An Empirical Study of Generating Texts for Search Engine Advertising

Hidetaka Kamigaito[*1], **Zhang Peinan**[*2], **Hiroya Takamura**[3] **and Manabu Okumura**[1]
[1]Tokyo Institute of Technology, [2]CyberAgent, Inc.
[3]National Institute of Advanced Industrial Science and Technology (AIST)
kamigaito@lr.pi.titech.ac.jp, zhang_peinan@cyberagent.co.jp
takamura.hiroya@aist.go.jp, oku@lr.pi.titech.ac.jp

Abstract

Although there are many studies on neural language generation (NLG), few trials are put into practice, particularly in the advertising domain. Generating ads with NLG models can help copywriters in their creation. However, few studies have adequately evaluated the effect of generated ads with actual serving because this requires a large amount of training data and a particular environment. In this study, we demonstrate a practical use case of generating ad-text with an NLG model. Particularly, we show how to improve the ads' impact, deploy models to a product, and evaluate the generated ads.

1 Introduction

Search engine advertising (SEA) displays ads relevant to the queries that users have searched on search engines as a part of the search results. On the ad creation side, ad copywriters develop keywords and ad-texts that are likely to be searched by users, and ad-texts are attractive to users based on the keywords and landing page (LP) contents. Advertisers or advertising agencies then submit these keywords and ad-texts to the ad delivery service. After submission, if the user's search queries and the submitted keywords match, the service selects an ad-text from the submissions through an auction and displays it to the user. SEA plays an important role in the advertising market because it can mutually satisfy users' and advertisers' respective demands. Advertising agencies have many copywriters on staff; however, manual creation will eventually reach its limit with the ever-increasing content. Therefore, auto-generating ads are expected to be a great support for ad creation.

Fill templates with words and phrases extracted from web search results or LPs are commonly used in ad-text generation (Bartz et al. (2008), Fujita et al. (2010), Fujita et al. (2011), Thomaidou et al. (2013)). These approaches create limited ad-text because they strongly rely on a pre-built list of templates or an LP containing ad-related texts. Hughes et al. (2019) proposed a method to incorporate a reinforcement learning (RL) framework into an end-to-end sequence-to-sequence (Seq2Seq) model for generating effective search engine ad-text considering feedback regarding ad effectiveness from ad delivery services.

Input	**LP Content**: Introducing popular insurance services in a ranking format! You can compare insurance services in the largest domestic comparison site of insurance A operated by B in various forms, such as by the order of request for information and the order of application by type of insurance. **Keyword**: [insurance, comparison] **Query**: [comparison of insurance services]
Output	**Title**: Which insurance is the best deal? Latest rankings. **Description**: This is the largest insurance comparison site. It is recommended for you from the top of the ranking order.

Table 1: Example of an input and its ad-text.

However, unlike typical natural language generation tasks, ad-texts are determined from input and characteristics such as previous ad delivery performance, the contexts in overall ads, and the relevance between the search queries and the results. Diversity is also an importance factor in this task because readers can be bored if a model generates ad-text that has already been used. Therefore, to make ad-texts more attractive, the model must generate ad-texts that were not used previously. In addition, the model must consider the ad-text's length because ads are presented in a limited space, which imposes a limitation on the length in practical usage. The addition of important keywords in the ad-text is also necessary to enhance user engagement because the search result page highlights the searched keyword in the user query and the ad-text. Furthermore, few ad-text generation models are used in real-world applications notwithstanding the commercial domain of advertising. Therefore, few studies have evaluated all end-to-end processes from generation to actual delivery.

Considering these requirements, we propose a

[*]Equal contribution.

Proceedings of NAACL HLT 2021: IndustryTrack Papers, pages 255–262
June 6–11, 2021.

method for generating ad-text by utilizing RL with rewards. This approach enables using ad-texts not included in the original training dataset through sampling from the training model. Therefore, we can expect improvements in diversity for generated ad-texts. We compared models specifically constructed for ad-text generation to determine the important factors for this task. We investigated our models in a real-world application, and performed an ad-delivery evaluation. The evaluation results showed that the proposed method improved the both human-rated attractiveness and relevance scores and the ad-delivery results compared to other approaches, while also maintaining diversity of the generated ad-texts.

