Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 1882 – 1896

doi:10.55730/1300-0632.3911

Generating ad creatives using deep learning for search advertising

Kevser Nur ÇOĞALMIŞ[1,*], **Ahmet BULUT**[2,3]

[1]Department of Computer Engineering, Faculty of Engineering and Natural Sciences, İstanbul Sabahattin Zaim University, İstanbul, Turkey

[2]Department of Computer Engineering, Faculty of Engineering, Acıbadem University, İstanbul, Turkey

[3]Carbon Health 300 California St. 7th Floor. San Francisco, CA 94104

Received: 24.11.2021 • **Accepted/Published Online:** 08.05.2022 • **Final Version:** 22.07.2022

Abstract: We generated advertisement creatives programmatically using deep neural networks. A landing page contains relevant text data, which can be used for generating advertisement creatives, i.e. ads. We treated the ad generation task as a text summarization problem and built a sequence to sequence model. In order to assess the validity of our approach, we conducted experiments on four datasets. Our empirical results showed that our model generated relevant ads on a template-based dataset with moderate hyperparameters. Training the model with more content increased the performance of the model, which we attributed to rigorous hyperparameter tune-up. The choice of word embedding used in the representation of the input altered the model's performance. When the source and the target shared common sequences during training, the model produced the best results.

Key words: Online advertising, ad creative generation, deep learning

1. Introduction

The search advertising market is brokered by major search engines such as Google, Microsoft, and Yahoo. Ad brokers charge advertisers based on the type of ads used. In our study, we focus on search advertising with keywords and text-based ads. In order to create a search campaign for advertising a product online, an advertiser needs to choose keywords that best describe the product. According to the keywords chosen, the broker determines which users get to see the product's ad. A search engine results page (SERP) is the search engine's response to a user's query. If the user's query matches verbally or semantically with any of the keywords in the campaign, then one of the ads in the corresponding campaign is displayed on the SERP. An ad that contains a marketing message and a target URL for the landing page of the advertised product is what gets shown in a SERP. More formally, the search engine as the ad broker determines whether a keyword K in a search campaign matches with a user query Q when the query is submitted. If the keyword K and the query Q are related, then the corresponding ad for K is eligible for display on the SERP to Q.

Advertisers create a set of keywords and a set of ad creatives. Crafting the right keywords and ad creatives is an arduous task that requires collaboration of online marketers, creative directors, copywriters, and linguists. Many parts of this craft are still manual and therefore not scalable especially for large e-commerce companies that have large inventories and large search campaigns. Furthermore, the marketing team has to experiment with different marketing messages from subtle to strong, with different keywords from broadly relevant (to the

*Correspondence: kevser.cogalmis@izu.edu.tr

product) to exactly relevant, with different landing pages from informative to transactional, and many other test variants. The failure to experiment quickly for finding what works results in marketing budget being wasted.

A company in the healthcare industry has on average five campaigns for each of its landing pages. A typical campaign contains five ad groups. Each ad group requires five ad creatives since Google recommends the creation of three to five ads in each ad group. Overall, a total of 125 creatives need to be written. This could easily overwhelm a small team of advertisers for a company on a growth path. The healthcare company in question requires 125 new creatives each week for its location specific campaigns. Each new location requires a set of custom marketing messages tailored to local market dynamics. One could use generic value propositions to design creative templates. These templates can then be customized at runtime by inserting custom keywords into the ad text dynamically. However, writing a creative vs. writing a creative template is essentially the same problem in theory as well as in practice. In order to help advertisers, we proposed a deep neural network to generate ad creatives automatically using the textual content found in landing pages. Figure 1 depicts our proposal. The landing page of a product contains sufficient promotional information, which could be used in the creation of ads relevant to the product. The main contributions of our study are as follows:

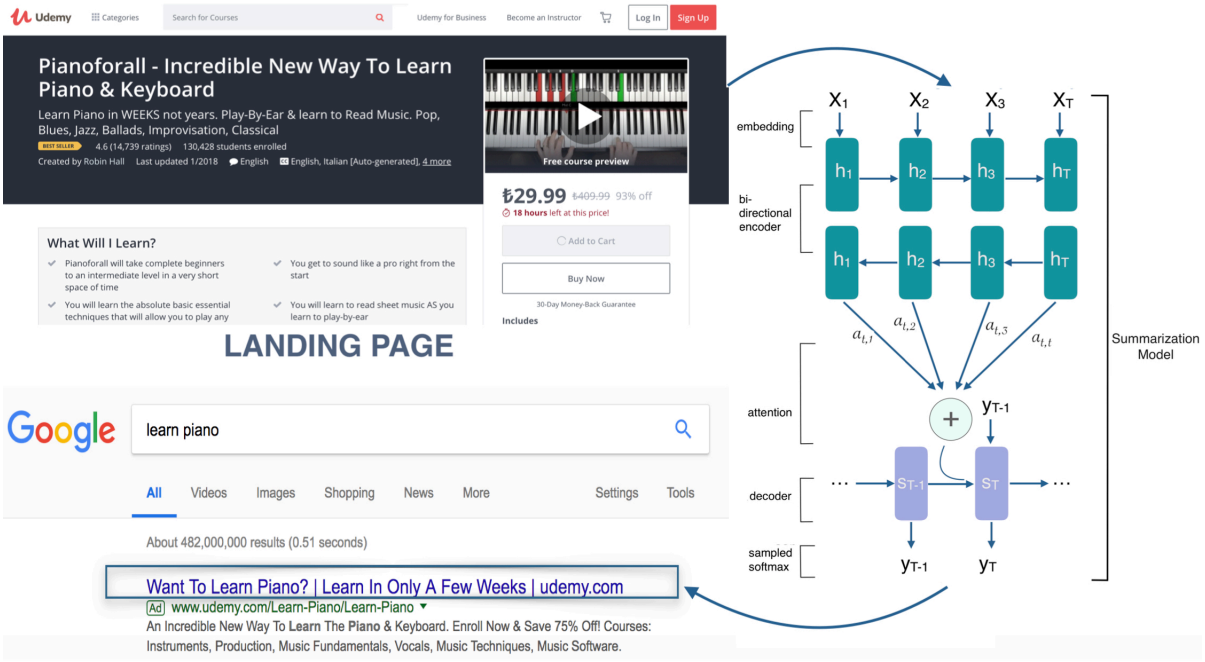

Figure 1. The abstractive summarization of a landing page into an ad creative with a summarization model.

