IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 March 2023, accepted 14 April 2023, date of publication 24 April 2023, date of current version 9 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3269505

APPLIED RESEARCH

Generating Campaign Ads & Keywords for Programmatic Advertising

AHMET BULUT[1,2] **AND ABDELRAHMAN MAHMOUD**[3]
[1]Department of Computer Science, Acıbadem University, 34752 İstanbul, Turkey
[2]Search Marketing, Carbon Health Inc., San Francisco, CA 94104, USA
[3]Oredata, Yıldız Technical University, Technopark, 34220 İstanbul, Turkey

Corresponding author: Ahmet Bulut (ahmet.bulut@acibadem.edu.tr)

This work was supported by the Turkish National Science Foundation (Tübitak) under Grant 119E031.

ABSTRACT Experimenting with different ads and keywords is usual practice in search marketing. Advertisers pause underperforming keywords and ads of a search campaign, and replace them with better alternatives. Therefore, new ads and keywords need to be produced easily for effective campaign management. We built GeNN for generating campaign ads and keywords programmatically. GeNN is based on language modeling. Using the existing keywords of a campaign as input, our GPT-2 based generator created novel keywords of good quality with a high number of expected clicks and conversions according to the forecast data provided by Google's keyword planner. Using the product landing page and sample ad copies as input, our GPT-2 based summarizer was able to generate production-ready ads. One of the ads that was tested for two weeks in a real search campaign had a CTR of 6% and converted real users. Finally, we compared GeNN's ad performance with a recent method based on two encoder-decoder RNNs being used in parallel; GeNN outperformed this method.

INDEX TERMS Deep learning, search marketing, text generation.

I. INTRODUCTION

Generative neural networks have gained popularity recently. They were used for generating textual content such as stories, poems, social media posts, and literature reviews [1]. In this work, we applied them in search advertising. Search advertising refers to the business of showing text based ads to search engine users whose search queries match with search keywords chosen by advertisers. A search campaign consists of multiple ad groups. Each ad group contains multiple ads and keywords that are related to each other. Figure 1 shows the structure of a search campaign. An ad contains a marketing message and a link to the landing page of the advertised product.

Advertisers are in constant need to find out prospective ads and keywords for each text-based search campaign. Starting with the initial campaign creation and then continuing with its ongoing management, there is a quest for the exploration of the most fruitful ads and keywords. Once found, they are exploited to their fullest potential. Underperforming ads and keywords are paused, and then replaced with better alternatives. Hence, the experimentation with various ads and keywords is usual practice in search marketing. In a competitive marketplace, the speed and the accuracy of this "search" process is therefore vital.

Google recommends the creation of three to five ads in each ad group. The ads should be both relevant to the advertised product and attractive to many users. The performance of an ad is measured by the rate at which users click through it and visit the linked landing page. In order to determine which of them performs well, various ads are tested for each product. For companies with hundreds of products, preparing these ads and then revising them according to performance needs is labor intensive. With a large number of campaigns to manage, advertisers tend to resort to a handful of ad templates. Generic value propositions are used in creating such templates. The templates are then customized at runtime by dynamic keyword insertion. However, the template based approach tends to result in suboptimal performance due to final ads ranking lower in the ad auction held by the ad broker. The good

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

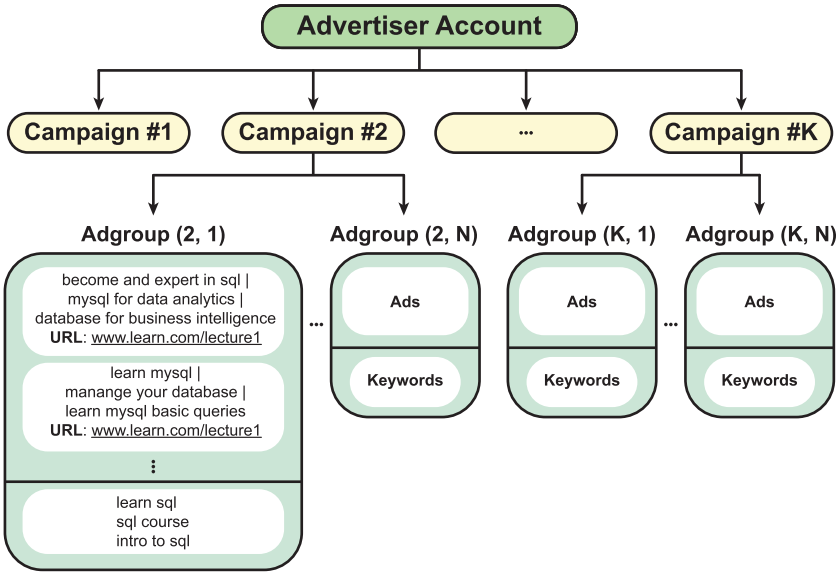

FIGURE 1. Advertisers create campaigns, which consist of multiple ad groups. Each ad group contains a set of related keywords and ads.

news is that one could reduce the burden by generating ads programmatically using the information found in landing pages [2].