Our contributions are as follows.

- We present a case study of ad-text generation as a real-world application. This study confirmed that the automatic generation of ad-texts using the RL-based encoder-decoder model is effective in actual advertisement creation.
- We propose a method for generating ad-text that utilizes RL with rewards to improve advertising performance. We performed automated evaluations on different types of metrics, as well as human evaluations involving crowdsourcing workers and professional copywriters. The results showed the usefulness of the methods.
- We describe how to incorporate our model into an ad-delivery service and performed an online evaluation to compare the performance of ad-texts generated by the model with traditional ads written by a human.

2 Generating Ad-text with RL

As shown in Table 1, Seq2Seq generates ad-texts from the given keywords and contents of LPs. Note that we concatenated these elements as sequences for each side by a separator symbol <SEP>. Figure 1 presents an overview of the proposed ad-text generation method. We use a model proposed by Paulus et al. (2018) as our Seq2Seq. In our method, Seq2Seq is trained using RL to capture useful features for generating effective ad-texts. In Sections 2.1 and 2.2, we describe each part of the proposed ad-text generation method.

2.1 RL for Seq2seq

In Seq2Seq, for the input sequence $\mathbf{x} = x_1 \cdots x_n$, the ad-text $\mathbf{y} = y_1 \cdots y_m$ is generated by maximizing the output probability, which is calculated by the following formula:

$$\mathbf{y} = \arg\max_{\mathbf{y}} \prod_{t=1}^{m} P(y_t|\mathbf{x}, y_{t-1} \cdots y_1). \quad (1)$$

For training Seq2Seq in consideration of the characteristics of an effective ad, we use RL in training. Because considering all possible outputs in the decoder is intractable, we use an approach based on a policy gradient method called self-critical sequence training (SCST) (Rennie et al., 2017), which can train models on sampled output sequences. In SCST, using $\mathbf{y}^s = y_1^s \cdots y_t^s$ (a sequence sampled from Seq2Seq) and $\hat{\mathbf{y}} = \hat{y}_1 \cdots \hat{y}_t$ (a sequence obtained by greedy decoding), the RL loss L_{rl} is calculated as follows:

$$L_{rl} = (r(\hat{\mathbf{y}}) - r(\mathbf{y}^s)) \sum_{t=1}^{m} logP(y_t^s|y_{t-1}^s \cdots y_1^s, \mathbf{x}), \quad (2)$$

where $r(\hat{\mathbf{y}})$ and $r(\mathbf{y}^s)$ are the rewards of $\hat{\mathbf{y}}$ and $\mathbf{y}^s$, respectively. It is difficult to optimize the model using only RL owing to its instability. We must, therefore, use maximum likelihood estimation (MLE), which maximizes the probability of the reference sequence $y_t^\star \cdots y_1^\star$ as follows:

$$L_{mle} = -\sum_{t=1}^{m} logP(y_t^\star|y_{t-1}^\star \cdots y_1^\star, \mathbf{x}). \quad (3)$$

Considering L_{rl} and L_{mle}, our final loss function L_{mix} is defined as follows:

$$L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{mle}, \quad (4)$$

where γ is a hyperparameter weighting the importance of L_{rl}.

2.2 Rewards

To explicitly capture the characteristics of effective ad-text, we use the following three rewards: fluency, relevance, and ad quality. These rewards are summed and incorporated in the loss function of Eq.(2) to enhance the effectiveness of the ad. Thus, the reward for the generated text $\mathbf{y}$ is calculated as follows:

$$r(\mathbf{y}) = r^F(\mathbf{y}) + r^R(\mathbf{x}, \mathbf{y}) + r^Q(\mathbf{x}, \mathbf{y}), \quad (5)$$

where $r^F(\mathbf{y})$ is a reward for fluency, $r^R(\mathbf{y})$ is a reward for relevance, and $r^Q(\mathbf{y})$ is a reward for ad quality. In Sections 2.2.1 through 2.2.3, we discuss these rewards in detail.

