Domain-Constrained Advertising Keyword Generation

Hao Zhou
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
tuxchow@gmail.com

Minlie Huang*
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
aihuang@tsinghua.edu.cn

Yishun Mao
Sogou Inc., Beijing, China
ps-adwr@sogou-inc.com

Changlei Zhu
Sogou Inc., Beijing, China
zhuchanglei@sogou-inc.com

Peng Shu
Sogou Inc., Beijing, China
shupeng203672@sogou-inc.com

Xiaoyan Zhu
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
zxy-dcs@tsinghua.edu.cn

ABSTRACT

Advertising (ad for short) keyword suggestion is important for sponsored search to improve online advertising and increase search revenue. There are two common challenges in this task. First, **the keyword bidding problem**: hot ad keywords are very expensive for most of the advertisers because more advertisers are bidding on more popular keywords, while unpopular keywords are difficult to discover. As a result, most ads have few chances to be presented to the users. Second, **the inefficient ad impression issue**: a large proportion of search queries, which are unpopular yet relevant to many ad keywords, have no ads presented on their search result pages. Existing retrieval-based or matching-based methods either deteriorate the bidding competition or are unable to suggest novel keywords to cover more queries, which leads to inefficient ad impressions.

To address the above issues, this work investigates to use generative neural networks for keyword generation in sponsored search. Given a purchased keyword (a word sequence) as input, our model can generate a set of keywords that are not only relevant to the input but also satisfy the domain constraint which enforces that the domain category of a generated keyword is as expected. Furthermore, a reinforcement learning algorithm is proposed to adaptively utilize domain-specific information in keyword generation. Offline evaluation shows that the proposed model can generate keywords that are diverse, novel, relevant to the source keyword, and accordant with the domain constraint. Online evaluation shows that generative models can improve coverage (COV), click-through rate (CTR), and revenue per mille (RPM) substantially in sponsored search.

*Corresponding author: Minlie Huang, aihuang@tsinghua.edu.cn.

CCS CONCEPTS

• **Information systems → Sponsored search advertising**; *Query log analysis*; Query suggestion.

KEYWORDS

ad keyword generation, domain constraint, reinforcement learning

ACM Reference Format:
Hao Zhou, Minlie Huang, Yishun Mao, Changlei Zhu, Peng Shu, and Xiaoyan Zhu. 2019. Domain-Constrained Advertising Keyword Generation. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3308558.3313570

1 INTRODUCTION

Advertising (ad for short) keyword suggestion is an important task for sponsored search which is one of the major types of online advertising and the major source of revenue for search companies. In sponsored search, a search engine first retrieves a set of advertisements whose keywords match a user issued query. It then ranks these advertising candidates according to an auction process by considering both the ad quality and the bid price of each ad [3]. Finally, the chosen advertisement is presented in a search result

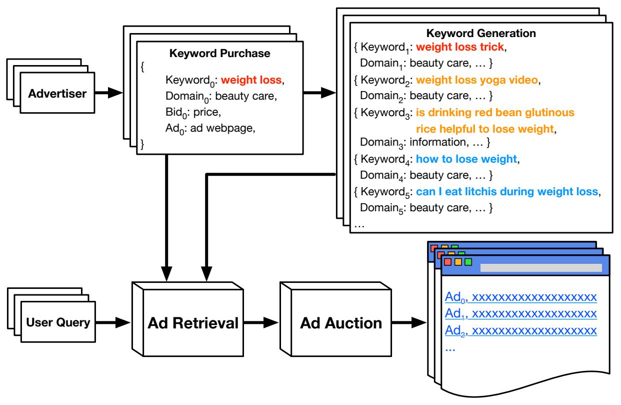

Advertiser
Keyword Purchase
{
Keyword$_0$: weight loss,
Domain$_0$: beauty care,
Bid$_0$: price,
Ad$_0$: ad webpage,
}
Keyword Generation
{ Keyword$_1$: weight loss trick,
Domain$_1$: beauty care, … }
{ Keyword$_2$: weight loss yoga video,
Domain$_2$: beauty care, … }
{ Keyword$_3$: is drinking red bean glutinous rice helpful to lose weight,
Domain$_3$: information, … }
{ Keyword$_4$: how to lose weight,
Domain$_4$: beauty care, … }
{ Keyword$_5$: can I eat litchis during weight loss,
Domain$_5$: beauty care, … }
…
User Query
Ad Retrieval
Ad Auction
Ad$_0$, xxxxxxxxxxxxxxxxxxx
Ad$_1$, xxxxxxxxxxxxxxxxxxx
Ad$_2$, xxxxxxxxxxxxxxxxxxx
…

Figure 1: The sponsored search with keywords generated by our model. The red/orange/blue keywords are the keywords which have more than 3/less than 3/no ads on the impression webpage respectively.

page. Therefore, ad keywords are vital for ads to gain access to the auction process and have the chance to be displayed on a search result page.

However, there are two common challenges that should be addressed in sponsored search. The first one is the **keyword bidding problem**: due to the Matthew effect [29], the hot ad keywords become too expensive for most of the advertisers, because too many advertisers bid on such keywords. As a result, many advertisers cannot survive in the auction process to get their desired ad impressions. As reported in [43], 55.3% advertisers have no ad impression, and 92.3% advertisers have no ad click at all, which is mainly caused by low bid price or improper keywords that they bid. The second issue is **inefficient ad impressions**: a substantial proportion of search queries, which are unpopular yet relevant to many ad keywords, have less competitive ads (46.6% search queries) even no ads (41.0% search queries) on their search result pages as reported in [43]. Because of the two reasons, the expectation of advertisers is not satisfied and the revenue of search engines is also not optimized.

To address these problems, several prior studies have been conducted in keyword generation or suggestion [2, 8, 13, 23, 30]. Most of these studies adopt matching methods based on the word co-occurrence between ad keywords [8] and queries [13]. However, these methods tend to suggest popular keywords to advertisers, which will deteriorate the bidding competition. In addition, these approaches cannot suggest novel ad keywords which do not appear in the corpus.

Recently, deep learning technologies have been applied in many natural language tasks, such as machine translation [39], ad keyword suggestion [15], and query rewriting [18]. However, it's not trivial to adapt these neural networks to the ad keyword generation task, due to two major challenges. **First**, the generated ad keywords should be diversified and relevant to the original keywords to cover more user queries, which is not supported by existing neural models applied in keyword and query generation tasks. **Second**, the generated ad keywords should satisfy many constraints in sponsored search. For instance, to provide relevant yet unexplored ads for users, it is necessary to satisfy the domain constraint which means that the generated keywords should belong to the domain of the source keyword or several appropriate domains. For instance, a keyword in the *health care* domain should only match the keywords from the same domain to ensure the ad quality, while a keyword from the *information* domain could match those from various domains, such as *entertainment* and *shopping*, to cover diverse user queries.

In this paper, we investigate to use generative neural networks in the task of ad keyword generation. Given the purchased keyword as input, our generative model can suggest a set of keywords based on the semantics of the input keyword, as shown in Figure 1. The generated keywords are diverse and even completely novel (the blue keywords) from those in the dataset. This generative approach can address the aforementioned problems in two ways. **First**, our model is able to generate diverse, novel keywords, instead of merely suggesting the existing popular keywords in the dataset, which can recommend keywords for advertisers to alleviate the keyword bidding problem and retrieve ads by keyword reformulation for sponsored search engines to address the inefficient ad impression issue. **Second**, to improve the quality of the generated keywords, we incorporate the domain constraint in our model, which is a key factor considered in sponsored search to display ads. Through capturing the domain constraint, our model learns both semantic information and domain-specific information of ad keywords during training, and is consequently able to predict the proper domain category and generate the ad keyword based on the predicted category. In addition, our model uses reinforcement learning to strengthen the domain constraint in the generation process, which further improve the domain correlation and the keyword quality.

