{x}, y_0^s, \cdots, y_{t-1}^s)$ quantifies how good the generated sentence $y_0^s, \cdots, y_{t-1}^s$ is according to the ultimate goal like the BLEU score or user satisfaction.

Semi-offline RL is most comparable to online and offline settings in the **fully observable** scenario shown in Fig. 1(c1). Fully observable means that the model input (observation) is equal to the state of the environment, meaning that the language model is aware of all information in the state when making a decision, i.e., $\hat{y}_t \sim p(\hat{y}_t|\mathbf{x}, s_t; \theta)$.

In this fully observable scenario, $p_m = 0$ is equivalent to the offline setting where the model optimizes its policy from a completely static dataset. Conversely, when $p_m = 1$, the configuration is in the online mode, allowing for dynamic exploration of the maximum search space. When $p_m \in (0, 1)$ is an intermidiate value, the semi-offline MDP balances between dynamic exploration of the search space and knowledge obtained from the static dataset, while also finding an equilibrium between exploration and time cost. More specifically, the time cost for semi-offline methods can be computed with the following proposition.

Proposition 1 (Time cost when fully observable)**.** *Considering the fully observable scenario in which all information in the states is observed by the language model to get sampled tokens. Let us denote the minimum number of FPs required to sample s_t as C_t and its expectation as $\mathbb{E}(C_t)$. We have*

$$C_t = \sum_{t'=0}^{t-1} M_{t'},\ \mathbb{E}(C_t) = tp_m \tag{4}$$

The proof is given in Appendix A.1. Proposition 1 shows that when $p_m \in (0, 1)$, we could optimize the reward with an intermediate optimization cost controlled by p_m while keeping the capability for exploring the environment.

In addition to the time cost, the intermediate methods whose $p_m \in (0, 1)$ provides an additional view for exploration. Compared with the offline setting, semi-offline methods only utilize the static data as initial points for exploration instead of seeing only the reward of static data points, thereby allowing the model to get a more comprehensive understanding about the final objective and identify better improvement directions. Compared with the online setting, semi-offline methods may find the optimal improvement directions with a fewer number of samples. This sampling efficiency is achieved by only exploring a vicinity of the static data point. Thus, the space to be explored for semi-offline methods ($|\mathcal{V}|^{Tp_m}$) is exponentially smaller than that of the online methods ($|\mathcal{V}|^{T}$), making it easier for the language model to understand the reward gain brought by different choices. Even though the exploration space is limited, it is possible that the knowledge explored in the vicinity of specific output text can be generalized to other output text considering the generalization ability of neural networks. This is verified by our experiments, which show that semi-offline usually performs equally good or better with much less time cost compared with existing online or offline methods (Sec. 4).

3.2. RL Setting with Minimum Optimization Cost

Next, we move towards achieving the minimum optimization cost while maintaining the effective exploration capability of the agent. In particular, we are interested in finding a semi-offline RL setting that could be optimized with only 1 FP per instance, so that even large pretrained language models could be optimized efficiently. Meanwhile, we hope that the agent could still freely decide the degree for exploration by choosing different values of p_m.

We can see from Propsition 1 that the time cost in the fully observable scenario cannot always be 1 FP for different values of p_m. In order to further speedup the optimization method, we must remove the condition of full observation, i.e. not requiring a_t to be a decision made after observing all information in s_t. This scenario can be formulated by using Partially Observable Markov Decision Process (POMDP).

Definition 2 (Semi-offline MDP when partially observable)**.** The MDP of the environment is the same with that in Def. 1. However, the agent in POMDP takes action a_t based on observation o_t, which does not contain all information in s_t:

$$a_t = \hat{y}_t \sim p(\hat{y}_t|o_t; \theta),\ M_t = 1 \tag{5}$$

o_t is a sub-sequence of $s_t = (\mathbf{x}, M_0, y^s_0, M_1 \cdots, y^s_{t-1}, M_t)$.

Losing information in s_t may significantly decrease the probability to achieve optimal results. To derive an optimal RL setting under the minimum time cost constraint, we consider two research questions:

RQ1. Which information has to be removed from the observation in order to meet the minimum time cost constraint?
RQ2. What information needs to be retained in the observation to maximize the performance?

RQ1 can be answered with the following proposition.

Proposition 2 (Information loss with minimum time cost)**.** *If s_T can always be sampled within 1FP for $\forall p_m \in [0, 1]$, then o_t must **not** contain any exact information about sampled tokens $\hat{y}_{t'}$, $\forall t \in [1, T)$ and $\forall t' \in [0, t-1]$.*

The proof is given in Appendix A.2. This proposition can be easily understood: if we want to achieve parallel generation of different tokens, then when one token $\hat{y}_t$ is generated, the information of another token $\hat{y}_{t'}$ generated at the same time can not be utilized for generating $\hat{y}_t$.

Proposition 2 allows us to define the maximum set of observations we can get at time step t when minimum time cost can be achieved, which is important for answering RQ2.

Definition 3 (Maximum observation with minimum time cost)**.** If s_T can always be sampled within 1FP for $\forall p_m \in [0, 1]$, the observation with the maximum information in s_t

is $o_t^{max} = (\mathbf{x}, M_0, y_0^o, \cdots, M_{t=1}, y_{t=1}^o, M_t)$, where

$$y_t^o = \begin{cases} NULL & M_t = 1 \\ y_t & M_t = 0 \end{cases} \tag{6}$$

Def. 3 is useful for answering RQ2 as it characterizes the asymmetric bias of RL methods, i.e., the error of the agent with unlimited data, according to the following lemma.