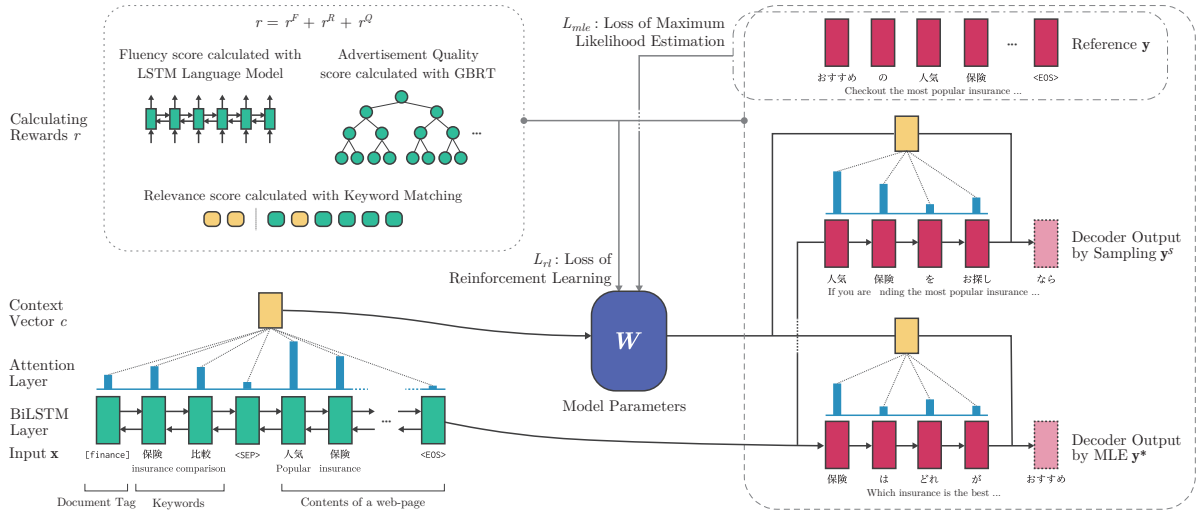

Figure 1: Overview of proposed ad-text generation method

2.2.1 Fluency

Fluency is an essential factor in generating natural language texts. In addition, the length limitation of the text must be considered, as space for advertising is limited. If the ad-text is truncated owing to space limitations, its fluency is significantly degraded. To address these problems, our fluency reward consists of two types of scores as follows:

$$r^F = s^{LM}(\mathbf{y}) + s^{Len}(\mathbf{y}), \tag{6}$$

where $s^{LM}(\mathbf{y})$ is a grammatical score and $s^{Len}(\mathbf{y})$ scores the fidelity of $|\mathbf{y}|$ to the given desired length. We use the function described in Eq. (10) of Zhang and Lapata (2017) as the first score $s^{LM}(\mathbf{y})$.

The second score, $s^{Len}(\mathbf{y})$, measures the appropriateness of the length of the generated text. The length of the generated text must not exceed the length limit. However, to maintain informativeness, it should not be significantly shorter than the limit. We incorporate these factors into s^{Len}. Let $\mathbf{y}_{title}$ be the title part of $\mathbf{y}$, $\mathbf{y}_{desc}$ be the description part of $\mathbf{y}$, C_{title} be the length limit of the title part, C_{desc} be the length limit of the description part, and s^l be a score function for each part of the generated text. The score s^{Len} is calculated as follows:

$$s^{Len}(\mathbf{y}) = \frac{s^l(\mathbf{y}_{title}, C_{title}) + s^l(\mathbf{y}_{desc}, C_{desc})}{2}, \tag{7}$$

where $s^l(\bar{\mathbf{y}}, C)$ is a function that returns $\exp(|\bar{\mathbf{y}}| - C)$ when $|\bar{\mathbf{y}}| \leq C$, whereas it returns 0 when $|\bar{\mathbf{y}}| > C$.

