Reinforcing Pretrained Models for Generating Attractive Text Advertisements

Xiting Wang[1], Xinwei Gu[1], Jie Cao[2], Zihua Zhao[1], Yulan Yan[2], Bhuvan Middha[2], Xing Xie[1]
[1]Microsoft Research Asia, [2]Microsoft Advertising
{xitwan, v-xinwgu, jimcao, v-zhaozi, yulanyan, bhumid, xing.xie}@microsoft.com

ABSTRACT

We study how pretrained language models can be enhanced by using deep reinforcement learning to generate attractive text advertisements that reach the high quality standard of real-world advertiser mediums. To improve ad attractiveness without hampering user experience, we propose a model-based reinforcement learning framework for text ad generation, which constructs a model for the environment dynamics and avoids large sample complexity. Based on the framework, we develop *Masked-Sequence Policy Gradient*, a reinforcement learning algorithm that integrates efficiently with pretrained models and explores the action space effectively. Our method has been deployed to production in Microsoft Bing. Automatic offline experiments, human evaluation, and online experiments demonstrate the superior performance of our method.

CCS CONCEPTS

• **Computing methodologies → Reinforcement learning**; **Natural language generation**; *Markov decision processes.*

KEYWORDS

Advertisement Generation; Pretrained Language Models; Reinforcement Learning; Natural Language Generation

ACM Reference Format:
Xiting Wang[1], Xinwei Gu[1], Jie Cao[2], Zihua Zhao[1], Yulan Yan[2], Bhuvan Middha[2], Xing Xie[1], [1]Microsoft Research Asia, [2]Microsoft Advertising, {xitwan, v-xinwgu, jimcao, v-zhaozi, yulanyan, bhumid, xing.xie}@microsoft.com, . 2021. Reinforcing Pretrained Models for Generating Attractive Text Advertisements. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467105

1 INTRODUCTION

Text advertisements contribute to a large portion of revenue for search engines, social media, and recommender systems. An attractive ad that is relevant to user tasks and interests can significantly increase the probability that a user clicks the ad [16]. Thus, advertisers usually make every effort to optimize the text displayed in the ads. While most ads created by advertisers are of high quality, designing ads manually for every product is demanding and economically inefficient, especially for large businesses with thousands of products. It becomes even more challenging when advertisers wish to create different ads for different user tasks or interests, or frequently refresh the ad content to avoid user fatigue [2]. Moreover, understanding what types of ads are attractive and lead to more user clicks can also be difficult for advertisers with less experience or working for small businesses.

To better automate the advertising process, a typical solution is to use a template with spots reserved for certain keywords (e.g., search queries) [5]. However, templates are rigid and still require human efforts [15]. Recently, Hughes et al. have proposed a Recurrent-Neural-Network-based model that takes product landing pages as inputs to generate text ads [15]. As a pioneer in generating ads in a data-driven way, this paper shows how challenging the task is: even though a sophisticated deep learning model has been developed, a considerable portion (>15%) of the generated ads are still reported to be nonsense, broken, or bad [15], which hinders the method from being applied in real-world advertising mediums. It is also unclear whether the generated ads are attractive to real users due to the absence of online experiments.

In this paper, we use text ads as a guiding example to show how attractive texts that reach the high standard of quality required by important real-world applications can be generated and demonstrate their positive impacts on real users with online experiments. To ensure both quality and attractiveness, we effectively leverage multiple types of data, ranging from unsupervised textual data (e.g., Wikipedia), supervised data (manually-created ads), and user feedback (user clicks). We show that the incorporation of multiple types of data can be conveniently achieved by using a three-phase training schema, which tightly integrates reinforcement learning (RL) [34] with pretrained language models [11, 17]. A key question is how we design an RL method that 1) works efficiently and effectively with large and complex pretrained language models and simultaneously 2) leverages user feedback effectively to improve attractiveness without hindering user experience.

Developing an aforementioned RL method is challenging for two reasons. First, RL methods for text generation typically require the model to decode sequences word by word during training [15, 23, 29]. This results in a slow and ineffective training process because 1) decoding is extremely computationally expensive for a large and complex pretrained model and 2) the decoding process results in a propagation of uncertainty, which makes the RL method spend too much time exploring suboptimal actions (*C1*). Second, when improving the attractiveness of the ads by using reinforcement learning, user experience can be easily hindered (*C2*). This is because RL usually requires a lot of interactions with the environment (or user feedback) before a good policy is learned [9]. During

KDD '21, August 14–18, 2021, Virtual Event, Singapore.
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467105

the learning process, users often have to investigate and provide feedback on potentially bad generations, but they may quickly abandon the system when they see bad results. Moreover, even if the RL method has learned to increase user clicks, it may encourage generating attractive ads with unfaithful claims like “free shipping".

To address the challenges, we propose a model-based reinforcement learning framework for text ad generation. In our framework, a model for the environment dynamics is constructed, which helps avoid large sample complexity and improve attractiveness without hindering user experience (*C2*). While model-based RL methods are commonly used in robotics applications to reduce sample complexity [9], we show how they can be tailored for modeling online user behaviors like user clicks and demonstrate how we avoid generating low-quality or unfaithful ads when improving attractiveness. Based on the framework, we develop *Masked-Sequence Policy Gradient* (*MPG*), an RL method that explores the action space effectively and is efficient even when integrated with a large and complex pretrained model (*C1*). This is achieved by replacing the computationally-expensive decoding process with masked sequence generation and using importance sampling to handle the discrepancy between the two generation processes.

Our model has been deployed in Microsoft Bing, a major search engine with more than 100 million monthly active users. In addition to automatic offline experiments, we also evaluate how the method impacts real users by conducting human evaluation and online experiments. Human evaluation shows that both the quality and attractiveness of the ads are improved compared with the baselines. Online experiments demonstrate that our method significantly increases click yield and revenue.

The contributions of the paper are as follows:

- We propose a model-based reinforcement learning framework for generating attractive and high-quality text advertisements. This framework does not hamper user experience during RL training and helps avoid generating attractive ads that contain unfaithful claims. Although we focus on ad generation in this paper, the framework can be generalized to many text generation tasks, e.g., generating attractive news headlines.
- We propose *Masked-Sequence Policy Gradient* (*MPG*), which integrates seamlessly with pretrained models. *MPG* explores the action space effectively and is efficient even when the policy network is a large and complex pretrained model.
- Our model has been deployed to production in Microsoft Bing, a search engine that has the second largest market share. Automatic offline experiments, human evaluation, and online experiments demonstrate the superior performance of our method.

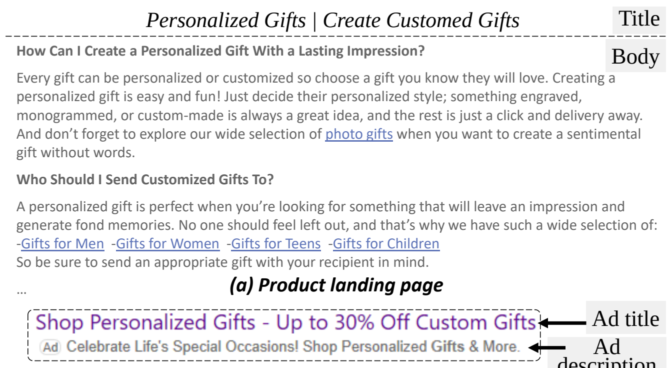
2 METHOD

2.1 Problem Statement

We formulate the problem of text ad generation as follows.

Input. The model input contains two major parts.

The first part is a product landing page, which is a webpage that describes the product to be advertised. As shown in Fig. 1(a), the landing page consists of a landing page title $\mathbf{x}^T = (x_1^T, ..., x_n^T)$ and a landing page body $\mathbf{x}^B = (x_1^B, ..., x_{n'}^B)$. Here, $x_i^T, x_{i'}^B \in \mathcal{V}$ correspond to a word token in the title and body, and $\mathcal{V}$ denotes the vocabulary.



Personalized Gifts | Create Customed Gifts

How Can I Create a Personalized Gift With a Lasting Impression?

Every gift can be personalized or customized so choose a gift you know they will love. Creating a personalized gift is easy and fun! Just decide their personalized style; something engraved, monogrammed, or custom-made is always a great idea, and the rest is just a click and delivery away. And don’t forget to explore our wide selection of photo gifts when you want to create a sentimental gift without words.

Who Should I Send Customized Gifts To?

A personalized gift is perfect when you’re looking for something that will leave an impression and generate fond memories. No one should feel left out, and that’s why we have such a wide selection of: -Gifts for Men -Gifts for Women -Gifts for Teens -Gifts for Children
So be sure to send an appropriate gift with your recipient in mind.

...

(a) Product landing page

Shop Personalized Gifts - Up to 30% Off Custom Gifts

Ad Celebrate Life's Special Occasions! Shop Personalized Gifts & More.

(b) Text advertisement

Figure 1: Example (a) landing page [1] and (b) text ad.

The second part $\mathbf{x}^U = (x_1^U, ..., x_{n''}^U)$ represents user tasks or interests. This part is optional. For ads displayed in search engines, $\mathbf{x}^U$ can be defined as the current search query of the user. Considering $\mathbf{x}^U$ allows us to generate ads that are tailored to the user. For scenarios in which user tasks or interests are absent, we can ignore $\mathbf{x}^U$ and generate ads based only on product landing pages.

Output. Text ad generation predicts an ad title $\mathbf{y}^T = (y_1^T, ..., y_m^T)$ and an ad description $\mathbf{y}^D = (y_1^D, ..., y_{m'}^D)$, as shown in Fig. 1(b). $y_i^T, y_{i'}^D \in \mathcal{V}$ are tokens in the ad title and ad description.