Various keywords from broad match to exact match need to be tried and tested. In order to advertise online courses for learning java, the keyword "online java course" is an obvious keyword. Since obvious keywords are generally used by many advertisers, the profit margins on such keywords tend be low. One could alternatively use less obvious keywords such as "java videos for newbies." Among the keywords related to learning java online in our dataset, the obvious keywords had 85 user conversions out of 6, 416 ad clicks, i.e., a conversion rate of 1.32%. The non-obvious keywords had 25 user conversions out of 1, 690 ad clicks, i.e., a conversion rate of 1.48%. The obvious keywords were 12% more expensive than the non-obvious ones. Hence, the non-obvious keywords had a lower cost per user conversion [3]. Since the online marketing budget is limited in most cases, it is important to come up with new keywords that are also more profitable.

A. OUR CONTRIBUTION

There are various studies in the search marketing literature that address either new keyword generation or new ad generation. However, it is not one or the other, but it is both that we need to tackle because a search campaign consists of both ads and keywords. We aimed to fill this gap with our framework called GeNN, which stands for **Ge**nerative **N**eural **N**etworks. The ability to generate keywords easily helps sustain campaign management efforts with least friction. With GeNN, it takes a few lines of code to generate new keywords, which can capture new clicks and conversions. Using the product landing page and sample ad copies as input, GeNN generates production-ready ads. We did a field study using an actual search campaign and got encouraging results.

Contrary to the existing works in the literature, GeNN is based mainly on language modeling. As such, all internal models in GeNN learn input patterns by treating the output as an adaptation of the input. This approach allowed us to address both keyword generation and ad generation under the same hood. For instance, the GPT-2 based generator for generating new keywords and the GPT-2 based summarizer for generating new ads are both language models in GeNN.

We provided the code of our framework as open source for re-producibility and for ease of adoption by the advertising community. It is wrapped into a Python package, which is publicly available at pypi.org/project/genn.

The main findings of our study are as follows:

1) GPT variants performed better than RNN variants on both of the generation tasks.
2) The high text-quality scores of all models implied that the generated ads and keywords were domain-relevant.
3) The forecast data provided by Google's keyword planner indicated that the keywords generated by GPT-2 are expected to get a higher number of unique user clicks and conversions compared to LSTM.
4) For new keyword exploration, we strongly recommend using GPT-2.
5) A select few of the generated ads were deployed in an actual search campaign of a healthcare company. They were tested for two weeks. One of these ads had a CTR of 6% and converted real users. This field study provided supporting evidence for the applicability of GeNN in practice.

II. STATE OF THE ART

In this section, the recent literature on the generation of ads and keywords are discussed.

A. GENERATION OF ADS

In order to generate an ad, relevant text is first extracted from a landing page, and then re-written in order to make it suitable for advertising. It is straightforward to extract unique text from a web page using rule-based information retrieval methods. The result is a short summary consisting of sentences that represent the main points in the original web page. Since the original text is cut in length to meet the character length limitations imposed on ads, the task boils down to text summarization. The key steps in such extractive text summarization systems are:

1) A sentence is represented as a vector of word counts, or as a vector of term frequency over inverse document frequency scores.
2) Each sentence is scored to quantify its relative importance in the whole text.
3) A subset of the sentences are chosen according to their importance to form the final summary [4].

In the selection of the final set of sentences, optimization strategies were shown to perform well [5]. TextRank constructs a web of sentences in order to compute the importance of each sentence iteratively as in PageRank [6]. Thomaidou et al. extracted the promotional text from product landing pages and used a traditional summarization method in order to shorten the summary to obey length limits. A call to action such as ''Order Now!'' was added to the end of each summary [7].

Hughes et al. used two encoder-decoder RNNs in parallel, one for generating the headlines of an ad, and the other for generating the descriptions of an ad. On a dataset containing landing page to ad pairs, their model learnt the association between the two [2]. Çoğalmış and Bulut proposed a bidirectional sequence-to-sequence model with attention mechanism in order to create ads from landing page content [8]. Terzioğlu et al. studied the generation of ads in the context of reinforcement learning. They proposed a generative adversarial network where the generator is an encoder-decoder LSTM with attention, and the discriminator is a single-layer uni-directional LSTM [9]. Using reinforcement learning, Wang et al. showed how the performance of pre-trained models could be improved further in generating high-quality text ads [10]. Yuan et al. studied the classification and the use of persuasive tactics in ad text, and predicted the promotional effectiveness of a given ad [11]. Such quantitative metrics are useful for the performance evaluation of a generative model in addition to the syntactic text quality scores.

B. GENERATION OF KEYWORDS

Joshi and Motwani used text snippets from search query results to construct a directed relevance graph called TermsNet with vertices denoting words and edges denoting the similarity between words [12]. On TermsNet, new keywords were suggested by traversing its edges in search of meaningful word associations. Wordy extended TermsNet by using both query results and web page contents in order to suggest a richer set of keywords [13]. Search engine query logs reveal the association between user queries. Search advertiser keyword logs reveal prospective advertisement keywords. By combining the word co-occurrences and the word associations found in such logs, Google's keyword planner suggests new keywords.

Chen et al. combined traditional query log mining with deep learning to generate new keywords [14]. Using query logs, they built two attention-based RNNs in order to model user behavior and suggest new keywords. He et al. utilized query rewriting for creating variants of an initial seed keyword [15]. In their approach, an encoder-decoder architecture was used to learn the mapping between the original keyword and its variants. Li et al. argued that simple encoder-decoder architectures are not suitable for text generation [16]; Zhou et al. proposed a latent variable network to alleviate this drawback [17]. A generative adversarial network consisting of an encoder-decoder generator and a discriminator RNN was used in generating rare queries [18].