2.2.2 Relevance

Effective ad-text is generally consistent with what it advertises. Therefore, we consider the relevance between the input text and output ad-text as a reward. In ad-text generation, the input text includes important keywords that should be emphasized in the generated ad-text. For the generated ad-text to be relevant to the input, the ad-text should contain keywords from the input text as important words[1]. Therefore, we focus on the use of important keywords in the generated ad-text for building a reward to measure relevance. In addition to the coverage of keywords, the positions of keywords in the generated ad-text are also important, because keywords should appear at the beginning of ad-text. Considering these factors, we calculate $r^R(\mathbf{x}, \mathbf{y})$, the reward of the relevance for input $\mathbf{x}$ and the generated ad-text $\mathbf{y}$ as follows:

$$r^R(\mathbf{x}, \mathbf{y}) = s^{cov}(\mathbf{x}, \mathbf{y}) + s^{pos}(\mathbf{x}, \mathbf{y}), \tag{8}$$

where $s^{cov}(\mathbf{x}, \mathbf{y})$ scores the coverage of keywords in $\mathbf{x}$ and $s^{pos}(\mathbf{x}, \mathbf{y})$ scores the position of keywords in $\mathbf{y}$. The first score $s^{cov}(\mathbf{x}, \mathbf{y})$ calculates the proportion of keywords in $\mathbf{x}$ that are covered by $\mathbf{y}$. The second score $s^{pos}(\mathbf{x}, \mathbf{y})$ calculates the average position of keywords in $\mathbf{y}$. To prevent this reward from reducing the coverage of keywords, we impose the length of the generated ad-text as a score of a keyword not included in the ad-text. Then,

[1] Actually, Google Ads recommends to include at least one of advertisement keywords as described in the support page: https://support.google.com/google-ads/answer/1704392?hl=en

$s^{pos}(\mathbf{x}, \mathbf{y})$ is calculated as follows:

$$s^{pos}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{|K|}\sum_{k \in K} s^{p}(k, \mathbf{y})\right), \quad (9)$$

where K is a set of keywords included in $\mathbf{x}$ and $s^{p}(k, \mathbf{y})$ is a function that returns a character-based position of k in $\mathbf{y}$ if $k \in \mathbf{y}$, whereas it returns $|\mathbf{y}|$ if $k \notin \mathbf{y}$.

2.2.3 Ad Quality

Ad quality score (QS) is an important factor indicating ad-text quality based on the delivery performance and keyword relevance. Because QS is reported by the ad delivery service as the effectiveness of the delivered ad-text, we cannot use it directly to evaluate the generated ad-text. Thus, we predict the QS from $\mathbf{x}$ and $\mathbf{y}$ in accordance with previous research (Hughes et al., 2019). We treat the predicted QS as $r^{Q}(\mathbf{x}, \mathbf{y})$, the reward of the ad-text's quality. To train a classifier to predict QS from the given $\mathbf{x}$ and $\mathbf{y}$, we prepare an ad dataset from our ad-text databases. This dataset consists of a title, description, and score that represent the quality of each ad. The QS ranges from 1 to 10, where a lower score corresponds to lower ad quality and a higher score indicates higher ad quality.

We develop a simple regression model to predict the QS from the title and description. In the model, the title and description are joined and encoded into embeddings by fastText (Bojanowski et al., 2017) and max-pooling of simple word embedding (SWEM) (Shen et al., 2018). After the encoding, we use the gradient boosting regression tree (GBRT) (Ke et al., 2017) to predict the QS. We developed a simple model because that the predicted quality score is used as a reward in RL, which requires a large computational time; therefore, efforts should be made to shorten it.

3 Experimental Settings

3.1 Dataset

We prepared a dataset that contained pairs consisting of an input and ad-text in Japanese for training and evaluating our models. This dataset consisted of eight clients (one real estate company, one health food company, one media service company, two cosmetic companies, one job recruiting company, and three financial companies). We carefully split the dataset into 713,928, 8,000, and 8,000 pairs for training, development, and the test set, respectively. In this dataset, we used the meta-description as the content of the LP. In addition to the ad-texts, we prepared Japanese Wikipedia articles[2] to pretrain the language model. The fine-tuning of LM was performed on the ad-text dataset. All texts in this dataset were tokenized by MeCab[3] with the Neologd dictionary (Sato et al., 2017).