2.2 Three-Phase Sequence Generation

We frame the text ad generation problem as a three-phase sequence-to-sequence generation task. Specifically, the input sequence is

$$\mathbf{x} = (x_1^U, ..., x_{n''}^U, v^S, x_1^T, ..., x_n^T, v^S, x_1^B, ..., x_{n'}^B) \tag{1}$$

where $v^S \in \mathcal{V}$ is a separator token *[SEP]* that distinguishes different parts of the input. Similarly, the output sequence is

$$\mathbf{y} = (y_1^T, ..., y_m^T, v^S, y_1^D, ..., y_{m'}^D) \tag{2}$$

The goal is to learn a function f such that $\mathbf{y} = f(\mathbf{x})$. f can be represented by using a neural sequence generation model. We represent f by using a Transformer-based pretrained language model, *UNILM* [11], which 1) models a deeper level of interactions between different parts of $\mathbf{x}$ and $\mathbf{y}$ and 2) has been shown effective in text generation tasks such as question answering and response generation. Other pretrained models for sequence-to-sequence generation, e.g., MASS [31], can also be used.

We train f so that it effectively leverages different types of data in different training phases. The basic idea is to ensure that f first uses unsupervised data to obtain fundamental capabilities (e.g., generate a fluent sentence) and then gradually learns abilities that are relevant to the downstream task (e.g., generate ads that lead to more clicks). This can be achieved conveniently by extending the two-phase training schema of pretrained models to a three-phase one:

Phase I. Pretraining. *UNILM* is pretrained by using unsupervised data such as Wikipedia [11]. This equips the model with fundamental natural language processing and generation abilities, e.g., learning about language models and distinguishing stop words [13].

Phase II. Fine-Tuning. *UNILM* is then fine-tuned with supervised data [11], which consists of human-written ads provided by advertisers. Fine-tuning maximizes the likelihood of generating human-written ads, or minimizes the masked cross entropy loss:

$$L_{XE} = -\sum_{\mathbf{x},\mathbf{y}\in X,Y}\sum_{t\in M}\log p_\theta(y_t|\mathbf{x}, \mathbf{y}_{<t\backslash M}) \quad (3)$$

Here, X, Y denote a preexisting ad corpus, M is a set of masked positions [11] for a training sample, $\mathbf{y}_{<t\backslash M} = \{y_i | i < t \,\&\, i \notin M\}$, and θ refers to the model parameters of *UNILM*.

Phase III. Reinforcement Learning. The first two phases cannot incorporate user feedback. For many real-world applications, user feedback like user clicks is the ultimate optimization goal. Integrating feedback during model training closes the open loop and provides a principled way to directly optimize user feedback. Thus, we employ reinforcement learning to further refine θ after fine-tuning, which maximizes the expected reward, or minimizes

$$L_{RL} = -\sum_{\mathbf{x}\in X}\mathbb{E}_{\hat{\mathbf{y}}\sim p_\theta(\cdot|\mathbf{x})} r(\hat{\mathbf{y}}|\mathbf{x}) \quad (4)$$

$r(\hat{\mathbf{y}}|\mathbf{x})$ denotes the reward of a predicted ad $\hat{\mathbf{y}}$ given $\mathbf{x}$. In this paper, we consider the reward as the click rate, which measures the percentage of impressed (displayed) ads that are clicked by users. The framework can generalize to other types of feedback, e.g., revenue.

Next, we introduce how L_{RL} can be efficiently and effectively optimized by using our **model-based RL framework** and **Masked-Sequence Policy Gradient** algorithm.

2.3 Model-Based RL Framework

The goal of reinforcement learning is to find an optimal *policy* (i.e., f with parameters θ) that leads to the maximum expected reward, or minimum L_{RL}. Directly minimizing L_{RL} is unrealistic in many real-world applications, because it needs user feedback for many predicted ads $\hat{\mathbf{y}}$. Since RL often requires lots of interactions with the environment to learn a good policy [9], users need to check many potentially bad $\hat{\mathbf{y}}$ during the learning process and may abandon the system before a good policy is obtained.

To solve this issue, we propose a model-based RL framework for text ad generation. Compared with model-free methods, model-based RL constructs a model of the environment dynamics, which serves as a simulation environment. The policy then interacts with the simulation environment to reduce sample complexity [9] and avoid harming experience of real users. While most model-based RL methods are designed based on physics or Gaussian processes [9], we construct a model for complex online user click behaviors. In particular, we formulate the environment dynamics of online clicks by using a Markov Decision Process (MDP) model and illustrate how to design the reward function in the model to avoid generating low-quality or unfaithful ads when improving attractiveness. Next, we introduce the MDP model and the reward function.

2.3.1 *MDP Model.*

A MDP is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S}\times\mathcal{A} \rightarrow \mathcal{S}$ denotes the transition function, and $\mathcal{R} : \mathcal{S}\times\mathcal{A} \rightarrow \mathbb{R}$ is the reward function.

- **State**: The initial state $\mathbf{s}_0$ is the input $\mathbf{x}$, i.e., $\mathbf{s}_0 = \mathbf{x}$. State $\mathbf{s}_t \in \mathcal{S}$ represents both the input sequence and the part of the ad that has been generated at time t: $\mathbf{s}_t = (x_1, ..., x_{|\mathbf{x}|}, v^S, \hat{y}_1, ..., \hat{y}_t)$.
- **Action**: Given state $\mathbf{s}_t$, the policy (*UNILM*) outputs an action $a_t \in \mathcal{A} = \mathcal{V}$ that represents the $(t+1)$-th token in the ad: $a_t = \hat{y}_{t+1}$.
- **Transition**: We consider a deterministic MDP, in which the next state $\mathbf{s}_{t+1}$ can be deterministically defined by using the current state and action: $\mathbf{s}_{t+1} = \mathcal{T}(\mathbf{s}_t, a_t) = (x_1, ..., x_{|\mathbf{x}|}, v^S, \hat{y}_1, ..., \hat{y}_{t+1})$.
- **Reward**: The reward function $\mathcal{R}(\mathbf{s}_t, a_t)$ approximates the online CTR we obtain after taking action a_t given state $\mathbf{s}_t$. Instead of using real user feedback, which may hinder user experience, we learn the reward function by using *off-policy* data [9]. Next, we will discuss in detail how we design the reward function.

2.3.2 *Reward Function.*

To avoid harming the experience of real users, we learn the reward function based on user clicks of human-written ads (i.e., $\mathbf{y}$), instead of asking users to investigate the ads generated by using the RL policy (i.e., $\hat{\mathbf{y}}$). Using such off-policy data may result in two issues. First, the reward function may be inaccurate for the generated ads. Suppose that a neural model $g(\mathbf{x}, \mathbf{y})$ that maps $\mathbf{x}, \mathbf{y}$ to $r(\mathbf{y}|\mathbf{x})$ has been learned. g may not be accurate for $(\mathbf{x}, \hat{\mathbf{y}})$, since it is trained on the domain of $(\mathbf{x}, \mathbf{y})$. The issue becomes more severe considering the fact that the RL policy can frequently interact with g to explore the result of different generated ads. The policy may easily find that some strange generations (e.g., ads that consist only of fragments of attractive phrases) can lead to an unreasonably large value of g and takes advantage of these generations to maximize the reward. As a result, we may end up with a strange policy that produces counterintuitive $\hat{\mathbf{y}}$ with large but misleading $g(\mathbf{x}, \hat{\mathbf{y}})$. Second, even if the generated ads are similar with human-written ads, RL may encourage generating attractive but unfaithful ads with phrases like "free shipping" because these ads can lead to a larger g.

To solve these issues, we design the reward function so that it evaluates generated ads from three aspects:

$$\mathcal{R}(\mathbf{s}_t, a_t) = \mathcal{R}_{\text{click}}(\mathbf{s}_t, a_t) + \lambda_h\mathcal{R}_{\text{human}}(\mathbf{s}_t, a_t) + \lambda_c\mathcal{R}_{\text{cons}}(\mathbf{s}_t, a_t) \quad (5)$$

where $\lambda_h, \lambda_c > 0$ are hyperparameters that balance the three terms on the right, and $\mathcal{R}_{\text{click}}$, $\mathcal{R}_{\text{human}}$, and $\mathcal{R}_{\text{cons}}$ are defined as follows.

$\mathcal{R}_{\text{click}}$ measures the probability of an ad being clicked:

$$\mathcal{R}_{\text{click}}(\mathbf{s}_t, a_t) = \begin{cases} 0 & \text{if } a_t \neq \textit{[EOS]} \\ g(\mathbf{x}, \hat{\mathbf{y}}, pos_1) & \text{if } a_t = \textit{[EOS]} \end{cases} \quad (6)$$

where *[EOS]* is a token that denotes the end of the sentence, and $g(\mathbf{x}, \hat{\mathbf{y}}, pos_1)$ is the probability that $\hat{\mathbf{y}}$ will be clicked when it is displayed in the first position (pos_1) in the page. Here, variable pos is added in the click prediction model g to remove the position bias [14, 20]. Given a training sample $(\mathbf{x}, \mathbf{y}, pos, r)$ where r is 1 if $\mathbf{y}$ is clicked or is 0 otherwise, we input $\mathbf{x}, \mathbf{y}, pos$ to BERT, and

$$g(\mathbf{x}, \mathbf{y}, pos) = \sigma(\mathbf{w}_1^T h(\textit{[CLS]})) \quad (7)$$

Here, $h(\textit{[CLS]})$ denotes the embedding of the first input token *[CLS]* in the last hidden layer of BERT, σ is the sigmoid function, and $\mathbf{w}_1 \in \mathbb{R}^d$ represents parameters to be learned. BERT is fine-tuned by minimizing the cross entropy loss: $-\sum[r\log g(\mathbf{x}, \mathbf{y}, pos) + (1-r)\log(1 - g(\mathbf{x}, \mathbf{y}, pos))]$. The negative class in the user click data is downsampled to obtain a 1:2 ratio of positive and negative samples. After that, we get 207,451,658 training samples and 5,183,013 testing samples. The evaluation results show that the click prediction model is accurate, with a test AUC (Area Under the ROC curve) of 0.832.