i. We proposed a sequence to sequence model for generating ads.

ii. We crafted ads datasets from landing pages of an online learning platform.

iii. Feeding the network with more and diverse data improved the model's performance by a factor of three.

iv. Using landing pages as source and allowing sequence overlaps with ads as target improved the quality of the generated ads.

v. Using GloVe word embedding in input representation improved model performance by 3% on a template-based ads dataset.

2. Related work

Choi et al. used the landing page content for ad selection via an extractive summarization method [1]. In a similar vein, Thomaidou et al. used a summarization technique to extract the textual content of a landing page in order to create keywords and ads [2]. In extractive summarization, one needs to identify the most important sentences in a given text to summarize. It is permissible to copy phrases and sentences from the original text into the summary. In abstractive summarization, however, one needs to generate novel words and phrases that do not occur in the original text. Thus, an abstractive model can capture the meaning in text and generate summaries that are closer to human generated ones [3]. As an illustrative example of abstractive summarization, a sequence to sequence model with a long short term memory (LSTM) was used in paraphrasing a given source text [4]. Kryściński et al. also used a sequence to sequence model with an additional pretrained language model in order to extract concise paraphrases for summarization [5].

Reisenbichler et al. used a pretrained language model called GPT-2 to generate text [6]. Such language models are adept in generating text that contain novel words. In practice, GPT variants are more suitable for keyword generation where novelty is preferred rather than for ad generation where consistency in messaging is preferred. Jin et al. used BART encoder-decoder model [7] for creating slogans [8]. The slogan generation from the corresponding description is analogous to abstractive summarization. The ROUGE scores they obtained with the pretrained abstractive model were lower compared to the ROUGE scores we obtained with our model. Hughes et al. proposed an RNN-based model for generating text ads from product landing pages as inputs [9]. It was a sophisticated deep learning model; however, more than 15% of the generated ads by this model were reported as nonsense, broken, or bad. Wang et al. showed how pretrained models could be enhanced by using reinforcement learning to generate attractive and high-quality text advertisements [10]. Since their method complements and integrates well with an underlying base model, which could be pretrained or not, we conjecture that their approach could be used in order to enhance the accuracy of our own model as well.

Yuan et al. proposed a model called CAPE for the use and classification of persuasive tactics in ad text as well as predicting the ad's promotional effectiveness, i.e. its click-through rate [11]. In addition to syntactic text quality scores, the expected promotional effectiveness of a given ad could be used as a quantitative metric in order to evaluate the performance of a generative model further. In fact, this approach was adopted by Wang et al. where they built a sequence to sequence model to generate social advertisements rather than more rigid search advertisements as in our case [12]. They leveraged the click-through data for measuring the fitness of each ad quantitatively.

3. Methodology

Our goal is to generate an ad from the target landing page content. We treated this task as a text summarization task. We created a sequence to sequence model using an encoder-decoder recurrent neural network (RNN) architecture. RNNs were shown to work well in modeling consumer preference [13, 14]. The input sequence is mapped to the output sequence using vectors of fixed length. The encoder reads input sequences of varying length and encodes them into vectors. These vectors retain the context information present in the input. The decoder emits the final word predictions using the output vectors of the encoder. We used an LSTM network for both the encoder and the decoder cells. Word embedding is used to transform our corpus into word vectors of 500 dimensions. A bidirectional LSTM is used in the encoding stage in order to retain the information present in the past. The mere reversal of the input in the same stage allows us to retain information in the future as

well [15]. For soft-alignment between each input word and output word, the attention mechanism is added into the model. At the final step, we use softmax sampling to compute probabilities over the target vocabulary.

Figure 2 depicts the structure of our summarization model consisting of encoder and decoder layers with word embedding to vectorize input sequences. There is an attention mechanism added subsequent to the encoder. A softmax layer subsequent to the decoder computes the final word probabilities.

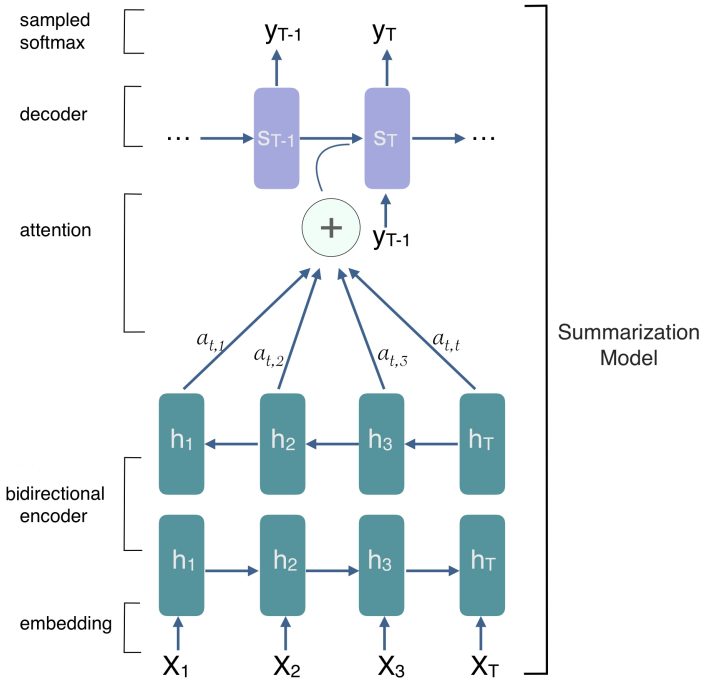

Figure 2. The illustration of our network architecture where $X_t s$ are sequences from the input document, $h_t s$ are hidden states in the encoder phase, $a_t s$ are part of the attention mechanism, $S_t s$ are the decoder states. $Y_t s$ are the output sequences that summarize a given input document.

