T.C.
İSTANBUL SABAHATTİN ZAİM UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Generating Ad Creatives using Deep Learning for Search Advertising

DOCTOR OF PHILOSOPHY

Kevser Nur Çoğalmış

İstanbul
July , 2020

T.C.
İSTANBUL SABAHATTİN ZAİM UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Generating Ad Creatives using Deep Learning for Search Advertising

DOCTOR OF PHILOSOPHY

Kevser Nur Çoğalmış

Supervisor
Assoc. Prof. Ahmet Bulut

İstanbul
July , 2020

THESIS APPROVAL

To Institute of Science and Technology,

This thesis has been accepted by our jury as DOCTORAL THESIS in Department of Computer Science and Engineering.

Chairperson of jury: Assoc. Prof. Ahmet Bulut

Member of jury: Asst. Prof. Abdullah Sönmez

Member of jury: Asst. Prof. Barış Arslan

Member of jury: Asst. Prof. Gökhan Erdemir

Member of jury: Asst. Prof. Hakan Gençoğlu

I certify that the signatures above belong to the professors mentioned above.

Director of the Institution
Prof. Dr. Ali GÜNEŞ

DECLARATION OF AUTHORSHIP

I Kevser Nur ÇOĞALMIŞ, declare that this thesis titled “Generating Ad Creatives using Deep Learning for Search Advertising” for a Doctor of Philosophy degree and the work presented in it are my own. I confirm that, I carefully followed scientific ethics and academic rules, obtained all information in this thesis within the framework of scientific morality and tradition, prepared the thesis in accordance with the rules of thesis writing, cited every direct or indirect quotation, and the works that I have used are those shown in the bibliography.

Kevser Nur ÇOĞALMIŞ

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Ahmet Bulut for his guidance, my family, and my friends for their support and motivation.

Kevser Nur Çoğalmış
2020

ABSTRACT

Generating Ad Creatives using Deep Learning for Search Advertising

Kevser Nur Çoğalmış

PhD., Department of Computer Science and Engineering

Supervisor: Assoc. Prof. Ahmet Bulut

July - 2020

We propose an automatic ad creative generation using a recurrent neural network. A landing page being advertised includes relevant text data, which can be used for generating ads. We formulated the ad creative generation task as a text summarization problem and built a sequence to sequence model with a Long Short Term Memory network. In this network, a source sequence is encoded and represented as a fixed-size vector, which represents the context of the sequence. The decoder uses the source encoding to predict the target sequence.

In order to assess the validity of the proposed approach, we experimented on four different datasets. The empirical evaluations have shown that the proposed model can generate relevant ad creatives on a template-based dataset D_{temp} with moderate hyper-parameters. Training the model with more content increased the performance of the model due to better hyper-parameter tune-up. Since the rich-in-context dataset D_{rich} includes $20x$ more unique words than D_{temp}, we expect that increasing the size of D_{rich} even further enables the network to generate more relevant ads. We observed that when the source and target shared common sequences during the training, the model produced the best ad creatives. Also, GloVe word embedding for input representation improved the performance of the network.

Keywords: Online Advertising, Automatic Ad Creative Generation, Sequence to Sequence Learning

ÖZET

Arama Motoru Reklamcılığı için Derin Öğrenme Kullanarak Reklam Oluşturma

Kevser Nur Çoğalmış

Doktora, Bilgisayar Bilimleri ve Mühendisliği

Tez Danışmanı: Doç. Dr. Ahmet Bulut

Temmuz - 2020

Bu çalışmada "tekrarlayan yapay sinir ağı" kullanarak reklam kreatifi oluşturmayı otomatikleştiren bir model oluşturduk. Reklamı yapılacak olan ürüne ait bir varış sayfası, o ürünle ilgili reklam oluşturmak için kullanılması mümkün olan ilgili metinleri içermektedir. Otomatik reklam kreatifi oluşturma problemini bir metin özetleme sorunu olarak ele aldık ve "Uzun Kısa Vadeli Hafıza Ağları" kullanarak dizi alıp dizi üreten bir model oluşturduk. Bu modelde, kaynak dizisi sabit boyutlu bir vektör olarak temsil edilir ve bu vektör o dizinin tüm kontekst içindeki anlamını ifade eder. Modele ait dekoder ise çıktı dizisini tahmin etmek için, girdi olarak modele sağlanmış olan bu vektörü kullanır.

Önerilen yaklaşımın geçerliliğini incelemek için temel olarak dört farklı veri seti üzerinde testler yaptık. Yapılan testlerin sonuçları, D_{temp} adlı şablona dayalı olarak oluşturulmuş olan bir veri kümesi kullanılarak, hiper parametrelerle oluşturulan modelin, kaliteli reklam kreatifleri oluşturabildiğini göstermiştir. Modelin içerik olarak daha zengin bir veri kümesi ile eğitilmesi ve daha iyi hiper-parametre ile çalıştırılması modelin performansını artırdı. İçerik açısından daha zengin olan veri kümesi D_{rich}, D_{temp}'den 20 kat daha fazla farklı kelime içerdiğinden, D_{rich} veri kümesinin boyutunu artırmanın, modelin daha kaliteli reklam kreatifleri oluşturmasını sağlaması beklenmektedir. Bununla birlikte modelin eğitilmesi aşamasında kaynak ve hedef veri kümeleri ortak diziler paylaştığında, modelin en iyi reklam öğelerini oluşturduğunu gözlemledik. Ayrıca, veri kümesinin sayısal olarak ifade edilmesi için GloVe kelime vektör modeli kullanıldığında, modelin performansı 3 kata kadar arttırıldı.

Anahtar Kelimeler: İnternet Reklamcılığı, Otomatik Reklam Kreatifi Oluşturma

TABLE OF CONTENTS

DECLARATION OF AUTHORSHIP ..... **i**

ACKNOWLEDGEMENTS ..... **ii**

ABSTRACT ..... **iii**

ÖZET ..... **iv**

LIST OF TABLES ..... **vii**

LIST OF FIGURES ..... **viii**

LIST OF SYMBOLS ..... **ix**

LIST OF ABBREVIATIONS ..... **x**

CHAPTER 1
INTRODUCTION **1**

CHAPTER 2
LITERATURE REVIEW **6**

CHAPTER 3
METHODOLGY **10**
3.1. Neural Networks and Recurrent Neural Networks 11
3.1.1. Long Short Term Memory Networks 13
3.1.2. Bidirectional Recurrent Neural Networks 14

CHAPTER 4
EXPERIMENT SETUP **15**
4.1. Dataset 15
4.2. OpenNMT 18
4.3. Google Colab Research 19
4.4. Training 19

CHAPTER 5

EVALUATION METRICS 21

5.1.ROUGE Score 21

5.2.BLEU Score 22

CHAPTER 6

EXPERIMENTS 23

6.1. Embeddings for Input Representation 23

6.1.1. Word2vec 23

6.1.2. FastText 26

6.1.3. Global Vectors for Word Representation (GloVe) 26

6.2. ROUGE Performance of Default embedding on D_{temp} and D_{rich} 27

6.3. ROUGE performance of Word2Vec, FastText and GloVe Embedding on D_{rich} 28

6.4. ROUGE performance of GloVe Embedding on D_{rich} 28

6.4.1. Assessing the Quality of Auto-Generated Creatives 29

6.4.2. Quality of Auto-generated Ad Creatives on Hand Crafted Dataset 32

6.4.3. Kappa Score of Auto-Generated Creatives from D^{+}_{rich} 37

CHAPTER 7

CONCLUSION 40

CHAPTER 8

FUTURE WORK 42

REFERENCES 44

APPENDIX 48

CV 48

LIST OF TABLES

Table 4.1: Sample rows from the template-based training dataset D_{temp} 16
Table 4.2: Sample rows from the content-based training dataset D_{rich} 17
Table 6.1: The effect of varyng model size and optimizer on ROUGE scores 28
Table 6.2: The side by side comparison of ground truth creatives with auto-generated creatives 29
Table 6.3: The effect of using various word embeddings on prediction performance quantified by ROUGE scores 30
Table 6.4: The effect of using GloVe word embedding on prediction performance quantified by ROUGE scores 30
Table 6.5: Two judges' assessment for the quality of auto-generated creatives from D_{rich} 31
Table 6.6: Sample rows from the training dataset D^{*}_{rich} 33
Table 6.7: Sample rows from the training dataset D^{+}_{rich} 34
Table 6.8: ROUGE and BLEU scores of our LSTM model with networks size $|RNN| = 512$ 35
Table 6.9: Auto-generated creatives from D^{+}_{rich} against human-generated creative ..... .36
Table 6.10: ROUGE and BLEU scores for varying the networks size RNN in {256,512} 38
Table 6.11: Two judges' assessment for the quality of auto-generated creatives from D^{+}_{rich} 39