To summarize, this paper makes the following contributions:

- This work investigates to use generative neural networks for keyword generation in sponsored search, which addresses the issues of keyword bidding and inefficient ad impressions.
- We present a novel model that incorporates the domain constraint in ad keyword generation. The model is able to predict a suitable domain category and generate an ad keyword correspondingly. A reinforcement learning algorithm is devised to adaptively utilize domain-specific information in keyword generation, which further improves the domain consistency and the keyword quality.
- We perform offline and online evaluation with the proposed model, and extensive results demonstrate that our model can generate diverse, novel, relevant, and domain-consistent keywords, and also improves the performance of sponsored search.

2 RELATED WORK

2.1 Keyword Generation

A variety of methods has been proposed for generating and suggesting the keywords for advertisements, as ad keywords play a critical role in sponsored search. Joshi and Motwani [23] collected text-snippets from search engine given the keyword as input, and constructed them as a graph model to generate relevant keywords based on the similarity score. Abhishek and Hosanagar [2] further improved the graph model, which computes the similarity score based on the retrieved documents. Chen et al. [8] applied concept hierarchy to keyword generation, which suggests new keywords according to the concept information rather than the co-occurrence of the keywords itself. Fuxman et al. [13] made use of the query-click graph to compute the keyword similarity for recommendation based on a random walk with absorbing states. Ravi et al. [32] introduced a generative approach, a monolingual statistical translation model, to generate bid phrases given the landing page, which performs significantly better than extraction-based methods. Recently due to the advances of deep learning, various neural network models have been applied to ad keyword suggestion. Grbovic et al. [15] proposed several neural language models to learn low-dimensional, distributed representations of search queries based on context and content of the ad queries within a search session. Zhai et al. [41] applied an attention network which is stacked on top of a recurrent neural network (RNN) and learns to assign attention scores to words within a sequence (either a query or an ad).

2.2 Query Generation

Another related research topic is query generation, which is widely studied and applied in organic search. It improves user experience by either expanding (reformulating) a user's query to improve retrieval performance, or providing suggestions through guessing the user intention, according to the user's behaviour pattern (query suggestion). Some previous studies adopt query logs to generate queries by handcrafted features such as click-through data [11, 25, 35, 42], session based co-occurrence [17, 20, 22] or query similarity [4, 6, 12]. Recently, artificial neural networks have been applied in query processing. A hierarchical recurrent encoder-decoder model [38] is introduced to query suggestion. He et al. [18] proposed a learning to rewrite framework consisting of a candidate generating phase and a candidate ranking phase for query rewriting. Song et al. [37] use an RNN encoder-decoder to translate a natural language query into a keyword query. An attention based hierarchical neural query suggestion model that combines a session-level neural network and a user-level neural network to model the short- and long-term search history of a user is proposed by Chen et al. [7]

2.3 Generative Neural Network

Recently, generative neural networks have been applied in many natural language tasks, such as machine translation [39], dialogue generation [34], and query rewriting [18]. Sutskever et al. [39] apply an end-to-end approach, a sequence to sequence (Seq2Seq) model, on machine translation tasks. Shang et al. [34] further introduce the Seq2Seq model to dialogue generation tasks with novel attention mechanisms. Although the Seq2Seq model is capable of generating a sequence with a certain meaning, it isn't suitable for diversified sequence generation as argued in [27]. Therefore, latent variable based models are proposed to address the diversity and uncertainty problem. Serban et al. [33] introduce latent variables to a hierarchical encoder-decoder neural network to explicitly model generative processes that possess multiple levels of variability. Zhou and Wang [45] propose several conditional variational autoencoders to use emojis to control the emotion of the generated text. Zhao et al. [44] use latent variables to learn a distribution over potential conversational intents based on conditional variational autoencoders, that is able to generate diverse responses using only greedy decoders.

3 MODEL

3.1 Background: Encoder-Attention-Decoder Framework

We first introduce a general encoder-attention-decoder framework based on sequence-to-sequence (Seq2Seq) learning [39], which is a widely used generative neural network. The encoder and decoder of the Seq2Seq model [39] are implemented with GRU [9, 10].

The encoder represents an input sequence $X = x_1x_2 \cdots x_n$ with hidden representations $\boldsymbol{H} = \boldsymbol{h}_1\boldsymbol{h}_2 \cdots \boldsymbol{h}_n$[1], which is briefly defined as below:

$$\boldsymbol{h}_t = \mathbf{GRU}(\boldsymbol{h}_{t-1}, \boldsymbol{e}(x_t)), \tag{1}$$

where $\boldsymbol{e}(x_t)$ is the embedding of the word x_t, and GRU is gated recurrent unit [9].

The decoder takes as input a context vector $\boldsymbol{c}_t$ and the embedding of a previously decoded word $\boldsymbol{e}(y_{t-1})$, and updates its state $\boldsymbol{s}_t$ using another GRU:

$$\boldsymbol{s}_t = \mathbf{GRU}(\boldsymbol{s}_{t-1}, [\boldsymbol{c}_{t-1}; \boldsymbol{e}(y_{t-1})]), \tag{2}$$

where $[\boldsymbol{c}_{t-1}; \boldsymbol{e}(y_{t-1})]$ is the concatenation of the two vectors, serving as input to the GRU network. The context vector $\boldsymbol{c}_t$ is designed to attend to the key information of the input sequence during decoding, which is a weighted sum of the encoder's hidden states as $\boldsymbol{c}_{t-1} = \sum_{k=1}^{n} \alpha_k^{t-1}\boldsymbol{h}_k$, and α_k^{t-1} measures the relevance between state $\boldsymbol{s}_{t-1}$ and hidden state $\boldsymbol{h}_k$. Refer to [5] for more details.

Once the state vector $\boldsymbol{s}_t$ is obtained, the decoder generates a token by sampling from the generation distribution $\boldsymbol{o}_t$ computed from the decoder's state $\boldsymbol{s}_t$ as follows:

$$\begin{aligned} y_t \sim \boldsymbol{o}_t &= P(y_t \mid y_1, y_2, \cdots, y_{t-1}, \boldsymbol{c}_t), && (3)\\ &= \mathrm{softmax}(\mathbf{W_o}\boldsymbol{s}_t). && (4) \end{aligned}$$

3.2 Task Definition and Overview

Our problem is formulated as follows: Given a purchased ad keyword $\boldsymbol{X} = (x_1, x_2, \cdots, x_n)$ (a word sequence)[2] and the corresponding domain category d_x obtained from a domain classifier, the goal is to predict a suitable target domain category d_y and generate a target keyword $\boldsymbol{Y} = (y_1, y_2, \cdots, y_m)$ (a word sequence) that is coherent with the domain category d_y. Essentially, the model estimates the probability: $P(Y, d_y|\boldsymbol{X}, d_x) = P(d_y|\boldsymbol{X}, d_x) \prod_{t=1}^{m} P(y_t|y_{<t}, d_y, \boldsymbol{X}, d_x)$. The domain categories are adopted from the sponsored search engine, which consists of k domain categories, such as *beauty care*, *shopping*, and *entertainment*.

Building upon the encoder-attention-decoder framework, we propose the Domain-Constrained Keyword Generator (DCKG) to generate diversified keywords with domain constraints using three mechanisms. **First**, DCKG incorporates a latent variable sampled from a multivariate Gaussian distribution to generate diversified

[1]Throughout the paper, a bold character (e.g., $\boldsymbol{h}$) denotes the vector representation of a variable (h).
[2]Throughout the paper, a keyword refers to a word sequence, but not a single word.