III. METHODOLOGY

A generative model estimates from a given sequence of words the probability of the next word among all possible words. The estimates are higher for words that appear more frequently at that certain position in the training data. For instance, a Recurrent Neural Network (RNN) is able to generate text [19]. It processes input text one word at a time. The output of the network is fed as input to the model in order to capture the temporal context present in the data.

RNNs were shown to work well in modeling user preference [20], [21]. An RNN retains information about the previous tokens in a given sequence, and it pays equal attention to all tokens. However, as the length of the sequence increases, it could forget important information due to its limited memory. A long short-term memory network (LSTM) has a local memory for persisting important information [22]. It captures what information to forget and what to retain at every time step. The gated recurrent unit (GRU) is a simpler RNN variant that changes the control mechanism of an LSTM [23]. With fewer number of parameters, it is faster to train but it encodes less context. Since search keywords consist of a handful of tokens, GRU is a suitable model in our problem setting. The transformer proposed by Vaswani et al. uses a series of attention-based encoders and decoders to process text [24]. The transformer outperformed its counterparts in many applications ranging from machine translation and text generation to abstractive summarization [25]. Contrary to RNNs, the transformer does not process text sequentially, and hence could be run in parallel on GPUs.

Figure 2 shows the pipelines for generating ads and keywords in GeNN.[1] RNNs and its variants including LSTM and GRU are used in generating keywords, and a transformer called GPT-2 is used in generating both ads and

[1] See Appendix V on how to use GeNN.

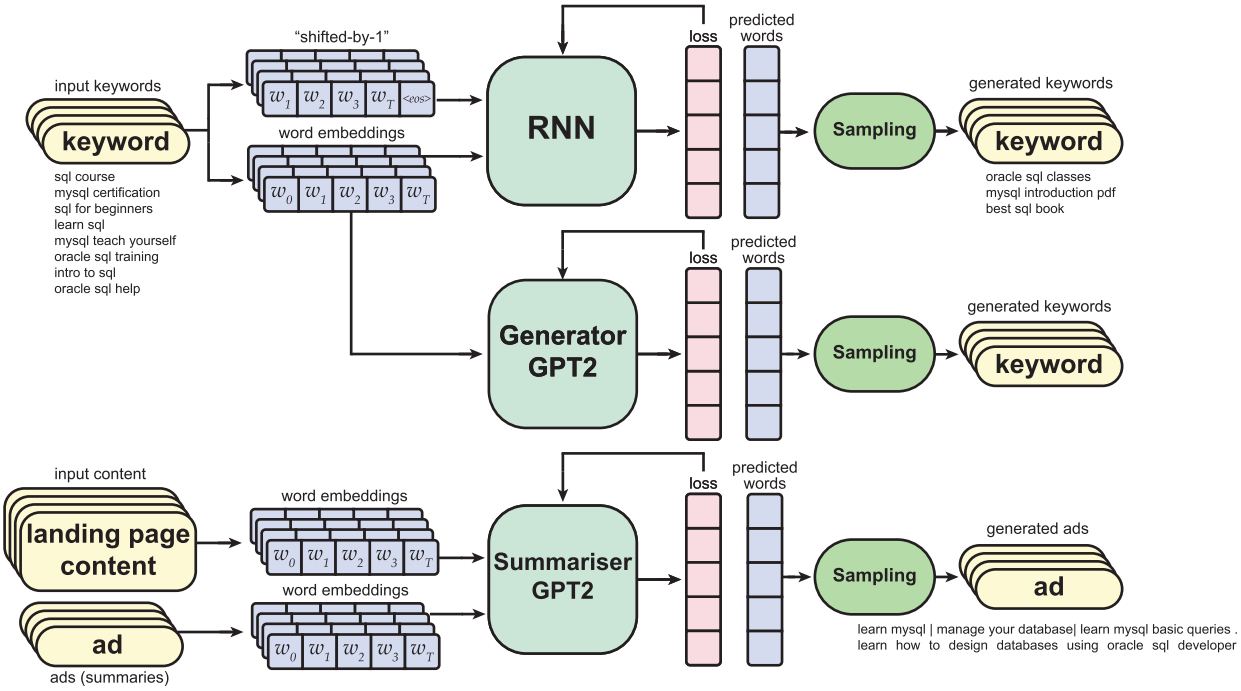

FIGURE 2. The pipelines for generating keywords and ads in GeNN.

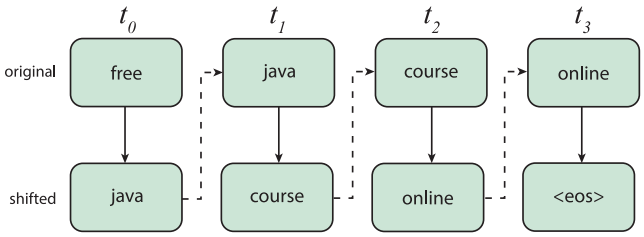

FIGURE 3. The illustration of an RNN with the keyword "free java course online" in the pipeline. At each time step, the keyword is shifted to the left.

keywords [26]. Using GeNN, a suitable GPT-2 model can easily be built for generating text or for summarizing text.

