Semi-Offline Reinforcement Learning for Optimized Text Generation

Changyu Chen [1 2] **Xiting Wang** [3] **Yiqiao Jin** [4] **Victor Ye Dong** [5] **Li Dong** [3] **Jie Cao** [5] **Yi Liu** [5] **Rui Yan** [1]

Abstract

In reinforcement learning (RL), there are two major settings for interacting with the environment: online and offline. Online methods explore the environment at significant time cost, and offline methods efficiently obtain reward signals by sacrificing exploration capability. We propose semi-offline RL, a novel paradigm that smoothly transits from offline to online settings, balances exploration capability and training cost, and provides a theoretical foundation for comparing different RL settings. Based on the semi-offline 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 approach is efficient and yields comparable or often better performance compared with state-of-the-art methods. Our code is available at https://github.com/ChangyuChen347/semi-offline-RL.

1. Introduction

Pretrained language models have achieved great success in improving text generation quality (Devlin et al., 2019; Liu et al., 2019). Recent research shows that a key for further improving pretrained 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 human feedback (Ouyang et al., 2022). Recent large pretrained models that incorporate reinforcement learning, e.g., InstructGPT (Ouyang et al., 2022), ChatGPT[1], and GPT-4[2], have demonstrated superior performance in aligning with user intent compared to GPT-3 (Zhao et al., 2023).

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 good generation can usually be found when the number of samples approaches infinity, it is empirically time-intensive to obtain even only a few samples from large pretrained language models. While there are many aspects of time cost in online RL (Wang et al., 2018; Zhao et al., 2020; Wang et al., 2022; Feng et al., 2022; Yang et al., 2022), this paper focuses on the forward propagations of 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 or ground-truth labels (Pang & He, 2020; Jaques et al., 2019; Serban et al., 2017; Zhu et al., 2023) for an input $\mathbf{x}$, as well as text pre-generated with beam search (Liu et al., 2022). By avoiding generating text in an autoregressive 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]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China [2]The work was done during the author's internship at Microsoft Research Asia. [3]Microsoft Research Asia, Beijing, China [4]Georgia Institute of Technology, Atlanta, USA [5]Microsoft, Redmond, USA. Correspondence to: Xiting Wang <xitwan@microsoft.com>, Rui Yan <ruiyan@ruc.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

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

[2] https://openai.com/research/gpt-4

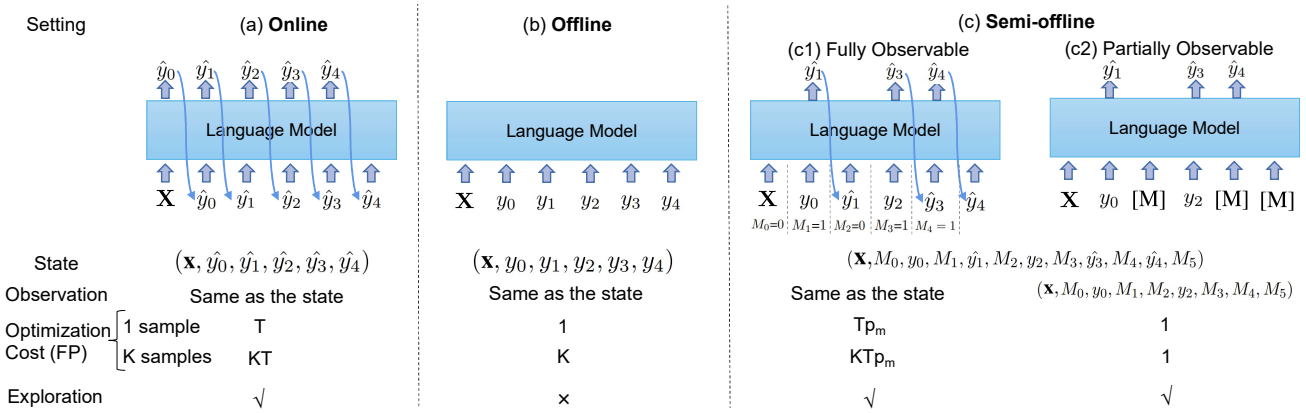

Figure 1. The comparison of different RL settings: (a) online methods explore the environment with a large optimization cost; (b) offline methods efficiently obtain reward signals by sacrificing the capability for exploration; (c) our proposed semi-offline setting enables exploration with minimum optimization 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), $0 \leq p_m \leq 1$ 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 is achieved with minimum optimization cost**? In this paper, we address 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 allowing for a smooth transition from offline to online settings. In particular, for a 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 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 minimal 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 is 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 evaluate 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 while improving efficiency.

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 (Hyun et al., 2022; Wang et al., 2022; Chen et al., 2022). 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, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, maps a state-action pair to a scalar value, and the transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$, 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: $s_t = (\mathbf{x}, y_0^s, \cdots, y_{t-1}^s)$. Here, $y_t^s \in \mathcal{V}$ is the output 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 with high variance in reward signals. To address this, methods such as actor-critic (Konda & Tsitsiklis, 1999; Bahdanau et al., 2016; Le et al., 2022), self-critic (Zhang et al., 2019b; Paulus et al., 2018; Li et al., 2019), and PPO (Schulman et al., 2017; Ouyang et al., 2022) have been developed. However, these methods still explore the environment at great 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 $s_t = (\mathbf{x}, y_0, \cdots, y_{t-1})$. RAML (Norouzi et al., 2016) can be viewed as one of the pioneering offline RL methods. It obtains the static dataset for offline learning by edit-distance sampling and weighs the samples according to the reward, resulting in a formulation of reward augmented maximum likelihood. SPG (Ding & Soricut, 2017) and ERPO (Tan et al., 2018) improve upon RAML (Norouzi et al., 2016) by exploring a larger region but introducing more decoding cost and they fall into the category of online methods. Offline methods are also 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. Unlike previous methods, ILQL (Snell et al., 2022) involves training a value network on a static dataset. During deployment, this value network is used to perturb the output distribution of another language model trained with supervised learning. While offline methods efficiently obtain reward signals by leveraging the static dataset, they sacrifice 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 and 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., $\hat{y}_t \sim p(\hat{y}_t|s_t;\theta)$.