Lemma 1 (Criteria for 0 asymmetric bias)**.** *According to Theorem 1 and Definition 2.4 in (François-Lavet et al., 2019), the additional error introduced by changing o_t^{max} to o_t when the data is unlimited is 0, if for $\forall t$ and $\forall s$*

$$p(s|o_t) = p(s|o_t^{max}) \tag{7}$$

Another measure for performance is the overfitting error, which depicts the additional error introduced due to limited data. The following lemma the criterion for achieving minimum overfitting error bound.

Lemma 2 (Criteria for minimum overfitting error bound)**.** *Accordingly to Theorem 3 in (François-Lavet et al., 2019), the overfitting error bound is minimized when the number of possible obervations $|O_t|$ is minimized, where each $o_t \in O_t$ is an element in set O_t.*

Lemmas 1 and 2 provide a theoretical foundation to decide whether a RL setting can achieve optimal performance. Lemma 1 shows that all information useful for predicting s should be kept in the observation, and Lemma 2 claims that the observation should contain as less information as possible to avoid overfitting. This allows us to rule out methods such as Scheduled Sampling (Bengio et al., 2015; Mihaylova & Martins, 2019), which does not contain the information of M_t and thus cannot satisfy Lemma 1. They also help exclude observations that include additional information such as y_t when $M_t = 1$, which fails to satisfy Lemma 2.

According to these lemmas, we define optimal RL setting under the minimum time cost constraint as follows:

Definition 4 (Optimal RL setting)**.** In the optimal RL setting, its observation o_t^* should satisfy

DP1. Minimum time cost: s_T can always be sampled within 1 FP.

DP2. Minimum asymmetric bias: o_t^* satisfies the criteria for 0 asymmetric bias given by Lemma 1.

DP3. Minimum overfitting error bound: o_t^* satisfies the criteria for overfitting error bound given by Lemma 2.

We then prove that the optimal RL setting in Def. 5 could be easily implemented by mixing static data and the mask token, as shown in Fig. 1(c2), where $[M]$ denotes a mask token. This masked observation setting fits naturally into existing pretrained language models and can explore K samples with only 1 FP. Formally, we define the RL setting with masked observations as follows.

Definition 5 (RL setting with masked observations)**.** The masked observation is defined as o_t^M=$x, y_0^M, y_1^M, \cdots, y_{t-1}^M$ where

$$y_t^M = \begin{cases} [M] & M_t = 1 \\ y_t & M_t = 0 \end{cases} \tag{8}$$

We then formally prove the optimality of masked observation with the following theorem.

Theorem 1 (Optimality of masked observation)**.** *o_t^M in Def. 5 is o_t^* in Def. 4.*

The proof of Theorem 1 is given in Appendix A.3.

3.3. Optimization

3.3.1. RL Loss for Solving MDP

POMDP defined in Def. 5 can be solved in the same way as traditional MDPs. Here we use policy gradient because pre-trained language models provide a natural initialization of the policy. Specifically, we adopt the REINFORCE with baseline (Williams, 1992) to reduce the variance among different trajectories. Accordingly, the policy is optimized with the following RL loss:

$$\begin{gathered} \mathcal{L}_{RL} = \frac{1}{K} \sum_{k=1}^{K} -(\mathcal{R}(Y^k) - b) \sum_t log\; p(a_t^k|o_t^M) \\ b = \frac{\sum_k \mathcal{R}(Y^k)}{K} \end{gathered} \tag{9}$$

where K is the number of samples, k denotes the sample index, $Y^k = (a_0^k, \cdots, a_{T-1}^k)$ is the k-th sampled sentence, and b is the baseline computed by averaging the rewards of sampled sentences to reduce the variance.

Analysis of optimization cost. We can easily see that the number of FPs needed for computing $\mathcal{L}_{RL}$ is always 1, irregardless of the number of samples. This is because that different samples are obtained by using the same observation o_t^M, thus can be obtained together with 1 FP.

Efficient estimation of token-wise rewards. Decomposing $\mathcal{L}_{RL}$ into token-wise rewards using the following proposition allows us to see that the RL setting with masked observations enable more efficient learning of token-wise rewards.

Proposition 3 (Token-level reward assignment)**.** *$\mathcal{L}_{RL}$ in Eq. 9 can be decomposed into token-wise reward $\mathcal{L}_t^i$:*

$$\mathcal{L}_t^i = -\frac{C_t^i}{K}\, log\, p(\mathcal{V}_i|o_t^M)(\mathbb{E}_{o=o_t^M, a=\mathcal{V}_i}\mathcal{R}(Y) - \mathbb{E}_{o=o_t^M}\mathcal{R}(Y)) \tag{10}$$

where $\mathcal{V}_i$ is the i-th word token in the vocabulary, and C_t^i denotes the number of samples that select the i-th word token $\mathcal{V}_i$ at time step t.

The proof is in Appendix A.4. Proposition 3 shows that estimating how well the i-th token performs at time t requires an accurate estimation of $\mathbb{E}_{o=o_t,a=\mathcal{V}_i}\mathcal{R}(Y) - \mathbb{E}_{o=o_t}\mathcal{R}(Y)$. This can be easily achieved in our semi-offline RL setting with masked observations, since o_t is always the same. In comparison, o_t is usually different for different samples in the online and offline RL setting. Thus, they may require more samples to accurately understand the contribution of a single token $\mathcal{V}_i$.