$\underline{\mathcal{R}_{\text{human}}}$ estimates how similar a generated ad is to a human-written ad. Large $\mathcal{R}_{\text{human}}$ ensures that $\mathcal{R}_{\text{click}}$ learned on the domain of human-written ads gives meaningful and accurate scores for the generated ads. For a given $\mathbf{x}$, we compute $\mathcal{R}_{\text{human}}$ by comparing each token of $\hat{\mathbf{y}}$ with that in $\mathbf{y}$:

$$\mathcal{R}_{\text{human}}(\mathbf{s}_t, a_t) = \mathbb{I}(a_t = y_{t+1}) \tag{8}$$

Note that $a_t = \hat{y}_{t+1}$. $\mathbb{I}$ is an indicator function, i.e., $\mathbb{I}(\text{true})$ is 1 and $\mathbb{I}(\text{false})$ is 0. We formulate $\mathcal{R}_{\text{human}}$ in this way because it can be efficiently and accurately optimized and is inherently related with the cross entropy loss L_{XE} (Appendix II).

$\underline{\mathcal{R}_{\text{cons}}}$ constrains the word or phrase usage to avoid generating attractive but unfaithful statements. Each constraint c is represented by a 3-tuple $(\mathbf{v}_c, l_c, \eta_c)$, in which $\mathbf{v}_c = (v_{c1}, ..., v_{ck})$ denotes a constrained phrase (e.g., "free shipping" or "official site"), $l_c > 0$ refers to the punishment weight for the constrained phrase, and η_c is a function that describes the condition under which generating $\mathbf{v}_c$ should be punished. $\eta_c(\mathbf{v}_c, \mathbf{x}, \mathbf{y})$ is 1 if $\mathbf{v}_c$ should be avoided given $\mathbf{x}$ and $\mathbf{y}$ and is 0 otherwise. For example, by setting $\eta_c(\mathbf{v}_c, \mathbf{x}, \mathbf{y})$ to $\mathbb{I}(\mathbf{v}_c \not\subseteq \mathbf{x} \;\&\; \mathbf{v}_c \not\subseteq \mathbf{y})$, we punish the generation of phrase $\mathbf{v}_c$ when it appears neither in the product landing page nor the human-written ads. Given a set of constraints, $\mathcal{R}_{\text{cons}}$ can then be defined as:

$$\mathcal{R}_{\text{cons}}(\mathbf{s}_t, a_t) = -\sum_c l_c \eta_c(\mathbf{v}_c, \mathbf{x}, \mathbf{y})[\delta(\mathbf{s}_{t+1}, \mathbf{v}_c) - \delta(\mathbf{s}_t, \mathbf{v}_c)] \tag{9}$$

where $\delta(\mathbf{s}_t, \mathbf{v}_c)$ is the number of times that $\mathbf{v}_c$ appears in $\mathbf{s}_t$.

We construct the set of constraints in a semi-automatic way. Specifically, we first find the most frequent words and bi-grams in the ad corpus, and then compute their impact on $\mathcal{R}_{\text{click}}$ by using LIME [30]. The words or bi-grams that have the largest average contribution to $\mathcal{R}_{\text{click}}$ according to LIME are then investigated manually to determine whether they will be included in the constraints. The constraints we use are provided in Appendix I.

2.4 Masked-Sequence Policy Gradient

Based on our proposed RL framework, the loss in Eq. (4) becomes:

$$L_{RL} = -\sum_{\mathbf{x} \in X} \mathbb{E}_{\hat{\mathbf{y}} \sim p_\theta(\cdot|\mathbf{x})} \mathcal{R}(\mathbf{x}, \hat{\mathbf{y}}), \quad \mathcal{R}(\mathbf{x}, \hat{\mathbf{y}}) = \sum_t \mathcal{R}(\mathbf{s}_t, \hat{y}_{t+1}) \tag{10}$$

Next, we discuss why minimizing L_{RL} by using existing RL algorithms may be problematic and introduce our proposed method, *Masked-Sequence Policy Gradient.*

2.4.1 Issues of Classical Policy Gradient. Policy gradient is widely used in natural language generation for optimizing the RL loss of a policy network [15, 23, 29]. Most existing methods compute $\nabla_\theta L_{RL}$ based on (a variant of) the following equation:

$$\nabla_\theta L_{RL} \approx -\sum_{\mathbf{x} \in X} (\mathcal{R}(\mathbf{x}, \hat{\mathbf{y}}) - \mathcal{R}(\mathbf{x}, \bar{\mathbf{y}})) \nabla_\theta \sum_t \log p_\theta(\hat{y}_{t+1}|\mathbf{x}, \hat{y}_1, ..., \hat{y}_t) \tag{11}$$

Each token $\hat{y}_t$ in $\hat{\mathbf{y}}$ is decoded by sampling from $p_\theta(\cdot|\mathbf{x}, \hat{y}_1, ..., \hat{y}_{t-1})$. $\bar{y}_t$ is a baseline added to reduce variance and is usually decoded by using $\texttt{argmax}$: $\bar{y}_t = \texttt{argmax}_{y'}\, p_\theta(y'|\mathbf{x}, \bar{y}_1, ..., \bar{y}_{t-1})$.

These classical methods result in two issues when integrated with pretrained models for ad generation.

The first issue pertains to **large computational cost**. Calculating the policy gradient according to Eq. (11) requires decoding sequences $\hat{\mathbf{y}}$ and $\bar{\mathbf{y}}$ token by token. This is computationally expensive because decoding each token requires performing a forward propagation through *UNILM*, which is a large, complex pretrained model. Overall, classical policy gradient performs $O(\frac{|X|T}{B})$ forward propagations at each training epoch, where $|X|$ is the number of training samples, T denotes the output sequence length, and B refers to the batch size. In comparison, the fine-tuning process requires only $O(\frac{|X|}{B})$ forward propagations per epoch. As a result, the RL phase is much slower than the already time-expensive fine-tuning phase. Empirically, fine-tuning takes about 11 hours/epoch for 6 million ads on NVIDIA V100 GPU, while the RL phase that uses classical policy gradient takes around 110 hours/epoch for the same data.

The second issue is **ineffective exploration**. Classical policy gradient adds randomness and explores the action space by sampling $\hat{y}_t$ from the distribution $p_\theta(\cdot|\mathbf{x}, \hat{y}_1, ..., \hat{y}_{t-1})$. Each time a token is sampled, a certain amount of uncertainty is introduced, which will propagate to later decoding steps. This propagation of uncertainty makes it difficult to control the quality of $\hat{\mathbf{y}}$. Suppose that a bad token $\hat{y}_t$ is sampled, it is likely that many tokens decoded after time t becomes inappropriate, since they all depend on $\hat{y}_t$. As a result, the RL method may spend too much time exploring suboptimal actions. This ineffective exploration not only results in slow improvement of the policy, it may also harm the model, since good tokens share the same update weight $\mathcal{R}(\mathbf{x}, \hat{\mathbf{y}}) - \mathcal{R}(\mathbf{x}, \bar{\mathbf{y}})$ with the bad tokens and are punished in the same way.

2.4.2 Masked Sequence Generation with Importance Sampling. To solve the above issues, we replace the decoding process in classical RL methods with masked sequence generation, and use importance sampling to handle the discrepancy between masked sequence generation and decoding.

Masked sequence generation is widely-used in the pretraining and fine-tuning phases of pretrained models to generate a sequence of tokens. Following *UNILM*, we randomly mask P_m (70% as in [11]) tokens in $\mathbf{y}$. Let M denote the set of masked positions and $\mathbf{y}_{<t\backslash M}$ denote $\{y_i | i < t \;\&\; i \notin M\}$. If $t \in M$, $\hat{y}_t$ is sampled from $p_\theta(\cdot|\mathbf{x}, \mathbf{y}_{<t\backslash M})$. If $t \notin M$, $\hat{y}_t$ is set to y_t. Similarly, $\bar{y}_t$ is computed by

$$\bar{y}_t = \begin{cases} \texttt{argmax}_{y'}\, p_\theta(y'|\mathbf{x}, \mathbf{y}_{<t\backslash M}) & \text{if } t \in M \\ y_t & \text{if } t \notin M \end{cases} \tag{12}$$

In this formulation, $\hat{y}_t$ and $\bar{y}_t$ no longer depends on previously decoded tokens. It enables us to compute sequence $\hat{\mathbf{y}}$ or $\bar{\mathbf{y}}$ by using only one (instead of T) forward propagation through *UNILM*. In total, masked sequence generation needs $O(\frac{|X|}{B})$ forward propagations each epoch, which is much more efficient compared with the $O(\frac{|X|T}{B})$ forward propagations needed for decoding. Moreover, since a previously sampled token will not affect the result of a future token, the uncertainty introduced by sampling will no longer propagate to future tokens. As a result, many suboptimal actions are avoided and the exploration strategy becomes much more effective.