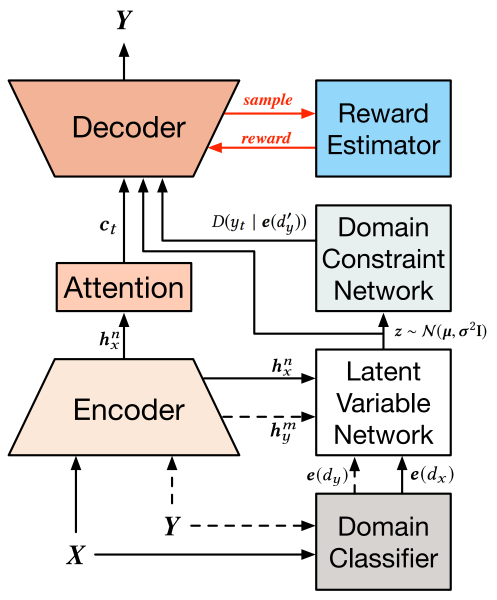


Figure 2: Overview of DCKG. The dashed arrow is used only in supervised training. The red arrow is used only in reinforcement learning. During reinforcement learning, DCKG infers a set of keyword samples and get their rewards from the reward estimator, which are used to further train our model to improve the quality of generated keywords.

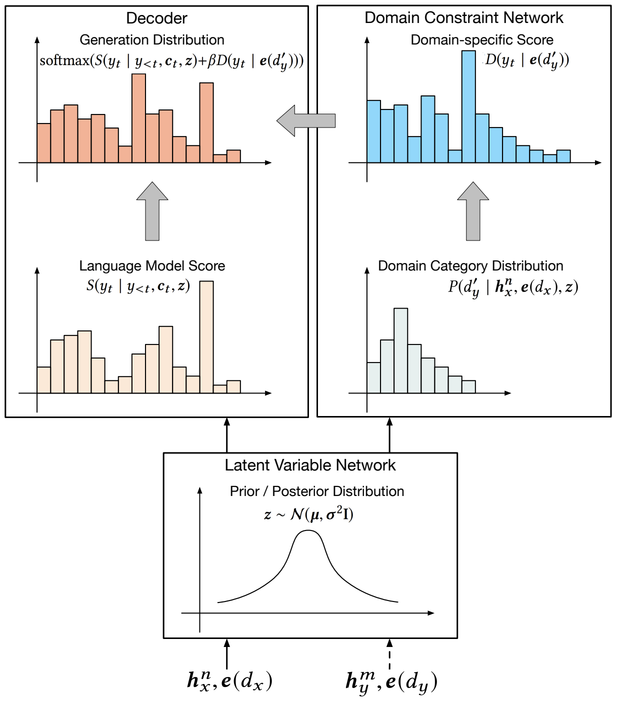
keywords. **Second**, a domain constraint network is proposed to facilitate generating domain-consistent keywords, which imposes more probability bias to domain-specific words. **Third**, DCKG further optimizes the decoder to adjust the word generation distribution with reinforcement learning.

An overview of DCKG is presented in Figure 2, which illustrates the dataflow of DCKG in supervised learning, reinforcement learning, and inference processes. In supervised learning, the source keyword $X = (x_1, x_2, \cdots, x_n)$ and the target keyword $Y = (y_1, y_2, \cdots, y_m)$ are fed to the encoder to generate the hidden representations h^n_x and h^m_y, meanwhile, they are fed to the domain classifier to obtain their domain categories d_x and d_y respectively. The domain categories are further converted to the domain embeddings $e(d_x)$ and $e(d_y)$ to encode the domain-specific information in DCKG. Then, the latent variable z is sampled from a multivariate Gaussian distribution, which is determined by a recognition network that takes the hidden representations and the domain embeddings as input. Given the latent variable z, the hidden representation of source keyword h^n_x, and the domain embedding $e(d_x)$, DCKG predicts the target keyword category d'_y and generate the domain-specific word score $D(y_t \mid e(d'_y))$. Finally, the decoder takes as input the context vector c_t generated by attention mechanism, the latent variable z, and the domain-conditioned word score $D(y_t \mid e(d'_y))$ to generate the target keyword Y.

During the inference process, DCKG has the input of only the source keyword $X = (x_1, x_2, \cdots, x_n)$ to generate a target keyword, conditioned on the latent variable sampled from a prior network which is approximated by the recognition network during supervised training.

During the reinforcement learning process, DCKG first infers a set of keyword samples. Then, these samples are fed to the reward estimator to get their rewards, considering both domain-specific and semantic information. Finally, these rewards are applied to train our model using reinforcement learning to further improve the quality of generated keywords.

3.3 Supervised Learning



Decoder
Generation Distribution
$\mathrm{softmax}(S(y_t \mid y_{<t}, c_t, z) + \beta D(y_t \mid e(d'_y)))$
Domain Constraint Network
Domain-specific Score
$D(y_t \mid e(d'_y))$
Language Model Score
$S(y_t \mid y_{<t}, c_t, z)$
Domain Category Distribution
$P(d'_y \mid h^n_x, e(d_x), z)$
Latent Variable Network
Prior / Posterior Distribution
$z \sim \mathcal{N}(\mu, \sigma^2 I)$
$h^n_x, e(d_x)$
$h^m_y, e(d_y)$

Figure 3: Computation flow of DCKG. The dashed arrow is used only in supervised learning to model the posterior distribution. During the inference process, the latent variable z is sampled from the prior distribution, and then fed to the domain constraint network and to the decoder. In the domain constraint network, the latent variable is used to predict the domain category distribution to obtain the target domain category d'_y, which is then used to compute the domain-specific score. In the decoder, the latent variable is used to compute the language model score. Finally, the language model score and the domain-specific score are combined to estimate the distribution for word generation.

We will concentrate on introducing the latent variable network, the domain constraint network, and the decoder network, as shown in Figure 3. The domain classifier is adopted from the sponsored search engine with parameters fixed during training. It generates a one-hot domain category distribution d_x for a keyword. A category is converted to a domain category embedding as follows:

$$e(d_x) = \mathbf{V_d} d_x, \tag{5}$$

where $\mathbf{V_d}$ is a random initialized domain category embedding matrix which will be learned automatically.

3.3.1 Latent Variable Network. Building upon the encoder-attention-decoder framework, our model introduces a latent variable to project ad keywords to a latent space. By this means, the model can generate diversified keywords conditioned on different latent variable sampled from the latent space. Specifically, we adopt the conditional variational autoencoder (CVAE) [36] as the latent variable network, which is successfully applied in language generation tasks [44, 45].

DCKG assumes the latent variable $\boldsymbol{z}$ follows a multivariate Gaussian distribution, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$. The latent variable network consists of a prior network and a recognition network to model the semantic and domain-specific information of ad keyword in the latent space.

In the training process, the recognition network takes as input the semantic representations $\boldsymbol{h}_x^n, \boldsymbol{h}_y^m$ and the domain embeddings $\boldsymbol{e}(d_x), \boldsymbol{e}(d_y)$ of the source keyword $\boldsymbol{X}$ and the target keyword Y, to approximate the true posterior distribution, as $q_\phi(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), \boldsymbol{h}_y^m, \boldsymbol{e}(d_y)) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$.