A. AD GENERATION

GeNN is able to generate ads using a GPT-2 summarizer, which is based on GPT-2 Small. Its vocabulary size is $50K$, and it has $117M$ parameters. GPT-2 is used in question answering, text summarization, and language translation simply by providing a task name [27]. Task names are prompts, which describe the task at hand and are provided alongside the input. GPT-2 performed well in zero-shot and few-shot learning using such prompts [28]. The prompt for text summarization is the abbreviation "`TL;DR`" as in:

"`source document TL;DR: summary`"

GPT-2 can recognize the mapping between input-output pairs in a summarization task and minimize the loss accordingly. In our case, the ads datasets are re-formatted as explicit input-output pairs where the input source is the landing page, and the summary output is the final ad creative.

B. KEYWORD GENERATION

The main objective in learning a language model is to predict the next token given the previous tokens. Initially, a seed word has to be provided for the model to start generating text. Since keywords are generally short, we use only one seed. We select this initial seed by random weighted sampling. First, a frequency distribution is obtained using the first token of each keyword. During generation, a seed is sampled from this distribution. This method results in a distribution of generated keywords that resembles the original, and it increases the likelihood of generating non-obvious keywords.

In order to train an RNN for generating keywords, the output loss at a given time step t should be minimized with respect to the output at time step $t-1$. Figure 3 shows the input and output at each time step during the training of an RNN. For a given keyword of length T, the loss is computed as

$$L = -\frac{1}{T}\sum_{t=0}^{T}\sum_{j=1}^{|V|} P(w_{j,t+1}) \log \hat{P}(w_{j,t+1} \mid w_{j,t}) \qquad (1)$$

where w represents a word, $P(w_{j,t+1})$ denotes the probability of the true word, and $|V|$ denotes the cardinality of the word vocabulary.

GeNN is able to generate keywords via RNN variants, i.e., LSTM and GRU, and GPT-2 models. In LSTM and GRU, the

keywords are shifted left by one as in Figure 3. For GPT-2, the task token is "`keyword:`." This token is inserted into the beginning of all training instances before they are fed into the model.

C. SAMPLING

The prediction of the next token at a given time step is not as simple as selecting the token with the highest probability. Such a greedy approach forces the model to repeat itself [29]. The repetition could be alleviated by randomizing the selection. Top-k sampling is one such approach [30]. Instead of always selecting the token with the maximum likelihood, it selects a subset of the top candidates and samples one according to a new normalized probability distribution. Zhu et al. noted that for top-k sampling to match the quality of human text, a large value of k should be used [31]. However as k grows, tokens with low probability could be selected especially when the perplexity of the model is low. In our setting, we set k to 5.

Nucleus sampling adjusts the value of k according to the perplexity of the model [29]. When the sequence to complete is "online free course in . . . ," the next token could be "java" or "javascript." Since the perplexity is low, fewer choices in the candidate pool is better. For the sequence "how to . . . ," the next token could be "write," "learn," or "code." The perplexity is higher, and the model should pick from a richer pool.

IV. RESULT ANALYSIS AND DISCUSSION

The performance of GeNN was benchmarked using standard evaluation metrics found in the language modeling literature. The ads generated via GeNN were deployed in an actual search campaign of a healthcare company for measuring its field performance. Furthermore, the field performance forecast data provided by Google was used in order to quantify the efficacy of keywords generated.

We evaluated the quality of ads and keywords generated both qualitatively and quantitatively. The Bilingual Evaluation Understudy (BLEU) is used in literature for evaluating the quality of generated text [32]. It was shown to reflect human evaluation for text quality [33], [34]. BLEU is defined as the fraction of n-grams in the generated text that also appear in the original data. Hence, BLEU is a measure of precision. Another widely used metric for text evaluation is Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [35]. In contrast to BLEU, ROUGE-n represents the ratio of n-grams found in the original data 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 original text. Together, BLEU and ROUGE quantify textual coherence and syntactic quality.

In order to evaluate the generated keywords further, we estimated their clickthrough rates and compared them with the actual values found in our keywords dataset.

TABLE 1. The samples from our ads datasets. The ads in D^{*}_{rich} and D^{+}_{rich} adhere to Google's ad format where the headlines are pipe-separated, and the descriptions are dot-separated.

Dataset	Landing page text	Ad
D_{rich}	create your own ios 10 apps . apply for ios developer jobs . choose the best design pattern for your app . monetize your skills . upload your own ios apps to the app store	the complete ios swift + objective c developer course . this course will teach you both the swift & objective c programming languages and how to build ios mobile apps
D_{temp}	beginner nikon digital slr (dslr) photography . beginner canon digital slr (dslr) photography . digital photography for beginners with dslr cameras .	the ultimate photography course for beginners . learn all the essentials of photography , and develop your skills to become an ultimate photographer yourself
D^{*}_{rich}	create your own ios apps . apply for ios developer jobs . choose the best design pattern for your app . monetize your skills . upload your own ios apps to the app store .	the complete ios swift | objective c developer course | create ios apps . learn the swift and objective c programming . learn how to build ios mobile apps .
D^{+}_{rich}	create your own ios apps . apply for ios developer jobs . choose the best design pattern for your app . monetize your skills . upload your own ios apps to the app store . the complete ios swift + objective c developer course . this course will teach you both the swift and objective c programming languages and how to build ios mobile apps .	the complete ios swift | objective c developer course | create ios apps . learn the swift and objective c programming . learn how to build ios mobile apps .