Since $\hat{y}_t$ and $\bar{y}_t$ do not depend on previously decoded tokens, masked sequence generation can be implemented in a *parallel* way so that all tokens in a sequence are computed simultaneously. Mathematically, we can also define the masked sequence generation policy θ' by considering a *sequential* process. Specifically, given a partially generated sequence $\mathbf{s}_t = (x_1, ..., x_{|\mathbf{x}|}, v^S, \hat{y}_1, ..., \hat{y}_t)$, we determine action $a_t = \hat{y}_{t+1}$ as follows. With a probability of P_m, a_t

is masked and set to y^{t+1}. With a probability of $1 - P_m$, a_t is not masked and is sampled according to *UNILM*, i.e., $p(a_t = \hat{y}_{t+1}|\mathbf{s}_t) = p_\theta(\cdot|\mathbf{x}, \mathbf{y}_{\leq t\backslash M})$. Accordingly, the policy θ' can be specified with

$$p_{\theta'}(a_t|\mathbf{s}_t) = (1 - P_m)\mathbb{I}(a_t = y_{t+1}) + P_m p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\backslash M}) \tag{13}$$

Importance sampling. Masked sequence generation can only be employed during training, since it uses human-written ads. During testing, we still generate ads through decoding. Such a discrepancy between training and testing causes exposure bias [27], which results in accumulation of errors during testing. As a consequence, it is difficult to obtain an optimal test reward. To solve this issue, we use important sampling [22], a method for estimating an expectation under one distribution (e.g., distribution of decoded sequences) given samples from another distribution (e.g., distribution of sequences generated with masks).

In particular, we consider two policies: θ and θ'. θ is the decoding policy used during testing, and θ' denotes the masked-sequence generation policy used during training. The RL loss, which optimizes the expected reward obtained during *testing*, can be written as:

$$\begin{aligned} L_{RL} &= -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{(\mathbf{s}_t,a_t)\sim p_\theta(\mathbf{s}_t,a_t)} \mathcal{R}(\mathbf{s}_t, a_t) \\ &= -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t\sim p_\theta(\mathbf{s}_t)} \mathbb{E}_{a_t\sim p_\theta(a_t|\mathbf{s}_t)} \mathcal{R}(\mathbf{s}_t, a_t) \\ &= -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)} \frac{p_\theta(\mathbf{s}_t)}{p_{\theta'}(\mathbf{s}_t)} \mathbb{E}_{a_t\sim p_{\theta'}(a_t|\mathbf{s}_t)} \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)} \mathcal{R}(\mathbf{s}_t, a_t) \\ &\approx -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}(\mathbf{s}_t)} \mathbb{E}_{a_t\sim p_{\theta'}(a_t|\mathbf{s}_t)} \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)} \mathcal{R}(\mathbf{s}_t, a_t) \end{aligned} \tag{14}$$

The third line is derived by using importance sampling, and the fourth line is a first-order approximation of the loss. This approximation trades off the variance of the off-policy correction while still correcting for the large bias of policy gradient [3].

Eq. (14) illustrates that even if different policies are used in training and testing, we can still ensure optimization of the test reward by considering the likelihood ratio $\beta_t = \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)}$ in training. In β_t, $p_\theta(a_t|\mathbf{s}_t) = p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t})$ and $p_\theta(a_t|\mathbf{s}_t)$ is defined as in Eq. (13).

2.4.3 Policy Gradient of the Modified RL Loss. Based on Eq. (14), we can then derive the policy gradient by

$$\nabla_\theta L_{RL} \approx -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} \mathbb{E}_{a_t\sim p_{\theta'}}\ \beta_t \mathcal{R}(\mathbf{s}_t, a_t) \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t) \tag{15}$$

A detailed derivation of Eq. (15) is given in Appendix II. The baseline can then be added to reduce variance:

$$\begin{aligned} \nabla_\theta L_{RL} &\approx -\sum_{\mathbf{x}\in X}\sum_{t\in M} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} \mathbb{E}_{a_t\sim p_{\theta'}} \beta_t\ \triangle\mathcal{R}_t \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t), \\ \triangle\mathcal{R}_t &= \mathcal{R}(\mathbf{s}_t, a_t) - \mathcal{R}(\mathbf{s}_t, \bar{a}_t), \quad \text{and} \quad \bar{a}_t = \bar{y}_{t+1} \end{aligned} \tag{16}$$

In Eq. (16), we only need to compute gradients for the masked tokens, because when $t \notin M$, we have $a_t = \bar{a}_t = y_{t+1}$, thus $\triangle\mathcal{R}_t = 0$.
Interpretation. In Eq. (16), the step size of $\nabla_\theta \log p_\theta(a_t|\mathbf{s}_t)$ is determined by $\beta_t\triangle\mathcal{R}_t$. Accordingly, the (log) probability of taking action a_t will be increased during optimization if $\triangle\mathcal{R}_t > 0$, i.e., if the reward of a_t is larger than the reward of the baseline action $\bar{a}_t$. β_t corrects the discrepancy between training and testing. By combining Eq. (13) with $\beta_t = \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)}$, we see that β_t is larger when $\mathbb{I}(a_t = y_{t+1}) = 0$. This means that the probability of taking action a_t will be increased faster if a_t is a novel (not a human-written) token that leads to a large reward gain $\triangle\mathcal{R}_t$.

3 EVALUATION

We evaluate our method by conducting automatic offline experiments, human evaluation, and online experiments.

3.1 Experimental Settings

3.1.1 Data. We train and test our method by using the ad data from Microsoft Bing. We consider two datasets. In the first dataset, each training sample consists of a product landing page ($\mathbf{x}^T$ and $\mathbf{x}^B$) and a text ad provided by the advertiser ($\mathbf{y}^T$ and $\mathbf{y}^D$). For this dataset, we ignore $\mathbf{x}^U$ (i.e., $\mathbf{x}^U = \emptyset$) to consider scenarios where user interests and tasks are unavailable. In the second dataset, each training sample additionally contains a search query that triggers the ad, which serves as $\mathbf{x}^U$ (i.e., $\mathbf{x}^U$ = Query). This dataset enables us to generate ads based on both the product landing pages and user queries. For both datasets, we filter out non-English language sequences and sequences that contain common HTML code elements. To prevent the model from copying only templates used for creating a large number of ads, we keep at most 3,000 samples for each advertiser domain. After data pre-processing, each dataset contains 6,066,249 samples. We then divide the samples into training, validation, and test sets of sizes 6,038,249, 14,000, and 14,000. In average, the source sequences ($\mathbf{x}$) of the two datasets contain 130.9 and 135.5 tokens (BERT tokenizer) respectively, and the target sequences ($\mathbf{y}$) contain 28.6 tokens.

3.1.2 Compared Methods. We compare our method with its variations and three baselines.
Our method and variations. We refer to our method as *UNILM*-based Masked Policy Gradient (***UMPG***) and evaluate two variants of our method. The first variant is ***UMPG-M***, which replaces the masked-sequence generation with the time-expensive decoding process. The second is ***UMPG-C***, which does not consider constrained phrases during reinforcement learning (i.e., $\lambda_c = 0$ and $\mathcal{R}_{\text{cons}}$ is ignored). The impact of $\mathcal{R}_{\text{human}}$ is evaluated by reporting the results at different values of λ_h for *UMPG*.
Baselines. We compare our method with three baselines. The first baseline is ***Transformer*** [33], which is widely-used for sequence-to-sequence generation. The second baseline, ***SCST*** [15] (*Self Critical Sequence Training*), is the state-of-art method for generating text ads. *SCST* leverages an LSTM-based attentive policy network and employs REINFORCE with baseline [29] to optimize the click rate of the generated text ads. The third baseline, ***UNILM* [11]**, is a pretrained model that effectively utilizes both unsupervised and supervised data for generating high-quality text.

3.1.3 Implementation Details. For both *UNILM* and our method, we leverage the base version of the pretrained model, in which the number of layers is 12 and the hidden state size is 768. We reuse most hyperparameters of *Transformer*, *SCST*, and *UNILM*. Important hyperparameters like learning rates and hidden state sizes are tuned on the validation set by using grid search and evaluated on the test set. We search the learning rate and the hidden state size of *Transformer* and *SCST* in the range of $[1e^{-4}, 3e^{-4}, 1e^{-3}, ..., 1e^{-1}]$

	$\mathbf{x}^U = \emptyset$				$\mathbf{x}^U$ = Query			
	Click Rate	Violation	LM	ROUGE-L	Click Rate	Violation	LM	ROUGE-L
Transformer	0.219	8.94%	-3.327	39.9	0.279	15.79%	-3.420	42.8
SCST	0.232	8.58%	-4.013	34.1	0.317	13.77%	-3.931	38.3
UNILM	0.290	7.42%	-3.319	44.6	0.376	14.04%	-3.302	48.5
UMPG-M	0.291	2.96%	-3.451	39.6	0.383	6.79%	-3.301	41.5
UMPG-C	0.357	84.00%	-3.113	41.3	0.558	89.23%	-3.269	41.8
UMPG (λ_h=0.2)	0.308	3.47%	-3.317	43.4	0.402	5.54%	-3.292	47.1
UMPG (λ_h=0.1)	0.312	0.49%	-3.305	43.4	0.419	4.16%	-3.278	46.2
UMPG (λ_h=0.05)	0.314	0.99%	-3.293	40.8	0.450	2.54%	-3.261	42.7
Imprv. of *UMPG* (λ_h=0.1)	+7.48%	+93.44%	+0.42%	-2.70%	+11.62%	+69.79%	+0.74%	-4.81%

Table 1: Evaluation results in terms of predicted click rate, constraint violation ratio, language model score, and ROUGE-L (similarity) between generated and human-written ads. We highlight results that are better than those of all baselines and show the improvement of our method (λ_h=0.1) over the best baseline. The predicted click rate of human-written ads is 0.371.

and [64, 128, 256, 512]. The learning rate of the pretrained models is searched in [$1e^{-7}$, $3e^{-7}$, $1e^{-6}$, $3e^{-6}$, $1e^{-5}$]. The Adam optimizer is leveraged during training. For our method, P_m and λ_c are set to 0.7 and 0.1. If not specifically mentioned, λ_h is set to 0.1. The batch size is set to 80. We use mixed precision and a 0.1 linear warmup schedule. All experiments are trained on NVIDIA V100 GPU.