LIST OF FIGURES

Figure 1.1: A query search snapshot 3
Figure 3.1: Our network architecture 11
Figure 3.2: Structure of Recurrent Neural Networks 12
Figure 3.3: Unfolded Recurrent Neural Network 12
Figure 3.4: Unfolded Recurrent Neural Network accross the time-steps 13
Figure 3.5: Structure of Bidirectional Recurrent Neural Network 14
Figure 4.1: The landing page of a piano course 18
Figure 4.2: Illustration of our proposed system 20
Figure 6.1: Word2vec embedding model 24
Figure 6.2: An example of Word2vec embedding model 24
Figure 6.3: CBOW and Skip-Gram Embeddings 25
Figure 6.4: Validation Perplexity of the first 6 models 35
Figure 8.1: Seq2seq GAN Model for Text Summarization 43

LIST OF SYMBOLS

$\|RNN\|$	: Size of the Recurrent Neural Network
D_{temp}	: Dataset with ad text that conforms to a well-defined set of templates
D_{rich}	: Dataset rich in Google context
D^{*}_{rich}	: Dataset with all ads in Google format and pre-processed
D^{+}_{rich}	: Dataset with original ad titles added to the content of landing pages
X_t	: Input sequence
o_t	: Output sequence
S_t	: State in a network
h_t	: Hidden state
i_t	: Input gate
f_t	: Forget gate
c_t	: Cell gate

LIST OF ABBREVIATIONS

AD	: **Ad**vertisement
SERP	: **S**earch **E**ngine **R**esult **P**age
LSTM	: **L**ong **S**hort **T**erm **M**emory
TF	: **T**erm **F**requency
IDF	: **I**nverse **D**ocument **F**requency
ROUGE	: **R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
BLEU	: **B**i**L**ingual Evaluation **U**nderstudy
RNN	: **R**ecurrent **N**eural **N**etwork
BRNN	: **B**idirectional **R**ecurrent **N**eural **N**etwork
NMT	: **N**eural **M**achine **T**ranslation
ADAM	: **Ada**ptive **M**oment Estimation
SGD	: **S**tochastic **G**radient **D**escent
GPU	: **G**raphics **P**rocessing **U**nit
GLOVE	: **Glo**bal **Ve**ctors for Word Representation
CBOW	: **C**ontinuous **B**ag **O**f **W**ords
SEQ2SEQ	: **Seq**uence **to Seq**uence

CHAPTER 1

INTRODUCTION

Internet usage is growing at a rapid rate. At that time of the year, 4.57 billion people are online on the Internet that is more than $\%7$ of the number of users last year (Kemp, 2020a). Internet users search the web for more relevant information, such as finding the best place to go on a vacation, buying a product with the best deal, and many others. Researches showed that online purchases in furniture, fashion, electronics categories were increased by $\%18$ from 2018 to 2019, and average spending worldwide was almost $\$500$ for an online shopper in a year (Kemp, 2020b).

Online advertising also called online marketing or Internet advertising, is a form of marketing and advertising which uses the Internet to deliver promotional marketing messages to consumers. Online advertising is the main source of income for search engines. According to eMarketer's February 2019 report, the total Internet advertising spending was $\$333$ billion in 2019, and it is estimated to reach $\$435$ billion with a steady increase by 2021 (Enberg, 2019). Google, which has the highest market share, has announced in its public earnings report that its search advertising revenue was $\$98$ billion in 2019. The search advertising market is brokered by major search engines such as Google, Microsoft, and Yahoo.

Paid search advertising also provides benefits to sellers. Product owners can increase brand awareness with online marketing and have the opportunity to discover their target customers. They desire to increase online conversions with a qualified ad creative. Ad creative is an object that a potential customer sees after a keyword search on a search engine. With online ad creatives, sellers can call the attention of potential customers faster and probably in a cheaper way than with printed ads.

In order to create a search network 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 the users that get to see the product's ad creative. A SERP (Search Engine Result Page) is the search engine's response to a

user's query. If the user's query matches verbally or semantically with a campaign keyword, then one of the ad creatives in the corresponding search campaign is displayed on the SERP. An ad creative that contains a marketing message and a target URL (the landing page for the advertised product) is what gets shown in the SERP. Figure 1.1 is a snapshot of response in a search engine after a query search.

Ad brokers charge advertisers based on the type of ads used and where those ads are shown. The main payment model in a search network (displaying text-based ads on search engine results pages) is the Cost Per Click (CPC) model. A survey was done by Wordstream showed the average CPC on Google is $\$0.58$, so when an ad takes 100 clicks with that CPC amount, it costs the advertiser $\$100$ (Kim, 2020). The main payment model in a display network (displaying "rich-media" ads on web pages) is the Cost Per a thousand Impressions (CPM) model. While in CPC advertisers are charged when a user clicks on ad, in CPM advertisers are charged when a user sees ad. In our study, we focus on online search campaigns consisting of keywords and text-based ads.

When an ad advertiser decides to make an advertisement on a search engine, she has to determine a well-chosen set of advertising keywords to describe her product. These keywords are used for matching process of an online customer's need with the product that is sold online. After the set of keywords were created, an ad creative is put online, and ad campaign of the product is made live by the ad broker.

When a campaign keyword matches a user's search query, created ad is eligible for entering into a bid-auction alongside with other matching ads. Ads are ranked by their bid times their quality score. The quality score of an ad is calculated using so many other metrics and algorithms by the ad broker. How many minutes a user spends on a landing page of a product directed by an ad creative with an associated keyword, how a visit ends on that landing page, the action of the user performs on that page like buying the product or subscribing for the product affects the quality score of an advertisement.

The advertiser is responsible to manage her advertisement budget correctly, so she must offer a bid for the keyword which can direct most clicks and can rank the ad on the top of the sponsored search result list. Matched ads are displayed in bid times quality score order to the user. Bidding for the highest price does not rank the ad creative of a product at the top of the sponsored search results because ad brokers care about users' satisfaction, experience, and feedbacks from their actions. If the user decides to click on one of the resulting ads, she will be re-directed to the destination URL designated by the advertiser. In return, the ad broker charges the advertiser