3.2 Models

The baselines are as follows:

Separated (**Sep**): This baseline trains the models for different clients separately. Therefore, it is necessary to build as many models as the number of clients, which can be significantly high. In addition, the diversity of the ad-text generated from this model was considerably low. Such unpractical features are not suitable for automatic ad-text generation; however, this model can generate ad-text that is highly similar to the given dataset. Therefore, we treated this model as a type of upper limit for generating ad-texts.

Mixed without Domain Tags (**Mix w/o tag**): This model was trained on the entire dataset with the MLE loss of Eq. (3).

Mixed (**Mix**): Similar to **Mix w/o tag**, this model was trained on the entire dataset with the MLE loss of Eq. (3). However, in this model, we included additional labels to identify the domain and the client of the input in both the training and the prediction steps. Particularly, we added the tags `<domain_id>` and `<client_id>` to the beginning of the input.

The methods employed are as follows:

Mixed with Rewards (**Mix+[Rewards]**): This model was trained on the mixed loss function, which is a combination of the RL loss and MLE loss, as defined in Eq. (4). The data format used in this model is similar to that in the **Mixed** model. We combined several rewards: reward of the fluency r^{F} in Eq. (6) (**Flu**), the reward of the relevance r^{R} in Eq. (8) (**Rel**), and reward of the ad quality r^{Q} discussed in Section 2.2.3 (**QS**). The combinations of **Flu**, **Rel**, and **QS** are represented by the + symbol. It should be noted that **Mix+QS** is the model proposed by Hughes et al. (2019); thus, it is categorized as a baseline.

Table 2 presents the parameter settings. We adapted to the settings used in previous research (Paulus et al., 2018) and choose λ as 0.98 for the re-

[2] https://dumps.wikimedia.org/jawiki/

[3] https://github.com/taku910/mecab

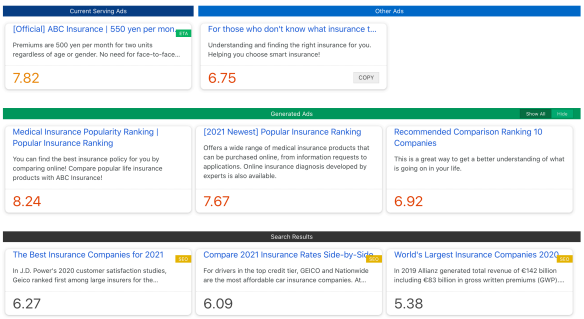

Figure 2: Screenshot of the ad-text generation tool displaying the current serving ads, generated ads, and search results corresponding to the keyword "insurance comparison." The scores in the figure are examples for clarity and are not equivalent to the scores used in the actual tool.

Seq2Seq	Dim	200	**LM**	Dim	200
	Vocab	cut word its freq. <2		Vocab	50,000
	Optimizer	Adam		Optimizer	SGD
	Learning rate	0.0001		Learning rate (LR)	20
	LR decay	-		LR decay	0.25
	Gradient normalization	0.5		Gradient normalization	0.25
	Dropout	0.3		Dropout	0.2
	Batch size	50		Batch size	20
	Epoch	10		Epoch (pretrain)	40
	γ	0.98		Epoch (fine-tuning)	10
	C_{title}	60		Max length	300
	C_{desc}	180			

Table 2: Parameter settings.

inforcement learning. If the method does not have any reward, we set λ as 0.0.

3.3 Automatic Evaluation

We used BLEU, F_1 scores of Rouge-1 (**R-1**), Rouge-2 (**R-2**), Rouge-L (**R-L**), and the following metrics for our automatic evaluation:

Estimated Quality Score (**EQ**): To measure ad quality for the generated ad-text, we used $r^Q(x, y)$ as discussed in Section 2.2.3.

Language Model Score (**LM**): To evaluate how a generated ad-text is grammatical, we used the language model for calculating s_{LM}.

Coverage of Keywords (**Cov**): To evaluate whether the keywords were included in the generated ad-text, we used the coverage score $s^{cov}(\mathbf{x}, \mathbf{y})$ defined in Section 2.2.2.

Position of Keywords (**Pos**): To investigate if important keywords appeared at the beginning of the generated ad-text, we used the position score $s^p(\mathbf{x}, \mathbf{y})$ used in Eq. (9).