3.2 Automatic Offline Evaluation

3.2.1 Criteria. We evaluate the generated ads in terms of four offline criteria. The first is the **click rate** $\mathcal{R}_{\text{click}}$ given by the click prediction model g. A large $\mathcal{R}_{\text{click}}$ indicates that the method can effectively generate ads that are considered attractive by the environment. The second criterion is the **violation** ratio, which measures the percentage of generated ads that violates any of the specified constraints. A large violation ratio means that many generated ads may contain unfaithful claims. The third criterion is the language model (**LM**) score, which evaluates whether the generated ads are natural and fluent text ads. To compute the LM score, we learn a masked language model of the human-written ads by using BERT [10] (data described in Sec. 3.1.1). Given an ad $\hat{\mathbf{y}}$ and a word token, the learned BERT model accurately predicts whether the token is likely to appear given the context $\hat{\mathbf{y}}$ (test AUC is 0.989). The LM score of an ad is computed based on the probability of all the generated words, i.e., $LM(\hat{\mathbf{y}}) = \frac{1}{T}\sum_{t=1}^{T}\log p_{\vartheta}(\hat{y}_t|\hat{\mathbf{y}}_{\leq T\setminus\{t\}})$, where ϑ denotes the model parameters of BERT. A large LM score means that the generated ads fit the language model of human-written ads. The fourth criterion is the **ROUGE-L** score [19], which is widely used in text summarization to measure the similarity between the generated text and the ground-truth. A large ROUGE-L score means that the generated ads are similar with the human-written ads.

3.2.2 Results. The results are shown in Table 1. We evaluate both the scenario when user tasks or interests $\mathbf{x}^U$ are absent (i.e., $\mathbf{x}^U = \emptyset$) and when $\mathbf{x}^U$ can be specified by using the search queries (i.e., $\mathbf{x}^U$ = Query). We have the following observations.

Overall performance. Our method (*UMPG*) consistently performs better than the three baselines (*Transformer*, *SCST*, *UNILM*) in terms of click rate, violation ratio, and LM scores. Take *UMPG* (λ_h=0.1) as an example. Compared with the best baseline, it improves the click rate by over 7%, violates much less factual constraints (>65%), and increases the language model score by at least 0.4%. When we specify user interests or tasks by using search queries ($\mathbf{x}^U$ = Query), the predicted click rate of the generated ads (>0.402) is even higher than that of the human-written ads (0.371). Improvement in terms of click rate and violation ratio demonstrates that our RL method can effectively generate ads that are considered attractive and faithful by the environment, while large LM scores illustrate that the generated ads fit the language model of human-written ads.

UMPG outperforms *Transformer* and *SCST* in terms of ROUGE-L, but has a lower ROUGE-L score compared with *UNILM*. This shows that our method can effectively leverage the model parameters of *UNILM* to incorporate supervised data, but tend to generate ads that diverge more from human-written ads compared with state-of-the-art supervised learning models. This is reasonable because our ultimate goal is not to maximize the similarity with human-written ads (supervised learning task), but to optimize ad attractiveness (click rate) and quality (RL task). We can see from Table 1 that diverging from the human-written ads (a lower ROUGE-L score) may sometimes lead to improved ad attractiveness and quality (e.g., larger click rate and smaller violation ratio). We further demonstrate in the human evaluation that the quality and attractiveness of our generated ads are indeed better than that of *UNILM*.

Impact of masked sequence generation. *UMPG* outperforms *UMPG-M*, which indicates that compared with traditional decoding process, the masked sequence generation is more effective in terms of exploring the action space. In addition, our method (training time is 24 hours/epoch) is much more efficient compared with *UMPG-M* (training time is 110 hours/epoch).

Impact of $\mathcal{R}_{\text{cons}}$. The results of *UMPG-C* illustrate that if we do not employ $\mathcal{R}_{\text{cons}}$ to constrain the word or phrase usage during reinforcement learning, the predicted click rate will largely increase at the expense of frequently violating factual constraints (violation ratio is larger than 80%). By checking the word distribution of the ads generated by *UMPG-C*, we find that almost every ad contains words like "official" and "free". On the one hand, this shows that our RL method can effectively leverage feedback from the environment to optimize the reward (click rate). On the other hand, this demonstrates that constraining the word or phrase usage is essential to avoid generating low-quality clickbaits.

Impact of $\mathcal{R}_{\text{human}}$. We study the impact of $\mathcal{R}_{\text{human}}$ by evaluating the results of *UMPG* at different values of λ_h. Table 1 shows that

	$\mathbf{x}^U = \emptyset$					$\mathbf{x}^U$ = Query				
	Language	Human	Accurate	Relevant	Overall	Language	Human	Accurate	Relevant	Overall
SCST	81.6%	87.4%	82.2%	92.0%	69.2%	86.5%	89.7%	88.8%	95.9%	76.9%
UNILM	97.8%	98.1%	93.4%	99.4%	90.9%	97.7%	**98.3%**	92.6%	99.4%	89.7%
UMPG	**97.9%**	**98.7%**	**94.8%**	**99.6%**	**92.9%***	**98.1%**	98.0%	**94.6%**	**99.8%**	**92.0%***

Table 2: Human evaluation of ad quality. We show the percentage of generated ads that are considered good by judges in terms of five criteria. For each method, 1,000 generated ads are evaluated. *UMPG* outperforms the baselines significantly in terms of the overall good ratio. Best results are highlighted in bold. Statistically significant improvement is marked by * (p-value<0.05).

smaller λ_h results in lower ROUGE-L, but improves the other three criteria. If ROUGE-L is too low, the generated ads may be irrelevant with the landing pages, even if all other three criteria are good. By setting λ_h to 0.1, we can generate attractive and high-quality ads that are relevant with the product landing pages. In the following experiments with real users, we will report the result of *UMPG* (λ_h=0.1) and demonstrate its superior performance.

3.3 Human Evaluation

In this experiment, we ask human judges to label ads in terms of quality and attractiveness. On average the judges have 12 months of experience in labeling text ads. They have been carefully trained to ensure that they correctly understand the tasks. Detailed guidance is provided before labeling and their initial labeling results have been checked to eliminate misunderstanding. We randomly order the ads so that the judges do not know which method is used to generate each ad. User interfaces and more details about the labeling guidance are provided in Appendix I.

3.3.1 Criteria. High quality and attractiveness are two of the most important goals for ad generation, hence the human-labeling-based evaluation are designed to cover these two aspects accordingly, in similar spirit as that of Hughes et al. [15].

To evaluate **quality**, the following four pivots are evaluated by judges: 1) **language**: language fluency and grammar correctness; 2) **human**-likeness: whether the ad looks like a human-written one; 3) **relevance**: whether the ad is relevant to the product landing page; 4) information **accuracy**: whether the information conveyed in the ad is accurate with regard to what on the advertiser's landing page / website. For example, if an ad claims "free shipping" but that is not promised by the advertiser, it shall be labeled as bad. An ad is considered as **overall** good if and only if all of the four pivots are labeled as good by the judges.

To evaluate **attractiveness**, we show two ads side-by-side, and let the judge choose which one looks more attractive. When $\mathbf{x}^U$ is set to the search query, the judges are asked to determine the attractiveness based also on the query. To eliminate biases caused by the layout position, we randomly determine the layout order (left or right) of the ads generated by using different methods.

3.3.2 Results. Tables 2 and 3 compare our method with the two most competitive baselines in terms of ad quality and attractiveness. For each method, 1,000 ads are evaluated. We compute whether an improvement is statistically significant by using paired t-test.

Quality. Table 2 shows that the ads generated by using our method have better or comparable quality according to multiple criteria compared with those generated by the baselines. The improvement in terms of overall good ratio is statistically significant, even though the ROUGE-L score of *UNILM* is larger than ours (Table 1). This further demonstrates that achieving a larger ROUGE-L score does not necessarily lead to better ad quality and that our model-based RL framework is effective for generating high-quality ads. The ads generated by using *SCST* have more quality issues compared with *UNILM* and *UMPG*, which indicates that effectively leveraging unsupervised data and pretrained models is important.

$\mathbf{x}^U$ =	$\emptyset$	Query
SCST better	31.1%	26.0%
UMPG better	**68.6%***	**72.7%***
Identical	0.3%	1.3%

(a) *UMPG* vs. *SCST*

$\mathbf{x}^U$ =	$\emptyset$	Query
UNILM better	44.1%	36.9%
UMPG better	**50.9%***	**60.2%***
Identical	5.0%	2.8%

(b) *UMPG* vs. *UNILM*

Table 3: Human evaluation of attractiveness. Judges choose between two ads generated by different methods. In each experiment, 1,000 labels are collected. Statistically significant difference is marked by * (p-value < 0.05).