In the inference process, the prior network takes only the semantic representation $\boldsymbol{h}_x^n$ and the domain embedding $\boldsymbol{e}(d_x)$ of the source keyword $\boldsymbol{X}$ as input, to sample latent variables from the prior distribution, $p_\theta(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x)) \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\sigma}'^2\mathbf{I})$. The prior/posterior distribution can be parameterized by neural networks such as a multilayer perceptron (MLP) as follows:

$$\begin{aligned} \left[\boldsymbol{\mu}, \boldsymbol{\sigma}^2\right] &= \mathbf{MLP}(\boldsymbol{h}_x^n, \boldsymbol{e}(d_x), \boldsymbol{h}_y^m, \boldsymbol{e}(d_y)), && (6)\\ \left[\boldsymbol{\mu}', \boldsymbol{\sigma}'^2\right] &= \mathbf{MLP}(\boldsymbol{h}_x^n, \boldsymbol{e}(d_x)), && (7)\end{aligned}$$

To alleviate the inconsistency between the prior distribution and the posterior distribution, we add the KL divergence term to the loss function as follows:

$$\mathcal{L}_1 = KL(q_\phi(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), \boldsymbol{h}_y^m, \boldsymbol{e}(d_y)) \,\|\, p_\theta(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x))) \quad (8)$$

3.3.2 Domain Constraint Network. The domain constraint network is designed to model the domain-specific information of ad keyword. The output of the network is further incorporated into the process of keyword generation to improve the quality of generated keywords in sponsored search. It plays two roles in our model: first, predicting an appropriate domain category given a latent variable; second, endowing the generated keyword with the target domain features.

Essentially, it predicts a domain category distribution conditioned on the latent variable sampled from a multivariate Gaussian distribution. Once the domain category distribution is determined, we can sample a target domain category by an argmax operation. However, argmax operation is non-differentiable and the training signal cannot be backpropagated. Inspired by the Gumbel-Max trick [16, 28], we adopt Gumbel-Softmax [19] as a differentiable substitute to generate a sample from the domain category distribution, which is defined as follows:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathbf{U}(0,1), && (9)\\ \boldsymbol{g} &= -\log(-\log(\boldsymbol{\epsilon})), && (10)\\ \boldsymbol{o}_d &= \mathbf{U_d}\,\mathbf{MLP}(\boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z), && (11)\\ P_{real}(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z) &= \mathrm{softmax}(\boldsymbol{o}_d), && (12)\\ P_{sample}(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z) &= \mathrm{softmax}((\boldsymbol{o}_d + \boldsymbol{g})/\tau) && (13)\end{aligned}$$

where $\boldsymbol{\epsilon}$ is a sample from the uniform distribution $\mathbf{U}(0,1)$, $\boldsymbol{g}$ is a sample from the Gumbel distribution $\mathbf{Gumbel}(0,1)$, $\boldsymbol{o}_d$ is the logits computed by a MLP and a projection matrix $\mathbf{U_d} \in \mathbb{R}^{k\times n}$; $P_{real}(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z)$ is the real distribution of the predicted domain category used in the inference process, $P_{sample}(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z)$ is the sample distribution used in the training process, and τ is a temperature used to adjust the shape of the sample distribution, which is annealed during training.

In supervised learning, we use the ground-truth domain category d_y of a keyword as the supervision signal in the loss function, such that the domain constraint network can predict the target domain category as expected, which is defined as follows:

$$\mathcal{L}_2 = -\mathbf{E}_{q_\phi(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), \boldsymbol{h}_y^m, \boldsymbol{e}(d_y))}[\log P_{real}(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z)] \quad (14)$$

Another task of the domain constraint network is to compute the domain-specific score of a generated word from the domain category distribution. The domain-specific score is added to the word generation distribution in the decoder to endow a generated keyword with desirable domain-specific features. When the target domain category distribution is obtained, the target domain embedding can be computed as follows:

$$\boldsymbol{e}(d_y') = \mathbf{V_d} P(d_y' \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), z), \quad (15)$$

where $\mathbf{V_d}$ is the domain category embedding matrix as introduced in Eq. 5. Subsequently, taking the target domain embedding as input, the domain word score is generated as follows:

$$D(y_t \mid \boldsymbol{e}(d_y')) = \mathbf{W_d}\,\mathbf{MLP}(\boldsymbol{e}(d_y')), \quad (16)$$

where $D(y_t \mid \boldsymbol{e}(d_y'))$ is the domain-specific score of a generated word, which models the domain-specific features of a target keyword, $\mathbf{W_d}$ is the domain word embedding matrix.

3.3.3 Decoder Network. The decoder of DCKG incorporates the latent variable and the domain word score to generate an ad keyword. Taking as input the latent variable $\boldsymbol{z}$ and the context vector $\boldsymbol{c}_t$, the language model score of a generated word, which captures the semantic information in a keyword, is generated as follows:

$$\begin{aligned} \boldsymbol{s}_t &= \mathbf{GRU}(\boldsymbol{s}_{t-1}, [\boldsymbol{c}_{t-1}; \boldsymbol{z}; \boldsymbol{e}(y_{t-1})]), && (17)\\ S(y_t \mid y_{<t}, \boldsymbol{c}_t, z) &= \mathbf{W_s}\boldsymbol{s}_t, && (18)\end{aligned}$$

where $S(y_t \mid y_{<t}, \boldsymbol{c}_t, z)$ is the language model score and $\mathbf{W_s}$ is the semantic word embedding matrix.

Finally, the decoder combines the language model score and the domain-specific score with a factor β and then normalizes the result to the word generation distribution, which is defined as follows:

$$P(y_t \mid y_{<t}, \boldsymbol{c}_t, z, \boldsymbol{e}(d_y')) = \mathrm{softmax}(S(y_t|y_{<t}, \boldsymbol{c}_t, z) + \beta D(y_t|\boldsymbol{e}(d_y'))), \quad (19)$$

where β is a domain constraint factor that controls the influence of the domain-specific information in the final word generation distribution, and is fixed to 1.0 during supervised training. The generation loss of the decoder is given as below:

$$\mathcal{L}_3 = -\mathbf{E}_{q_\phi(z \mid \boldsymbol{h}_x^n, \boldsymbol{e}(d_x), \boldsymbol{h}_y^m, \boldsymbol{e}(d_y))}[\sum_{t=1}^{m} \log P(y_t \mid y_{<t}, \boldsymbol{c}_t, z, \boldsymbol{e}(d_y'))]. \quad (20)$$

3.3.4 *Loss Function.*

The final loss to be minimized in supervised learning is the combination of the KL divergence term $\mathcal{L}_1$, the domain prediction loss $\mathcal{L}_2$, and the generation loss $\mathcal{L}_3$:

$$\mathcal{L} = \max(\delta, \mathcal{L}_1) + \mathcal{L}_2 + \mathcal{L}_3, \tag{21}$$

where the max operation and the factor δ are used to balance the KL divergence term and other loss function for better optimization, which is known as the free bits method in [24].

3.4 Reinforcement Learning

One major disadvantage of DCKG described above is the domain constraint factor β is fixed for all the keywords in any domain. However, the optimal factor should be determined by the semantic and domain-specific information of a keyword dynamically. A lower β value leads to keywords containing less domain-specific features, while a higher value results in keywords that are less fluent or relevant but contain more domain-specific features, as shown in Section 4.4. Therefore, we propose a reinforcement learning algorithm that is able to learn different β values for different keywords.

3.4.1 *Policy Network.*

To explore suitable β values for different keywords, we first define a value space $\mathcal{B}$ which contains feasible β values. The main idea is to choose the best β value by a policy network to achieve the maximum reward with respect to the evaluation metrics, which can be implemented in three steps: first, generate a set of keywords with different β values sampled from $\mathcal{B}$, given the same source keyword and latent variable; second, obtain the reward of each keyword using the reward estimator; third, update the policy network to choose the β value that leads to a maximum reward. The policy network $\pi_\psi(\beta \mid X, z)$, parameterized by ψ, is formally given below:

$$\begin{aligned} \boldsymbol{o}_b &= [\boldsymbol{h}_x^n; \boldsymbol{e}(d_x); z; \boldsymbol{e}(d'_y)], & (22) \\ \pi_\psi(\beta \mid X, z) &= \text{softmax}(\mathbf{W_b}\,\mathbf{MLP}(\boldsymbol{o}_b)), & (23) \end{aligned}$$

where $\mathbf{W_b} \in \mathbb{R}^{b\times n}$ is a matrix projecting the input vector $\boldsymbol{o}_b$ to the action space $\mathcal{B}$.