A. PRELIMINARIES

1) ADS DATASETS

There are four ads datasets used in this study. Table 1 shows a sample row from each dataset. Specifically,

1) $\boldsymbol{D_{rich}}$ contains 4795 rows. Each row is a pair of landing page content and the corresponding ad.
2) $\boldsymbol{D_{temp}}$ contains 3363 rows. In contrast to $\boldsymbol{D_{rich}}$, the ads in $\boldsymbol{D_{temp}}$ adhere to a small number of ad templates. An example ad in $\boldsymbol{D_{temp}}$ is `online sql course. quality videos by domain experts. why wait ? learn sql now.`, where the word "sql" could be replaced with other words such as "java" or "photography" for creating different ads.
3) $\boldsymbol{D^{*}_{rich}}$ is a modified version of $\boldsymbol{D_{rich}}$, in which the ads are rewritten by a domain expert so that they adhere to Google's following ad format:

   Title #1 | Title #2 | Title #3.
   Description #1 . Description #2

   A text ad in Google has up-to three headlines, each containing 30 characters at most. Headlines are separated by a pipe symbol, i.e. |. In addition, the ad has up-to two descriptions, each containing 90 characters at most.
4) $\boldsymbol{D^{+}_{rich}}$ is a modified version of $\boldsymbol{D^{*}_{rich}}$, in which the landing page title is also included in the landing page content.

2) KEYWORDS DATASET

The search keywords dataset contains 52K keywords in 260 campaigns. There is a row of data per keyword, which

includes keyword match type, campaign and ad group identifiers, ad impressions, ad clicks, click-through rate (CTR), which is the rate of user clicks per ad impression, average cost-per-click, average position, advertisement cost, conversions, cost per converted click, click conversion rate, quality score, bounce rate, conversion value, and return on investment.

3) DATA PRE-PROCESSING

Keywords and ads are first separated into individual words called tokens. Each unique token is then assigned a unique id. This mapping of tokens to ids becomes the vocabulary of the dataset. By simple tokenization, the city name "Los Angeles" is split up into two independent tokens as "Los" and "Angeles." However, they should be treated as a single token. The named-entity recognition would capture such semantic relationships between words and produce a single token instead [36].

4) VECTORIZATION

The simple mapping of words to ids does not encode word context. Therefore, similar words such as "motel" and "hotel" would be as equally distant in the id space as any other word pair in the vocabulary. In order to preserve word context, words could be embedded into a vector space of a fixed dimension where each word is represented as a unique vector of its contexts. In a large text dataset, each word appears in a large number of contexts, and its meaning tends to be reflected in its embedding. The pre-trained word embeddings on large datasets were shown to perform well in sentence classification and language translation [37], [38]. A widely used set of vectors is Global Vectors (GloVe) [39]. An alternative method for learning word embeddings is fastText [40]. GeNN supports both GloVe and fastText.

B. AD EVALUATION

Table 2 shows the input, the ground-truth ad, and the generated ad side by side for a randomly selected landing page from each dataset. The observed quality of the generated ads was encouraging for field deployment. Therefore, a select few of the generated ads were deployed in an actual search campaign of a healthcare company. They were tested for two weeks. One of these ads had a CTR of 6% and converted real users.

All four ads datasets were split into training, validation, and test sets with 70 : 15 : 15 ratios respectively. We tuned GPT-2 on each dataset and reported the final ROUGE and BLEU performance. The results are reported in Table 3. We treated ROUGE as a measure of quality and BLEU as a measure of perplexity. GPT-2 generated phrases that were not present in the training data but were common in the language especially when its perplexity was low. This is because the model was originally trained on a much larger dataset consisting of 8 million public web pages. The domain dependent structure of the data in D^{*}_{rich} and D^{+}_{rich} improved the ad quality significantly compared to D_{rich} as indicated by higher ROUGE scores.

TABLE 2. The input landing page, the ground-truth ad, and the generated ad for a randomly selected data instance from each dataset. The generated ad is grammatically correct, coherent, and obeys the length limits.

Dataset	Landing page text	Ground-truth ad	Generated ad
D_{rich}	write and compose music using proven techniques and tips . create their own music using software , or good old pencil and paper . program music and metaphors . character sketches . writing through canon . form . binary , ternary , and rondo forms . starting with a plan . graphic form representations	music composition bundle : composition film scoring 1 2 . learn music composition techniques through music composition lessons , music composition software , and techniques	music theory, character sketches, and rondo. learn how to use basic musical notation, create your own musical ideas, and much more!
D_{temp}	build responsive real world websites with html5 and css3 . html5 and css3 for beginners : create a website from scratch . learn html in 2 hours ! learn html 5 with hands on example step by step . a web development crash course in html5 and css3	learn html online . self paced html courses . starting from $9 . signup today .	on demand html courses. the best price for the best value. learn at your own pace. start now!
D^{*}_{rich}	work in 3d space . create basic animation . create basic 3d models . work with materials and textures . basic lighting and render setups including image based lighting	3ds max mentor | animation in 3ds max . guided bootcamp on the 3ds max basics from 3d modeling to animation . learn basic lighting and render setups including image based lighting .	the complete guide to 3d space and photoshop | 3d scene for beginners. master basic scene creation and lighting, render with materials and textures.
D^{+}_{rich}	learn how to shoot in manual mode . learn how to get more / less depth of field . learn how to get more / less movement . learn how to talk to your camera , and how it talks to you . digital photography : shooting in manual . learn the skills , settings , and build confidence to shoot your dslr in manual mode for better and more creative photos .	digital photography | learn shooting in manual . learn the skills , settings , and build confidence to shoot your dslr in manual mode for better photos .	digital photography | shooting in manual. learn the skills, settings, and build confidence to shoot your dslr in manual mode for better and more creative photos.