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 intermediate 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 can 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 can 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 accelerate 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 as 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:

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

o_t is a sub-sequence of $s_t = (\mathbf{x}, M_0, y_0^s, M_1 \cdots, y_{t-1}^s, M_t)$.

Losing information in s_t may significantly decrease the probability of achieving 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 $\hat{y}_{t'}$, for $\forall t' \in [0, t-1]$ and $\forall t \in [1, T)$.*

The proof is given in Appendix A.2. This proposition can

be easily understood: when aiming for parallel generation of different tokens, the generation of one token $\hat{y}_t$ should not rely on the information of another token $\hat{y}_{t'}$ generated simultaneously.

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

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

With Def. 3, we can answer RQ2 by characterizing the asymmetric bias of RL methods, which refers to the error of an agent with unlimited data. This characterization is achieved by using 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 shows 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 observations $|O_t|$ 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 little redundant 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 minimum overfitting error bound criteria in Lemma 2.

We then prove that the optimal RL setting in Def. 4 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

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

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 pretrained 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:

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

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, regardless of the number of samples. This is because different samples are obtained by using the same observation o_t^M, and 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 enables 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 loss $\mathcal{L}_t^i$ of the i-th token in the vocabulary at time step t:*

$$\begin{aligned}&\mathcal{L}_{RL} = \textstyle\sum_t \sum_{i=1}^{|\mathcal{V}|} \mathcal{L}_t^i \\ &\mathcal{L}_t^i = -\frac{C_t^i}{K} \log\, p(\mathcal{V}_i|o_t^M)(\frac{\sum_{k=1,\mathcal{V}_i=a_t^k}^K \mathcal{R}(Y^k)}{C_t^i} - b) \\ &b = \frac{\sum_{k=1}^K \mathcal{R}(Y^k)}{K}\end{aligned} \tag{10}$$

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

The proof is in Appendix A.4. Proposition 3 shows that we evaluate how well the token $\mathcal{V}_i$ performs at time t by computing $\frac{\sum_{k=1,\mathcal{V}_i=a_t^k}^K \mathcal{R}(Y^k)}{C_t^i} - \frac{\sum_{k=1}^K \mathcal{R}(Y^k)}{K}$, which estimates $\mathbb{E}_{Y\sim p(Y|o=o_t^M,a=\mathcal{V}_i)}\mathcal{R}(Y) - \mathbb{E}_{Y\sim p(Y|o=o_t^M)}\mathcal{R}(Y)$, the expected advantage of generating $\mathcal{V}_i$ under observation o_t^M.

The accurate estimation of this expected advantage requires a large number of samples under the same observation o. This can be easily achieved in our semi-offline RL setting with masked observations as o_t^M remains constant for different sampled tokens. In comparison, observation o 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

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

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

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

$$\mathcal{L} = \mathcal{L}_{MLE} + \lambda \mathcal{L}_{RL} \tag{12}$$

where $\lambda > 0$ is a hyperparameter.

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 **SQuAD** (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 statistical information about these datasets can be found in Appendix C. We have also experimented with other tasks such as advertisement generation, and the results can be found in Appendix G.

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-large (406M). For SQuAD and XSum we use T5-large (737M) and Pegasus-large (570M) (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**) (Le et al., 2022). 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 rewards of the multiple samples as the baseline. The RL loss of AVG is also a variant of the contrastive loss used by BRIO (Liu et al., 2022)[3].

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.

[3]The proof of deriving BRIO to AVG 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.

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	27.70	47.46	24.69	39.53
	M-FT	N	45.10	21.76	41.86	53.09	28.17	49.02	27.43	54.30	27.82	47.65	24.85	39.56
OFFLINE	GOLD	N	45.51	22.10	42.30	53.18	28.90	49.11	27.20	54.43	27.59	47.75	24.92	39.70
	BRIO	NK	47.83	23.75	44.65	53.98	29.11	49.56	27.17	54.53	27.64	48.91	**25.71**	**40.60**
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	**48.54**	**24.40**	**45.35**	**54.27**	29.19	**50.57**	**27.79**	**54.95**	**28.32**	**49.02**	25.37	40.52

To ensure a fair comparison, we employ M-FT as the initialization method and keep the base model the same for ours and all RL methods. For BRIO and ours, we use the same static dataset. For more implementation details of how we collect trajectory for online, offline, and our semi-offline method, one can refer to Appendix B.

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 of the reward and other hyperparameters, one can refer to Appendix B. In addition to these metrics, we have also tested how our method performs when optimizing other metrics such as factuality (Appendix G) and whether humans consider the generated text to be better (Appendix H).

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 performs 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.

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

MODELS	XSUM		
	R-1	R-2	R-L
BRIO	48.91	25.71	40.60
OURS	49.02	25.37	40.52
BRIO+OURS	**49.23**	**25.98**	**41.01**

Third. our method performs as well as or better than state-of-the-art methods that are much more costly than ours (e.g., 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 to 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.

We also provide human evaluation and results on more metrics in Appendix H.

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. 4. The experiments are 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.

Table 3. Ablation study on variants that do not satisfy Lemma 1 or Lemma 2.