In Figure 2, X_t denotes an input word embedding, i.e. the vector representation of each word in the dataset, and S_t denotes the hidden state that corresponds to the memory of the network at time t. It is calculated from the input at time t and the hidden state at time $t-1$. Finally, o_t denotes the output of the current state at time t. In the forward pass, S_t accumulates the information at each time step while the model parameters W, U, and V remain constant. In the backward pass, these model parameters are updated with the gradient descent algorithm. More formally,

$$
\begin{aligned}
i_t &= \sigma(U_i x_t + W_i s_{t-1} + b_i) \\
f_t &= \sigma(U_f x_t + W_f s_{t-1} + b_f) \\
\tilde{c_t} &= \tanh\left(U_c x_t + W_c s_{t-1} + b_c\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \\
o_t &= \sigma(U_o x_t + W_o s_{t-1} + b_o) \\
s_t &= o_t \odot \tanh c_t,
\end{aligned}
$$

where i_t refers to the input gate, f_t refers to the forget gate, and $\tilde{c}_t$ refers to the input modulation gate [16]. The previous cell state c_{t-1} is modulated by the forget gate f_t, and the input gate i_t is modulated by the input modulation gate $\tilde{c}_t$ in order to compute the current cell state c_t. With these gates, the network learns what part of the input to emphasize, and what information to forget retained in its memory.

4. Experiment setup

4.1. Ads datasets

For the experiments, we used the data from an online learning marketplace where professionals from various fields create online classes and tutorials. We used two datasets. The first one is called D_{temp}, which contains ads written according to a set of ad templates. In D_{temp}, the source document is the set of course titles related to a given course while the target document is the ad written by advertisers for the given course. In D_{temp}, the target document contains template phrases that occur in many other ads, e.g., "this is the best place to learn" and "start learning at your own pace". A source and target document pair from D_{temp} is shown in Table 1. The second dataset is called D_{rich}, which is richer in both content and context. The custom description of each course present in its landing page is the source document, and the page title is the corresponding target document. A source and target document pair from D_{rich} is shown in Table 2. The dataset D_{temp} has $3,363$ pairs while D_{rich} has $4,795$ pairs. In D_{temp}, there are $2,632$ and 408 unique words in the source and the target documents, respectively. In D_{rich}, there are $11,172$ and $7,290$ unique words in the source and the target documents, respectively.

Table 1. A sample row from the template-based ads dataset D_{temp}. For a given course, its source document contains the titles of twelve other courses similar to it, and its target document is the ad written for it.

Source document	Target document
corporate finance - a brief introduction. hline personal finance - the core four of personal finance. introduction to finance, accounting, modeling and valuation. how to start a successful career in finance?. personal finance masterclass - easy guide to better finances. start-up financial modeling for non-finance professionals. power bi: create a power bi finance dashboard with tutorials. finance and accounting for startups - become empowered!. classification-based machine learning for finance. accounting and finance: bookkeeping & financial statements. debt and equity financing strategies for your company. accounting and finance for bankers - a comprehensive study.	improve finance skills?. this is a great place to learn. improve your finance skills today.

Table 2. A sample row from the content-based ads dataset D_{rich}. For a select course, its source document contains the relevant text found in the body of its landing page, and its target document is the title of that landing page.

Source document	Target document
learn how to grow your pinterest following . understand the pinterest platform advertising . learn how to start a pinterest board that succeeds . learn how to use pinterest messages for marketing . learn the analyze and measure your pinterest efforts . learn how to optimize your pins for the pinterest smart feed . learn how to market yourself and your business on pinterest platform. learn how to get resource to guide your pinterest	a professional course of pinterest marketing and promotions . pinterest marketing: for planning, collecting, discussing, and sharing ideas, common interests and directing traffic.

Figure 3 depicts the landing page of a piano course. The landing page contains a section called "what you'll learn" shown in the blue rectangle. This section contains phrases that describe the course content and the learning outcomes. We used this section to compile the source documents in D_{rich}. As for the target

documents in D_{rich}, the title and subtitles of the course shown in the red rectangle were used in crafting an ad for the course.