We use the REINFORCE algorithm [40], a policy gradient method, to optimize the parameters by maximizing the expected reward of a generated keyword as follows:

$$\mathcal{J}(\psi) = \mathbb{E}_{\beta\sim\pi_\psi(\beta|X,z)}[R(X, z, \beta)], \tag{24}$$

where $R(X, z, \beta)$ is the normalized reward of a generated keyword. It is noteworthy that the policy network cannot be optimized through supervised learning, as the ground-truth β value is not observable.

3.4.2 *Reward Estimator.*

The reward estimator is designed to provide a reward balancing the domain-specific information and the semantic information for a generated keyword. To estimate the reward, given the source keyword X, the latent variable z, and the target domain category d'_y as the same input, we first sample a set of β values $\{\beta_1, \beta_2, \cdots, \beta_k\}$, and infer a set of keyword samples $\{Y_1, Y_2, \cdots, Y_k\}$ based on different β samples using the proposed model. Then, we use the domain classifier to predict the domain category d_{y_i} of each keyword Y_i. The agreement γ between d'_y and d_{y_i} is treated as an evaluation metric of the domain-specific information, which is defined as follows:

$$\gamma_i = \begin{cases} 1, & \text{if } d'_y = d_{y_i} \\ 0, & \text{if } d'_y \neq d_{y_i} \end{cases} \tag{25}$$

Subsequently, we use the generation probabilities computed by a language model and DCKG as an evaluation metric of the semantic information. Finally, the min-max normalization is applied to rescale the rewards $\{r_1, r_2, \cdots, r_k\}$ for each β_i and Y_i. This process is defined as follows:

$$\begin{aligned} r_i &= \gamma_i(\lambda P_{LM}(Y_i) + (1-\lambda)P_{DCKG}(Y_i)), & (26) \\ R_i(X, z, \beta_i) &= \frac{r_i - \min(\{r_i\})}{\max(\{r_i\}) - \min(\{r_i\})}, & (27) \end{aligned}$$

where r_i is the reward for each β_i and Y_i, γ_i is the domain category agreement, P_{LM} and P_{DCKG} are the generation probabilities modeled by a language model and our DCKG respectively, $\lambda \in (0, 1)$ is a weight to balance these two generation probabilities.

And the gradient is approximated using the likelihood ratio trick [14] as:

$$\nabla\mathcal{J}(\psi) = R(X, z, \beta)\nabla\log\pi_\psi(\beta \mid X, z), \tag{28}$$

This RL component encourages the model to choose for each keyword an optimal factor β that will lead to both higher domain category agreement and generation quality. Through reinforcement learning, we can further improve the quality of keywords generated by DCKG, resulting in more relevant, fluent, and domain-specific keywords.

4 EXPERIMENTS

4.1 Dataset

Table 1: Statistics of the dataset.

Pairs		Details	Keywords	Queries
Training	43,756,585	Number	4,419,555	19,203,972
		Length	3.96	5.40
Validation	10,000	Number	9,504	9,997
		Length	3.73	5.21
Test	10,000	Number	9,474	9,996
		Length	3.74	5.21

We sampled about 40 million query logs from Sogou.com, and each sample consists of a <*ad keyword, user query*> pair. When the search engine retrieves ads based on their owners' purchased keywords, the ad keyword must be relevant to the user query. Therefore, we treat user queries as target keywords to train our model. In other words, the input to our model is a purchased ad keyword (X), and the expected output is a user query (Y). We randomly sampled 10,000 pairs for validation and test. The statistics are presented in Table 1.

4.2 Implementation Details

Our model was implemented with Tensorflow [1]. The encoder and decoder do not share parameters and each has 4-layer GRU networks with 1024 hidden neurons in each layer. The word embedding size is set to 1,024. The vocabulary size is limited to 40,000. The MLP is implemented as a one-layer linear transformation with a tanh activation function. The parameters of the MLP and the embedding matrix share the same hidden size $n = 1,024$. All the parameters and embeddings are randomly initialized and tuned during end-to-end training.

We adopted top $k = 25$ frequent domain categories produced by the domain classifier, which is a key component used in the sponsored search engine. The domain classifier is a support vector machine [21] trained on a large human-labeled corpus. The accuracy of the 25-class classification is 92.65%. The temperature τ is set to 3.0 at the beginning, and is annealed to 0.1 during supervised learning. The factor δ is set to 5.0. The value space $\mathcal{B}$ consists of $b = 21$ values in the range $[0, 5]$ with an interval of 0.25. The generation probability factor λ is set to 0.9.

Our model is trained in two stages: supervised learning is first used to model the semantic information and the domain-specific information of keywords; reinforcement learning is then used to adaptively utilize domain-specific information in the generation process. We used the Adam optimizer with a mini-batch size of 100. The learning rate is 0.0001 for supervised learning and 0.00001 for reinforcement learning. The models were run at most 10 epochs for supervised learning and 2 epochs for reinforcement learning. The training process of each model took about a week on a Tesla V100 GPU machine.

4.3 Baselines

As aforementioned, this paper is the first work to address the domain constraint factor in ad keyword generation. We did not find closely-related baselines in the literature.

Although retrieval-based approaches proposed in previous studies are able to recommend relevant keywords from the dataset, these methods cannot generate novel keywords and thus deteriorate the bidding competition problem. Furthermore, as the dataset consists of <*ad keyword, user query*> pairs, retrieval-based methods can only suggest keywords which are already covered by the search engine. Consequently, the coverage and revenue of the search engine cannot be improved more than those methods which can generate novel keywords. Due to these reasons, we didn't consider retrieval-based methods as our baselines.

Nevertheless, we chose two suitable generative models as our baselines:

- A modified Seq2Seq model [39]. First, the target domain category is predicted by the domain constraint network and is embedded into a category vector. Then, the vector serves as input to each decoding position.
- A modified CVAE model [36] considering the domain constraint information, which is implemented similarly to DCKG but without reinforcement learning.

4.4 Automatic Evaluation

We evaluated the quality of keywords generated by different models using several automatic metrics. To evaluate the performance in metrics except for perplexity[3], we generated 10 keywords for each purchased ad keyword using different models (for DCKG and CVAE, keywords were generated conditioned on 10 sampled latent variables using greedy search; for Seq2Seq, keywords were inferred using beam search with $beamsize = 10$). Due to the uncertainty caused by the latent variable in our model and the CVAE baseline, we repeatedly tested all the models for 10 times. The averaged performance, the standard deviation, and the significance test results were reported for these models [4]. The bold number indicates the best performing model, which outperforms all other models significantly (2-tailed t-test, $p - value < 0.01$).

4.4.1 *Metrics.*

- Perplexity: We adopted perplexity to evaluate the generation quality with respect to grammar and fluency. The perplexity here is computed by the generation distribution in the models themselves.
- Perplexity$_{LM}$: We trained a language model on our dataset, and used this language model to calculate the perplexity of keywords generated by different models.
- Accuracy: To evaluate whether the model can generate domain-constrained keywords, we adopted the domain accuracy as the agreement between the expected domain category (as predicted by the model, see Eq. 12) and the domain category of a generated keyword predicted by the domain classifier.
- Distinct-*n*: We calculated the proportion of distinct *n*-grams to all the *n*-grams in generated keywords to evaluate the diversity.
- Novelty$_{ALL/AD}$-n: To evaluate the novelty of the generated keywords, we counted the distinct *n*-grams that don't appear in all the corpus (ALL) or not in the purchased ad keyword set (AD). Novelty$_{ALL}$-n means the percentage of totally novel *n*-grams that have not been observed in the corpus.