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	48.54	24.40	45.35	54.27	29.19	50.57	27.79	54.95	28.32	49.02	25.37	40.52
-MASK	47.95	24.00	44.85	54.19	28.65	50.23	27.78	54.92	28.28	48.80	25.34	40.40
-MASK, p_m=1	47.43	23.67	44.48	53.64	28.50	50.16	27.57	54.94	28.07	48.88	25.37	40.49
+NOISY MASK	48.14	24.03	45.03	54.01	29.07	50.22	27.50	54.91	28.00	48.55	25.27	40.34
+ALL	47.90	23.98	44.82	53.95	28.89	50.04	27.65	54.74	27.98	48.71	25.22	40.41
+PRE	48.34	23.86	45.30	54.04	29.00	50.17	27.68	54.71	28.10	48.44	25.13	40.17

Table 4. 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

In contrast, our method is more memory efficient as it only uses one target for each instance.

4.4. Ablation Study

This ablation study investigates 1) the impact of using an MDP that does not satisfy Lemma 1 or Lemma 2, and 2) the effect of using different offline datasets for training.

4.4.1. DESIGN OF MDP: LEMMA1

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). In this variant, we obtain the trajectory by Scheduled Sampling with a fixed choosing probability of 0.4 (Bengio et al., 2015; Mihaylova & Martins, 2019);

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. 3, All methods that violate Lemma 1 have a certain degree of performance drop. +NOISY MASK partially removes the mask information while -MASK and -MASK, p_m=1 completely remove the mask information. 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 a large variance of the reward signal and poor optimization results.

4.4.2. DESIGN OF MDP: LEMMA2

For Lemma 2, we design two new baselines that incorporate additional information by adding the complete sequence of the offline static data, denoted as Y^d, to the current observation, represented as $o_t = o_t \cap Y^d$. . We empirically validate if this will result in overfitting and a drop in results on the test set.

1. **+ALL**: we replace the input source X in the encoder as a concatenated sequence X + [END] + Y^d.

2.**+PRE**: we modify the decoder input by adding the embedding of the corresponding token in the target sequence Y_i^d to the embedding of the token [M] at the i-th position, i.e., Embedding([M]) is replaced with Embedding([M])+Embedding(Y_i^d).

If we ignore the difference in input method of Y^d, $|O_t|$ of +ALL is larger than that of +PRE: for the t-th position, +ALL can see the full sequence $Y^d = (Y_1^d, \cdots, Y_{|Y^d|}^d)$, while +PRE can only see the prefix $(Y_1^d, \cdots, Y_t^d)$. The results from Tab. 3 show that +ALL and +PRE are both worse than ours.

4.4.3. 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+**. For our experiment, we follow BRIO (Liu et al., 2022), using diverse beam search to get the dataset, and more details can be referred to in Appendix B.

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-)	48.54	24.40	45.35	54.27	29.19	50.57	27.79	54.95	28.32	49.02	25.37	40.52

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.

Table 6. Win rate of sampled text compared with greedy.

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. 6. 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.

For the optimization of text similarity towards ground truth, we use the aforementioned pre-decoded results as the static dataset following BRIO. In contrast, we can directly use ground truth as the dataset to in turn remove the pre-decoding cost in more general tasks. We test the results of optimizing factuality for summarization and click-through rate for textual advertisement. The results are provided in Appendix G.

4.5. Sensitivity Analysis

Tab. 7 shows how different numbers of samples impact model performance. We observe that when the same number of samples is used, we usually have 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 significant benefit can be seen by continuing to increase the number of samples, e.g., 128.

Table 7. 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	46.10	22.31	42.85
	16	16X	47.83	23.75	44.65
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	48.54	24.40	45.35

We give results of sensitivity for the choice of mask rate and the weight λ in Appendix F. For mask rate, we use a default mask rate 0.4 for the main experiment. As it performs well across various tasks. However, tuning the mask rate for the specific task can bring better results (Tab. 12). For example, by setting the mask rate to 0.8, we can get a better performance on SAMSum. For the weight λ, one should tune this parameter as long as the weight is not too large or too small, the method can work well (Tab. 13).

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.

6. Acknowledgement

This work was supported by National Natural Science Foundation of China (NSFC Grant No. 62122089), Beijing Outstanding Young Scientist Program NO.BJJWZYJH012019100020098, and Intelligent Social Governance Platform, Major Innovation & Planning Inter-disciplinary Platform for the "Double-First Class" Initiative, Renmin University of China.

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. In *ICLR*, 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. *NeurIPS*, 28, 2015.

Chen, C. H., Wang, X., Yi, X., Wu, F., Xie, X., and Yan, R. Personalized chit-chat generation for recommendation using external chat corpora. *KDD*, 2022.

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

Ding, N. and Soricut, R. Cold-start reinforcement learning with softmax policy gradient. In *NeurIPS*, 2017.

Feng, C., Lian, D., Wang, X., Liu, Z., Xie, X., and Chen, E. Reinforcement routing on proximity graph for efficient recommendation. *TOIS*, 41:1–27, 2022.

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

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, pp. 70, 2019.

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. *NeurIPS*, 28, 2015.

Hyun, D., Wang, X., Park, C., Xie, X., and Yu, H. Generating multiple-length summaries via reinforcement learning for unsupervised sentence summarization. In *EMNLP*, 2022.

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:1907.00456*, 2019.

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

Laban, P., Schnabel, T., Bennett, P. N., and Hearst, M. A. Summac: Re-visiting nli-based models for inconsistency detection in summarization. *ACL*, 10:163–177, 2021.

Le, H., Wang, Y., Gotmare, A. D., Savarese, S., and Hoi, S. CodeRL: Mastering code generation through pretrained models and deep reinforcement learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 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 *ACL*, 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 *EMNLP*, 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: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. Scheduled sampling for transformers. *ACL 2019*, pp. 351, 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. In *EMNLP*, pp. 1797–1807, 2018.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. Reward augmented maximum likelihood for neural structured prediction. *NeurIPS*, 29, 2016.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *NeurIPS*, 35:27730–27744, 2022.

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 *ICLR*, 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 *EMNLP*, pp. 2383–2392, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *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:1709.02349*, 2017.