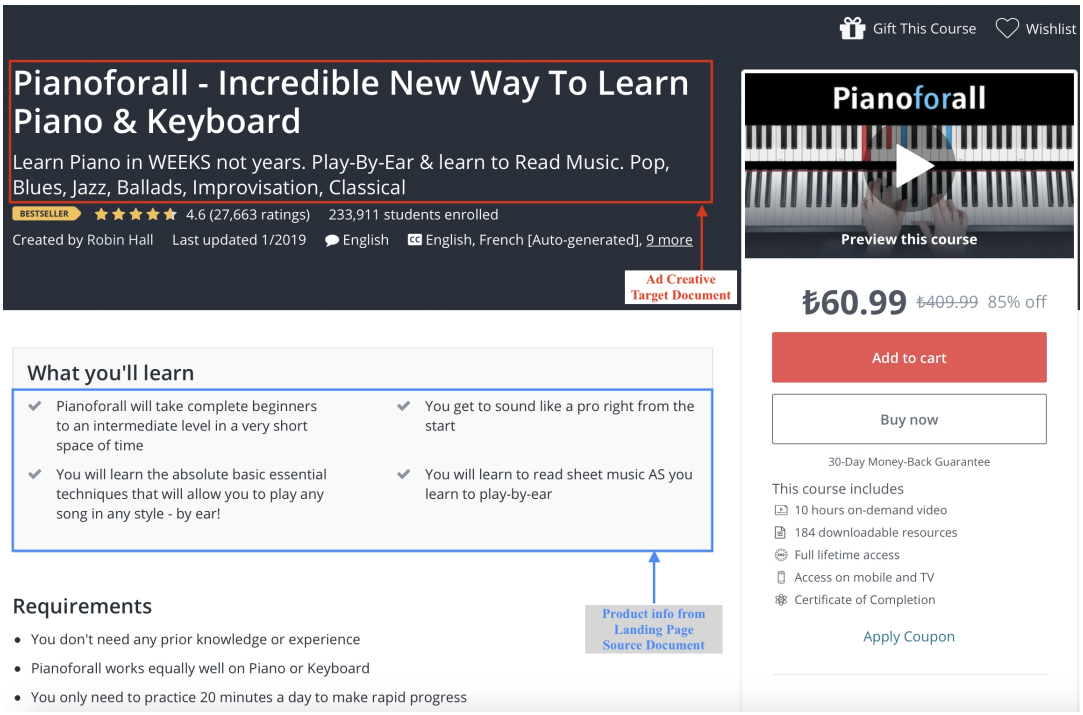

Figure 3. A sample landing page. In the generation of our dataset D_{rich}, the text in the blue rectangle was used as the "source document", and the text in the red rectangle was used as the "target document".

4.2. OpenNMT and Google Colab research

An open source neural machine translation toolkit called OpenNMT is widely used in sequence to sequence modeling [17]. OpenNMT is available in LuaTorch, PyTorch, and Tensorflow. The toolkit is mainly used for language translation and text summarization. With OpenNMT, one can create various types of neural networks such as RNN, bidirectional RNN, and convolutional neural network. We built our models using OpenNMT, and ran all our experiments on Google's Colab. Colab is a cloud service provided by Google, and it supports running Python code on GPUs.

4.3. Training

There are multiple hyperparameters to tune in any sequence to sequence network. A small network with a small number of layers and with moderate hyperparameters was created for D_{temp} because the dataset is relatively simple and it includes many repetitive sentences. Using the same hyperparameters is not sufficient for D_{rich} because capturing the interconnected relationships present in this dataset is not as straightforward as in D_{temp}. Therefore, we performed hyperparameter optimization to obtain the best results. We set the maximum source sequence length to 360, and the target sequence length to 50 based on the longest input and output. The dropout was set to 0.3, which determines the percentage of nodes to be ignored temporarily in the hidden states in order to avoid overfitting [18]. We set the beam size to 5.0 in order to control the search depth.

We built a bidirectional RNN using an LSTM with attention as was shown earlier in Figure 1. We conducted experiments using stochastic gradient descent (sgd) and adaptive moment estimation (adam) optimizers [19]. We varied the number of hidden states in each RNN from 128 to 512 neurons. In each model, both encoder and decoder have two hidden layers. Since each sequence in the output sentence is represented as a one-hot vector, the number of neurons in the output layer is equal to the size of the output vocabulary V_o. Each output neuron represents the probability of a word. If we would always choose the most probable one as in greedy decoding, we could bail out prematurely. Assume that all possible outcomes correspond to paths of a tree, the depth of which is the number of output words l_o, and each node has V_o branches. Choosing the most probable word at each time may not produce the best possible output. Since there are $V_o^{l_o}$ possible paths, the time complexity is exponential. Fortunately, the beam search decoding reduces the number of paths to a scalable $k \times V_o \times l_o$ where k is the number of best possible words.

4.4. Evaluation metrics

There exist metrics that consider whether generated summaries are grammatically correct or whether the sentences are well-structured. Such metrics only address syntactical quality. On the other hand, content-based metrics measure text quality by comparing the words in the generated sentences with the words in the human-generated sentences. The Bilingual Evaluation Understudy (BLEU) is used in literature for evaluating the quality of generated text [20]. BLEU is defined as the fraction of n-grams in the generated text that also appear in the ground-truth text. Hence, BLEU is a measure of precision. Another widely used metric for measuring text quality is Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [21]. In contrast to BLEU, ROUGE-n represents the ratio of n-grams found in the ground-truth text that also appear in the generated text, and therefore, is a measure of recall. ROUGE-L corresponds to the longest overlapping subsequence between the generated text and the ground-truth text. Together, BLEU and ROUGE could quantify textual coherence and syntactic quality.

5. Empirical results

We ran experiments on both of our datasets D_{temp} and D_{rich}, and varied the hyperparameters for network creation and input representation. We split all datasets into train, validation, and test sets with a split ratio of $70:15:15$. We present our empirical findings next.

Table 3. The effect of varying model size and optimizer on ROUGE scores. The experiments were conducted on D_{temp} and D_{rich}. $|RNN|$ represents the size of the network while *opt* corresponds to the optimizer used.

	Model parameters	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
D_{temp}	$\|RNN\| = 128$, opt = sgd	43.0	22.3	42.1
	$\|RNN\| = 512$, opt = sgd	**45.8**	**25.5**	**45.2**
	$\|RNN\| = 128$, opt = adam	43.3	21.3	42.7
	$\|RNN\| = 512$, opt = adam	44.6	23.8	43.7
D_{rich}	$\|RNN\| = 128$, opt = sgd	14.9	2.4	13.7
	$\|RNN\| = 512$, opt = sgd	14.0	1.8	12.8
	$\|RNN\| = 128$, opt = adam	13.5	2.8	12.8
	$\|RNN\| = 512$, opt = adam	**20.6**	**4.7**	**19.0**

5.1. The performance of default embedding on D_{temp} and D_{rich}

In our first set of experiments, we used random weights in the input representation. Table 3 shows the ROUGE performance of four different models for varying network size and for varying the optimizer. As shown by the results in Table 3, increasing the size of the network improved ROUGE-1 score by 2.8% and by 1.3% for sgd and adam optimizers, respectively, on D_{temp}.