Format Correctness (**FC**): This metric measures the number of generated ad-texts that followed the appropriate format. We verified whether the generated ad-texts contained the title and description parts, and that the lengths of these parts did not exceed the length limit. The percentage of ad-texts with the appropriate format was reported.

Diversity (**Div**): This metric evaluates the diversity of the generated ad-text by calculating the percentage of generated ad-texts excluded from the training dataset for each model.

3.4 Human Evaluation

We hired 10 annotators through Lancers[4], a crowdsourcing service in Japan, and 3 professional copywriters to evaluate the quality of ad-texts that were generated by the seven models: ad-text written by a human (**Reference**), **Sep**, **Mix**, **Mix+QS**, **Mix+Flu+QS**, **Mix+Flu+Rel**, and **Mix+Flu+QS+Rel**. It should be noted that, for this human evaluation, we omitted some models that performed poorly in the automatic evaluation.

We generated ad-texts from randomly sampled 240 ads in the test set and performed two tasks to evaluate three criteria: (i) fluency, (ii) attractiveness, and (iii) relevance. In the first task, annotators were instructed to evaluate the fluency of a displayed single ad-text.In the second task, the annotators were instructed to select one of the two ad-texts displayed side-by-side from the perspective of the ad-texts' attractiveness and relevance to the keywords[5]. We generated 11,760 questions in total, including 1,680 questions for the first task and 10,080 questions for the second task.

In the first task, fluency scores were obtained by calculating the percentage of annotators who answered "yes" to all questions. All answers were represented as a relation of each pair such as "win," "lose," and "tie" in the second task. We then scored each method's performance through these relation pairs using TrueSkill™(Herbrich et al., 2007), a widely used rating system that incorporates a Bayesian inference algorithm[6]. This algorithm treats the performance of each method as a standard distribution, where the mean μ represents the performance, and variance σ represents confidence. These are updated by repeating the pairwise comparison through the annotators' evaluations. We used the μ value of each method as the final result.

3.5 Deployment and Ad-delivery Evaluation

We deployed the baselines and our models to an ad-text generation tool, as shown in Figure 2. In

[4] https://www.lancers.jp

[5] There was also the option "same," but it was explicitly stated in the tutorial that this was deprecated.

[6] Following the official settings, the parameters were initialized as follows: $\mu = 25.0, \sigma = 8.33$. The draw probability was set to the ratio of the number of "same" options selected out of all answers.

Model	R-1	R-2	R-L	BLEU	EQ	LM	Cov	Pos	FC	Div
Sep	36.4	20.2	36.4	44.2	64.1	10.6	55.7	11.0	99.2	19.7
Mix w/o tag	36.8	17.9	36.8	43.7	64.0	10.4	51.6	10.4	99.3	36.0
Mix	35.6	18.5	35.6	44.7	64.3	12.4	**55.6**	10.6	99.5	40.6
Mix+QS	35.8	18.0	35.8	44.2	64.6	**13.4**	53.1	**10.2**	**99.6**	42.0
Mix+Flu	36.8	17.9	36.8	44.9	64.2	12.1	54.4	10.8	**99.6**	35.3
Mix+Rel	34.6	17.3	34.6	42.9	64.6	12.2	52.8	11.3	99.3	**45.1**
Mix+Flu+QS	36.3	16.9	36.3	**45.4**	64.5	12.9	52.2	10.8	99.3	37.4
Mix+Flu+Rel	**37.5**	**20.5**	**37.5**	44.4	**64.9**	13.3	53.2	11.9	99.4	40.8
Mix+Flu+Rel+QS	35.3	18.4	35.2	43.1	**64.9**	13.1	55.2	11.8	99.4	42.7

Table 3: Results of automatic evaluation.