3.3.2. Overall Optimization Loss

We follow existing paradigm for RL training. Specifically, pretrained language models are first fine-tuned with the ground-truth labels to ensure a good starting point for RL training. This is achieved by optimizing the MLE loss

$$\mathcal{L}_{MLE} = \sum_{t\in|y^{gt}|} log\, p(y_t^{gt}|x, y_1^{gt}, , \cdots, y_{t-1}^{gt}) \qquad (11)$$

where y_{gt} is the ground-truth in the dataset.

We then perform RL training by simultaneously considering both the MLE loss and the RL loss. The MLE loss is considered here to prevent the policy from drifting away from the original dataset, which may lead to a reduction in generation quality. This is achieved by minimizing

$$\mathcal{L} = \mathcal{L}_{MLE} + \lambda\mathcal{L}_{RL} \qquad (12)$$

where $\lambda > 0$ is a hyperparameter. During this phase, we replace some tokens in $y_1^{gt}, , \cdots, y_{t-1}^{gt}$ with $[M]$ so that the model can be better adapt to masks in the inputs.

4. Experiment

4.1. Experimental Setup

4.1.1. Datasets

We conduct experiments on 1) a summarization dataset **CNN/DM** (Hermann et al., 2015), where the goal is to generate summaries for news articles; 2) a dialogue summarization dataset **SAMSum** (Gliwa et al., 2019), in which the focus is summarizing dialogues instead of news articles; 3) a natural question generation dataset **NQG** (Rajpurkar et al., 2016), where the task is to generate questions that can be answered by a specific segment of an article; 4) an extreme summarization dataset **XSum** (Narayan et al., 2018), which focus on generating highly abstractive summaries for news articles from the BBC. More information about these datasets can be found in Appendix C.

4.1.2. Compared Methods

Base models. We fine-tune (**FT**) pre-trained language models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) for each task. Specifically, for CNN/DM and SAMSum we use BART and for SQuAD and XSum we use T5 and Pegasus (Zhang et al., 2020) respectively.

Additionally, we fine-tune the tasks with masks (**M-FT**) to study the influence of involving masks during training. This method is similar to FT but with the added step of randomly masking a portion of the tokens in the targets, giving the model the ability to predict the next token when given the special token [M] on these downstream tasks.

Online methods. The online methods we compare include the online generation of single-sample methods Self-Critic (**SC**) (Paulus et al., 2018; Li et al., 2019) and Actor-Critic (**AC**) (Konda & Tsitsiklis, 1999). Both methods are optimized using REINFORCE with baseline (Sutton et al., 1999), where the baseline for SC is the greedy decoding result of the agent, and AC uses a quality scoring critic model to compute the reward. Average baseline (**AVG**) is a multiple-sample approach, in which its RL loss using the average baseline is a variant of the contrastive loss used by BRIO (Liu et al., 2022)[2].

Offline methods. The offline methods we compare are **GOLD** (Pang & He, 2020) and **BRIO** (Liu et al., 2022), where GOLD uses the original ground truth as the static dataset in offline training, and BRIO uses the generated results of the BASE model as the static dataset.

4.1.3. Metrics

Following (Liu et al., 2022; Bengio et al., 2015), We use ROUGE scores including **R-1**, **R-2**, and **R-L** (Lin & Hovy, 2003) to evaluate the results on the summarization tasks: CNN/DM, SAMSum, and XSum. For SQuAD, we follow (Ushio et al., 2022) and adopt the BLEU score **B-4** (Papineni et al., 2002), ROUGE scores (Lin & Hovy, 2003), and METEOR score **MTR** (Banerjee & Lavie, 2005). We directly use the corresponding metrics as the reward to be optimized. For more implementation details, one can refer to Appendix B.

4.2. Overall Performance

Tab. 1 shows the overall performance of different models as well as their optimization cost (FP). We have three main observations by analyzing the table.

First, our performance is significantly higher compared with other methods with the same FPs (i.e., FP=N). This is because that our method is the only one that has the exploration capability when FP is N. Other methods that have the same optimization cost are either base models or offline methods that only consider one sample.

Second, methods that only utilize one sample usually per-

[2]Proof can be found in Appendix D.

Table 1. Overall performance. FP denotes the number of forward propagations required for optimization, N is number the instances consumed by the model, K is the number of samples used, and L is the sentence length. '$*$' indicates our reproduced results.

Groups	Models	FP	CNN/DM			SAMSum			SQuAD			XSum		
			R-1	R-2	R-L	R-1	R-2	R-L	B-4	R-L	MTR	R-1	R-2	R-L
Base	FT	N	44.16	21.28	40.90	53.32	28.53	49.03	27.21	54.13	27.7	47.46	24.69	39.53
	M-FT	N	45.10	21.76	41.86	53.11	28.74	48.98	27.43	54.30	27.82	47.65	24.85	39.56
Offline	GOLD	N	45.40	22.01	42.25	53.18*	28.90*	49.11*	27.20*	54.43*	27.59*	45.85	22.58	37.65
	BRIO	NK	47.78	23.55	44.57	53.98*	29.11*	49.56*	27.17*	54.53*	27.64*	**49.07**	**25.59**	40.40
Online	SC*	NT	45.45	21.85	42.16	53.47	28.54	48.99	27.14	54.36	27.58	47.90	24.95	39.73
	AC*	NT	45.71	22.07	42.42	53.41	28.29	48.90	27.35	54.48	27.62	47.88	24.92	39.71
	AVG*	NTK	48.28	24.16	45.00	54.10	29.21	49.58	27.50	54.79	27.77	48.48	25.21	40.23
Semi	Ours	N	**48.56**	**24.26**	**45.41**	**54.14**	**29.28**	**50.33**	**27.78**	**54.93**	**28.37**	48.61	25.56	**40.51**

Table 2. The results of combining the loss of BRIO and OURS.