4.4.2 *Results.*

The results are shown in Table 2 and Table 3, from which we made the following observations:

First, DCKG obtains the best performance in domain accuracy, distinct, and novelty. The improvement in domain accuracy shows that our model can enhance the domain consistency of the generated keywords, thanks to the reward we applied in reinforcement learning. DCKG can generate more diverse and novel keywords than baselines, which can be attributed to the latent variable network we adopted and the domain constraint we considered. Our model has better perplexity and perplexity$_{LM}$ than Seq2Seq but slightly worse than CVAE, because DCKG is tuned to satisfy the domain constraint more by reinforcement learning than fit to the language generation probability. Nevertheless, the disadvantage in perplexity and perplexity$_{LM}$ harms the grammaticality of the generated keywords but less influences the overall performance, as shown in the following experiments.

Second, the novelty scores indicate that our model can generate more novel *n*-grams than the baselines. The generative models can generate novel keywords which have not been purchased yet so

[3]We generated the ground-truth target keyword using greedy search for calculating the perplexity.

[4]For the deterministic Seq2Seq model, we tested 10 models which achieve the top 10 performance in the validation set. These models differ in different training steps.

Table 2: Automatic evaluation.

Model	Perplexity		Perplexity$_{LM}$		Accuracy		Distinct-2		Distinct-3		Distinct-4	
	avg.	std.	avg.	std.	avg.(%)	std.(%)	avg.(%)	std.(%)	avg.(%)	std.(%)	avg.(%)	std.(%)
Seq2Seq	18.82	0.033	12.26	0.022	76.84	0.054	39.09	0.028	46.78	0.033	53.02	0.041
CVAE	**10.76**	0.021	**8.24**	0.023	77.40	0.057	69.58	0.051	83.04	0.060	90.32	0.065
DCKG	11.21	0.025	8.94	0.034	**84.61**	0.080	**71.07**	0.077	**84.26**	0.065	**91.26**	0.051

Table 3: Novelty evaluation.

Model	Novelty$_{ALL}$-2		Novelty$_{ALL}$-3		Novelty$_{ALL}$-4		Novelty$_{AD}$-2		Novelty$_{AD}$-3		Novelty$_{AD}$-4	
	avg.	std.	avg.	std.	avg.(%)	std.(%)	avg.(%)	std.(%)	avg.(%)	std.(%)	avg.(%)	std.(%)
Seq2Seq	0.54	0.005	5.33	0.015	15.53	0.020	2.65	0.010	12.40	0.026	24.73	0.019
CVAE	2.01	0.016	16.03	0.058	38.38	0.119	8.18	0.024	33.49	0.085	57.64	0.099
DCKG	**2.60**	0.021	**18.23**	0.056	**41.48**	0.114	**9.34**	0.035	**36.01**	0.095	**60.40**	0.106

that the bidding competition problem can be alleviated as aforementioned. For retrieval-based methods, the Novelty$_{ALL}$ scores, which count n-grams that are never observed in the corpus, will be zero since they can only suggest existing keywords from the corpus.

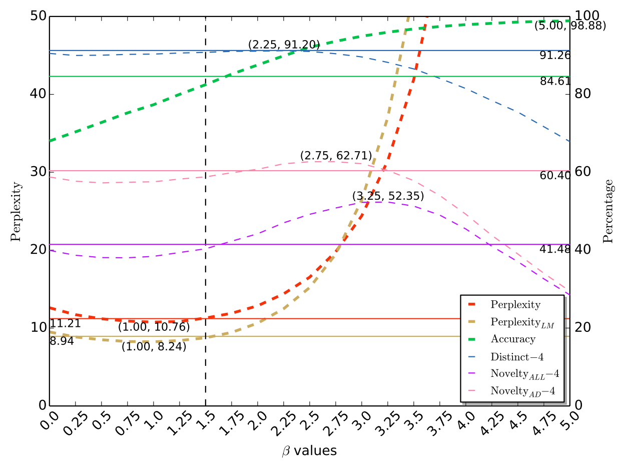


Figure 4: Performance curve with varying β. The dashed curve is the performance of DCKG with manually fixed β in different metrics. The bold curves (perplexity, perplexity$_{LM}$, accuracy) show the change of rewards which are used reinforcement learning (see Eq. 26). The horizontal line is the performance of DCKG with dynamic β tuned by RL. The dashed vertical line is an auxiliary line to compare DCKG with fixed β = 1.5 to DCKG with dynamic β optimized by RL. Left axis: Perplexity and Perplexity$_{LM}$. Right axis: Accuracy, Distinct-4, Novelty$_{ALL}$-4, and Novelty$_{AD}$-4.

4.4.3 *Effect of β.* In order to investigate the influence of the domain constraint factor β in DCKG, we tested the performance of DCKG with different fixed $\beta \in \mathcal{B}$. The results are shown in Figure 4. As we can see, the domain accuracy of DCKG improves continuously with the increase of β (the green curve), indicating that β is critical to satisfying the domain constraint in keyword generation because larger β applies more domain-specific information in keyword generation. DCKG achieves the best performance in perplexity and perplexity$_{LM}$ with $\beta = 1$ since this value is the default β value during supervised learning. However, when β becomes larger, DCKG performs worse in the perplexity metrics. It is thus important to balance the domain constraint and the language generation quality. Therefore, we apply reinforcement learning to find an optimal β for different instances dynamically.

It is noteworthy that DCKG with the dynamically determined β outperforms DCKG with the manually adjusted β. For example, although the performance of DCKG in perplexity and perplexity$_{LM}$ is approximately equal to DCKG with the fixed $\beta = 1.5$, the domain accuracy of DCKG outperforms DCKG with the fixed $\beta = 1.5$ (**84.61** vs. 82.49). Moreover, the novelty score and the diversity score of DCKG are also better than DCKG with the fixed $\beta = 1.5$, for instance, the Distinct-4 of DCKG is better than DCKG with any fixed $\beta \in \mathcal{B}$ (**91.26** vs. 91.20), indicating that the domain constraint can further improve the quality of generated keywords.

4.5 Manual Evaluation

In manual evaluation, we tested whether the generated keywords are relevant to the source keyword and consistent with the expected domains, such that they are suitable to be suggested to advertisers. We manually evaluated the relevance and the grammaticality of keywords generated by different models. We randomly sampled 100 purchased ad keywords from the test set, and generated 10 keywords for each purchased ad keyword using different models as in automatic evaluation. Five experienced annotators were recruited to judge the relevance (*Relevant/Irrelevant*) of 3000 pairs of generated and purchased keywords. The grammaticality of a generated keyword is also considered during annotation, as bad grammaticality leads to the ambiguity in the meaning and thus results in irrelevance rating. We used majority voting to obtain the final label of annotation. Moreover, as domain accuracy is critical to measuring how well the models satisfy domain constraint, we also reported the automatic measure in these tables. The bold number indicates the best performing model, which outperforms all other models significantly (2-tailed sign test, $p - value < 0.05$).

4.5.1 *Metrics.*

- Precision (P): Precision is defined as the ratio of the number of relevant keywords generated to the number of total keywords generated.
- Recall (R): Recall is the proportion of relevant keywords generated by a model to all the relevant keywords pooled from all models' results. Similar to [23], the relevant keywords from all the models are pooled together and are treated as the entire relevant set. This metric is useful to compare the ability to cover the potential relevant keywords.
- Accuracy (A): Similar to the preceding section, this metric is the agreement between the expected domain category, predicted by the model, and the domain category of a generated keyword, predicted by the domain classifier (SVM).
- F-measures: We adopted four F-measures to quantify the overall performance including F(PR), F(PA), F(RA), and F(PRA). Each F-measure is defined as the harmonic mean of all factors, where P/R/A indicate precision/recall/accuracy respectively.