Varying the optimizers increased ROUGE-1 score by 0.3% for RNN models of size 128, but decreased it by 1.2% for RNN models of size 512. We recommend the use of moderate hyperparameters with such simpler datasets as D_{temp} for practical reasons. The ROUGE scores on D_{rich} indicate that moderate hyperparameters are not sufficient on more complex datasets. As shown in Table 3, ROUGE scores on D_{rich} were lower compared to D_{temp}. Since D_{temp} contains repetitive template-based phrases, e.g., "this is great place to learn" and "start learning at your own pace", the model was able to learn the patterns and performed well. In order to improve performance on D_{rich}, we enriched it with more content around similar topics.

Table 4. The side by side comparison of the ground truth creatives with the generated creatives. The results were obtained on datasets D_{temp} and D_{rich} on models of varying size $\{128, 512\}$.

The samples were obtained on D_{temp}.

Source document for D_{temp}
the complete ruby on rails developer course. dissecting ruby on rails 5 - become a professionaldeveloper. professional ruby on rails developer with rails 5. learn to code with ruby. professional rails code along. ruby on rails: superhero generator. angular 2 + rails 5 bootcamp. comprehensive ruby programming. 8 beautiful ruby on rails apps in 30 days & tdd - immersive. advanced ruby programming: 10 steps to mastery. ruby on rails foundations. ruby on rails: the complete full stack
Ground-truth ad creative
online ruby on rails. quality videos by domain experts. why wait? learn ruby on rails now.
Generated ad creative by a model with $\|RNN\|$**=128, opt=adam**
online ruby on rails. this is a great place to learn. improve your ruby on rails skills.
Generated ad creative by a model with $\|RNN\|$**=512, opt=adam**
online ruby on rails. the best price for the best value. learn at your own pace. start now!

The samples were obtained on D_{rich}.

Source document for D_{rich}
learn the four components required to perform ethical hacking . learn the basics of port scanning and how you identify running services . learn the basics of vulnerability scanning to identify vulnerable hosts . learn the basics of exploitation with metasploit .
Ground-truth ad creative
hacking for beginners . perform your first hack in the next 2 hours !
Generated ad creative by a model with $\|RNN\|$**=512, opt=adam**
the complete ethical hacking course for beginners . learn the basics of ethical hacking , ethical hacking , and more !

We compared the ads generated by our models with the ground truth ads in Table 4. Although the predictions on D_{rich} were suboptimal in terms of their ROUGE scores, the model generated concise creatives. Surprisingly, the model was able to learn call-to-action phrases such as "learn how to" and "learn the basics of", which call for users to act. The performance of models with sizes 128 and 512 approximated the ground truth creatives on D_{temp}. The models learned the structure of ground truth ads, which consist of three sentences. The first sentence is about the course itself. The second and third sentences are generic marketing phrases independent of the course. Hence, generating the first sentence is more difficult than generating the other two.

Though the last two sentences are different for each summary, the models predicted the first sentence in the ground truth ad correctly as shown in Table 4. The RNN models with sizes of 128 and 512 had similar ROUGE performance on D_{temp}, and they generated similar creatives qualitatively as well.

5.2. The performance of GloVe embedding on D_{rich}

GloVe measures contextual word cooccurrence of words that are related or analogous [22]. GloVe considers how often words cooccur in context by using the inner product of word frequencies. We created GloVe word embedding on D_{rich}. Table 5 reports the ROUGE scores of the generated creatives under different parameter settings. The use of GloVe increased ROUGE-1 and ROUGE-L scores by 3%, and improved the quality of the generated creatives.

Table 5. The effect of using GloVe word embedding on model performance quantified by ROUGE scores.

Dataset is D_{rich}	**GloVe parameters**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
$\|RNN\| = 512$, opt = adam	dim = 300, $\min(w_c) = 1$, iter = 10, $\|W\| = 5$	22.4	5.3	21.0
	dim = 300, $\min(w_c) = 1$, iter = 15, $\|W\| = 15$	**23.1**	**5.4**	**21.3**
	random weight embedding	20.6	4.7	19.0

5.3. Qualitative assessment of ad creatives

For qualitative assessment, the generated creatives were rated using a scale of three grades by two human judges. The goal is to quantify the match between the generated ad creatives and the ground truth ad creatives. As shown in Table 6, 55% of the ad creatives were judged as "relevant while" 32% were judged as "not relevant".

Table 6. Two judges assessed the quality of the generated creatives on D_{rich}.

		2nd **Judge**			
		Relevant	Not relevant	Not sure	Total
1st **Judge**	Relevant	247	8	14	269
	Not relevant	5	136	17	158
	Not sure	16	12	26	54
	Total	268	156	57	481

A single human judge may not assess the quality of the generated creatives reliably because of subjectivity. In order to mitigate subjectivity, we considered the agreement between two judges, and computed Cohen's kappa statistic κ. It was computed from the individual judgments shown in Table 6 as follows [23]:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)},$$

where

- $P(A)$ denotes the maximum-likelihood estimate of the probability of agreement between the two judges.
- $P(E)$ denotes the maximum-likelihood estimate of the probability of agreement between them due to chance.

- The kappa statistic computed from $P(A)$ and $P(E)$ is 0.73.