Models	XSum		
	R-1	R-2	R-L
BRIO	49.07	25.59	40.40
Ours	48.61	25.56	40.51
BRIO+Ours	**49.23**	**25.98**	**41.01**

Table 3. Optimization efficiency on SQuAD and SAMSum. Time is measured in minutes.

Models	#data	SQuAD			SAMSum	
		B-4	R-L	time	R-L	time
Ours	8K	27.66	54.80	14.7	49.75	8.9
	16K	27.64	54.75	29.3	49.96	18.0
BRIO	8K	27.42	54.36	18.8	49.39	9.3
	16K	27.50	54.46	37.5	49.35	19.0
Avg	8K	27.62	54.70	121.0	49.09	135.8
	16K	27.58	54.72	243.0	49.49	271.0

forms worse than multi-sample methods, even when they are time-expensive online methods such as SC and AC). This demonstrates the necessity to obtain more reward signals by exploring the environment. Our method is the only one that can increase the number of samples without increasing the number of FPs required for optimization.

Third. our method performs better or equally good than state-of-the-art methods that are much more expensive than ours (e.g., BRIO and AVG). We achieve the best result on three datasets, demonstrating the superiority of our semi-offline setting in addition to the significantly reduced optimization cost. Although our result in XSum is only similar with BRIO, we show that our method still brings additional benefits. More specifically, Tab. 2 shows that combining our method with BRIO results in improved performance.

4.3. Optimization Efficiency

To fully evaluate the performance of our method, it is important to consider not only the number of FPs, but also the real time cost during optimization. To this end, we fix the number of instances and compared the optimization speed as well as model performance in Tab. 3. The experiments were run on a machine with an Nvidia A40 GPU (memory: 48 GB) using a learning rate of 1e-6 and a batch size of 8 for all compared methods. The results show that our method not only has the lowest training time consumption, but also the best optimization speed on both the SQuAD and SAMSum datasets. It is also worth noting that BRIO and AVG use multiple target texts for the same source text for one instance, which leads to increased **memory usage** on GPUs. In contrast, our method is more memory efficient as it only uses one target for each instance.

4.4. Ablation Study

We conduct an ablation study to investigate 1) the impact of using an MDP that does not satisfy Lemma 1, and 2) the effect of using different offline datasets for training.

4.4.1. Design of MDP

As mentioned in Sec.3, failing to satisfy Lemma 1 will result in suboptimal results. We evaluate the correctness of this statement by devising the following variants:

1. **-Mask**: remove the mask information, and the p_m of the environment is consistent with the main experiment (0.4);

2. **-Mask, p_m=1**: also remove the mask information and p_m of the environment is set to 1;

3. **+Noisy Mask**: with mask information included, the model receives part of the wrong mask information, i.e. the environment tells the model that $M_t = 0$ when in fact the environment's $M_t = 1$;

As shown in Tab. 4, settings that completely remove the

Table 4. Ablation study on variants that do not satisfy Lemma 1.

Models	CNN/DM			SAMSum			SQuAD			XSum		
	R-1	R-2	R-L	R-1	R-2	R-L	B-4	R-L	MTR	R-1	R-2	R-L
Ours	48.56	24.26	45.41	54.14	29.28	50.33	27.78	54.93	28.37	48.61	25.56	40.51
Mask	47.66	23.42	44.32	53.33	28.68	49.38	27.13	54.53	28.14	48.31	25.20	40.11
Mask, p_m=1	47.76	23.34	44.46	53.43	28.53	49.31	27.38	54.68	28.02	48.24	24.96	40.07
Noisy Mask	47.87	23.40	44.98	53.77	28.70	50.13	27.69	54.98	28.20	48.20	25.08	40.25

Table 5. Performance of using different static datasets.

Models	CNN/DM			SAMSum			SQuAD			XSum		
	R-1	R-2	R-L	R-1	R-2	R-L	B-4	R-L	MTR	R-1	R-2	R-L
Ours (data+)	47.18	23.51	43.78	53.49	28.60	49.79	27.80	54.86	28.14	48.29	25.28	40.27
Ours (data-)	48.56	24.26	45.41	54.14	29.28	50.33	27.78	54.93	28.37	48.61	25.56	40.51

Table 6. Sensitivity of # sample on CNN/DM.

Models	CNN/DM				
	# sample	FP	R-1	R-2	R-L
BRIO	2	2x	46.02	22.21	42.71
	4	4x	46.1	22.31	42.85
	16	16x	47.78	23.55	44.57
Ours	2	1x	46.09	22.59	43.03
	4	1x	47.05	23.33	43.87
	16	1x	48.31	23.91	45.15
	64	1x	48.56	24.26	45.41

mask information (-Mask and -Mask, p_m=1) result in the worst performance. This is because without mask information, the model is unable to account for how the environment mixes the dataset and model predictions, leading to inaccurate estimation of the reward for current actions, resulting in large variance of the reward signal and poor optimization results. Partially removing the mask information (+Noisy Mask), as opposed to completely removing it, is better in terms of estimating the reward, as it allows for a portion of signals to be received correctly.