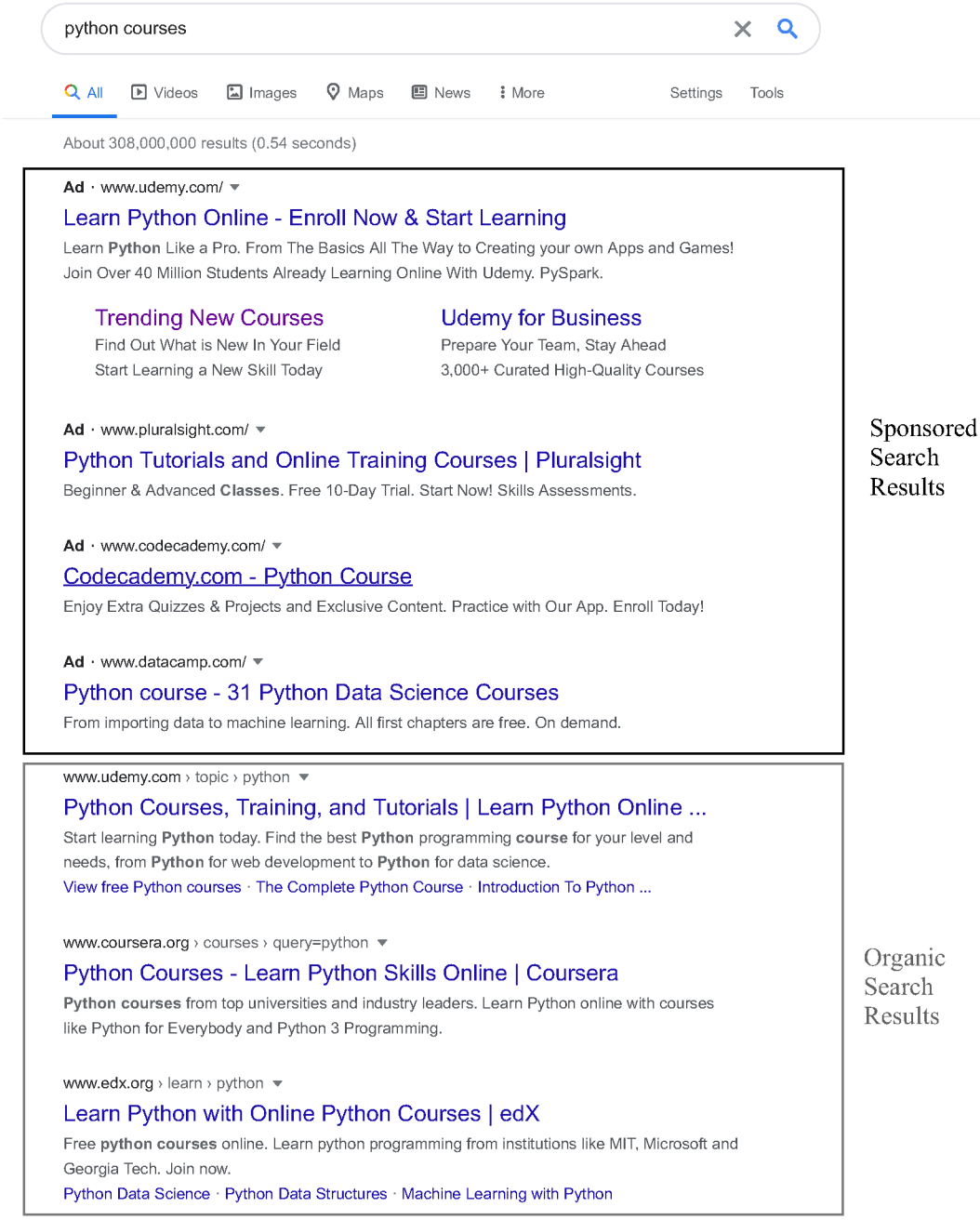

Figure 1.1: The figure illustrates result page of a search engine when a user searches for a keyword phrases or query. Top of the page listed sponsored search results that designed with ads, and rest of the list is organic page results.

depends on which payment model is used.

In summary, the search engine as the ad broker determines whether a keyword K in an online search campaign matches with a user query Q when the query is submitted. If the keyword K and the query Q are related (relevant to each other), then the corresponding ad creative 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 the collaboration of online marketers, creative directors, copywriters, and possibly linguists. And yet many parts of this craft are still manual and therefore not scalable, especially for large e-commerce companies that have big inventories, and big search campaigns. Furthermore, the craft is inherently experimental, which means that the marketing team has to experiment with different marketing messages from subtle to strong, with different keywords from broadly relevant (to the 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 the marketing budget being spent with poor returns. Based on a survey, 75% of companies have difficulties finding an expert for optimizing a landing page and an ad creative campaign (of Ecommerce Statistics, 2020).

For rapid experimentation, new and alternative keywords and creatives have to be generated automatically. In this study, we focused on the generation of ad creatives alone and proposed a solution based on deep learning. To help the advertiser, we formulate ad generation process as a text summarization task and figure an ad creative as a summary of a landing page of the product. We automate the ad creation process by using a recurrent neural network. We assume that the landing page has enough information about the product in order to create a relevant ad creative. The main contributions of our study are as follows:

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

ii. We created an advertisement dataset by using landing pages of products in an online learning platform.

iii. Feeding the network with more and diverse data improved the model's performance by a factor of 3.

iv. Using a landing page as source and allowing word/sequence overlap with an advertisement improved the quality of the generated ads.

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

vi. By creating a less complex network, we accomplished almost the same success rate with a more complex network.

The rest of this paper is organized as follows: the related works found in literature are presented first. Next, we describe our methodology. Subsequently, we present our experimental setup and the results we obtained. Finally, we conclude with key takeaways for researchers, and practitioners interested in automated ad generation.

CHAPTER 2

LITERATURE REVIEW

There is an extensive body of research on keyword generation rather than the creative generation of textual ads. A set of keywords is generated by an advertiser to create a match link between a search query and an ad creative. When a keyword or a bid phrase and a query are relevant, a search engine directs a user to a landing page of a product. Ravi et al. used both a translation model and a language model to generate relevant bid phrases automatically (Ravi et al., 2010). They converted landing pages to a bag of words vectors to extract essential information for the keyword generation task and created candidate keywords by using several algorithms with a translation model. Since keywords are generated automatically, with another translation model, they ranked these keywords to attract more users' attention. They also demonstrated that 96% of the landing pages had at least one ad phrase that was not present on the landing page.

Joshi and Motwani noted that bidding on non-obvious yet relevant words (ios phone instead of iPhone) is a more economical (Joshi & Motwani, 2006). In order to find non-obvious words, they used directed semantic similarity graphs and search engines to find out the relevancy of terms. They showed that the non-obvious keywords received as many user clicks as their expensive alternatives and defined a measurement as "non-obviousness" to measure the performance of relevant, but non-obvious and less expensive keywords. They used Inverse Document Frequency (idf) score to eliminate words that are general and common (Manning, Raghavan, & Schutze, 2008).

Yih et al. extracted prospective keywords from web pages (Yih, Goodman, & Carvalho, 2006). They used several sources of information such as meta tags, and title section of a web page and search query logs of a search engine to achieve the number of occurrences for each word. They run logistic regression algorithm with 40 features including term frequency (tf) and idf values to identify suitable keywords and showed the algorithm's performance depended on the quality of the chosen features.

Choi et al. used the landing page content for ad selection (Choi et al., 2010). They identified parts of a web page that can be used for this purpose. They used the tf-idf algorithm for the selection (Manning et al., 2008) and applied an extractive summarization method. The authors proved that their method of selecting the landing page region for an ad context results in $\%8.5$ performance improvement in the baseline method, and using the content of the landing page can enhance ad quality.

Berger and Mittal argued that extractive text summarization methods such as Choi's would not perform well since landing pages do not share a similar structure with ad creative (Berger & Mittal, 2000; Choi et al., 2010). To address this issue, they proposed a non-extractive summarization model based on statistical machine translation. Sun et al. summarized web pages by using click-through data obtained from a search engine, thereby linking web pages with user queries (Sun et al., 2005). The authors stated that the words of a query contains the essence of the web page content and can be used as its summary.

Fujita et al. suggested a system that uses both search query logs and promotional text contents in order to generate bid phrases, ad titles, descriptive texts for ads (Fujita et al., 2010). Their system created bid phrases for each selected shop. For each keyword, a title was created based on a template. Finally, the system created nine types of shop-specific descriptive texts which do not depend on the generated bid phrases and titles. They looked at the precision and CTR (click-through rate) of listing ads to measure the efficiency of the proposed system (Help, 2020).

Thomaidou et al. built a system that extracts the textual content of a landing page and uses that information to create keywords and ad creative text by using a summarization technique (Thomaidou, Liakopoulos, & Vazirgiannis, 2014). They extracted the most important sentences from the landing page via a Bayesian classifier and with the help of ad-texts they designed a campaign creation module. Their system also used a genetic algorithm for budget optimization of an online marketing campaign to maximize profit.