Attractiveness. Table 3 demonstrates that the ads generated by using our method are significantly more attractive compared with those generated by the baselines. This is consistent with the automatic offline evaluation, which shows that our method achieves much larger predicted click rate. Example ads generated by using different methods are shown in Table 4. We can see that compared with *UNILM*, our method generates ads with more attractive phrases and phrases that are relevant with the user interests or tasks.

3.4 Online Evaluation

In the online evaluation stage, we deployed our model in Dynamic Search Ads (DSA) of Microsoft Bing, which is a major search engine that has the second largest market share and has more than 100 million monthly active users. DSA allows for minimum advertiser effort when creating ads. In particular, the advertisers share their landing pages. The DSA infrastructure then generates ads for landing pages by using different models. In DSA, the ads are generated offline when user queries are unknown (i.e., $\mathbf{x}^U = \emptyset$).

3.4.1 Criteria. Three major online criteria used by the search engine are Revenue, Click, and MClick. **Revenue** represents the earnings that accrue for every 1000 impressed Search Engine Results Pages (SERPs). **Click** yield is defined as the total clicks divided by the total number of impressed SERPs. **MClick** (mainline click yield) is the click yield of the mainline position of the SERP. Different from the bottom ads, which are shown after the regular search results, ads in the mainline position are shown before the regular

	Landing Page 1 https://www.keeganlegal.com/ $\mathbf{x}^U = \emptyset$	**Landing Page 2** https://www.autojapanonline.com/ $\mathbf{x}^U$ = "kia garage santa rosa"	**Landing Page 3** https://www.1stmedfinancial.com/ $\mathbf{x}^U$ = "bank america vet practice loans"
SCST	Aggressive Defense Attorney - We're Local for Any Location *We'll Fight Tooth & Nail to Ensure Your Rights, Freedom & Future Are Protected.*	Kia Repair Shop for Car - Free Shipping - Shop Online *Call Us for Service, Repair Your Car, Truck, or SUV. Repair Your Car in the Area.*	Veterinary Practice Loans - We're on Lexus of Plano *Veterinary Financing Available for Veterinary Financing*
UNILM	Get the Answers You Need Now - Call Our BUI Defense Firm *We Will Help You Understand Your Rights & Will Fight Tirelessly to Protect Them!*	Kia Repair Specialists - Asian Auto Repair *Factory Trained Technicians. Same Day Service. Great Prices!*	Looking for Vet Practice Loans? - See Why Vets Love 1st Med *We're Here to Help You Find the Right Loan for Your Veterinary Practice.*
UMPG	BUI Defense Attorney - Aggressive & Experienced Help *Are You Facing BUI Charges? We Make Your Fight Our Own and Won't Back Down!*	Kia Garage Specialists - Voted Best Alternative in Santa Rosa *Same Day Service at Low Prices with Warranty in Cars.*	Bank of America Veterinary Loans - See Why Vets Love Us. *Get the Best Rates & No Closing Costs. Over $1m in Closed Practice Transactions!*

Table 4: Example ads generated by using different methods. *SCST* sometimes generates ads with language or grammar issues. Compared with *UNILM*, our method generates more attractive phrases and phrases that are relevant with the user query $\mathbf{x}^U$.

	Impressions	△Revenue	△Click	△MClick
Base2 vs. *Base1*	>3.5 million	+1.15%	+1.63%*	+0.81%
Ours vs. *Base2*	>3.5 million	+1.68%*	+1.31%*	+1.84%*

Table 5: Online experiment result. △Revenue, △Click, and △MClick refer to gain in revenue per thousand impressions, click yield, and mainline click yield, respectively. Statistically significant difference is marked by * (p-value < 0.05).

search results and contribute a majority of the revenue. DSAs are displayed both in the mainline position and at the bottom.

3.4.2 *Compared methods.* Methods to be deployed online need to be carefully evaluated by judges first to ensure good user experience. The standard guideline is that the quality of the generated ads should be similar with or better than the quality of human-written ads. Among all baselines, only methods that leverage pretrained language models reach this high standard for deployment. Accordingly, we deploy and compare three methods:

- *Base1* is the method previously used to generate DSAs. It employs *UNILM* for generating ads, together with a few heuristic extractive approaches (e.g., using the titles of the product landing pages as the ad titles). Among multiple candidate ads for a landing page, the online infrastructure picks the one that results in the largest predicted click probability. The predicted click probability is computed by using an online click prediction model.
- *Base2* leverages *UniLMv2* [4] to generate ads, in addition to the approaches in *Base1*. Similar with *Base1*, the ads with the largest predicted click probability are displayed to users.
- *Ours* includes ads generated by using *UMPG*, in addition to the set of ads generated by *Base2*. As with *Base1* and *Base2*, ads with the largest predicted click probability are displayed.

3.4.3 *Results.* Table 5 shows the online experiment results after the flight runs for 8.8 days. We get over 3.5 million impressions for each compared method. As shown in the table, even though *Base2* already achieves good improvement by combining advanced pretrained models, *ours* still improves over *Base2* significantly (t-test, p-value<0.05) in terms of revenue and clicks by applying the proposed Masked-Sequence Policy Gradient to *UNILM*. This demonstrates the effectiveness of integrating reinforcement learning with pretrained models, which enables effective utilization of unsupervised, supervised, and user feedback (click) data. Due to its good performance, *ours* has been deployed in production to serve the main traffic.

4 RELATED WORK

4.1 Text Generation in Online Advertising

There are two major generation tasks in online advertising: bid phrase generation [28] and text ad generation [8, 15]. For each ad, bid phrases are the search queries that will trigger the ad. Automatically generating bid phrases aims to expand the manually-chosen bid phrase set to more accurately match the corresponding ads [6, 18]. Compared with bid phrase generation, which focuses on relevance or match accuracy, generating ads also requires optimizing fluency and attractiveness of the generated sentences: it is important that the text ads consist of natural language sentences that attracts users by revealing the advantages of the products.

Pioneer works on text ad generation rely on pre-defined templates to generate readable and attractive sentences [5, 12, 32]. Example templates include "<*product name*> with <*feature set*> <*price*>" and "<*feature set*> <*price*>" [32]. Sophisticated methods are developed to extract key phrases and fill them appropriately in the templates. Compared with creating each ad manually, template-based ad generation methods can largely reduce human effort. However, templates are usually rigid, lack diversity, and cannot adapt in the manners that a human would [15].

Recently, Hughes et al. propose a data-driven method that learns to write text ads from existing examples [15]. The model employs LSTMs and attention layers to encode product landing pages and decode text ads. Moreover, REINFORCE with baseline [29] is leveraged to generate attractive ads that potentially have larger click rate. This method achieves a certain degree of success in automatic generation of text ads. However, over 15% text ads it generates are labeled as non-sense, broken, or bad [15], which fails to meet the high quality standard for production. We show that further improvement of ad quality and attractiveness can be achieved by effectively combining different types of data (e.g., the unsupervised data) and tightly integrating reinforcement learning with pretrained models. We also develop a reinforcement learning method, *MPG*, that more effectively explores the action space and works efficiently with large and complex pretrained models. Because of its good performance, our method is deployed in a major search engine.

4.2 Pretrained NLP Models

Pretrained language models have achieved great success in many natural language processing (NLP) tasks. While bidirectional pretrained language models like BERT [10] and RoBERTa [21] are very effective for natural language understanding (NLU) tasks such as sentiment classification and natural language inference, left-to-right pretrained language models like GPT [7, 24, 25] and XLNet [35] may also be used for natural language generation (NLG). Researchers have also proposed sequence-to-sequence pretrained models, e.g., MASS [31] and T5 [26], to better encode input sequence during sequence generation. Recently, *UNILM* [4, 11] has been proposed, which unifies all three types of models. Jointly pretrained with unidirectional, bidirectional, and sequence-to-sequence language model objectives, *UNILM* has proven effective for both NLU and NLG. In this paper, we use *UNILM* as a guiding example to show how large and complex pretrained models can be effectively and efficiently integrated with RL for text ad generation. Our method can also be applied to other pretrained language models or other text generation tasks that benefit from user feedback.

5 CONCLUSION

We show how pretrained models can be enhanced by using deep RL to generate attractive and high-quality text advertisements. To leverage user feedback effectively without hindering user experience, we propose a model-based RL framework for text ad generation. Based on the framework, we propose an RL algorithm, *Masked-Sequence Policy Gradient*, which integrates efficiently with large and complex pretrained models and explores the action space effectively. We have conducted extensive experiments to demonstrate the superiority of our method. Because of its good performance, our method has been deployed to production in Microsoft Bing.

REFERENCES

[1] 2021. Example Landing Page. https://crepusculedespharaons.com/personalized-gifts-find-send-custom-gifts-at-personal-creations.html. Accessed Jun. 1, 2021.

[2] Zoë Abrams and Erik Vee. 2007. Personalized ad delivery when ads fatigue: An approximation algorithm. In *International Workshop on Web and Internet Economics*. Springer, 535–540.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *ICML*.

[4] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*. 642–652.

[5] Kevin Bartz, Cory Barr, and Adil Aijaz. 2008. Natural language generation for sponsored-search advertisements. In *ACM Conference on Electronic Commerce*. 1–9.

[6] Andrei Broder, Evgeniy Gabrilovich, Vanja Josifovski, George Mavromatis, and Alex Smola. 2011. Bid generation for advanced match in sponsored search. In *WSDM*. 515–524.