Snell, C., Kostrikov, I., Su, Y., Yang, M., and Levine, S. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

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

Tan, B., Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. P. Connecting the dots between mle and rl for sequence generation. *ArXiv*, abs/1811.09740, 2018.

Ushio, A., Alva-Manchego, F., and Camacho-Collados, J. Generative Language Models for Paragraph-Level Question Generation. In *EMNLP*, Abu Dhabi, U.A.E., December 2022.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D. J., and Batra, D. Diverse beam search: Decoding diverse solutions from neural sequence models. *ArXiv*, abs/1610.02424, 2016.

Wang, X., Chen, Y., Yang, J., Wu, L., Wu, Z., and Xie, X. A reinforcement learning framework for explainable recommendation. *ICDM*, pp. 587–596, 2018.

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

Wang, X., Liu, K., Wang, D., Wu, L., Fu, Y., and Xie, X. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. *Proceedings of the ACM Web Conference 2022*, 2022.

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

Yang, R., Wang, X., Jin, Y., Li, C., Lian, J., and Xie, X. Reinforcement subgraph reasoning for fake news detection. *KDD*, 2022.

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

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675, 2019a.

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

Zhao, K., Wang, X., Zhang, Y., Zhao, L., Liu, Z., Xing, C., and Xie, X. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. *SIGIR*, 2020.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhu, B., Dang, M., and Grover, A. Scaling pareto-efficient decision making via offline multi-objective rl. In *ICLR*, 2023.

A. Proofs

A.1. Proof of Proposition 1

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

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

where $M_{t'} \in \{0, 1\}$ denotes whether the next token $y^s_{t'}$ will be determined according to the agent's generation ($M_{t'} = 1$) or the static dataset ($M_{t'} = 0$).

Proof. Our proof consists of three steps.

Step 1. We first prove $C_t \geq \sum_{t'=0}^{t-1} M_{t'}$ by using mathematical induction.

Let us consider the scenario when $t = 0$, and verify that $C_0 = 0$. As s_0 only contains the source $\mathbf{x}$, no FPs are needed, thus $C_0 = 0$ holds. We can then easily see that for $t = 1$, $C_1 = M_0 \geq \sum_{t'=0}^{0} M_{t'}$.

Next, let us consider the scenario when $t > 1$. Assume that $C_{t-1} \geq \sum_{t'=0}^{t-2} M_{t'}$. There are two cases:

- If $M_{t-1} = 1$, deriving $a_{t-1} \sim p(a_{t-1}|s_{t-1}; \theta)$ needs 1 FP. In this case, $C_t = C_{t-1} + 1 = C_{t-1} + M_{t-1} \geq \sum_{t'=0}^{t-1} M_{t'}$,
- If $M_{t-1} = 0$, C_t can not be less than C_{t-1}, $C_t \geq C_{t-1} = C_{t-1} + M_{t-1} \geq \sum_{t'=0}^{t-1} M_{t'}$.

In either case, $C_t \geq \sum_{t'=0}^{t-1} M_{t'}$ holds.

Step 2. In the second step, 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 also be derived with 0 FPs. What left is $\{y^s_{t'} : y^s_{t'} = \hat{y}_{t'}, M_{t'} = 1\}$, whose size is $\sum_{t'=0}^{t-1} M_{t'}$, so we can also get it with at most $\sum_{t'=0}^{t-1} M_{t'}$ FPs. Thus, $C_t \leq \sum_{t'=0}^{t-1} M_{t'}$.

Step 3. Combing 1 and 2, we have $C_t = \sum_{t'=0}^{t-1} M_{t'}$. Accordingly, $\mathbb{E}(C_t) = \mathbb{E}(\sum_{t'=0}^{t-1} M_{t'}) = \sum_{t'=0}^{t-1} \mathbb{E}(M_{t'}) = \sum_{t'=0}^{t-1} p_m = tp_m$. □

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 $\hat{y}_{t'}$, for $\forall t' \in [0, t-1]$ and $\forall t \in [1, T)$.

Proof. We prove that if o_t contains information about a sampled token, then we need at least 2FP to compute s_T.

- For $t \in [1, T)$, suppose that o_t contains the exact information of some $\hat{y}_{t'}$. Then, we need at least 1FP to derive o_t (in order to get $\hat{y}_{t'}$).
- At time t, with probability $p_m > 0$, one needs to compute $a_t = \hat{y}_t$ given o_t. This requires an additional FP.
- Thus, with probability $p_m > 0$, we need at least 2FP=1FP (computing $\hat{y}_{t'}$)+1FP (computing a_t) to derive s_{t+1}. Considering that s_{t+1} is a subsequence of s_T, we also need at least 2FP to derive s_T.

□

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. To prove Theorem 1, we prove o^M_t in Def. 5 satisfies DP1-DP3 in Def. 4 as follows.

DP1. Here we prove that we can compute a_t=$\hat{y}_t$ within 1FP.

To get a_t=$\hat{y}_t$, we need to compute $p(\hat{y}_t|o^M_t; \theta)$ and sample a_t from it: a_t=$\hat{y}_t \sim p(\hat{y}_t|o^M_t; \theta)$. Please note that o^M_t does not contain any information about sampled tokens (does not violate Proposition. 2) and can be derived with 0FP. Thus, $p(\hat{y}_t|o^M_t; \theta)$ can be computed by using 1FP. No matter how many actions a_t we wish to sample from $p(\hat{y}_t|o^M_t; \theta)$, it can all be done without further FPs as long as $p(\hat{y}_t|o^M_t; \theta)$ is determined. Thus, we only need 1FP to get $a_t = \hat{y}_t$.