Extractive summarization techniques extract the most important sentences from the text as a summary while abstractive summarization methods can create novel words, and sentences that are not in the original text. Extractive summarization is easier, more accurate, and grammatically correct because it copies phrases or sentences from the original text. On the other hand, abstractive summarization is harder because phrases that do not occur in the original document need to be generated. An abstractive model understands the whole text and can generate summaries closer to what humans can do (See, Liu, & Manning, 2017).

Traditional extractive summarization methods cannot produce phrases that do not occur in the source document. Fortunately, a sequence to sequence model with Long Short Term Memory (LSTM) can paraphrase the source text in order to perform abstractive summarization (Nallapati, Zhou, Gulcehre, Xiang, & Nogueira dos santos, 2016). Nallapati et al. stated that even there are similarities between an abstractive summarization problem and a machine translation problem, they are not the same because of fixed-length of an output. In abstractive summarization, the idea is expressing key concepts of a given document with a short summary whether the source is long or short. The researchers used similar neural networks and achieved state-of-the-art performance on two different corpora. They argued that their model, which has attention on the hidden state as an encoder, solved some critical problems in summarization such as modeling keywords, and emitting words that rarely occur in training. They achieved 25.5 in terms of ROUGE score with their baseline model and increased the ROUGE score to 35.5 after improving their model.

Cho et. al used an RNN encoder-decoder architecture to create an English to French translation framework (Cho et al., 2014). Their system used a variable-length source sequence as an input representation and an encoder converted these sequences to a fixed-length vector. The decoder in the network converted the fixed-length vector to variable-length sequences as output. Their model aims to maximize the conditional probability of an output sequence given an input sequence. They achieved a BLEU score of 33.3 in their baseline methods.

Sutskever et. al used the same dataset with Cho's study in their work (Sutskever, Vinyals, & Le, 2014). They created multi-layered LSTM to perform the translation and represented input sequences with a fixed-sized vector in the encoder stage. Another LSTM layer was used in the decoder stage to generate auto-translated sequences. They remarked that LSTM can work on long sentences without any problem. Their work achieved a BLEU score of 34.8.

Bahdanau et al. discovered that assigning weighted attention to each input would yield more accurate results and shed light on why the algorithm outputs a particular word (Bahdanau, Cho, & Bengio, 2014). They pointed out any encoder-decoder system suffers from feeding a network with a fixed-length vector. Their proposed method weights an important part of the source data to predict the target sequence. This is a very important step for interpreting the results of a neural network.

Facebook AI researchers proposed abstractive sentence summarization with a sequence to sequence model (Rush, Chopra, & Weston, 2015). Using the attention-based model, they were able to interpret the result of the summaries. They used several

encoder mechanisms. In the basic model, input sentences were represented as a bag of words and a convolutional encoder to improve the performance of bag of words representation by considering local interactions between words. They also created an attention-based encoder to smooth representation of the input. They achieved the highest ROUGE scores as 28.18 and 31.00 on two different corpora.

Kryściński et al. proposed a novel sequence to sequence model that supports the decoding process with an additional pre-trained language model in order to extract more concise paraphrases for text summarization (Kryściński, Paulus, Xiong, & Socher, 2018). Adding this external component helped the decoder for focusing on compacting and extracting the source document. They also offered a metric to increase abstraction level in the summaries by combining policy gradient and maximum likelihood estimation.

Lee et al. proposed a control mechanism on sequence to sequence model using the generative adversarial network (GAN) (Lee, Gao, & Zhang, 2018). In their study, a GAN based model was used in expanding and diversifying the keywords. They created a sequence to sequence generator to create keywords from a user query and an RNN discriminator to improve the correctness of the generated keywords.

CHAPTER 3

METHODOLOGY

We generate an ad creative from the landing page text targeted by the ad. We formulate our ad creative generation task as a text summarization problem. Advancement in language modeling with deep learning outperformed the standard translation algorithms (Wu et al., 2016). Neural machine translation and summarization algorithms perform well given that the amount of data is large enough. They both use sequence to sequence models because input size and output size are not fixed in most cases.

We create a sequence to sequence model using an encoder-decoder recurrent neural network (RNN) architecture. We map the input sequence to the output sequence using fixed-sized length vectors. Input sequences of varying length are read by the encoder and encoded into fixed-sized vectors. These vectors contain context information present in the inputs. The decoder emits the predicted results using the output vectors of the encoder.

We used a variant of RNN that is an LSTM network for both of the encoder, and the decoder cells and word embedding to vectorize our corpus. For practical reasons, we initialized embedding with random weights, used word vectors of 500 dimensions, and then let the algorithm learn the weights. Bidirectional LSTM is used at the encoding phase to provide all information available in the past as well as the future by reversing the order of the input (Schuster & Paliwal, 1997). For soft-alignment between each input word and output word, the attention mechanism is added on top of this model. At the final step, we use softmax sampling to compute probabilities over the target vocabulary.

Figure 3.1 depicts the structure of a summarization model consisting of encoder, and decoder layers where word embedding is used to represent input sequences. On the top of encoder, there is an attention mechanism; on the top of decoder, a softmax layer is applied to calculate overall word probabilities.

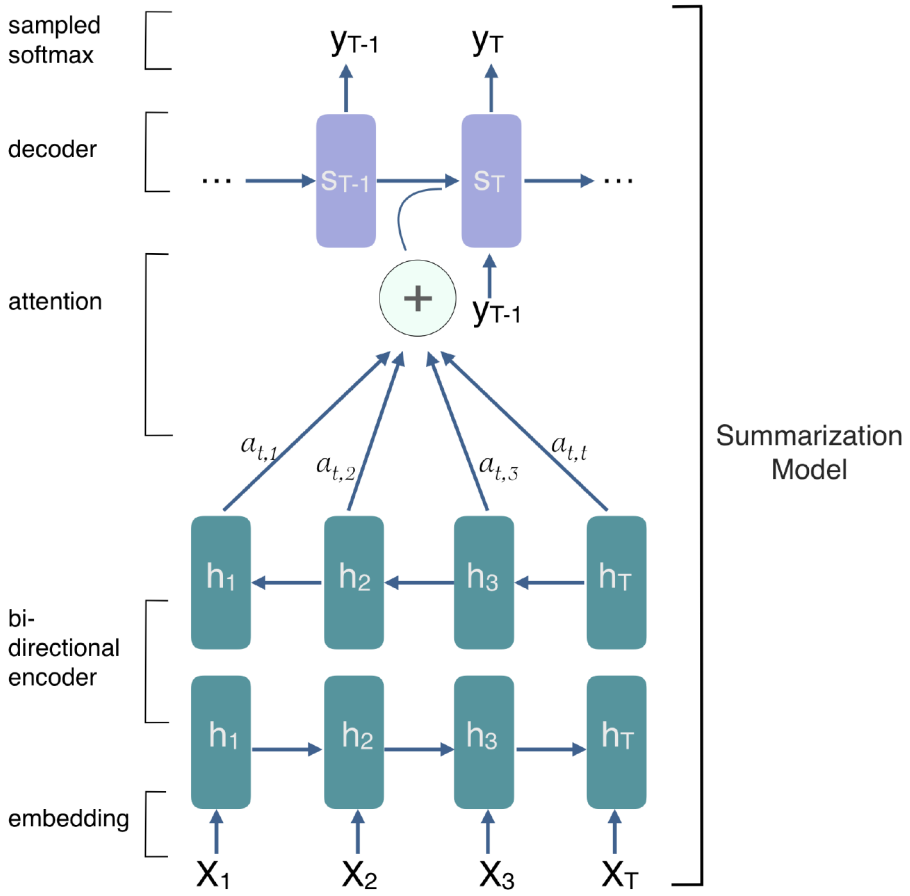

Figure 3.1: The figure illustrates our network architecture where $X_t s$ are sequences from input document, $h_t s$ are hidden states in the encoder phase, $a_t s$ are part of the attention mechanism, $S_t s$ are states in the decoder. $Y_t s$ are the output sequences that summarize a given input document.

3.1 Neural Networks and Recurrent Neural Networks

A neural network is fed with numerical input and produces an output or multiple outputs according to its internal wiring. They have to be fed with a fixed size input vector and produce a fixed size output vector.