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[8] Zhangming Chan, Yuchi Zhang, Xiuying Chen, Shen Gao, Zhiqiang Zhang, Dongyan Zhao, and Rui Yan. 2020. Selection and Generation: Learning towards Multi-Product Advertisement Post Generation. In *EMNLP*. 3818–3829.

[9] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML*. 1052–1061.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

[11] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*.

[12] Atsushi Fujita, Katsuhiro Ikushima, Satoshi Sato, Ryo Kamite, Ko Ishiyama, and Osamu Tamachi. 2010. Automatic generation of listing ads by reusing promotional texts. In *International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*. 179–188.

[13] Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. 2019. Towards a deep and unified understanding of deep neural models in nlp. In *ICML*. 2454–2463.

[14] Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang, and Yuzhou Zhang. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *RecSys*. 452–456.

[15] J Weston Hughes, Keng-hao Chang, and Ruofei Zhang. 2019. Generating Better Search Engine Text Advertisements with Deep Reinforcement Learning. In *KDD*. 2269–2277.

[16] Bernard J Jansen and Marc Resnick. 2005. Examining searcher perceptions of and interactions with sponsored results. In *Workshop on Sponsored Search Auctions*.

[17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

[18] Mu-Chu Lee, Bin Gao, and Ruofei Zhang. 2018. Rare query expansion through generative adversarial networks in search advertising. In *KDD*. 500–508.

[19] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[20] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *International Conference on World Wide Web Companion*. 689–698.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[22] A Rupam Mahmood, Hado P van Hasselt, and Richard S Sutton. 2014. Weighted importance sampling for off-policy learning with linear function approximation. In *NeurIPS*. 3014–3022.

[23] Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *ICLR*.

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018). https://openai.com/blog/language-unsupervised/

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* 21 (2020), 1–67.

[27] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

[28] Sujith Ravi, Andrei Broder, Evgeniy Gabrilovich, Vanja Josifovski, Sandeep Pandey, and Bo Pang. 2010. Automatic generation of bid phrases for online advertising. In *WSDM*. 341–350.

[29] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*. 7008–7024.

[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should I trust you?" Explaining the predictions of any classifier. In *KDD*. 1135–1144.

[31] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *ICML*.

[32] Stamatina Thomaidou, Ismini Lourentzou, Panagiotis Katsivelis-Perakis, and Michalis Vazirgiannis. 2013. Automated snippet generation for online advertising. In *CIKM*. 1841–1844.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.

[34] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[35] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

SUPPLEMENTARY MATERIAL

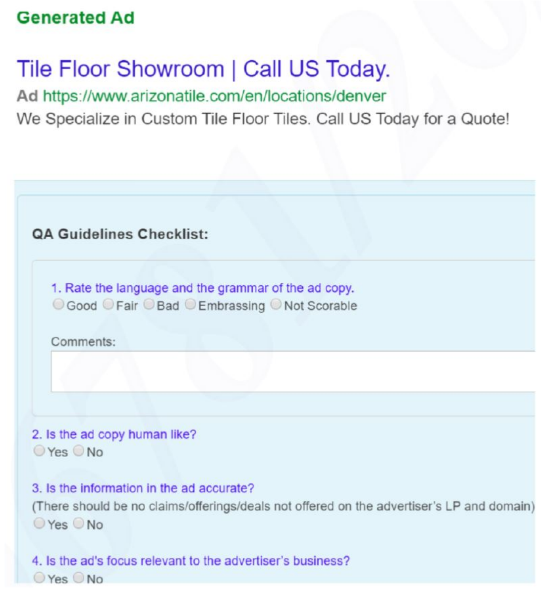
Appendix I. Implementation Details

Constraint list. The list of constraints we use to avoid generating attractive but unfaithful ads is shown in Table 6. The constraints can be divided into two categories.

Category I includes phrases about concrete facts, e.g., "free ship" or "lowest price". Making false claims about these facts can easily annoy users and lead to severe consequences. Thus, we only allow the generation of these phrases if they have been mentioned in the product landing page title ($\mathbf{x}^T$) or body ($\mathbf{x}^B$), and assign a large punish weight l_c if the constraint is violated.

Category II includes phrases that are less sensitive, e.g., "official" and "luxury". Whether these phrases are fact or not for a product is more ambiguous. Even if the product landing page does not contain these phrases, it is highly possible that they are still true given the content of the landing page. For example, even if a landing page does not contain the word "official", we can generate "Official Olukai® Webstore" in the ad if "Olukai" appears frequently in the product landing page. As a result, we allow the generation of these words if they appear in the product landing page ($\mathbf{x}^T$ or $\mathbf{x}^B$) or the human-written ads ($\mathbf{y}$). By maximizing $\mathcal{R}_{\text{cons}}$ during RL training, the model automatically learns to judge whether a phrase in Category II can be used in an ad based on the content of the landing page.

Human evaluation: user interface and labeling guideline. User interfaces for labeling ad quality and ad attractiveness are shown in Figs. 2 and 3, respectively. In addition to the user interfaces, a detailed guidance about each task is provided to the judges before they start labeling ads. The guidance includes an overall description of the task, clarifications about what is important, and labeling examples. A part of the guidance for labeling ad attractiveness is given in Fig. 4. The judges are asked to carefully read the guidance and their initial labeling results are verified to avoid misunderstanding.



Generated Ad

Tile Floor Showroom | Call US Today.

Ad https://www.arizonatile.com/en/locations/denver

We Specialize in Custom Tile Floor Tiles. Call US Today for a Quote!

QA Guidelines Checklist:

1. Rate the language and the grammar of the ad copy.

○ Good ○ Fair ○ Bad ○ Embrassing ○ Not Scorable

Comments:

2. Is the ad copy human like?

○ Yes ○ No

3. Is the information in the ad accurate?

(There should be no claims/offerings/deals not offered on the advertiser's LP and domain)

○ Yes ○ No

4. Is the ad's focus relevant to the advertiser's business?

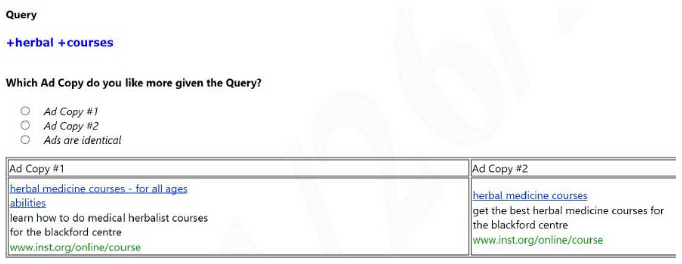
○ Yes ○ No

Figure 2: User interface for labeling ad quality.

Appendix II. Proof and Theoretical Analysis

Relationship between $\mathcal{R}_{\text{human}}$ and L_{XE}. Here, we show that optimizing the reinforcement learning loss for $\mathcal{R}_{\text{human}} = \mathbb{I}(a_t = y_{t+1})$ by using Eq. (16) is closely related to optimizing the cross-entropy loss L_{XE}. When $\mathcal{R}$ is $\mathcal{R}_{\text{human}}$ and when a simple baseline method that obtains a zero reward for $\forall t \notin M$ is used, we can compute Eq. (16) accurately by enumerating all possible actions:

$$\begin{aligned}
\nabla_\theta L_{RL} \approx & -\sum_{\mathbf{x}\in X}\sum_{t\in M} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} \mathbb{E}_{a_t\sim p_{\theta'}} \beta_t \triangle\mathcal{R}_t \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t), \\
= & -\sum_{\mathbf{x}\in X}\sum_{t\in M} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} \sum_{a_t} p_{\theta'}(a_t|\mathbf{s}_t)\beta_t \triangle\mathcal{R}_t \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t), \\
= & -\sum_{\mathbf{x}\in X}\sum_{t\in M} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} p_{\theta'}(y_{t+1}|\mathbf{s}_t)\beta_t \nabla_\theta \log p_\theta(y_{t+1}|\mathbf{s}_t) \\
= & -\sum_{\mathbf{x}\in X}\sum_{t\in M} \mathbb{E}_{\mathbf{s}_t\sim p_{\theta'}} p_{\theta}(y_{t+1}|\mathbf{s}_t) \nabla_\theta \log p_\theta(y_{t+1}|\mathbf{s}_t) \qquad (17) \\
\approx & -\sum_{\mathbf{x}\in X}\sum_{t\in M} p_{\theta}(y_{t+1}|\mathbf{s}_t) \nabla_\theta \log p_\theta(y_{t+1}|\mathbf{x}, \mathbf{y}_{\le t\setminus M}) \\
\approx & -\sum_{\mathbf{x},\mathbf{y}\in X,Y}\sum_{t\in M} \mu \nabla_\theta \log p_\theta(y_{t+1}|\mathbf{x}, \mathbf{y}_{\le t\setminus M}) \\
= & \mu\nabla_\theta L_{XE}
\end{aligned}$$



Query

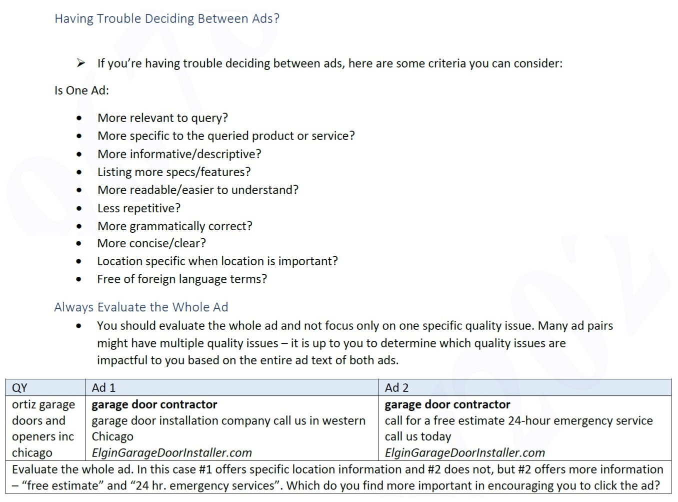
+herbal +courses

Which Ad Copy do you like more given the Query?