DP2. To prove that we achieve the minimum asymmetric bias, we need to prove $p(s|o^{max}_t) = p(s|o^M_t)$ according to Lemma 1. Since the state s is a trajectory of tokens, we have the following two equations:

$$p(s|o^M_t) = \prod_{t'} p(y^s_{t'}|o^M_t) \tag{14}$$

$$p(s|o^{max}_t) = \prod_{t'} p(y^s_{t'}|o^{max}_t) \tag{15}$$

Then, we consider if the t'-th token is a generated token or a token from the static data, in order to compute $p(y^s_{t'}|o^M_t)$ and $p(y^s_{t'}|o^{max}_t)$. When $y^M_{t'}$ = [M] or $M_t = 1$, $y^s_{t'}$ is derived by sampling from $p(y^s_{t'}|x, y_{<t'\setminus m})$. When $y^M_{t'} \neq$ [M] or $M_t = 0$, $y^s_{t'}$ is the same as the input $y^M_{t'}$, i.e., the same as the static token $y_{t'}$ from dataset. Thus, we have:

$$p(y^s_{t'}|o^M_t) = \begin{cases} p(y^s_{t'}|o^M_{t'-1}) = p(y^s_{t'}|x, y_{<t'\setminus m}) & \text{if } y^M_{t'} = \text{[M]} \\ p(y^s_{t'} = y^M_{t'}) = 1 & \text{if } y^M_{t'} \neq \text{[M]} \\ 0 & \text{otherwise} \end{cases}$$

$$p(y^s_{t'}|o^{max}_t) = \begin{cases} p(y^s_{t'}|o^{max}_{t'-1}) = p(y^s_{t'}|x, y_{<t'\setminus m}) & M_t = 1 \\ p(y^s_{t'} = y^M_{t'}) = 1 & M_t = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

According to Eq. (A.3) and Eq. (16), we have $p(y^s_{t'}|o^M_t) = p(y^s_{t'}|o^{max}_t)$. Together With Eq. (14) and Eq. (15), we have $p(s|o^{max}_t) = p(s|o^M_t)$.

DP3. According to Lemma 2, we can prove that we achieve the minimum overfitting error bound by proving that the observation space of o^M_t (i.e., $|O^M_t|$) is minimum for all observations that satisfy DP1 and DP2. We prove this by illustrating that removing features in o^M_t will result in the violation of DP2.

First, the size of the observation space is $|O^M_t| = (|\mathcal{V}| + 1)^{t+1}$. Take $t = 0$ for an instance. We have $o^M_t = (x, M_0, y^M_0)$, and $|O^M_t| = |\mathcal{V}| + 1$, i.e., the observation space includes tokens from the vocabulary and a token [M]. Then we remove information from $o^M_t = (x, M_0, y^M_0)$ and prove that DP2 will be violated.

1. If y^M_0 is removed from the observation, then the new observation becomes $\phi'(o^{max}_t) = (x, M_0)$. In this case,

$$p(y^s_{t'}|\phi'(o^{max}_t)) = \begin{cases} p(y^s_{t'}|x, M_0) & y^M_{t'} = \text{[M]} \\ \frac{1}{|\mathcal{V}|} & y^M_{t'} \neq \text{[M]} \end{cases} \tag{17}$$

We have $p(y^s_{t'}|\phi'(o^{max}_t)) \neq p(y^s_{t'}|o^{max}_t)$ according to Eqs. (16) and (17), thus DP2 is violated.

2. If M_0 is removed from the observation, then the new observation becomes $\phi''(o^{max}_t) = (x, y^M_0)$. In this case,

$$p(y^s_{t'}|\phi''(o^{max}_t)) = \begin{cases} p_m p(y^s_{t'}|x, \tilde{y}_{<t}) + (1 - p_m) & \text{if } y^M_{t'} = y^s_{t'} \\ p_m p(y^s_{t'}|x, \tilde{y}_{<t}) & \text{if } y^M_{t'} \neq y^s_{t'} \end{cases}$$

We have $p(y^s_{t'}|\phi''(o^{max}_t)) \neq p(y^s_{t'}|o^{max}_t)$ according to Eqs. (16) and (A.3), thus DP2 is violated.

In summary, $|O^M_t|$ is the minimum one among all that satisfy DP1 and DP2, since removing any information in o^M_t will result in the violation of DP2.

□

A.4. Proof of Proposition 3

Proposition 3 (Token-level reward assignment) $\mathcal{L}_{RL}$ in Eq. 9 can be decomposed into token-wise loss $\mathcal{L}^i_t$ of the i-th token in the vocabulary at time step t:

$$\begin{aligned} &\mathcal{L}_{RL} = \textstyle\sum_t \sum_{i=1}^{|\mathcal{V}|} \mathcal{L}^i_t \\ &\mathcal{L}^i_t = -\frac{C^i_t}{K} \log\, p(\mathcal{V}_i|o^M_t)\left(\frac{\sum_{k=1,\mathcal{V}_i=a^k_t}^{K} \mathcal{R}(Y^k)}{C^i_t} - b\right) \\ &b = \frac{\sum_{k=1}^{K} \mathcal{R}(Y^k)}{K} \end{aligned}$$

Proof. We first write down Eq. (9):

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

where $\mathcal{L}_t$ is the loss for some specific time step t:

$$\mathcal{L}_t = \tfrac{1}{K}\textstyle\sum_{k=1}^{K} -(\mathcal{R}(Y^k) - b) \log\, p(a^k_t|o^M_t)$$

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

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

where $\mathcal{L}^i_t$ satisfies

$$\begin{aligned} \mathcal{L}^i_t &= \tfrac{1}{K}\textstyle\sum_{k=1,\mathcal{V}_i=a^k_t}^{K} -(\mathcal{R}(Y^k) - b) \log\, p(a^k_t|o^M_t) \\ &= \tfrac{1}{K}\textstyle\sum_{k=1,\mathcal{V}_i=a^k_t}^{K} -(\mathcal{R}(Y^k) - b) \log\, p(\mathcal{V}_i|o^M_t) \end{aligned}$$