A kappa score between 0.4 and 0.6 indicates moderate agreement whereas a score between 0.6 and 0.8 indicates satisfactory agreement [24]. Therefore, the quality of the ad creatives our model generated could be deemed as satisfactory. Furthermore, we observed that the model performed well for courses that are well represented in the training dataset while it performed poorly for courses that lack sufficient representation.

5.4. Assessment on handcrafted datasets

We handcrafted two new datasets from D_{rich}. First, the ad titles were reformatted according to Google's ad format, which has restrictions on the length of each ad unit as follows: "Headline 1 (30) | Headline 2 (30) | Headline 3 (30) . Description 1 (90) . Description 2 (90)". Second, brand new ad creatives were created by a domain expert and added into the pool of ads. In the first dataset D^{*}_{rich}, all ads are in Google's ad format and each ad went through a rigorous editorial process to make sure it is in the correct format. In the second dataset D^{+}_{rich}, the ad titles were added verbatim to the content of landing pages as source documents. The same target ad creatives were used in both D^{*}_{rich} and D^{+}_{rich}. Both datasets contain 3732, 458, and 467 document pairs for training, validation, and test respectively. Tables 7 and 8 show sample document pairs.

Table 7. The source and target document pairs in D^{*}_{rich}, which contains ad creatives in Google's ad format.

Source document #1	**Target document #1**
create and write advanced functions . create charts and buttons sort and filter with regular and advanced filters . build pivot tables and calculated columns and financial models . build pivot use randomness for model prediction	learning microsoft excel | introduction to mastery excel .using excel to build advanced functions tables and use randomness for model prediction
Source document #2	**Target document #2**
understand the purpose of webpack in a modern web app . build custom boilerplate projects to serve es2015 javascript . deploy webpack based projects to aws , heroku , and more . enhance the performance of web apps by leveraging webpack's ecosystem of plugins . enhance code organization through the use of es2015 js modules .	webpack guide for beginner | master webpack 2 with web apps . as you deploy web apps supported by babel , code splitting , and es2015 modules . deploy webpack based projects to aws , heroku , and more .

Table 8. The source and target documents pairs in D^{+}_{rich}, which contains ad creatives in free-form.

Source document #1	**Target document #1**
create and write advanced functions . create charts . and buttons sort and filter with regular and advanced filters . build pivot tables and calculated columns . use randomness for model prediction . learning microsoft excel : introduction to mastery . using excel to build advanced functions .	learning microsoft excel | introduction to mastery excel . using excel to build advanced functions and financial models . build pivot tables and use randomness for model prediction .
Source document #2	**Target document #2**
understand the purpose of webpack in a modern web app . build custom boilerplate projects to serve es2015 javascript . deploy heroku , and more . enhance the performance of web apps by leveraging webpack's ecosystem of plugins . enhance code organization through the use of es2015 js modules webpack 2 : the complete developer's guide . master webpack 2 as you deploy web apps supported by babel , code splitting , and es2015 modules	webpack guide for beginner | master webpack 2 with web apps . as you deploy web apps supported by babel , code splitting , and es2015 modules . deploy webpack based projects to aws , heroku , and more .

5.4.1. ROUGE and BLEU performance

Since GloVe embedding had the highest ROUGE scores, we used GloVe with its best settings on D^{*}_{rich} and D^{+}_{rich}. We worked with RNNs of size 512 and used "adam" as the optimizer with a learning rate of 0.001. All models were trained 5000 times, and the interim model was saved at every 250 steps. The running time was 17 mins on D^{+}_{rich} and 33 mins on D^{*}_{rich}. In order to generate the final ad creatives, we used the interim model that had the lowest perplexity. Figure 4 shows the validation perplexity of the first six models on both datasets. The perplexity started at 20% on D^{+}_{rich}, and decreased gradually till a point beyond which it started to increase. This particular behaviour shows the reliability of the model. The perplexity started at 105% on D^{*}_{rich}, and the lowest perplexity was obtained by the $2nd$ model, which was saved at the $500th$ step. The perplexity grew exponentially thereafter. An ad creative generated by a select model is shown in Table 9 along with the ground truth creative. In general, all generated creatives were in the right ad format and were relevant to the source context.

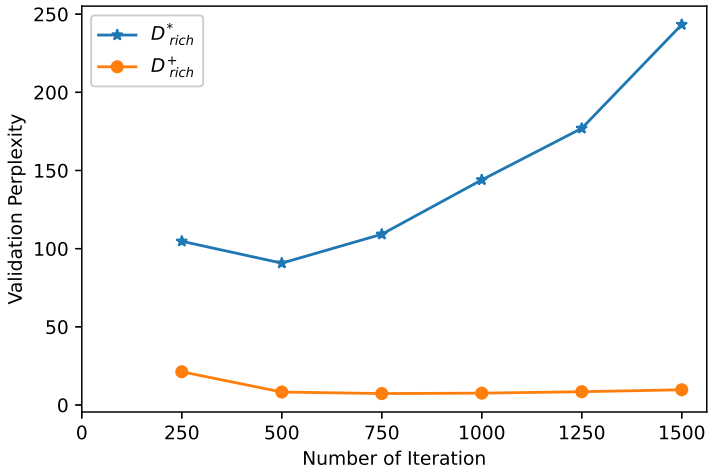

Figure 4. The validation perplexity of the first six models on D^{*}_{rich} and D^{+}_{rich}. We used the validation perplexity in order to select the model to use without overfitting in generating creatives.

Table 9. A generated creative vs. the corresponding ground-truth creative from D^{+}_{rich}.

Source document for D^{+}_{rich}
run ruby interactively at the command prompt . understand basic principles of ruby programming . learn ruby programming the easy way lite . learn to program in ruby , in a step by step easy to follow hands on course .
Ground truth ad creative
learn ruby programming | the easy way lite | application with ruby . learn to program in ruby , in a step by step .
Generated ad creative by a model with $\|RNN\|$=512, opt=adam
learn ruby programming | the easy way lite . learn to program in ruby , in a step by step easy to follow hands on course .