4.4.2. Different Options of Offline Dataset

In this part, we investigate the effect of using different static datasets for model optimization. Consider K candidate targets obtained using diverse beam search or top-p sampling, etc. We sort them by metrics such as ROUGE and collect all sentences with the lowest metric among the K candidates as **data-** and the highest metric as **data+**.

As shown in Tab. 5, using data- as the dataset gives better results than data+. From an optimization perspective, we believe that data- is easier to sample useful signals, because the probability that it learns about how to further improve the sentence is higher.

4.5. Sensitivity Analysis

Tab 6 shows how different number of samples impact the model performance. We observe that when the same number of samples is used, we usually have with better results compared with BRIO. This is probably due to the fact that we can more efficiently estimate the reward of each token, according to Proposition 3. Furthermore, in contrast to BRIO, which requires more memory and FP cost to increase the number of samples, our sampling does not introduce additional memory and FP cost. So our method allows for a larger number of samples, e.g. 64, which results in further improved performance. No clear benefit can be seen by continuing to increase the number of samples, e.g., 128.

5. Conclusion

We propose semi-offline reinforcement learning, a novel paradigm that bridges the gap between online and offline RL, and provides a theoretical foundation for comparing different RL settings. Our semi-offline RL approach achieves a balance between effective exploration and minimum optimization cost. Extensive experiments show that our semi-offline RL approach is effective in various text generation tasks and datasets, and yields comparable or usually better performance compared to the state-of-the-art methods.

References

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Banerjee, S. and Lavie, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*, 2005.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

François-Lavet, V., Rabusseau, G., Pineau, J., Ernst, D., and Fonteneau, R. On overfitting and asymptotic bias in batch reinforcement learning with partial observability. *Journal of Artificial Intelligence Research*, 65:1–30, 2019.

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://www.aclweb.org/anthology/D19-5409.

Goodrich, B., Rao, V., Liu, P. J., and Saleh, M. Assessing the factual accuracy of generated text. In *KDD*, pp. 166–175, 2019.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *NIPS*, 12, 1999.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Le, H., Wang, Y., Gotmare, A. D., Savarese, S., and Hoi, S. C. H. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *ArXiv*, abs/2207.01780, 2022.

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, 2020.

Li, S., Lei, D., Qin, P., and Wang, W. Y. Deep reinforcement learning with distributional semantic rewards for abstractive summarization. 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. 6038–6044, 2019.

Lin, C.-Y. and Hovy, E. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, pp. 150–157, 2003.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Liu, Y., Liu, P., Radev, D., and Neubig, G. Brio: Bringing order to abstractive summarization. In *ACL*, pp. 2890–2903, 2022.

Mihaylova, T. and Martins, A. F. T. Scheduled sampling for transformers. *ArXiv*, abs/1906.07651, 2019.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback.

Pang, R. Y. and He, H. Text generation by learning from demonstrations. In *ICLR*, 2020.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pp. 311–318, 2002.

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

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., Ke, N. R., et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *NIPS*, 12, 1999.

Ushio, A., Alva-Manchego, F., and Camacho-Collados, J. Generative Language Models for Paragraph-Level Question Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, U.A.E., December 2022. Association for Computational Linguistics.

Wang, X., Gu, X., Cao, J., Zhao, Z., Yan, Y., Middha, B., and Xie, X. Reinforcing pretrained models for generating attractive text advertisements. KDD '21, pp. 3697–3707, 2021.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pp. 11328–11339. PMLR, 2020.

Zhang, Z., Pu, J., Zhuang, L., Zhou, W., and Li, H. Continuous sign language recognition via reinforcement learning. In *ICIP*, pp. 285–289. IEEE, 2019.

A. Proofs

A.1. Proof of Proposition 1

Proposition 1 (FP cost,full observation)

Considering the fully observable scenario in which all information in the states is observed by the language model to get sampled tokens. Let us denote the minimum number of FPs required to sample s_t as C_t and its expectation as $\mathbb{E}(C_t)$. We have

$$C_t = \sum_{t'=0}^{t-1} M_{t'},\ \mathbb{E}(C_t) = tp_m \tag{13}$$

Proof. 1. First we prove $C_t \geq \sum_{t'=0}^{t-1} M_{t'}$ by using mathematical induction.

When $t=0$, $C_0=0$, as s_0 only contains the source X, no FPs are needed.

When $t=1$, $C_1 = M_0 \geq \sum_{t'=0}^{0} M_{t'}$

for $t>1$, Assume that $C_{t-1} \geq \sum_{t'=0}^{t-2} M_{t'}$.

If $M_{t-1}=1$, the computing of $a_{t-1} \sim p(a_{t-1}|s_{t-1};\theta)$ needs 1 FP.

$C_t = C_{t-1}+1 = C_{t-1}+M_{t-1} \geq \sum_{t'=0}^{t-1} M_{t'}$,

if $M_{t-1}=0$,

C_t can not be less than C_{t-1}, $C_t \geq C_{t-1} = C_{t-1} + M_{t-1} \geq \sum_{t'=0}^{t-1} M_{t'}$

2. Second we prove $C_t \leq \sum_{t'=0}^{t-1} M_{t'}$

we can get all M_t with 0 FP, the tokens $\{y^s_{t'} : y^s_{t'} = y_t, M_{t'}=0\}$ are from the dataset and can be also gotten with 0 FPs.