TABLE 3. The ROUGE and BLEU scores of ads generated by GPT-2 on each dataset. The addition of ad titles in D^{+}_{rich} improved the performance of the model dramatically.

Dataset	Sampling	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
D_{rich}	Nucleus	21.4	4.8	21.0	7.3
	Top-5	21.7	5.0	21.7	6.3
D_{temp}	Nucleus	38.1	19.1	38.0	8.8
	Top-5	36.1	18.0	36.0	10.6
D^{*}_{rich}	Nucleus	31.1	10.8	31.3	5.0
	Top-5	31.1	10.2	31.9	5.9
D^{+}_{rich}	Nucleus	60	46.1	62	6.0
	Top-5	55.8	40.1	58.6	5.0

The presence of ad titles in D^{+}_{rich} enriched the source context and improved the ad quality the most. The scores on the template based D_{temp} are not comparable to the rest. Encouraged by the data, the model avoided exploration and

TABLE 4. The ROUGE and BLEU scores comparison of ads generated by GeNN and Hughes et al.'s model [2] on all four ads datasets.

Dataset	Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
D_{rich}	GeNN	**21.4**	**4.8**	**21.0**	**7.3**
	Hughes	15.0	4.0	18.0	0.2
D_{temp}	GeNN	**38.1**	**19.1**	**38.0**	**8.8**
	Hughes	16.0	8.0	16.0	0.01
D^{*}_{rich}	GeNN	**31.1**	**10.8**	**31.3**	**5.0**
	Hughes	21.0	4.0	19.0	0.1
D^{+}_{rich}	GeNN	**60**	**46.1**	**62**	6.0
	Hughes	34.0	19.0	34.0	**6.5**

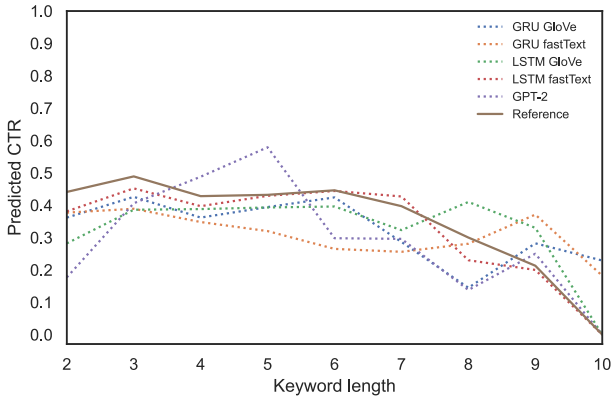

FIGURE 4. The expected CTRs of the generated keywords of varying length. The ground-truth data is shown by the solid line.

instead exploited a small number of ad templates. This resulted in the highest BLEU scores. We compared the performance of GeNN with the model proposed by Hughes et al [2]. As shown in Table 4, GeNN performed better in all cases with the only exception being the BLEU performance on D^{+}_{rich}. The domain specific information present in the dataset is exploited better when the base model is not already pre-trained on the content of public web pages, which contain data from other domains as well. This is a manifestation of the tradeoff between exploration vs. exploitation.

We tested the capability of GeNN in generating different ad copies for the same landing page. A viable model should generate multiple choices for the same input. GPT-2 achieved a high level of generalization on all datasets except on D_{temp} where rigid adherence to ad templates was expected. On average, the GPT-2 model generated 8 distinct ad copies in a batch of 10 ad copies.

TABLE 5. The ROUGE-L and BLEU scores for a batch of 300 generated keywords. Nucleus sampling was used for word selection; fastText was used for word vectorization.

Generative model	ROUGE-L	BLEU
RNN	0.810	0.890
LSTM	0.992	0.999
GRU	0.998	0.989
GPT-2	0.993	0.999

C. KEYWORD EVALUATION

Table 5 shows the BLEU and ROUGE-L scores for a batch of 300 keywords generated via GeNN. The results indicate that

TABLE 6. The best and worst keywords according to Google's performance forecast. The top keywords were shorter and concise whereas the less attractive keywords were relatively longer. The word embedding used was GloVe.

Generative model	Generated keyword	CTR forecast by Google
LSTM	best way to learn javascript	7.0%
	advanced java free online courses	0%
GRU	java programming language	13.6%
	introduction to learn java free	0%
GPT-2	how to learn java online	**19.0%**
	introduce yourself to java	0%

TABLE 7. The monthly performance forecast by Google's keyword planner. The keywords generated by GPT-2 were expected to drive a higher number of clicks and conversions.

Generative model	Clicks forecast by Google	Conversions forecast by Google
LSTM with fastText	10.34	1.00
GPT-2	**720.13**	**55.00**

all models except RNN were able to generate relevant keywords. The performance of a naive RNN model was subpar compared to the other models. The best and worst keywords of the winning models according to their expected CTRs are shown in Table 6. We observed that the best keywords were short and concise whereas the less attractive keywords were relatively longer.