As one $\mathcal{V}_i$ may appear in multiple Y. We can assume in these samples, what we do is fix $\mathcal{V}_i$ at time step t, and do sampling at other positions. We regard the sample results of other positions 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.

$$\mathcal{L}^i_t = -\log\, p(\mathcal{V}_i|o^M_t) \tfrac{1}{K}\textstyle\sum_{k=1,\mathcal{V}_i=a^k_t}^{K} (\mathcal{R}(Y^k) - b)$$

From Eq. 9, we have

$$b = \frac{\sum_{k=1}^{K} \mathcal{R}(Y^k)}{K}$$

Let $C^i_t = \sum_{k=1}^{K} \mathbb{I}(\mathcal{V}_i = a^k_t)$, then we get the formulation of Eq. (10):

$$\mathcal{L}^i_t = -\frac{C^i_t}{K} \log\, p(\mathcal{V}_i|o^M_t)\left(\frac{\sum_{k=1,\mathcal{V}_i=a^k_t}^{K} \mathcal{R}(Y^k)}{C^i_t} - b\right)$$

We compute $\frac{\sum_{k=1,\mathcal{V}_i=a_t^k}^{K}\mathcal{R}(Y^k)}{C_t^i} - \frac{\sum_{k=1}^{K}\mathcal{R}(Y^k)}{K}$ to estimate $\mathbb{E}_{Y\sim p(Y|o=o_t^M,a=\mathcal{V}_i)}\mathcal{R}(Y) - \mathbb{E}_{Y\sim p(Y|o=o_t^M)}\mathcal{R}(Y)$:

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

$$\lim_{K\to\infty}\frac{\sum_{k=1}^{K}\mathcal{R}(Y^k)}{K} = \mathbb{E}_{Y\sim p(Y|o=o_t^M)}\mathcal{R}(Y)$$

$$\lim_{K\to\infty,C_t^i\to\infty}\frac{\sum_{k=1,\mathcal{V}_i=a_t^k}^{K}\mathcal{R}(Y^k)}{C_t^i} = \mathbb{E}_{Y\sim p(Y|o=o_t^M,a=\mathcal{V}_i)}\mathcal{R}(Y)$$

The term $\mathbb{E}_{Y\sim p(Y|o=o_t^M,a=\mathcal{V}_i)}\mathcal{R}(Y) - \mathbb{E}_{Y\sim p(Y|o=o_t^M)}\mathcal{R}(Y)$ is the expected advantage of generating $\mathcal{V}_i$ under observation o_t^M. □

B. Implementation Details

Table 8. Implementation details. For CNN/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	1E-6	1E-6	3E-6	2E-6
λ(WEIGHT OF $\mathcal{L}_{RL}$)	20	5	4	2
# SAMPLE	64	64	16	64
p_m(MASK RATE)	0.4	0.4	0.4	0.4
REWARD	ROUGE	ROUGE	BLEU-4 + ROUGE	ROUGE

Here, we introduce how we get the trajectories for compared methods and our method.

Offline We evaluate two offline methods: GOLD, BRIO. For GOLD, we follow their original setting (Pang & He, 2020) and use the ground truth labels as the samples. For BRIO, we follow its settings (Liu et al., 2022) to obtain the samples. In particular, given a base model trained with only supervised learning (more details in Sec. 4.1.2, "Base models"), we obtain samples from the base model by using Diverse Beam Search (Vijayakumar et al., 2016). This produces K (=16) diverse output sentences for each input.

Semi-offline ground-truth labels or pre-generated results can both be used as our dataset. In our experiment, we use the same pre-generated results with BRIO. BRIO uses all outputs as the samples for offline learning, while we only select one as the initial exploration point for semi-offline training. Based on the experiment results in Sec. 4.4.3, we decide to use the sample with the lowest reward (i.e., DATA-) as our static data, since it leads to a higher probability of learning about improvement directions. Then we mask the static data as the input of our model. Our model predicts the action distribution of each mask position, and samples these distributions simultaneously to generate the final trajectories. For example, given an offline data point consisting of four tokens: A, B, C, and D, we randomly mask it and obtain for example A [M1] B [M2]. The model then calculates the action distribution for [M1] and [M2]. We sample tokens from the two distributions independently multiple times, resulting in multiple trajectories.

Online In the online approach, the trajectory is generated via real-time sequential decoding and sampling. We employ the same decoding parameters as those utilized during evaluation. We follow previous work to set these hyperparameters (Gliwa et al., 2019; Liu et al., 2022; Ushio et al., 2022). For example, we set a beam size of 4, minimum length of 56, maximum length of 142, and other relevant parameters for the CNN/DM using BART.

C. Datasets Statistics

Table 9. Statistical information on the datasets.

-	CNN/DM	SAMSUM	SQUAD	XSUM
# TRAIN	287, 113	14, 732	75, 722	204, 045
# DEV	13, 368	818	10, 570	11, 332
# TEST	11, 490	819	11, 877	11, 334
$\lvert Source\rvert$	781	124	148	431
$\lvert Target\rvert$	56	23	11	23

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.

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

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 $\mathcal{L}_{ij}$ for each pair of samples (i, j). The function $max(0, .)$, together with the sorted results, gives a desired ordering condition: when $f(S_j) - f(S_i) + \lambda_{ij} > 0$, $M_j > M_i$ should hold. It means when the order of the sum of loglikelihood $< f(S_j), f(S_i) >$ disobeys the order of quality $< M_i, M_j >$, we should rerank S_i and S_j. Then we have:

$$\mathcal{L}_{ij} = \mathbb{I}(M_i > M_j\ \&\ f_i < f_j + \lambda_{ij})(\frac{\log P_j}{|s_j|^\alpha} - \frac{\log P_i}{|s_i|^\alpha}) \tag{19}$$

where $\mathcal{L}_{ctr} = \sum_i \sum_{j>i} \mathcal{L}_{ij}$. f_i represents $f(S_i)$ and $\log P$ represents $\log\ p_{g_\theta}(s_t|D, S < t; \theta)$ for simplification.