Table 4: Manual evaluation.

Model	Seq2Seq	CVAE	DCKG
Precision	0.948	0.954	**0.970**
Recall	0.325	0.332	**0.343**
Accuracy	0.761	0.774	**0.847**
F(PR)	0.483	0.491	**0.503**
F(PA)	0.808	0.832	**0.892**
F(RA)	0.440	0.454	**0.480**
F(PRA)	0.534	0.548	**0.574**

4.5.2 Annotation Statistics. We calculated the statistics to measure inter-rater agreement. For relevance rating, the percentage of the pairs that all 5 judges gave the same label is 85.1%, the percentage of the pairs that 4 judges gave the same label (4/5 agreement) amounts to 8.8%, and the percentage for 3/5 agreement is 6.1%. We further computed the free-marginal kappa [31] to measure inter-rater consistency. The free-marginal kappa is 0.856, indicating adequate inter-rater agreement.

4.5.3 Results. The results are shown in Table 4. As it can be seen, DCKG obtains the best performance in all the metrics, indicating that our model can generate relevant and grammatical keywords. The best precision shows that more relevant keywords are generated by DCKG. Our model also achieves the best recall, indicating that DCKG is able to cover more potential user queries than other models. In addition, DCKG outperforms other models in domain accuracy, indicating that incorporating the domain constraint appropriately can enhance the quality of generated keywords. Moreover, DCKG ranks highest for all the four F-measures, showing that our model is superior considering all these factors.

4.6 Online Evaluation

In online evaluation, we examined whether the sponsored search engine can retrieve and present more relevant ads using the keywords generated by our model. We applied the generative models in the keyword reformulation method to facilitate the ad retrieval process of sponsored search (see Figure 1). Specifically, we collected 5 million purchased ad keywords, and then used each model to generate 10 keywords for each purchased keyword. We used these generated keywords as the reformulated keywords to retrieve original ads. The original keyword reformulation method, based on handcrafted rules and templates, is one of the retrieval methods in the ad retrieval process of the sponsored search engine, which retrieves about 2% unique ads. We added the keywords generated by our models to the keyword reformulation method of the sponsored search engine, which is called the enhanced keyword reformulation method. To make a fair comparison, the enhanced keyword reformulation methods of all models run ten days in our A/B online test system on Sogou.com, where 10% user queries are selected into the test system. We compared the performance of the enhanced keyword reformulation method with the original keyword reformulation method, and reported the relative gain in percentage for all the metrics as the final result. The bold number indicates the best performing model, which outperforms all other models significantly (2-tailed t-test, $p - value < 0.05$).

4.6.1 *Metrics.*

- Coverage (COV): Coverage is defined as the percentage of web pages that contain at least one ad.
- Click-through rate (CTR): Click-through rate is the ratio of page views that lead to a click to the total number of page views.
- Revenue per mille (RPM): Revenue per mille is the revenue of the search engine per one thousand page views.

4.6.2 Results. The results are shown in Table 5. As it can be seen, DCKG obtains the best performance in all the metrics. Results show that keywords generated by DCKG can cover more user queries than those by other models. The click-through rate scores indicate that the new ads which are matched with ad keywords from DCKG lead to more user clicks, because the generated keywords (user queries) are of higher quality and more relevant. The revenue is improved because the click-through rate increases, which verifies that our model can contribute more ad impressions in sponsored search.

Although the Seq2Seq model can also retrieve more ads, the improvements in all metrics are the lowest in the three models, as the diversity and the quality of the generated keywords are lower than ours. CVAE outperforms Seq2Seq because CVAE is able to model diversity with latent variables. However, as CVAE incorporate the domain constraint in a fixed way, it cannot balance the semantic information and the domain-specific information in the generation process. DCKG further improves the diversity and the quality of generated keywords by tuning the β factor dynamically with reinforcement learning. The results of online evaluation agree with those of automatic evaluation, indicating that DCKG can generate the most diverse, novel and domain consistent keywords among the three models.

4.7 Case Study

Several sample keywords generated by different models are shown in Table 6. We can see that keywords generated by Seq2Seq have

Table 5: **Online evaluation. The number reported in each metric is the increased proportion compared to the original keyword reformulation method of the sponsored search engine, which retrieves about 2% unique ads.**

Day	Seq2Seq			CVAE			DCKG		
	COV(%)	CTR(%)	RPM(%)	COV(%)	CTR(%)	RPM(%)	COV(%)	CTR(%)	RPM(%)
1	39.51	44.92	21.96	53.64	59.14	24.86	58.52	63.26	30.52
2	44.30	49.99	24.99	58.22	60.38	28.49	55.16	60.88	33.49
3	46.38	50.02	22.52	51.53	54.04	30.30	58.07	58.78	28.24
4	47.32	46.22	16.78	51.12	57.54	34.64	58.24	61.68	28.23
5	46.41	50.25	20.56	49.82	52.58	21.72	59.40	60.85	33.78
6	45.27	47.06	17.36	51.61	51.53	24.95	55.20	58.17	21.68
7	41.09	45.37	26.00	51.54	55.39	24.12	54.36	59.58	24.81
8	36.90	49.98	17.42	51.53	55.96	23.15	53.38	58.29	27.41
9	35.89	42.85	19.25	51.92	61.53	27.00	50.60	63.24	37.75
10	39.46	49.36	25.07	56.17	60.54	28.06	53.86	57.61	22.06
avg.	42.25	47.60	21.19	52.71	56.86	26.73	**55.68**	**60.23**	**28.80**

Table 6: Sample keywords generated by different models with domain accuracy (Acc.) and relevance (Rel.).

Model	Domain		Keyword		Acc.	Rel.
	Original	Translated	Original	Translated		
Source	美容保健	beauty care	减肥减肥	weight loss weight loss	-	-
Seq2Seq	美容保健	beauty care	抖音上的减肥的舞蹈视频	dance video on weight loss	0	1
	美容保健	beauty care	抖音上的减肥的舞蹈视频教程	dance video tutorial on weight loss	1	1
	美容保健	beauty care	减肥的人可以吃什么水果不容易	what fruit is not easy for people who want to lose weight	1	0
	美容保健	beauty care	减肥的人可以吃什么水果不容易胖	what fruit is not easy to fat for people who want to lose weight	1	1
CVAE	信息	information	减肥的时候吃的热量低的食物	low calorie food when losing weight	1	1
	信息	information	减肥吃什么肉会胖	what meat will lead to fat if I eat during weight loss	0	1
	美容保健	beauty care	减肥期间吃什么主食最好	what is the best staple food during weight loss	1	1
	电脑网络	Internet	一天中什么时候减肥最好	when is the best time to lose weight in a day	0	1
DCKG	美容保健	beauty care	减肥瑜伽视频	weight loss yoga video	1	1
	美容保健	beauty care	减肥小妙招	weight loss trick	1	1
	美容保健	beauty care	减肥能吃荔枝吗	can I eat litchis during weight loss	1	1
	信息	information	喝红豆薏米减肥吗	is drinking red bean glutinous rice helpful to lose weight	1	1

lower diversity than other models since it tends to generate keywords with the same prefix in similar meanings using beam search [26], such as the 3rd and 4th keywords generated by Seq2Seq. Moreover, it can only predict one target domain category since it is a deterministic model, which cannot handle the task of generating different keywords from multi-domain categories.