○ Ad Copy #1
○ Ad Copy #2
○ Ads are identical

Ad Copy #1	Ad Copy #2
herbal medicine courses - for all ages abilities learn how to do medical herbalist courses for the blackford centre www.inst.org/online/course	herbal medicine courses get the best herbal medicine courses for the blackford centre www.inst.org/online/course

Figure 3: User interface for labeling ad attractiveness.



Having Trouble Deciding Between Ads?

- If you're having trouble deciding between ads, here are some criteria you can consider:

Is One Ad:

- More relevant to query?
- More specific to the queried product or service?
- More informative/descriptive?
- Listing more specs/features?
- More readable/easier to understand?
- Less repetitive?
- More grammatically correct?
- More concise/clear?
- Location specific when location is important?
- Free of foreign language terms?

Always Evaluate the Whole Ad

- You should evaluate the whole ad and not focus only on one specific quality issue. Many ad pairs might have multiple quality issues – it is up to you to determine which quality issues are impactful to you based on the entire ad text of both ads.

QY	Ad 1	Ad 2
ortiz garage doors and openers inc chicago	**garage door contractor** garage door installation company call us in western Chicago *ElginGarageDoorInstaller.com*	**garage door contractor** call for a free estimate 24-hour emergency service call us today *ElginGarageDoorInstaller.com*
Evaluate the whole ad. In this case #1 offers specific location information and #2 does not, but #2 offers more information – "free estimate" and "24 hr. emergency services". Which do you find more important in encouraging you to click the ad?		

Figure 4: Part of the guidance for labeling ad attractive.

	$\mathbf{v}_c$	l_c	η_c
Category I	"free ship", "lowest price", "lowest cost", "best", "first", "largest", "latest", "permanently", or any word that contains a number	3	$\mathbb{I}(\mathbf{v}_c \not\subseteq \mathbf{x}^T \;\&\; \mathbf{v}_c \not\subseteq \mathbf{x}^B)$
Category II	"official", "top", "experienced", "luxury", "instant"	1	$\mathbb{I}(\mathbf{v}_c \not\subseteq \mathbf{x}^T \;\&\; \mathbf{v}_c \not\subseteq \mathbf{x}^B \;\&\; \mathbf{v}_c \not\subseteq \mathbf{y})$

Table 6: The constraints we use to compute $\mathcal{R}_{\text{cons}}(\cdot)$.

The 2nd line is derived by computing the expectation $\mathbb{E}_{a_t \sim p_{\theta'}}(\cdot)$ through enumerating all possible a_t, which is more accurate compared with approximating $\mathbb{E}_{a_t \sim p_{\theta'}}(\cdot)$ by sampling several a_t. The 3nd line holds because $\triangle\mathcal{R}_t = \mathbb{I}(a_t = y_{t+1})$ is not 0 only when a_t is y_{t+1}. The 4rd line can be derived since $\beta_t = \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)}$ for $\forall a_t$. The 5th line is true considering that $p_\theta(a_t|\mathbf{s}_t) = p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t}) \approx p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t\setminus M}) = p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})$. The 6th line can be derived by considering that θ has been trained with human-written ads during the fine-tuning phase. Thus, $p_\theta(y_{t+1}|\mathbf{s}_t)$, the probability of generating the human-written token, is usually large with a relatively small variance (≈0.04 in practice). Thus, we may ignore the variance and approximately consider $p_\theta(y_{t+1}|\mathbf{s}_t)$ as a constant μ. Then, we have $\nabla_\theta L_{RL} \approx \mu \nabla_\theta L_{XE}$, i.e., optimizing the reinforcement learning loss for $\mathcal{R}_{\text{human}}$ according to Eq. (16) is similar to optimizing the cross-entropy loss L_{XE} by using gradient descent.

Policy gradient of the modified RL loss. Considering the modified RL loss defined in Eq. (14) in the paper:

$$L_{RL} = -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \mathbb{E}_{a_t \sim p_{\theta'}(a_t|\mathbf{s}_t)} \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)} \mathcal{R}(\mathbf{s}_t, a_t) \tag{18}$$

We now prove that the gradient of this loss can be approximately computed by using Eq. (15) in the paper.

Proof.

$$\begin{aligned}
\nabla_\theta L_{RL} &\approx -\nabla_\theta \sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \mathbb{E}_{a_t \sim p_{\theta'}(a_t|\mathbf{s}_t)} \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)} \mathcal{R}(\mathbf{s}_t, a_t) \\
&\approx -\nabla_\theta \sum_{\mathbf{x}\in X}\sum_{t<T} \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{a_t \sim p_{\theta'}(a_t|\mathbf{s}_t^{(n)})} \frac{p_\theta(a_t|\mathbf{s}_t^{(n)})}{p_{\theta'}(a_t|\mathbf{s}_t^{(n)})} \mathcal{R}(\mathbf{s}_t^{(n)}, a_t) \\
&= -\nabla_\theta \sum_{\mathbf{x}\in X}\sum_{t<T} \frac{1}{N}\sum_{n=1}^{N} \sum_{a_t} p_{\theta'}(a_t|\mathbf{s}_t^{(n)}) \frac{p_\theta(a_t|\mathbf{s}_t^{(n)})}{p_{\theta'}(a_t|\mathbf{s}_t^{(n)})} \mathcal{R}(\mathbf{s}_t^{(n)}, a_t) \\
&= -\sum_{\mathbf{x}\in X}\sum_{t<T} \frac{1}{N}\sum_{n=1}^{N} \sum_{a_t} \nabla_\theta p_\theta(a_t|\mathbf{s}_t^{(n)}) \mathcal{R}(\mathbf{s}_t^{(n)}, a_t) \qquad (19) \\
&= -\sum_{\mathbf{x}\in X}\sum_{t<T} \frac{1}{N}\sum_{n=1}^{N} \sum_{a_t} p_\theta(a_t|\mathbf{s}_t^{(n)}) \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t^{(n)}) \mathcal{R}(\mathbf{s}_t^{(n)}, a_t) \\
&\approx -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \sum_{a_t} p_\theta(a_t|\mathbf{s}_t) \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t) \mathcal{R}(\mathbf{s}_t, a_t) \\
&= -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \sum_{a_t} p_{\theta'}(a_t|\mathbf{s}_t) \beta_t \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t) \mathcal{R}(\mathbf{s}_t, a_t) \\
&= -\sum_{\mathbf{x}\in X}\sum_{t<T} \mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \mathbb{E}_{a_t \sim p_{\theta'}(a_t|\mathbf{s}_t)} \beta_t \nabla_\theta \log p_\theta(a_t|\mathbf{s}_t) \mathcal{R}(\mathbf{s}_t, a_t)
\end{aligned}$$

The 2nd and 6th lines are derived by approximating the expectation $\mathbb{E}_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)}(\cdot)$ through sampling. Here, $\mathbf{s}_t^{(n)}$ is the n-th sample from the distribution $p_{\theta'}(\mathbf{s}_t)$. The 5th line holds because $\nabla_x f(x) = f(x)\nabla_x \log f(x)$. The 7th line is true since $\beta_t = \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)}$. □

Approximation of β_t. β_t can be accurately computed by:

$$\beta_t = \frac{p_\theta(a_t|\mathbf{s}_t)}{p_{\theta'}(a_t|\mathbf{s}_t)} = \frac{p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t})}{(1-P_m)\mathbb{I}(a_t = y_{t+1}) + P_m p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})} \tag{20}$$

We can compute β_t for $\forall t$ by performing two forward propagations through *UNILM*. We can more efficiently compute β_t by considering that $p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t}) \approx p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t\setminus M}) = p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})$. Thus,

$$\begin{aligned}
\beta_t &= \frac{p_\theta(a_t|\mathbf{x}, \hat{\mathbf{y}}_{\leq t})}{(1-P_m)\mathbb{I}(a_t = y_{t+1}) + P_m p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})} \\
&\approx \frac{p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})}{(1-P_m)\mathbb{I}(a_t = y_{t+1}) + P_m p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})} \\
&= \frac{1}{\frac{1-P_m}{p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})}\mathbb{I}(a_t = y_{t+1}) + P_m} \qquad (21) \\
&\approx \frac{1}{(1-P_m)\mu'\mathbb{I}(a_t = y_{t+1}) + P_m} \\
&= \frac{1}{P_m}[1 - \beta'\mathbb{I}(a_t = y_{t+1})]
\end{aligned}$$

The 4th line can be derived by ignoring the small variance of $\frac{1}{p_\theta(a_t|\mathbf{x}, \mathbf{y}_{\leq t\setminus M})}$ (≈0.13 in practice) and consider it as a constant. $\beta' = \frac{(1-P_m)\mu'}{(1-P_m)\mu' + P_m}$ in the 5th line is a constant that determines how much we lower the weight for human-written tokens. We find that approximating β_t by using Eq. (21) sufficiently balances efficiency and effectiveness in practice.