Table 10 shows the ROUGE and BLEU scores obtained on both datasets. When we used the raw ad titles in the source, and the human-generated ad creatives in the target as in D^{+}_{rich}, we obtained the highest ROUGE and BLEU scores. A ROUGE score of 40 is regarded as a good performance for a summarization task. Our model scored 62.7 in ROUGE-1 and 61.9 in ROUGE-L.

Table 10. ROUGE and BLEU scores of our model with network size $|RNN| = 512$, with *adam* as the optimizer, and with a learning rate of 0.001 on the datasets D^{*}_{rich} and D^{+}_{rich}.

	Model parameters	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**BLEU**
D^{*}_{rich}	$\|RNN\| = 512$, opt = adam, LR = 0.001	27.6	8.1	26.1	6.0
D^{+}_{rich}	$\|RNN\| = 512$, opt = adam, LR = 0.001	**62.7**	**48.0**	**61.9**	**41.4**

5.4.2. Qualitative assessment of ad creatives

For qualitative assessment of generated creatives on $D^{+}{}_{\text{rich}}$, the creatives were rated using a scale of three grades by two human judges. The goal was to quantify the quality of the generated creatives according to domain experts. As shown in Table 11, %77 of the ad creatives were judged as relevant while %13 were judged as "not relevant". The resulting kappa score was 0.8. We can state that the ad creatives generated from $D^{+}{}_{\text{rich}}$ had higher quality than the ad creatives generated from D_{rich}. Thus, we increased the model performance dramatically by enriching the dataset.

Table 11. Two judges assessed the quality of the creatives generated from $D^{+}{}_{\text{rich}}$. The judges used three labels as "relevant", "not relevant", and "not sure" during their assessment.

		2nd **Judge**			
		Relevant	Not relevant	Not sure	Total
1st **Judge**	Relevant	352	1	18	371
	Not relevant	1	57	9	67
	Not sure	5	5	19	29
	Total	358	63	46	467

We experimented with variations of our model and measured ROUGE and BLEU scores in each case. The results are summarized in Table 12. Decreasing the learning rate decreased the creative quality. The default learning rate of 0.001 with adam optimizer worked well on our datasets. Decreasing the network size to 256 from 512 resulted in a 5% decrease in ROUGE scores. When the computational resources are scarce, smaller networks should be preferred. The last column of Table 12 reports model confidence. In each case, the validation accuracy was less than the train accuracy, which means that the selected model was not overfitting. Furthermore, the crossentropy of the selected model, i.e. the loss over the number of words, was under 1.0 and the perplexity was under 10.0 in validation and training stages.

5.5. The comparison with baseline methods

We compared our model with three baseline models including RNNs and Transformer. Table 13 shows the ROUGE and BLEU scores of each model. Our model achieved the highest ROUGE scores on D^{+}_{rich}. Among the baselines, the RNN with attention performed better than the other two.

6. Conclusion

In this study, we treated ad generation as text summarization such that an ad corresponds to the summary of a given landing page. We observed that our bidirectional LSTM model with attention performed well in generating

ads. In order to validate our approach, we conducted experiments with four datasets. The experiments on the template-based dataset D_{temp} showed that our model generated ad creatives of good quality with moderate hyperparameters. On a content richer dataset D_{rich}, the model generated precise ad creatives, but its complexity increased. To improve the performance of the model on D_{rich}, we experimented with various word embeddings. We observed that GloVe word embedding performed the best. With GloVe, we observed a 3% increase in ROUGE-1 and ROUGE-L scores. The creatives generated from D_{rich} were validated by two judges with an interrater agreement score of 0.73.

Table 12. ROUGE and BLEU scores for varying network parameters.

Model parameters	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	Stats
$\|RNN\| = 512$, opt = adam LR = 0.001, stepSize = 5000, modelStep = 750	**62.7**	**48.0**	**61.9**	**41.4**	val. perp. = 7.3, val. acc. = 71.7 train acc. = 84.8, train ppl. = 2.1, xent = 0.7
$\|RNN\| = 512$, opt = adam LR = 0.0001, stepSize = 5000 modelStep = 4750	58.7	41.7	57.5	35.5	val. perp. = 11.3, val. acc. = 65.7 train acc. = 85.4, train ppl. = 2.2 xent = 0.7
$\|RNN\| = 256$, opt = adam LR = 0.001, stepSize = 5000 modelStep = 1250	57.1	41.3	56.3	30.4	val. perp. = 13.2, val. acc. = 66.0 train acc. = 83.2, train ppl. = 2.7 xent = 1.0
$\|RNN\| = 256$, opt = adam LR = 0.0001, stepSize = 2500 modelStep = 2500	32.0	11.2	30.3	7.1	val. perp. = 6441, val. acc. = 37.4 train acc. = 39.5, train ppl. = 44.2 xent = 3.7
$\|RNN\| = 256$,opt = adam LR = 0.0001, stepSize = 5000 modelStep = 5000	50.1	31.1	48.8	24.1	val. perp. = 20.9, val. acc. = 56.2 train acc. = 66.1, train ppl. = 7.2 xent = 1.9
$\|RNN\| = 256$, opt = adam LR = 0.0001, stepSize = 10000 modelStep = 7000	57.5	41.1	56.4	34.7	val. perp. = 16.6, val. acc. = 63.2 train acc. = 81.4, train ppl. = 3.0 xent = 1.1