Traditional neural networks do not have memory and can not remember the decisions made in a previous step. This is where recurrent neural networks (RNN) can be a solution. An RNN can be trained with a sequence of vectors with different sizes which is known as $seq2seq$ modeling and has a feedback loop as shown in Figures 3.2 and 3.3.

RNN is a variant of neural networks where the input of the current state depends on the output of the previous state (Karpathy, Johnson, & Fei-Fei, 2015). More formally,

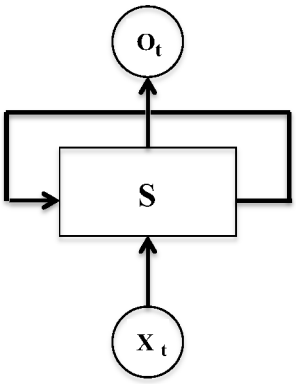

Figure 3.2: Basic structure of recurrent neural network with a single node where X_t is an input, S is a network cell, O_t is an output.

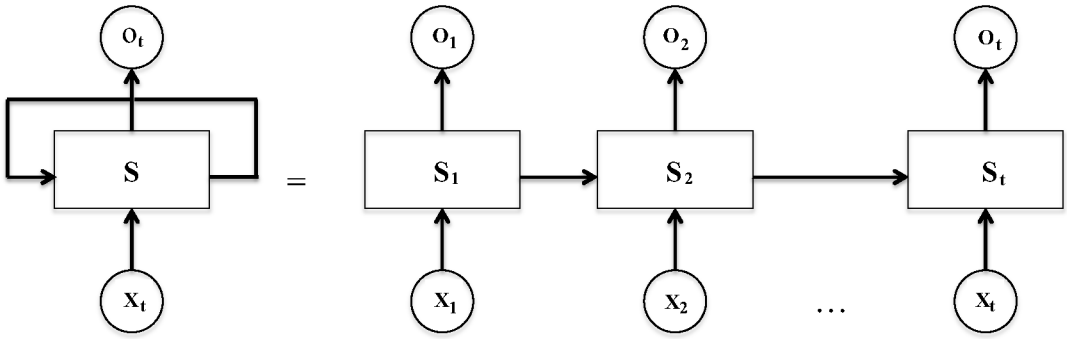

Figure 3.3: Unfolded recurrent neural network where X_t is an input, S is a network cell, O_t is an output.

$$S_t = \sigma(WS_{t-1} + X_tU) \tag{3.1}$$

$$o_t = softmax(S_tV) \tag{3.2}$$

where X_t is an input word embedding, which is a representation of each word in the dataset, and S_t is the hidden state that corresponds to the "memory" of the network at time step *t* and is calculated from the input at time step *t* and the hidden state at time step *t-1*. Finally, o_t is the output of the current state at time step *t*.

In the forward pass, S_t is accumulating the information through each time-step while model parameters (U, V, W) stay the same. In the backward pass, model parameters are updated with the gradient descent algorithm. Figure 3.4 (LeCun, Bengio, & Hinton, 2015) shows the internals of the network. Each neuron gets its input from another neuron in previous time step. In each time step, the parameters (U, V, W) stay the same.

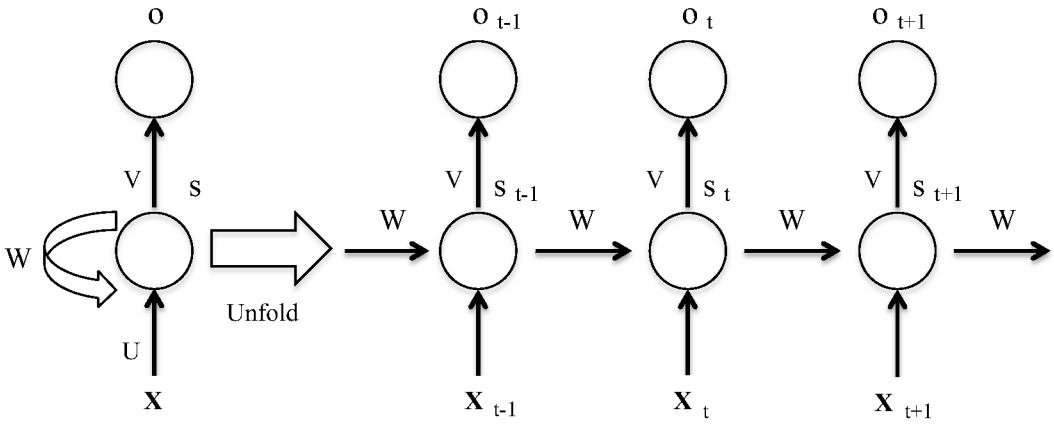

Figure 3.4: Unfolding recurrent neural network across the time-steps where X_t is an input, S is a network cell, O_t is an output.

3.1.1 Long Short Term Memory Networks

An RNN can remember the next word, but we may need to know more than just the next word. The network should know about the context to widen its memory and generate creatives that fit well to the context, but may include different words from any direct input content. Long Short Term Memory (LSTM) networks keep long term dependencies to remember the context. An LSTM is divided into two parts as an encoder and a decoder. Information from the input data is transferred as a context from encoder to decoder. Context maintaining history is computing with a softmax function. In the decoder part, each output is an input to the next node in the sequence. For an LSTM network, we have,

$$i_t = \sigma(U_i x_t + W_i s_{t-1} + b_i) \tag{3.3}$$

$$f_t = \sigma(U_f x_t + W_f s_{t-1} + b_f) \tag{3.4}$$

$$\tilde{c_t} = \tanh\left(U_c x_t + W_c s_{t-1} + b_c\right) \tag{3.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{3.6}$$

$$o_t = \sigma(U_o x_t + W_o s_{t-1} + b_o) \tag{3.7}$$

$$s_t = o_t \odot \tanh c_t \tag{3.8}$$

where i_t refers to an input gate, f_t refers to a forget gate and $\tilde{c}_t$ refers to an input

modulation gate and these gates are formulated as in RNNs (Zaremba, Sutskever, & Vinyals, 2014; Olah, 2015). According to the equation 3.6, a previous cell state $c_{t\text{-}1}$ is modulated by the forget gate f_t, and the input is modulated by the input modulation gate $\tilde{c}_t$ in order to calculate the current cell state c_t. With these gates, LSTM learns how much of the information preserved in previous states should be forgotten and how much of the given input should be emphasized.

3.1.2 Bidirectional Recurrent Neural Networks

Bidirectional Recurrent Neural Networks (BRNN) has an extended version of an RNN structure (Schuster & Paliwal, 1997). A BRNN allows training the model in a forward and a backward pass. Basically, each state is divided into two cells as forward and backward components as shown in Figure 3.5. Each output in each state at time t is calculated with the activation function of $\overrightarrow{S}$ and $\overleftarrow{S}$. This structure makes it possible to take information from the future. For a text generation task, a BRNN makes it more likely to learn next words correctly in a context where even two given sentences start with the same sequences.

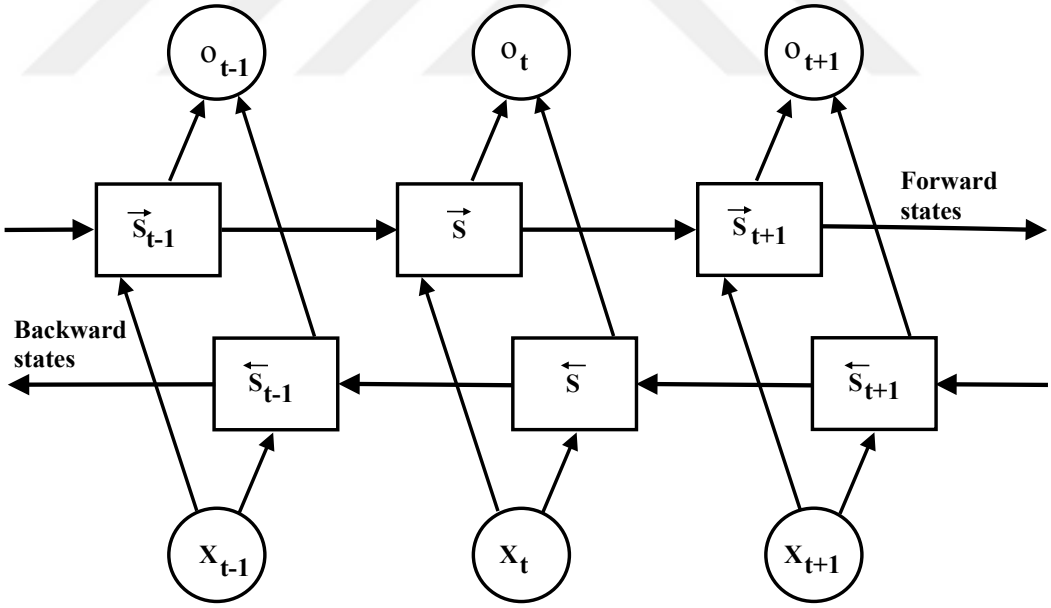

Figure 3.5: Structure of a bidirectional recurrent neural network (brnn) where X_t is an input, S is a network cells in forward and backward passes, O_t is an output.