	Copywriter			Crowdsourcing		
Model	(1) Flu.	(2) Att.	(3) Rel.	(1) Flu.	(2) Att.	(3) Rel.
Reference	87.5	25.5	24.4	75.6	26.8	29.1
Sep	82.1	25.2	24.0	65.7	24.6	28.4
Mix	**83.3**	25.1	23.7	**64.5**	23.8	26.1
Mix+QS	80.8	25.0	23.7	64.0	23.2	26.5
Mix+Flu+QS	81.7	**25.3**	22.8	64.3	24.4	26.6
Mix+Flu+Rel	77.5	24.2	23.7	60.9	24.8	26.2
Mix+Flu+QS+Rel	81.2	23.9	**24.3**	62.7	**25.4**	**26.9**

Table 4: Results of human evaluation. Flu., Att., and Rel. refer to Fluency, Attractiveness, and Relevance, respectively.

addition to the generated ad-texts, the tool also displays currently serving ads, other ads in the search result, and the non-ad search results on the same screen. Besides, we predict the quality scores for all items using the method described in Section 2.2.3. This tool aids copywriters to edit the generated ad-text effectively and obtain a comprehensive understanding of market trends.

We also performed an ad-delivery evaluation using the deployed tool. We gathered the ad-texts generated by the different models[7], including those written by copywriters, into the same ad-group per product and served each ad-group for one to two weeks. In total, we served 104 ads, 11 ad-groups, and 1,568 keywords for three weeks. In the result section, we show the number of impressions, click-through rates (CTRs), and costs averaged by ad-groups from the serving result. The Impression is the number of times an ad is displayed, the CTR is the percentage of clicks that out of impressions, and the Cost is the budget spent; the higher, the better for all metrics.

[7]**Mix+Flu+QS** was not included because it had a high percentage of output overlap with the other models, and we filtered out ad-texts that could cause legal problems.

4 Results and Discussions

4.1 Automatic Evaluation

Table 3 presents the results of the automatic evaluation. **Mix** achieved the best diversity among the three models **Sep**, **Mix**, and **Mix w/o tag**. The diversity of **Mix w/o tag** was also better than that of **Sep**. This result indicates that a dataset covering many domains is useful for improving diversity. Furthermore, a comparison between **Mix** and **Mix w/o tag** illustrated that discriminating domains and clients is useful in terms of diversity. **Mix+Flu+Rel** improved **R-1**, **R-2**, and **R-L** by 0.7, 2.0, and 0.7 points, respectively, whereas **Mix+Flu+QS** achieved the best BLEU score. Because Rouge uses the F_1 score and BLEU uses the precision of overlapped words between references and system outputs, we conclude that **Rel** generated ad-texts with words that were not included in the references. This is consistent with the improvements of **Mix+Rel** and **Mix+Flu+Rel** in **Div**. In **EQ**, **Rel** improved **EQ** similar to **QS**. This result is consistent with our expectation that Rel is an important factor for the effectiveness of ad-texts. In both **LM** and **Pos**, **QS** achieved the best scores. This result indicates that **LM** and **Pos** are correlated with **EQ**. In **Cov**, only **Mix+Flu+Rel+QS**

achieved a score comparable to that of **Mix**. The results for each metric suggest the importance of a combination of rewards. However, the importance of each metric in ad-text generation is uncertain. To clarify this concept, we performed additional human evaluations.

4.2 Human Evaluation

Table 4 presents the results of these human evaluations. Our methods achieved better scores than all other methods in terms of the attractiveness and relevance criteria by both the copywriter's and crowdsourcing's evaluations. Particularly, **Mix+Flu+QS+Rel** improved attractiveness and relevance scores by 1.6 and 0.4 points, respectively. Considering the attractiveness evaluation results, there are some differences between the annotations by copywriters and crowdsourcing workers. This is due to the stance of each annotator, where copywriters evaluate ad-texts as editable sources. By contrast, crowdsourcing workers treat ad-texts as a part of completed ads. In other words, copywriters rate an ad-text highly if they regard that they can fix it to a good one, whereas crowdsourcing workers evaluate ad-texts in their current form. This trend also appears in the fluency task, as presented in Table 4; overall, the score of the fluency task in the copywriter section is higher than that in the crowdsourcing section. **Mix** produced the best score for the fluency task; however, the proposed **Mix+Flu+QS** had a highly competitive result with a difference of just 0.2 points.