The left is $\{y^s_{t'} : y^s_{t'} = \hat{y}_{t'}, M_{t'}=1\}$, the size of it is $\sum_{t'=0}^{t-1} M_{t'}$, we can get it with at most $\sum_{t'=0}^{t-1} M_{t'} FPs$.

3. combing 1 and 2, we have $c_t = \sum_{t'=0}^{t-1} M_{t'}$.

$E(C_t) = \sum_{t'=0}^{t-1} \mathbb{E}(M_{t'}) = \sum_{t'=0}^{t-1} p_m = tp_m$ □

A.2. Proof of Proposition 2

Proposition 2 (Information loss with minimum time cost)

If s_T can always be sampled within 1FP for $\forall p_m \in [0,1]$, then o_t must **not** contain any exact information about sampled tokens $\hat{y}_{t'}$,

Proof. For $t \in [1,T)$, suppose that o_t contains the exact information of some $\hat{y}_{t'}$. Then, there's a time cost of at least 1FP in O_t.

Then at t, with probability $p_m > 0$ one needs to compute $a_t = \hat{y}_t$ given o_t.

Then getting s_{t+1} requires at least $2FP$=1FP(computing $\hat{y}_{t'}$)+1FP(computing a_t),

s_{t+1} is a subsequence of s_T, so s_T also requires at least 2FP □

A.3. Proof of Theorem 1

Theorem 1 (optimality of masked observation)

o^M_t in Def. 5 is o^*_t in Def. 4

Proof. 1. Proving we can computing: a_t=$\hat{y}_t$ within 1FP

a_t=$\hat{y}_t \sim p(\hat{y}_t|x, y^m_0, \cdots, y^m_{t-1};\theta) = p(\hat{y}_t|o^M_t;\theta)$

The computation of $p(\hat{y}_t|o^M_t;\theta)$ doesn't need the exact $\hat{y}_{t'}, t' < t$. It does not violate Proposition. 2.

2. proving bias=0:

$p(s|o^M_t) = \prod_{t'} p(y^s_{t'}|o^m_t)$

$$p(y^s_{t'}|o^m_t) = \begin{cases} p(y^s_{t'}|o^m_{t'-1}) = p(y^s_{t'}|x, y_{<t'\backslash m}) & y^m_{t'} = \text{[M]} \\ y^m_{t'} = y_t & y^m_{t'} \neq \text{[M]} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

$p(s|o^{max}_t) = \prod_{t'} p(y^s_{t'}|o^{max}_t)$

$$p(y^s_{t'}|o^{max}_t) = \begin{cases} p(y^s_{t'}|o^{max}_{t'-1}) = p(y^s_{t'}|x, y_{<t'\backslash m}) & M_t = 1 \\ y^m_{t'} = y_t & M_t = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

$p(y^s_{t'}|o^m_t) = p(y^s_{t'}|o^{max}_t)$ and $p(s|o^{max}_t) = p(s|o^M_t)$

3. proving $|O^M_t|$ is minimum for all $\phi(o^{max}_t)$ satisfying 1 and 2.

First, we have $|O^M_t| = (|\mathcal{V}|+1)^t$

Taking t=1 for an instance,

1. removing y^M_0, $\phi'(o^{max}_t) = (x, M_0, M_1)$

$$p(y^s_{t'}|\phi'(o^{max}_t)) = \begin{cases} p(y^s_{t'}|x, M_0, M_1) & y^m_{t'} = \text{[M]} \\ \frac{1}{|\mathcal{V}|} & y^m_{t'} \neq \text{[M]} \end{cases} \tag{16}$$

$p(y^s_{t'}|\phi'(o^{max}_t)) \neq p(y^s_{t'}|o^{max}_t)$

2. removing M_0, $\phi^{''}(o_t) = (x, y^M_0, M_1)$

$$p(y^s_{t'}|\phi^{''}(o_t)) = \begin{cases} p_m p(y^s_{t'}|x, \tilde{y}_{<t}) + (1-p_m) & y^m_{t'} = y^s_{t'} \\ p_m p(y^s_{t'}|x, \tilde{y}_{<t}) & y^m_{t'} \neq y^s_{t'} \end{cases} \tag{17}$$

$p(y^s_{t'}|\phi''(o^{max}_t)) \neq p(y^s_{t'}|o^{max}_t)$

□

A.4. Proof of Proposition 3

Proposition 4 (Token-level expected reward assignment) Using Eq. 9 as our loss function, we can assign the expected reward to the specific token for each action $\mathcal{V}^i$ and observation o_t:

$$\mathcal{L}_t^i = -\frac{C_t^i}{K}\, log\, p(\mathcal{V}_i|o_t^M)(\mathbb{E}_{o=o_t^M,a=\mathcal{V}_i}\mathcal{R}(Y) - \mathbb{E}_{o=o_t^M}\mathcal{R}(Y))$$

Proof. We first write down Eq. 9:

$$\begin{aligned}\mathcal{L}_{RL} &= \tfrac{1}{K}\textstyle\sum_{k=1}^{K} -(\mathcal{R}(Y^k)-b)\sum_t log\, p(a_t^k|o_t^M)\\ &= \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\sum_t -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)\\ &= \tfrac{1}{K}\textstyle\sum_t\sum_{k=1}^{K} -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)\end{aligned} \tag{18}$$

After swapping the enumeration order of t and k, we then remove the enumeration of t and consider the loss $\mathcal{L}_t$ for some specific t:

$$\mathcal{L}_t = \tfrac{1}{K}\textstyle\sum_{k=1}^{K} -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)$$