CHAPTER 4

EXPERIMENT SETUP

4.1 Dataset

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} since it contains ad text that conforms to a well-defined set of templates. In D_{temp}, the source text is a set of product titles for a given search keyword, and the target text is an actual ad creative created by advertisers. In this dataset, the target text contains generic marketing phrases that occur in many ads, e.g. "this is the best place to learn", and "start learning at your own pace". Two samples of source and target documents from D_{temp} are shown in Table 4.1.

The second dataset is called D_{rich} since this dataset is richer in content and context. We used the custom description of each course present in its landing page as the source text and the page title as the target text. D_{temp} has 3363 entries while raw D_{rich} has 4795 entries. The source, and target document samples from D_{rich} are shown in Table 4.2. We split both datasets into train, validation, and test sets with a split ratio of $70 : 15 : 15$. There are 2632, and 408 unique words in the source, and the target documents respectively in D_{temp}, while there are 11172, and 7290 unique words in D_{rich}.

Figure 4.1 is a snapshot from a landing page of a course that we used to create D_{rich}. Title and sub-titles of the course are selected as ad creative for this course. A landing page of each course contains a section named "What you'll learn" with several phrases in order to explain the content of the course and learning outcomes after the course is completed. We used "What you'll learn" sections as source documents for compiling D_{rich}.

Source Document #1	**Target Document #1**
online marketing business : create your agency , step-by-step .	
internet marketing classroom .	
quick and dirty marketing secrets - get viral website traffic .	
marketing : how i skyrocket sales by 633% in 12 hours of work .	
rapidly dominate social media.	internet marketing .
how to start an online business in 30 days - step by step .	the best price for the best value .
digital marketing : networking and entreprenuership guide .	learn at your own pace . start now !
imagine the possibilities for an internet business .	
video editing with screenflow for internet marketers .	
drive thousands to your website with zero seo knowledge .	
teach online : how to market and sell online courses .	
internet marketing plan + free software .	
Source Document #2	**Target Document #2**
corporate finance - a brief introduction .	
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 .	improve finance skills .
start-up financial modeling for non-finance professionals .	this is a great place to learn .
power bi : create a power bi finance dashboard with tutorials .	improve your finance skills today .
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 .	
Source Document #3	**Target Document #3**
build responsive real world websites with html5 and css3 .	
full responsive website design with solid html5 & pure css3 .	
create amazing website using bootstrap 4 , html5 & css3 .	
html5 and css3 for beginners : create a website from scratch .	
html5 masterbuild superior websites & mobile apps new 2017 .	on demand html5 course .
a web development crash course in html5 and css3 .	the best price for the best value .
learn html 5 with hands on example step by step .	learn at your own pace . start now !
html5 and css3 essential training .	
build modern responsive website with html5 , css3 & bootstrap .	
html5 specialist : comprehensive html5 training .	
introduction to html5 : learn to create html5 websites .	
html5 and css3 build modern responsive websites	

Table 4.1: Sample rows from the template-based training dataset D_{temp}. The source is the titles of twelve relevant courses and the target is the corresponding ad creative.

Source Document #1	**Target Document #1**
you will be able to design awesome professional websites in photoshop . you will be able to design ui elements in photoshop . you will learn various cool tips tips and tricks in photoshop for web designing you will get the psd templates designed in this course . you will learn the step by step procedure of web designing	the ultimate web designing course in photoshop. over 20 psd files, 3 mega web design projects included.
Source Document #2	**Target Document #2**
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 .
Source Document #3	**Target Document #3**
you will learn how to communicate meaning and purpose to your team about your company or department's mission . you will know the best way to express genuine caring about people in a professional setting . you will understand how to maintain an external focus so that you aren't blindsided by competition . you will have a framework to analyze the big picture so you remain ahead of industry changes . you will gain the skills for clear , open and honest communication with your team	top five leadership behaviors . strong leaders don't just happen : learn these critical leadership behaviors to gain the trust of your team .

Table 4.2: Sample rows from the content-based training dataset D_{rich}. The source corresponds to the relevant text found in the landing page of the selected course and the target is the title of the corresponding landing page.

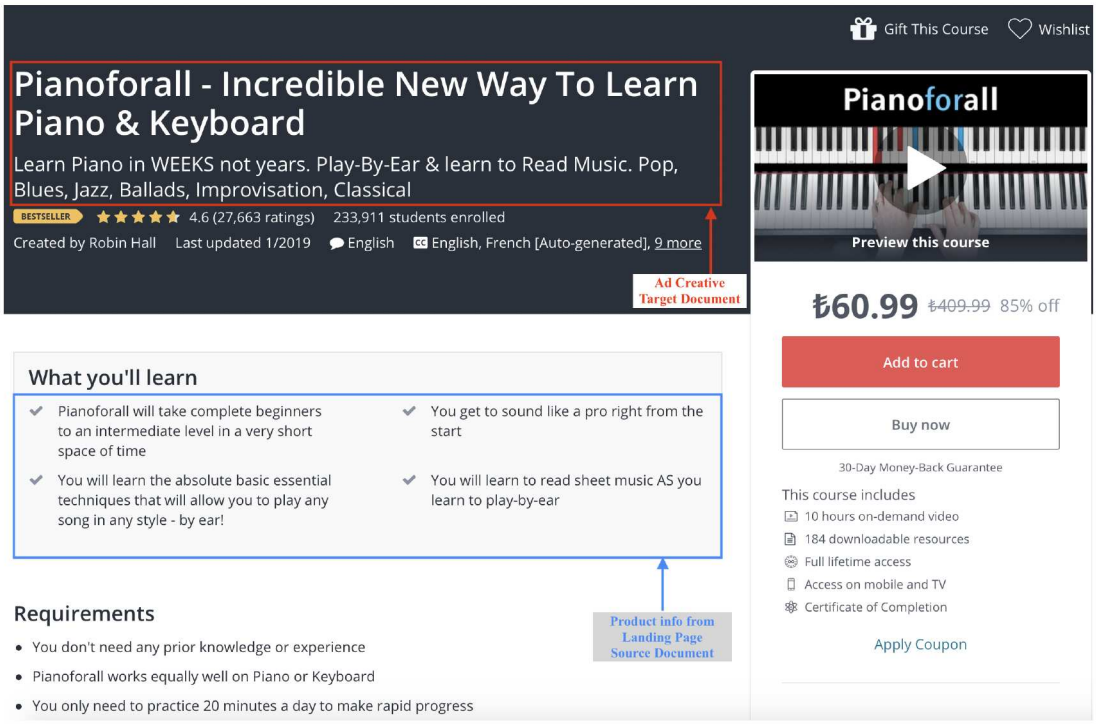

Figure 4.1: The landing page of a piano course used in the generation of our dataset D_{rich}. The part in the red rectangle was used as "target document" and the section in the blue rectangle as "source document".