4.3 Ad-Delivery Evaluation

Table 5 presents the result of the ad-delivery evaluation. **Mix+Flu+Rel** achieved the best score in terms of impression and cost, whereas **Sep** achieved the best score in CTR. Because **Sep** and **Reference** have similar ad-texts, their CTRs are almost identical. These results indicate that considering the relevance between ad-texts and user queries is important to enhance user recognition for the ad-texts.

Based on these results, we used linear regression to investigate if the evaluation metrics are related to each ad-delivery evaluation metric. Figure 3 shows the results. With regard to impression and cost, considering the coverage of keywords in generated ad-texts is important. In CTR, it is necessary to focus on the fluency rated by crowdsourcing workers. The attractiveness of crowdsourcing workers is counterproductive. This is because people avoid clicking on an ad-text that looks like an ad-text. Based on the result, we conclude that the keyword-related automatic evaluation metric and evaluations via crowdsourcing are important for generating effective ad-texts.

[7]CW and CS denote the copywriter and the crowdsourcing results, respectively.

Model	Impression	CTR (%)	Cost
Reference	1.33	7.82	23.79
Sep	3.96	7.98	47.43
Mix	4.71	5.18	78.80
Mix+QS	2.77	**7.01**	51.89
Mix+Flu+Rel	**5.06**	4.10	**86.01**
Mix+Flu+QS+Rel	1.75	5.53	61.48

Table 5: Results of ad-delivery evaluation.

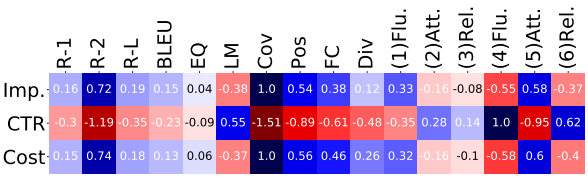

Figure 3: The weights for each metric. All weights are scaled by maximum values for each row.

5 Conclusion

In this paper, we proposed several rewards based on RL, which can consider the various characteristics of ad-texts. In experiments, ad-texts generated from Seq2Seq incorporated with these rewards achieved better automatic, human, and ad-delivery evaluation results than the basic Seq2Seq methods. Our analysis showed that considering results from the keyword-related automatic evaluation metric and the fluency by crowdsourcing workers is important for generating effective ad-texts. As further work, we plan to consider diversity as a reward to generate more diverse ad-texts.

Acknowledgement

The ad-delivery results used in this paper were served and aggregated by the AI Tech Studio of CyberAgent, Inc. We would like to take this opportunity to thank them.

References

Kevin Bartz, Cory Barr, and Adil Aijaz. 2008. Natural language generation for sponsored-search advertisements. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, EC '08, page 1–9, New York, NY, USA. Association for Computing Machinery.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Atsushi Fujita, Katsuhiro Ikushima, and Satoshi Sato. 2011. Automatic generation of listing ads and assessment of their performance on attracting customers: a case study on restaurant domain. *Journal of Information Processing*, 56(6):2031–2044.

Atsushi Fujita, Katsuhiro Ikushima, Satoshi Sato, Ryo Kamite, Ko Ishiyama, and Osamu Tamachi. 2010. Automatic generation of listing ads by reusing promotional texts. In *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, ICEC '10, page 179–188, New York, NY, USA. Association for Computing Machinery.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. Trueskill™ : A bayesian skill rating system. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 569–576. MIT Press.

J. Weston Hughes, Keng-hao Chang, and Ruofei Zhang. 2019. Generating better search engine text advertisements with deep reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, page 2269–2277, New York, NY, USA. Association for Computing Machinery.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

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

S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195, Los Alamitos, CA, USA. IEEE Computer Society.

Toshinori Sato, Taiichi Hashimoto, and Manabu Okumura. 2017. Implementation of a word segmentation dictionary called mecab-ipadic-neologd and study on how to use it effectively for information retrieval (in japanese). In *NLP*, pages NLP2017–B6–1. The Association for Natural Language Processing.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Melbourne, Australia. Association for Computational Linguistics.

Stamatina Thomaidou, Ismini Lourentzou, Panagiotis Katsivelis-Perakis, and Michalis Vazirgiannis. 2013. Automated snippet generation for online advertising. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, page 1841–1844, New York, NY, USA. Association for Computing Machinery.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.