We can change this desired ordering condition so that we can derive the formulation of RL loss of AVG:

1. To remove $f_i < f_j + \lambda_{ij}$ in $\mathcal{L}_{ij}$, we consider that no matter whether the ordering of f is correct, if the model has a stochastic policy, we should keep increasing 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 signal $M_i - M_j$.

Then,

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

We replace $\mathbb{I}(M_i > M_j \;\&\; f_i < f_j + \lambda_{ij})$ with the new term $M_i - M_j$ in Eq.(19):

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

For the j-th sample, $\mathcal{L}_j = -\frac{2N}{|s_j|^\alpha} \log P_j (M_j - \frac{\sum_i M_i}{N})$.

$\frac{\sum_i M_i}{N}$ 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 coefficient.

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

E. 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. We give a case of the parallel prediction in Tab. 10. The repetition "will will" and "Friday Friday" happen when many [M] tokens are connected together. Long masked sequence is difficult for a parallel decoder, since it needs to generate all masked tokens together in one forward propagation. However, we believe that using a multi-layer transformer model as the base model can alleviate this issue. The generative model can be regarded as a stack of two pre-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 the POMDP theory.

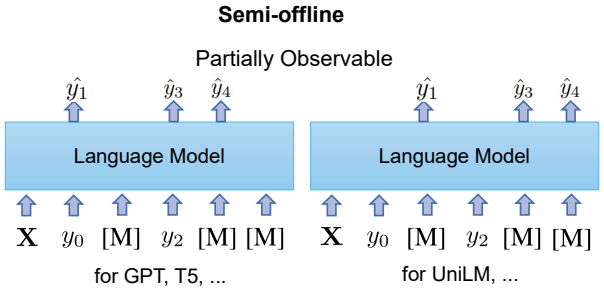

Figure 2. Our method can be applied to different architectures built on Transformer.

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.

For the majority of conditional generation tasks, our method can be applied and optimize the reward. We also acknowledge the limitation of our method in employing sampling-based techniques for generating coherent long-form text. While our methods may be highly effective in optimizing local aspects of a particular metric, we concede that the optimization of long dependencies remains a challenging task due to the potential for incoherent trajectories.

Table 10. Case 1. A very long masked sequence may result in repetition.

Original data	Martha will be in Cracow on Thursday morning and will stay there until Friday afternoon.
Masked data	Martha will be in Cracow on [M] [M] [M] [M] [M] [M] [M] [M] [M]
Prediction	Martha will be in Cracow on Thursday morning and will will be stay Friday Friday

Table 11. Case 2. Tokens generated at time may be inconsistent with static data given after , due to the unidirectional attention in the decoder.

Original data	Hee wants to move in with a man she knows for 2 months to another state. Jane strongly disagrees.
Masked data	[M] wants to move in with a [M] she [M] [M] 2 [M] [M] another state. Jane [M] disagrees [M]
Prediction	Hee wants to move in with a guy she knows for 2 months. another state. Jane is disagrees.

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. We show a case for this irrelevance in Tab. 11. the end of the predicted sentence ". another state. Jane is disagrees." is not fluent. Such a broken sentence is

generated because the model fail to foresee "another" and "disagrees" when generating the token before them. Please note that this fluency issue only happens during training, and the model will learn to avoid such broken generation based on the low reward. During testing, we use sequential decoding to avoid this issue. 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 experimented with the general case, where p_m randomly determines the sequence of masks, and have achieved good results with the current experimental settings.

We suggest further experimentation with mask replacement, such as masking only the end of sentences or specific parts 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 can be considered, but would require changes to the model structure and inference method. Or one can use another auxiliary bidirectional model as the behaviour policy to get multiple samples, but it increases the FPs to NK for computing the logits of these multiple samples like BRIO. In our method, the generative models we consider are not bidirectional, and the method doesn't affect the model structure or require more FPs.

Limited exploring space Semi-offline methods indeed result in a smaller exploration space, which we mentioned at the end of Sec. 3.1 with a quantitative comparison: "the space to be explored for semi-offline methods ($|\mathcal{V}|^{Tp_m}$) is exponentially smaller than that of online methods ($|\mathcal{V}|^T$)". Our method may lead to potential suboptimal solutions due to the incapability to thoroughly investigate the entire space.

We wish to highlight here that this limitation may be largely alleviated by the generalization ability of deep models, and that the semi-offline setting with a small exploration space may yield more benefits than issues. This is discussed in Sec. 3.1: "(the smaller exploration space makes) 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 well or better with much less time cost compared with existing online or offline methods (Sec. 4)."

The good performance of BRIO also demonstrates that the generalization ability of neural networks may be leveraged to avoid exploring every point.

Negative societal impact While our text generation method has exhibited promising results in text generation, it is important to consider its potential negative impact on society. The generated text could be utilized for spreading misinformation, reinforcing negative biases, or serving other malicious purposes. Given the risk of misinformation, it is crucial to establish safeguards such as fact-checking mechanisms. In addition, we recommend incorporating fairness principles into the reward function to mitigate potential biases in the generated content. Furthermore, addressing nefarious use cases requires the implementation of monitoring systems to prevent misuse and protect public discourse.

Hard to optimize length Furthermore, we have identified that our approach may not be suitable for length optimization. The generation is performed by providing a pre-defined generation length based on the offline data. It is not clear whether our method could effectively make a target sequence longer or shorter, which we are interested in investigating in the future.

F. Sensitivity Analysis

Table 12. Sensitivity of mask rate.