Figure 4 shows how the expected CTR of the generated keywords varies by keyword length. For each keyword generated, its expected CTR was estimated from the true CTRs of its near neighbors. The ratio of the sum of clicks to the sum of impressions of neighbors was used as the expected CTR. The near neighbors of a given keyword were identified using locality-sensitive hashing [41]. In an individual run, each model was trained from scratch, and was allowed to generate a batch of 300 keywords. The average performance across five runs was reported. All models were able to capture the patterns present in the data, but LSTM with fastText traced the true CTRs better compared to the other models. This is because it generated keywords that closely resembled the existing keywords in the dataset. The lack of novelty in LSTM with fastText was confirmed by the monthly clicks and conversions forecasts of Google's keyword planner as shown in Table 7. Its keywords had less additive value over the clicks and conversions received by the existing keywords in the campaign. On the contrary, GPT-2 exploited the keyword space better and created novel keywords of better quality that were expected to generate a higher number of clicks and conversions.

V. CONCLUSION, IMPLICATIONS AND LIMITATIONS

We built GeNN to generate keywords and ads programmatically. The high BLEU and ROUGE-L scores of the generated ads and keywords implied that they were relevant to the

target domain. According to Google's keyword planner, the keywords generated by GPT-2 generator were expected to get a higher number of unique user clicks and conversions than the keywords generated by other models. Therefore, we strongly recommend the use of GPT-2 model for new keyword exploration.

The ads generated by GPT-2 summarizer were coherent, and they adhered to Google's ad format. In a specific field study, we observed that the generated ads performed well in an actual search campaign, and converted real users.

We plan to extend our work by factoring in cost per user acquisition during the exploration of prospective ads and keywords. This is important in practice when the operating marketing budget is limited.

APPENDIX A
HOW TO USE GeNN

For reproducibility, all methods mentioned in this paper are wrapped into a Python package called **Ge**nerative **N**eural **N**etworks. GeNN is a high-level interface for our PyTorch implementations of LSTM, GRU, and GPT-2. It is available at pypi.org/project/genn and can be installed via `pip install genn`. The following code snippets illustrate the usage of GeNN.

The module `Preprocessing` handles parsing files, tokenizing keywords, creating the random seed distribution, and creating shifted input-output pairs. To import the modules:

```
from genn import Preprocessing
from genn import LSTMGenerator, GRUGenerator
from genn import GPT2, GPT2Summarizer
```

A. GENERATING KEYWORDS WITH LSTM

```
# Do pre-processing
ds = Preprocessing("keywords.txt")

# Initialize an LSTM generator with default fastText
lstm = LSTMGenerator(ds, nLayers=1,
                     batchSize=16, epochs=10)

# Train the model
lstm.run()

# Generate 300 keywords
print(lstm.generate_document(300))
```

B. GENERATING KEYWORDS WITH GRU

```
# Use the most frequent first token as seed
ds = Preprocessing("keywords.txt", seedParams={})

# Initialize a GRU generator with GloVe
gru = GRUGenerator(ds, nLayers = 1,
                       embeddingType = "glove",
                       glovePath = "glove.6B.50d",
                       batchSize = 16,
                       epochs = 10)

# Train the model
gru.run()

# Generate 300 keywords using top-3 selection
print(gru.generate_document(300,
                selection = "topk",
                k = 3))
```

C. GENERATING KEYWORDS WITH GPT-2

```
# Initialize a GPT-2 Medium generator
gpt2 = GPT2("keywords.txt",
            taskToken = "keyword:",
            variant = "medium",
            epochs = 10)

# Train the model
gpt2.run()

# Generate 300 keywords using nucleus with p=0.8
print(gpt2.generate_document(300, p = 0.8))
```

D. GENERATING ADS WITH GPT-2

```
# Initialize a GPT-2 summarizer.
summ = GPT2Summarizer("ads.txt",
                      epochs = 10,
                      batch_size = 4)

# Train the model
summ.run()

# Generate 10 ads from a landing page
src_doc = "This is the landing page to summarize."
print(summ.summarize_document(n=10, source=src_doc))
```

ACKNOWLEDGMENT

The authors would like to thank Dr. Kevser Nur Çoğalmış from İstanbul Sabahattin Zaim University for giving them permission to use the ads datasets. The work was completed while Abdelrahman Mahmoud was pursuing his master's degree at Marmara University under the supervision of Dr. Bulut. The authors thank Fahed Şabellioğlu for his outstanding contributions to GeNN at its inception.

REFERENCES

[1] Z. Yang, H. Bai, Z. Luo, Y. Xu, W. Pang, Y. Wang, Y. Yuan, and Y. Yuan, "PaCaNet: A study on CycleGAN with transfer learning for diversifying fused Chinese painting and calligraphy," 2023, *arXiv:2301.13082*.

[2] J. W. Hughes, K.-H. Chang, and R. Zhang, "Generating better search engine text advertisements with deep reinforcement learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 2269–2277.

[3] C. Kim, S. Park, K. Kwon, and W. Chang, "How to select search keywords for online advertising depending on consumer involvement: An empirical investigation," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 594–610, 2012.

[4] A. Nenkova and K. McKeown, "A survey of text summarization techniques," in *Mining Text Data*. Boston, MA, USA: Springer, 2012, pp. 43–76.

[5] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, and C. A. Mehdiyev, "MCMR: Maximum coverage and minimum redundant text summarization model," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 14514–14522, 2011.