Then we add the enumeration of the actions, and $\mathcal{V}_i$ denotes the i-th action. When sampling Y, for time step t, one $\mathcal{V}_i$ may appear in multiple Y.

$$\begin{aligned}\mathcal{L}_t &= \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\sum_{i=1,\mathcal{V}_i=a_t^k}^{|\mathcal{V}|} -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)\\ &= \tfrac{1}{K}\textstyle\sum_{i=1}^{|\mathcal{V}|}\sum_{k=1,\mathcal{V}_i=a_t^k}^{K} -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)\end{aligned}$$

We swapped the enumeration again and fix both i and t to derive the $\mathcal{L}_t^i$

$$\begin{aligned}\mathcal{L}_t^i &= \tfrac{1}{K}\textstyle\sum_{k=1,\mathcal{V}_i=a_t^k}^{K} -(\mathcal{R}(Y^k)-b) log\, p(a_t^k|o_t^M)\\ &= \tfrac{1}{K}\textstyle\sum_{k=1,\mathcal{V}_i=a_t^k}^{K} -(\mathcal{R}(Y^k)-b) log\, p(\mathcal{V}_i|o_t^M)\end{aligned}$$

As one $\mathcal{V}_i$ may appear in multiple Y. We can assume in these samples, what we do is to fix $\mathcal{V}_i$ at time step t, and do sampling at other positions. We regard the sample results as different contexts for $\mathcal{V}_i$ at time step t. Then we can compute the expected reward of $\mathcal{V}_i$ using these samples.

$$\mathcal{L}_t^i = \tfrac{1}{K} - log\, p(\mathcal{V}_i|o_t^M)\textstyle\sum_{k=1,\mathcal{V}_i=a_t^k}^{K}(\mathcal{R}(Y^k)-b)$$

Let $C_t^i = \sum_{k=1}\mathbb{I}(\mathcal{V}_i = a_t^k)$

$$\mathcal{L}_t^i = -\tfrac{C_t^i}{K}\, log\, p(\mathcal{V}_i|o_t^M)\Big(\frac{\sum_{k=1,\mathcal{V}_i=a_t^k}^{K}\mathcal{R}(Y^k)}{C_t^i} - b\Big)$$

If $C_t^i \to \infty$, we have

$$\mathcal{L}_t^i = -\tfrac{C_t^i}{K}\, log\, p(\mathcal{V}_i|o_t^M)(\mathbb{E}_{o=o_t,a=\mathcal{V}_i}\mathcal{R}(Y) - b)$$

From Eq. 9, we have

$$b = \frac{\sum_k \mathcal{R}(Y^k)}{K}$$

As $K \geq C_t^i$, so $K \to \infty$

$$b = \mathbb{E}_{o=o_t}\mathcal{R}(Y)$$

It is the expected reward without any position fixed for o_t

$$\mathcal{L}_t^i = -\tfrac{C_t^i}{K}\, log\, p(\mathcal{V}_i|o_t^M)(\mathbb{E}_{o=o_t,a=\mathcal{V}_i}\mathcal{R}(Y) - \mathbb{E}_{o=o_t}\mathcal{R}(Y))$$

□

B. Implementation details

Table 7. Implementation details. For CMM/DM,SAMSum,and Xsum the ROUGE score is the average of ROUGE-1, ROUGE-2 and ROUGE-L. For Squad. the Reward is the average of BLEU-4 and ROUGE-L.

-	CNN/DM	SAMSUM	SQUAD	XSUM
BATCH SIZE	16	16	16	16
LEARNING RATE	3E-6	1E-6	3E-6	2E-6
λ(WEIGHT OF $\mathcal{L}_{RL}$)	20	3	4	1
# SAMPLE	64	64	64	64
p_m(MASK RATE)	0.4	0.4	0.4	0.4
REWARD	ROUGE	ROUGE	BLEU-4 + ROUGE	ROUGE

C. Datasets Statistics

Table 8. Statistical information on the datasets.

-	CNN/DM	SAMSUM	SQUAD	XSUM
# TRAIN	287,113	14732	75,722	204,045
# DEV	13,368	818	10,570	11,332
# TEST	11,490	819	11,877	11,334
$\lvert Source\rvert$	822.3	120.8	149.4	429
$\lvert Target\rvert$	57.9	23.4	11.5	23.3

D. Deriving AVG. from BRIO

We introduce how to get the RL loss of AVG. from the contrastive loss of BRIO (Liu et al., 2022). We first give the original loss function in BRIO.

$$\begin{aligned}L_{ctr} &= \textstyle\sum_i\sum_{j>i} max(0, f(S_j) - f(S_i) + \lambda_{ij})\\ f(S) &= \frac{\sum_{t=1}^{l} log\, p_{g_\theta}(s_t|D,S<t;\theta)}{|s|^\alpha}\end{aligned}$$

there are N samples, S_i is the i-th sample, f(S) is the normalized sum of the loglikelihood of tokens in S, —S— denote the length of S, α and λ are two hyperparameters. the sentences are sorted by some quality metric M, and $M_i > M_j$

Let's consider the loss for each pair of sentences (i,j). The function max(0, .), together with the sorted results, gives a condition of $f(s_j) - f(s_i) + \lambda_{ij} > 0$ and $M_i < M_j$. It means when the order of the sum of loglikelihood disobeys the order of quality, we should rerank $S_i and S_j$ Then we have:

$$\mathcal{L}_{ij} = \mathbb{I}(M_i > M_j \& f_i < f_j + \lambda_{ij})(\frac{logP_j}{|s_j|^\alpha} - \frac{logP_i}{|s_i|^\alpha})$$

We can relax this condition, so we can derive the formulation policy gradient.