4.2 OpenNMT

OpenNMT is an open source Neural Machine Translation (NMT) toolkit to perform sequence to sequence modeling (Klein, Kim, Deng, Senellart, & Rush, 2017). OpenNMT is available in LuaTorch, PyTorch, and Tensorflow. The toolkit is mainly used for language translation and summarization. With OpenNMT we can create different types of neural networks such as a recurrent neural network, a bidirectional recurrent neural network, and a convolutional neural network. OpenNMT provides opportunities to use different kinds of word embeddings to represent input data. With several parameters, the number of layers or size of RNN can be set. Also choice of parameters makes it possible to create batches with a given number of samples and epochs to train a model with an epoch at each step, so GPU can be used for experiments to decrease computation time and resource allocation. We used Python version of OpenNMT for building our models.

4.3 Google Colab Research

Google Colab Research is a free cloud service, which supports running a Python code on a GPU. We ran all our experiments on Colab Research Notebooks.

4.4 Training

There are multiple hyperparameters to fine-tune in any sequence to sequence network. We chose moderate hyperparameters to create a small network with a small number of layers for D_{temp} because the data is relatively simple and includes many repetitive sentences. However, using the same hyperparameters is not sufficient for D_{rich}, because exploring the connections between sentences in this dataset is not as straightforward as in D_{temp}. Therefore, we performed hyperparameter optimization in order to obtain the best results.

Before training a model, pre-processing was done on the model to set some parameters about input and output data. Model pre-processing created vocabulary dictionary from the datasets. Embeddings were converted to torch to fit in Python. Also, we set the maximum source sequence length to 360, and target sequence length to 100 after considering the longest input, output, and Google's Ad standards. For training the model, dropout value was set to 0.3 to specify the percentage of nodes to be ignored temporarily in the hidden states in order to avoid overfitting (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). We set the beam size to 5.0 in order to audit how deep the search goes in a search tree. In each 250^{th} step, a checkpoint and a model were saved with validation perplexity and validation accuracy for that model.

We built a bidirectional recurrent neural network using an LSTM with an attention mechanism as shown in Figure 4.2. We conducted experiments using "stochastic gradient descent (sgd)" and "adam" optimizers (Kingma & Ba, 2014). Researchers stated that for attention models and recurrent networks adam optimizer works better than sgd, but because sgd is simpler and works faster, we also use sgd as optimizer in our first experiments. We varied the size of hidden states in each RNN from 128 to 512 neurons. In each model, the decoder and the encoder have two hidden layers.

Since the output is represented as a one-hot vector, the number of neurons at the output layer is equal to the size of the output vocabulary V_{o}. Each output neuron represents the probability of the word. If we would always choose the most probable one, we might bail out prematurely, an approach called greedy decoding. If we think

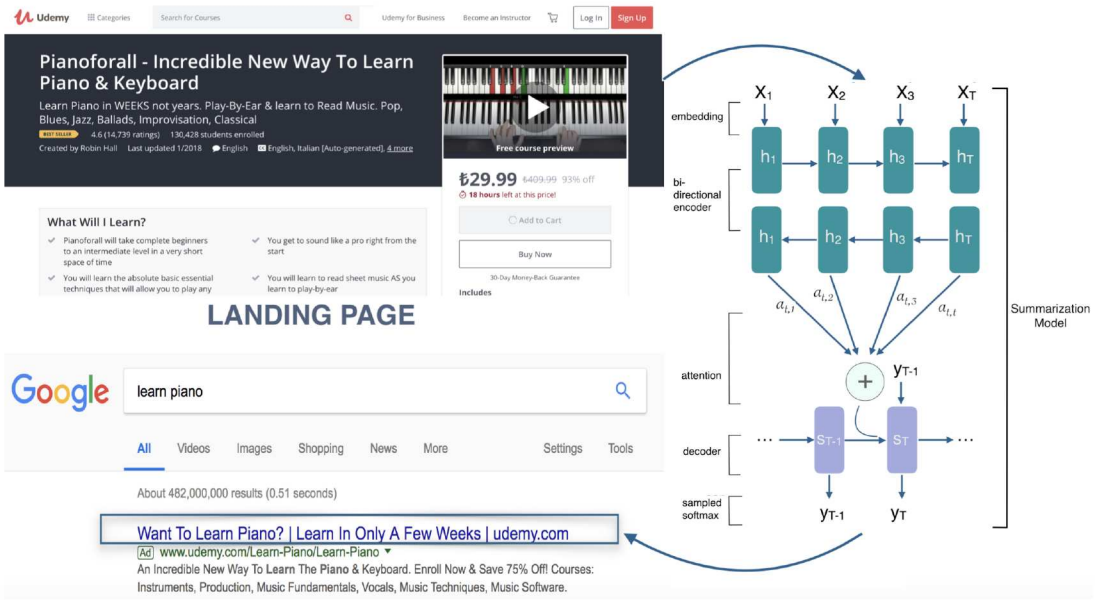

Figure 4.2: The figure illustrates our proposed system. An input landing page shown in the upper left is converted to source sequences that are represented as $X_t s$. X_ts are encoded in the hidden states $h_t s$. After the attention mechanism is applied, the encoded input is decoded to $S_t s$. Finally, softmax is applied to generate outputs $Y_t s$, the sequences that are used as final "Ad Creatives".

all the possible outcomes as paths of a tree, then the depth of this tree would be the number of output words l_o and each node would have V_o branches. Choosing the most probable word at each time step would not give the best possible output. Since there are ${V_o}^{l_o}$ possible paths, the time complexity is exponential and not scalable. Fortunately, BeamSearch decoding reduces the number of paths to a scalable $k \times V_o \times l_o$ where k is the number of the best possible words.

CHAPTER 5

EVALUATION METRICS

In summarization tasks, we need formal mechanisms to evaluate the quality of auto-generated summaries. Some metrics only address syntactical quality and consider whether generated summaries are grammatically correct or whether the sentences are well-structured. There are also metrics to evaluate the quality of the generated content such as cosine similarity, longest common subsequence, unit overlap, Recall-Oriented Understudy for Gisting Evaluation (ROUGE), and Bilingual Evaluation Understudy (BLEU) (Lin, 2004; Steinberger & Ježek, 2012). Content-based metrics evaluate summaries by comparing the words of auto-generated sentences and human-generated sentences. We used ROUGE and BLEU scores to evaluate our auto-generated creatives.

5.1 ROUGE Score

Recall-Oriented Understudy for Gisting Evaluation named as ROUGE is a measure that is used to quantify the resemblance of an automatically generated summary to a human-generated summary (Lin, 2004). It measures phrase overlaps between a human-generated summary and an auto-generated summary. Its variants are ROUGE-1, ROUGE-2, and ROUGE-L, which stand for unigram, bigram, and the longest common subsequences overlap respectively. It is a measurement of recall since it quantifies how many sequences in the human-generated summaries are also in the auto-generated summaries. The ROUGE score is calculated as follows (Steinberger & Ježek, 2012):

$$ROUGE\text{-}n = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} count(gram_n)} \tag{5.1}$$

where *ReferenceSummaries* is a set of human-generated summaries, $count_{match}(gram_n)$ is the maximum number of n-gram overlaps between a candidate summary and a refer-

ence summary, and $count(gram_n)$ is the number of n-grams in the reference summary.

5.2 BLEU Score

BLEU or the Bilingual Evaluation Understudy is used in summarization tasks to measure the performance of a model (Papineni, Roukos, Ward, & Zhu, 2002). BLEU score quantifies the overlap between auto-generated summaries and human-generated summaries. It is a measurement of precision since it quantifies how many sequences in the auto-generated summaries are also in the human-generated summaries. BLUE score varies on a scale of 1 to 100 and a summary is scored 100, if and only if the auto-generated summary and the human-generated summary (reference summary) have an exact match, so any BLEU score more than 70 may indicates overfitting.

CHAPTER 6

EXPERIMENTS

We ran experiments on both datasets D_{temp}, and D_{rich} and varied hyperparameters for network creation and input representation. We explained several models for input representation and present our empirical findings next.