Table 13. The ROUGE and BLEU scores of our model and three baselines on D^{+}_{rich}. The baselines are RNN with attention, RNN without attention, and Transformer. All models had a similar configuration.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
Transformer	28.3	7.6	26.5	6.5
RNN without attention	22.6	5.8	21.5	4.5
RNN with attention	56.9	42.5	55.9	32.8
Our Model	**62.7**	**48.0**	**61.9**	**41.4**

We enriched D_{rich} even further with information found in the landing pages and by reformatting the ads. Hence, we uplifted the ROUGE scores of the model and increased the observed quality of the generated ads. The resulting datasets $D^{*}{}_{\text{rich}}$ and $D^{+}{}_{\text{rich}}$ include the same creatives as their target documents, but $D^{+}{}_{\text{rich}}$ has relatively richer source documents. The visual inspection of the generated creatives hints at their potential for field deployment. The ad creatives generated from $D^{+}{}_{\text{rich}}$ were validated by two judges with an interrater agreement score of 0.79. Our model generated creatives in Google's ad format. Furthermore, it performed better than three other baselines.

One limitation in our problem domain is that it is hard to find datasets for research purposes. Additionally, the generated creatives must be evaluated using quantitative and qualitative metrics. The qualitative metrics

are the hardest to garner since one has to work with real advertisers for them to annotate and to evaluate the final output. This of course is susceptible to subjectivity, but active learning may come as remedy.

Another limitation is the lack of field deployment. Our generated ads were not deployed in real advertisement campaigns. It is utterly vital to get actual feedback from users, but this would mean for companies to deploy the programmatically generated ads and to spend advertising dollars on them. Any such endeavor is inherently experimental, and companies are reluctant to get involved in such initiatives.

Acknowledgment

This project is funded by the Scientific and Technological Research Council of Turkey (TUBİTAK) under grant number 119E031.

References

[1] Choi Y, Fontoura M, Gabrilovich E, Josifovski V, Mediano M et al. Using landing pages for sponsored search ad selection. In: Proceedings of the 19th International Conference on World Wide Web; 2010. pp. 251-260.

[2] Thomaidou S, Liakopoulos K, Vazirgiannis M. Toward an Integrated Framework for Automated Development and Optimization of Online Advertising Campaigns. In: Intelligent Data Analysis 2014; 18 (6): 1199-1227.

[3] See A, Liu P, Manning C. Get To The Point: Summarization with Pointer-Generator Networks. In: Association for Computational Linguistics; 2017.

[4] Nallapati R, Zhou B, Nogueira dos Santos C, Gülçehre C, Xiang B. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning; 2016. pp. 280-290.

[5] Kryściński W, Paulus R, Xiong C, Socher R. Improving Abstraction in Text Summarization. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2018. pp. 1808-1817.

[6] Reisenbichler M, Reutterer T, Schweidel D, Dan D. Frontiers: Supporting Content Marketing with Natural Language Generation. Marketing Science 2022; 41 (3): 441â 452. doi: 10.1287/mksc.2022.1354

[7] Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL); 2020. pp. 7871-7880.

[8] Jin Y, Bhatia A, Wanvarie D, Le PTV. Towards Improving Coherence and Diversity of Slogan Generation. In: Natural Language Engineering; 2022: 1-33.

[9] Hughes JW, Chang K, Zhang R. Generating Better Search Engine Text Advertisements with Deep Reinforcement Learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2019: 2269-2277.

[10] Wang X, Gu X, Cao J, Zhao Z, Yan Y et al. Reinforcing Pretrained Models for Generating Attractive Text Advertisements. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining; 2021: 3697-3707.

[11] Yuan Y, Xu F, Cao H, Zhang G, Hui P et al. Persuade to Click: Context-aware Persuasion Model for Online Textual Advertisement. In: IEEE Transactions on Knowledge and Data Engineering; 2021. doi: 10.1109/TKDE.2021.3110724

[12] Wang Y, Huang H, Yan Y, Liu X. Quality-Sensitive Training! Social Advertisement Generation by Leveraging User Click Behavior. In: Proceedings of the 2019 World Wide Web Conference (WWW '19); 2019. pp. 2045-2055. doi: 10.1145/3308558.3313536

[13] Chambua J, Niu Z, Zhu Y. User Preferences Prediction Approach Based on Embedded Deep Summaries. Expert Systems with Applications 2019; 132 (C): 87-98. doi: 10.1016/j.eswa.2019.04.047

[14] Koehn D, Lessmann S, Schaal M. Predicting Online Shopping Behaviour from Clickstream Data using Deep Learning. In: Expert Systems with Applications; 2020. doi: 10.1016/j.eswa.2020.113342

[15] Schuster M, Paliwal KK. Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing 1997; 45 (11): pp. 2673-2681. doi: 10.1109/78.650093

[16] Aggarwal CC. Neural Networks and Deep Learning. Springer Press, 2018. doi: 10.1007/978-3-319-94463-0

[17] Klein G, Kim Y, Deng Y, Senellart J, Rush A. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In: Proceedings of 2017 Association for Computational Linguistics (ACL), System Demonstrations; 2017: 67-72.

[18] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. The Journal of Machine Learning Research 2014; 15 (1): 1929-1958.

[19] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. In: Proceedings of 3rd International Conference on Learning Representations (ICLR); 2015.

[20] Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: A method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics; 2002: 311-318. doi: 10.3115/1073083.1073135

[21] Lin CY. ROUGE: A Package for Automatic Evaluation of Summaries. In: Text Summarization Branches Out, Association for Computational Linguistics; 2004: 74-81.

[22] Pennington J, Socher R, Manning CD. GloVe: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014: 1532-1543.

[23] Cohen J. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 1960; 20 (1): pp. 37-46. doi: 10.1177/001316446002000104

[24] Altman DG. Practical Statistics for Medical Research. CRC Press, 1990. doi: 10.1002/sim.4780101015