In comparison, CVAE and DCKG are able to predict multiple domain categories as well as generate diverse keywords according to these domain categories. However, in CVAE, the domain constraint is not addressed properly in some cases, as it takes a fixed way to combine the domain-specific information with the semantic information. For example, although the 6th generated keyword, "what meat will lead to fat if I eat during weight loss", is relevant to the source keyword, it is inconsistent with its domain category "information" since the domain-specific information is less considered than the semantic information in this case. By contrast, DCKG is able to choose a proper domain constraint factor to balance the domain-specific information and the semantic information for each instance. Hence, the generated keywords are both relevant to the source keyword and consistent with their domain categories.

In order to validate the influence of the domain constraint factor β in the generated keywords, we presented several sample keywords generated by DCKG with different β values in Table 7. Take the 6th keyword in Table 6 as an example. Given the source keywords "weight loss weight loss" and its domain category "beauty care", our model predicts a target domain category "information", and then generates keywords with different β values while other parameters are fixed.

Table 7: Sample keywords conditioned on different β with domain accuracy (Acc.) and relevance (Rel.).

β	Keyword		Acc.	Rel.
0.00	Original	减肥瘦身应该吃什么	0	1
	Translated	what should I eat to lose weight and slim		
1.00	Original	减肥吃什么肉会胖	0	1
	Translated	what meat will lead to fat if I eat during weight loss		
2.00	Original	减肥吃牛肉有什么好处	1	1
	Translated	the benefits of eating beef during weight loss		
3.00	Original	减肥吃牛肉有什么好处	1	1
	Translated	the benefits of eating beef during weight loss		
4.00	Original	减肥公司调查报告范文	1	0
	Translated	weight loss company survey report essay		
5.00	Original	减肥公司调查调查信息	1	0
	Translated	weight loss company survey survey information		

As shown in Table 6, when β equals to 0.00, the target keyword belongs to the source domain category "beauty care" but violates the target domain category "information" since the keyword is generated regardless of the domain-specific information. With the increase of β, the domain accuracy improves since more domain-specific information is involved in the model, but the relevance decreases in some instances because the model faces with the compatibility of controlling semantic and domain-specific information. For example, the keywords generated when $\beta = 2.00/3.00$, are both consistent to the target domain category and relevant to the source keyword. However, when β increases to 4.00/5.00, the generated keywords are not relevant to the source keyword anymore, as a high β value will harm the quality of generated keywords severely.

5 CONCLUSION

In this paper, we investigate the use of generative neural networks for ad keyword generation. We propose a Domain-Constrained Keyword Generator (DCKG) to generate diversified and domain-consistent keywords. A latent variable network is designed to model the diversity and a domain constraint network to satisfy the domain constraint in supervised learning. To further leverage the domain-specific information in keyword generation, we propose a reinforcement learning algorithm to adaptively utilize domain-specific information in keyword generation. Offline evaluation shows that the proposed model can generate keywords that are diverse, novel, relevant to the source keyword, and accordant with the domain constraint. Online evaluation shows that generative models can improve coverage (COV), click-through rate (CTR), and revenue per mille (RPM) substantially in sponsored search.

ACKNOWLEDGMENTS

This work was jointly supported by the National Science Foundation of China (Grant No.61876096/61332007), and the National Key R&D Program of China (Grant No. 2018YFC0830200).

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR* abs/1603.04467 (2016).

[2] Vibhanshu Abhishek and Kartik Hosanagar. 2007. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the ninth international conference on Electronic commerce.* ACM, 89–94.

[3] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. 2006. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM conference on Electronic commerce.* ACM, 1–7.

[4] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. 2008. Simrank++: query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment* 1, 1 (2008), 408–421.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473 (2014).

[6] Lidong Bing, Wai Lam, Tak-Lam Wong, and Shoaib Jameel. 2015. Web query reformulation via joint modeling of latent topic dependency and term context. *ACM Transactions on Information Systems (TOIS)* 33, 2 (2015), 6.

[7] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2018. Attention-based Hierarchical Neural Query Suggestion. *arXiv preprint arXiv:1805.02816* (2018).

[8] Yifan Chen, Gui-Rong Xue, and Yong Yu. 2008. Advertising keyword suggestion based on concept hierarchy. In *Proceedings of the 2008 international conference on web search and data mining.* ACM, 251–260.

[9] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. 1724–1734.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555 (2014).

[11] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web.* ACM, 325–332.

[12] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *Proceedings of the 14th ACM international conference on Information and knowledge management.* ACM, 696–703.

[13] Ariel Fuxman, Panayiotis Tsaparas, Kannan Achan, and Rakesh Agrawal. 2008. Using the wisdom of the crowds for keyword generation. In *Proceedings of the 17th international conference on World Wide Web.* ACM, 61–70.

[14] Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM* 33, 10 (1990), 75–84.

[15] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval.* ACM, 383–392.

[16] Emil Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series* 33 (1954).

[17] Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on.* IEEE, 1443–1454.

[18] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* ACM, 1443–1452.

[19] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[20] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.* ACM, 445–454.

[21] Thorsten Joachims. 1998. *Making large-scale SVM learning practical.* Technical Report. Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

[22] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web.* ACM, 387–396.

[23] Amruta Joshi and Rajeev Motwani. 2006. Keyword generation for search engine advertising. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on.* IEEE, 490–496.

[24] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems.* 4743–4751.

[25] Kenneth Wai-Ting Leung, Wilfred Ng, and Dik Lun Lee. 2008. Personalized concept-based clustering of search engine queries. *IEEE transactions on knowledge and data engineering* 20, 11 (2008), 1505–1518.

[26] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL*. 110–119.

[27] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A Simple, Fast Diverse Decoding Algorithm for Neural Generation. *CoRR* abs/1611.08562 (2016).

[28] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. 3086–3094.

[29] Robert K Merton. 1968. The Matthew effect in science: The reward and communication systems of science are considered. *Science* 159, 3810 (1968), 56–63.

[30] Dandan Qiao and Jin Zhang. 2015. A Novel Keyword Suggestion Method to Achieve Competitive Advertising on Search Engines.. In *PACIS*. 142.

[31] Justus J Randolph. 2005. Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa. *Online submission* (2005).

[32] Sujith Ravi, Andrei Broder, Evgeniy Gabrilovich, Vanja Josifovski, Sandeep Pandey, and Bo Pang. 2010. Automatic generation of bid phrases for online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 341–350.

[33] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues.. In *AAAI*. 3295–3301.

[34] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *ACL*. 1577–1586.

[35] Marc Sloan, Hui Yang, and Jun Wang. 2015. A term-based methodology for query reformulation understanding. *Information Retrieval Journal* 18, 2 (2015), 145–165.

[36] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. 3483–3491.

[37] Hyun-Je Song, A Kim, Seong-Bae Park, et al. 2017. Translation of Natural Language Query Into Keyword Query Using a RNN Encoder-Decoder. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 965–968.

[38] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 553–562.

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. 3104–3112.

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

[41] Shuangfei Zhai, Keng-hao Chang, Ruofei Zhang, and Zhongfei Mark Zhang. 2016. Deepintent: Learning attentions for online advertising with recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1295–1304.

[42] Wei Vivian Zhang and Rosie Jones. 2007. Comparing click logs and editorial labels for training query rewriting. In *WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges*.

[43] Ying Zhang, Weinan Zhang, Bin Gao, Xiaojie Yuan, and Tie-Yan Liu. 2014. Bid keyword suggestion in sponsored search based on competitiveness and relevance. *Information Processing & Management* 50, 4 (2014), 508–523.

[44] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 654–664.

[45] Xianda Zhou and William Yang Wang. 2017. MojiTalk: Generating Emotional Responses at Scale. *arXiv preprint arXiv:1711.04090* (2017).