1. To remove this condition, when the ordering is correct, if the model has a random policy, we should keep optimizing the probability of the best action to get a better return.

2. $\mathbb{I}(M_i > M_j)$ gives a discrete signal, which we can replace with a continuous $M_i - M_j$

Then,

$$I(M_i > M_j \& f_i < f_j + \lambda_{ij}) \approx M_i - M_j$$

We can calculate the loss with the new condition,

$$\begin{aligned}
&\mathcal{L} = \textstyle\sum_{i,j}(M_i - M_j)(\frac{logP_j}{|s_j|^\alpha} - \frac{logP_i}{|s_i|^\alpha}) \\
&= \textstyle\sum_{i,j}(\frac{logP_j}{|s_j|^\alpha})(M_i - M_j) + \sum_{i,j} -(\frac{logP_i}{|s_i|^\alpha})(M_i - M_j) \\
&= \textstyle\sum_{i,j} -(\frac{logP_j}{|s_j|^\alpha})(M_j - M_i) + \sum_{i,j} -(\frac{logP_i}{|s_i|^\alpha})(M_i - M_j) \\
&= 2\textstyle\sum_{ij} -\frac{logP_j}{|s_j|^\alpha}(M_j - M_i) \\
&= 2\textstyle\sum_i \sum_j -\frac{logP_j}{|s_j|^\alpha}(M_j - M_i) \\
&= 2\textstyle\sum_j -\frac{logP_j}{|s_j|^\alpha}(\sum_i M_j - \sum_i M_i) \\
&= 2\textstyle\sum_j -\frac{logP_j}{|s_j|^\alpha}(NM_j - \sum_i M_i) \\
&= 2N\textstyle\sum_j -\frac{logP_j}{|s_j|^\alpha}(M_j - \frac{\sum_i M_i}{N})
\end{aligned}$$

For the j-th sample, $\mathcal{L}_j = -\frac{2N}{|s_j|^\alpha} logP_j(M_j - \frac{\sum_i M_i}{N})$ $\frac{\sum_i M_i}{N}$ can be regarded as the baseline averaging the reward of N samples. It is in a formulation of REINFORCE with baseline and is the same as the loss we use in Eq. 9 if we ignore the coefficients.

For our compared method AVG. in Sec. 4, we use this training loss for optimization.

E. Ablation on the using of different datasets

Table 9. Data Condition. Win rate of sampled results compared to the greedy results.

MODELS	CNN/DM	SAMSUM	SQUAD	XSUM
OURS (DATA+)	27 %	15%	13%	11%
OURS (DATA-)	32 %	19%	18%	16%

To provide a numerical analysis, we calculate the proportion of sampled sentences that are better than the greedy decoding result (i.e. better than the current policy) in Tab. 9. We found that DATA- is more likely to sample better sentences for improving the current strategy. Even though we fix the data as input, the optimization is not only for these sentences. The mask information given by our environment each time is random and does not allow the model to see a complete and fixed sentence, which may represent more abstract semantics and prevent overfitting, as per Lemma 2. Additionally, even though we only perform exploration on this data, the generalization ability of the neural model also facilitates the results on the test set.

F. Discussion of Limitations

Parallel prediction of future tokens: One potential limitation of our method is that the parallel prediction of future tokens may result in a lack of fluency in the sentences upon sampling, i.e., the two tokens predicted simultaneously may lack correlation. However, we believe that using a multi-layer transformer model as the base model can address this issue. Our generative model is a stack of two already trained K-layer transformer models, resulting in a $2K$-layer model. The first K-layers of the model make their own predictions for generation, while the last K-layers take into account predictions from the previous time step, leading to more informed predictions. By only estimating the action distribution information from the previous time step, the model effectively models the unknown state through estimation in the intermediate layers of the transformer, similar to a belief function.

In terms of FP, assume the unit changes from the whole model to one layer for a K-layer transformer model. Therefore, 1 model-level FP is equivalent to K layer-level FPs. While according to Theorem 1, the model cannot access the sequential self-generated information under the 1-FP setting, at the layer level, the higher layer can access some of the sequential information from the lower layer during the K FPs.

Relevance of tokens from dataset and model prediction: Another possible disadvantage of our approach is that if data replacement is performed after a period of model generation, the current data may not be relevant to the previous model generation. In light of this mismatch, our approach can be seen as optimizing each fragment of the target. However, as each fragment is predicted by the token from the same data, the correlation between each fragment is partially preserved. We have currently experimented with the general case, where p_m randomly determines the sequence of masks, and have achieved good results in the current experimental settings.

We suggest further experimentation with mask replacement, such as masking only the end of sentences or specific parts in advertisements for better results, considering that inter-sentence associations are weaker than intra-sentence associations. For practical applications, such as advertisements generation, the adjective or numerical parts of the sentences could be masked and optimized to generate more attractive or factual descriptions. Meanwhile, the generative models considered in our experiments are not bidirectional, and the optimization method does not affect the model structure. In this sense, the use of bidirectional models could be considered, but would require changes to the model structure and inference method. In our experiments, the generative models we consider are not bidirectional, and the optimization method does not affect the model structure.