[6] C. Mallick, A. K. Das, M. Dutta, A. K. Das, and A. Sarkar, "Graph-based text summarization using modified textrank," in *Soft Computer Data Analytics*. Singapore: Springer, 2019, pp. 137–146.

[7] S. Thomaidou, M. Vazirgiannis, and K. Liakopoulos, "Toward an integrated framework for automated development and optimization of online advertising campaigns," 2012, *arXiv:1208.1187*.

[8] K. N. Çoğalmiş and A. Bulut, "Generating ad creatives using deep learning for search advertising," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 30, no. 5, pp. 1882–1896, Jan. 2022.

[9] S. Terzioğlu, K. N. Çoğalmiş, and A. Bulut, "AD creative generation using reinforced generative adversarial network," *Electron. Commerce Res.*, vol. 323, pp. 134–155, May 2022.

[10] X. Wang, X. Gu, J. Cao, Z. Zhao, Y. Yan, B. Middha, and X. Xie, "Reinforcing pretrained models for generating attractive text advertisements," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (ACM SIGKDD)*, Aug. 2021, pp. 3697–3707.

[11] Y. Yuan, F. Xu, H. Cao, G. Zhang, P. Hui, Y. Li, and D. Jin, "Persuade to click: Context-aware persuasion model for online textual advertisement," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 22, pp. 1938–1951, Dec. 2022.

[12] A. Joshi and R. Motwani, "Keyword generation for search engine advertising," in *Proc. 6th IEEE Int. Conf. Data Mining, Workshops (ICDMW)*, 2006, pp. 490–496.
[13] V. Abhishek and K. Hosanagar, "Keyword generation for search engine advertising using semantic similarity between terms," in *Proc. 9th Int. Conf. Electron. Commerce*, Aug. 2007, pp. 89–94.
[14] W. Chen, F. Cai, H. Chen, and M. de Rijke, "Attention-based hierarchical neural query suggestion," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 1093–1096.
[15] Y. He, J. Tang, H. Ouyang, C. Kang, D. Yin, and Y. Chang, "Learning to rewrite queries," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. (CIKM)*, 2016, pp. 1443–1452.
[16] J. Li, W. Monroe, and D. Jurafsky, "A simple, fast diverse decoding algorithm for neural generation," 2016, *arXiv:1611.08562*.
[17] H. Zhou, M. Huang, Y. Mao, C. Zhu, P. Shu, and X. Zhu, "Domain-constrained advertising keyword generation," in *Proc. World Wide Web Conf.*, May 2019, pp. 2448–2459.
[18] M.-C. Lee, B. Gao, and R. Zhang, "Rare query expansion through generative adversarial networks in search advertising," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 500–508.
[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[20] J. Chambua, Z. Niu, and Y. Zhu, "User preferences prediction approach based on embedded deep summaries," *Expert Syst. Appl.*, vol. 132, pp. 87–98, Oct. 2019.
[21] D. Koehn, S. Lessmann, and M. Schaal, "Predicting online shopping behaviour from clickstream data using deep learning," *Expert Syst. Appl.*, vol. 150, Jul. 2020, Art. no. 113342.
[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[23] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, and H. Schwenk, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. 2014 Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.
[25] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating Wikipedia by summarizing long sequences," 2018, *arXiv:1801.10198*.
[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[27] R. Bommasani, "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.
[28] T. B. Brown, "Language models are few-shot learners," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates Inc., 2020, pp. 13–16.
[29] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2019, *arXiv:1904.09751*.
[30] A. Holtzman, J. Buys, M. Forbes, A. Bosselut, D. Golub, and Y. Choi, "Learning to write with cooperative discriminators," 2018, *arXiv:1805.06087*.
[31] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A benchmarking platform for text generation models," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2018, pp. 1097–1100.
[32] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*. New York, NY, USA: Association for Computational Linguistics, 2002, pp. 311–318.
[33] G. Doddington, "Automatic evaluation of machine translation quality using N-gram co-occurrence statistics," in *Proc. 2nd Int. Conf. Hum. Lang. Technol. Res.*, San Francisco, CA, USA, 2002, pp. 138–145.
[34] D. Coughlin, "Correlating automated and human assessments of machine translation quality," in *Proc. Mach. Transl. Summit IX, Papers*, New Orleans, LA, USA, 2003.
[35] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop ACL Text Summarization Branches Out*, Jul. 2004, pp. 74–81.
[36] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," Tech. Rep., 2017.
[37] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*.
[38] Y. Qi, D. Singh Sachan, M. Felix, S. Janani Padmanabhan, and G. Neubig, "When and why are pre-trained word embeddings useful for neural machine translation?" 2018, *arXiv:1804.06323*.
[39] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[40] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.
[41] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1998, pp. 604–613.

AHMET BULUT received the B.S. degree in computer engineering from Bilkent University, Ankara, in 2000, and the Ph.D. degree in computer science from the University of California at Santa Barbara, in 2005. He is currently a Professor in computer science with Acıbadem University, İstanbul, Turkey. He has more than 15 years of field experience in search marketing acquired at multiple companies, including Udemy and most recently Carbon Health.

ABDELRAHMAN MAHMOUD received the B.S. degree in computer science and engineering from İstanbul Şehir University, in 2020, and the M.S. degree in computer engineering from Marmara University, in 2022. He is currently a Data Scientist with Oredata, Turkey.