MASK RATE	R-1	R-2	R-L
OURS (p_m=0.1)	53.75	28.92	49.54
OURS (p_m=0.2)	53.96	28.82	49.89
OURS (p_m=0.3)	53.93	29.10	50.10
OURS (p_m=0.4)	54.27	29.19	50.57
OURS (p_m=0.5)	54.21	29.10	50.45
OURS (p_m=0.6)	54.46	29.11	50.43
OURS (p_m=0.7)	54.64	**29.69**	50.89
OURS (p_m=0.8)	**54.70**	29.25	**51.00**
OURS (p_m=0.9)	54.19	29.40	50.61
OURS (p_m=1) = NAT	53.55	28.87	49.54

Tab. 12 shows tuning the mask rate can bring better results on SAMSum than the default mask rate (0.4). Note that when $p_m = 1$, it becomes an online RL method with Non-autoregressive Transformer (NAT) that never uses static data during training, where p_m is the probability that we use the generated token (to perform exploration) instead of leveraging the static data point (to find a good initial point). The results also show the importance of using semi-offline training ($0 < p_m < 1$) instead of the online one ($p_m = 1$). We can see that the semi-offline setting outperforms the online one (NAT) with a wide variety of p_m. Moreover, the performance first increases with increasing p_m and then decreases when p_m becomes too large, which further demonstrates the necessity to balance exploration and the effective leverage of offline static dataset with a semi-offline setting. For the weight λ, we show how our method and the most com-

Table 13. Sensitivity of the interpolation weight on SAMSum and SQuAD. Here we report the R-L scores. Results on other criteria are similar.

WEIGHT	SAMSUM		SQUAD	
	OURS	AVG	OURS	AVG
0 (BASE)	48.98	48.98	54.30	54.30
0.1	48.76	49.04	54.50	54.26
1	49.72	49.18	54.75	54.68
2	50.35	49.58	54.87	**54.79**
3	50.33	**49.65**	54.84	54.73
4	50.45	49.25	**54.95**	54.65
5	**50.57**	49.03	54.92	54.71
10	50.31	49.06	54.54	54.36

petitive baseline AVG perform with varying interpolation weight (Tab. 13). As shown in the table, our method is better than AVG for most of the weight values. Moreover, our method yields comparable results as reported in the paper for a wide range of weight values (1-5). For both AVG and our method, the performance first increases with increasing interpolation weight, and then drops after the weight becomes too large. This trend verifies the necessity to balance the MLE and RL loss and shows a clear pattern that helps better understand the relationship between performance and weight.

G. Experiments on Other Rewards and Tasks

To show our method can be applied to optimize other rewards in addition to text similarity towards ground truth (e.g., ROUGE and BLEU), we experiment with two other rewards.

Table 14. Optimizing Factuality on CNN/DM. $*$ indicates the metrics directly optimized during training.

	R-1	R-2	R-l	Fact*
BASE	45.10	21.76	41.86	15.57
BRIO	44.23	21.18	40.92	17.54
AVG	44.23	21.24	40.93	17.68
OURS	44.60	21.66	41.91	18.30

Table 15. Optimizing CTR for advertisement generation. $*$ indicates the metrics directly optimized during training.

	R-1	R-2	R-L	Distinct	CTR*
Base	34.22	11.44	**28.21**	0.578	0.1998
OURS	**34.51**	**11.45**	**28.21**	**0.590**	**0.2086**

1. **Optimizing factuality**. We evaluate how our method performs when optimizing factuality rather than ROUGE or BLEU. This is achieved by using a model that measures factuality (Laban et al., 2021) as the reward function. In this setting, we directly use ground truth as the static dataset. Tab. 14 shows that our method achieves the best factuality score. For the ROUGE scores that are not directly optimized, we still achieve the best or second-best performance.

2. **Optimizing click-through rate (CTR)**. We add an advertisement generation task (Wang et al., 2021), in which the goal is to generate a textual advertisement based on the textual description on the product website. The reward is the click-through rate given by a click prediction model. We also use ground-truth as the static dataset. Tab. 15 shows that our method can achieve good results in a diverse set of metrics, including ROUGE, Distinct, and CTR. Here, Distinct measures the ratio of distinct uni-grams in an output.

H. Additional Criteria including Human Evaluation Scores.

For SAMSum, we add other criteria of quality that have not been directly optimized, including human evaluation score and BertScore (Zhang et al., 2019a) that measures the similarity to ground-truth in the latent space. In human evaluation, we give the guidelines by following (Gliwa et al., 2019). In particular, we ask workers to score the overall quality from -1, 0, and 1 (e.g., 1 stands for "it is understandable and gives a brief overview of the text," while -1 means that "a summarization is poor, extracts irrelevant information or does not make sense at all"). We sample 50 instances in the test set, ask 3 workers to score the outputs of different models for each instance, and report the average score here. As shown in Tab. 16, in addition to the ROUGE scores which we directly optimize, we also perform the best in human evaluation and BertScore when compared with the most competitive baseline (AVG) and the base model.

Table 16. More metrics and human evaluation on SAMSum. $*$ indicates the metrics directly Optimized during training.

	Human	BertScore	R-1*	R-2*	R-L*
Base	0.52	0.5637	53.09	28.17	49.02
AVG	0.57	0.5745	54.10	**29.21**	49.58
OURS	**0.63**	**0.5762**	**54.27**	29.19	**50.57**

Table 17. More metrics and human evaluation on SQuAD. $*$ indicates the metrics directly Optimized during training.

	Grammar	Understanding	Correctness	B-4*	R-L*	MTR
Base	**2.91**	2.95	**2.71**	27.43	54.30	27.82
AVG	2.89	2.94	2.69	27.50	54.79	27.77
OURS	**2.91**	**2.97**	**2.71**	**27.79**	**54.95**	**28.32**

For SQuAD, we follow (Ushio et al., 2022) to include a humane evaluation based on grammaticality, understandability and correctness with a 3-point scale. In Tab. 17, the human evaluation in SQuAD show a slight improvement. The reason may be that the base model can do this task well, as its score is already close to the perfect score of 3.