6.1 Embeddings for Input Representation

Creating an ad creative is a kind of natural language processing (NLP) problem and in NLP tasks words are need be represented in some way. Also, for deep learning algorithms, data must be represented by using numeric values. Embeddings are ways of converting textual data based on their positions in a given context. Word embedding is a vector representation of any word in a given document. The vector is used to measure semantic and syntactic similarity of each word in that particularly given context. There are several well-known ways to create a word embedding. The following sections give some notion about common embedding models such as Word2vec, GloVe, and FastText.

6.1.1 Word2vec

Word2vec embedding model uses a single hidden layer neural network (Mikolov, Chen, Corrado, & Dean, 2013). Each word is represented by using one-hot vector which only one cell is set to value one that represents the index of the word, rest is set to zero. All neurons in a hidden layer are linear, and the network is a fully connected neural network. In the network, the number of input neurons is equal to the number of words in the given vocabulary, and the size of hidden layer determines the dimension of word vectors. If V denotes the number of vocabulary and N denotes dimensionality of word vectors, then the input of hidden layer is going to be a randomly initialized

matrix with a size of $V \times N$ and each row represents a word from vocabulary as shown in Figure 6.1.

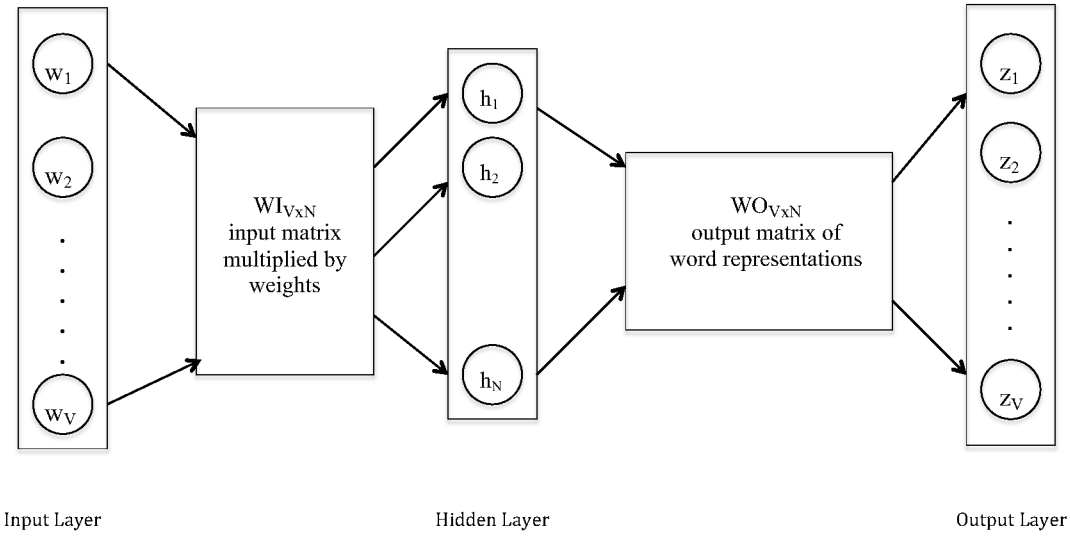

Figure 6.1: Working mechanism of Word2vec embedding model where w_{i} is an input, h_{i} is a hidden state, and z_{i} is an output.

Figure 6.2 demonstrates a well-known example of Word2vec. Word vectors of "man", "woman" and "king" are aligned according to the distance among themselves. When we start with a movement from word "king" with the same direction and amount from "man" to "woman", we can find the vector of word "queen", so subtracting addition of "man" and "woman" word vectors from "king" vector gives the "queen" vector.

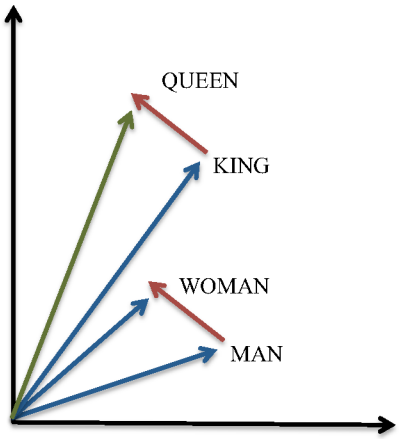

Figure 6.2: An example of Word2vec embeddings model where from vector representation of "man, woman and king" vectors, "queen" vectors is calculated.

6.1.1.a Continuous Bag of Words (CBOW)

CBOW is a kind of Word2vec embedding. CBOW predicts a target word by using its context vector, and context vector is represented by neighbor words of the target word (Mikolov et al., 2013). Obviously, neighbors of output word is an input, and a hidden layer is the multiplication of weights by one-hot vector of words. For example: "cat" and "tree" words are context words for "climbed" as the target word in a context. CBOW replicated input by the context words times to project hidden layer and likewise, output layer adds context number of times division after the hidden layer. CBOW has a low computation cost and works well on small datasets.

6.1.1.b Skip-Gram

Skip-Gram has a reverse logic of CBOW. Creation of a network is as same as in CBOW, but skip-gram predicts the context words from a given target word (Mikolov et al., 2013). In skip-gram a single target word is an input, a hidden layer is as same as in CBOW, and context words are the output. Figure 6.3 illustrates structure of CBOW and skip-gram models. Skip-Gram can perform well on synonym words and large datasets but has high computation cost.

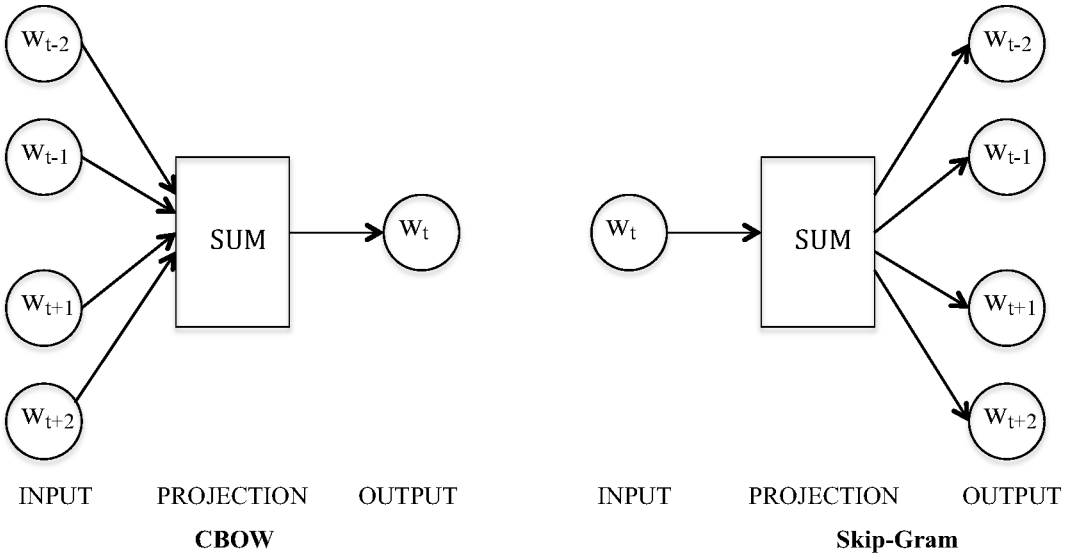

Figure 6.3: Illustration of CBOW and Skip-Gram embedding models where input of CBOW is an output of Skip-Gram and output of CBOW is an input of Skip-Gram model.

6.1.2 FastText

FastText word representation simply relies on log-likelihood idea like in the other embedding models, but other embeddings models are treated each word as a single vector, which prevents to learn from apparently different, but semantically same, and just another version of the same root word with prefix or suffix next to it (Bojanowski, Grave, Joulin, & Mikolov, 2017). In FastText representation, the smallest unit is a character, whereas words are the smallest unit in other embedding models. Sum of the subwords which are n-grams of characters, creates a vector representation of a word. For example if $n = 3$ and a word "where" is a target, then all n-grams are created which have length between three and six such as $< wh, whe, her, ere, re >$ and so on. By using FastText as word embeddings, rare words can be learned by a model. Even a word does not appear in the given context, the word can has a word representation vector. With FastText vector representation, words can share the similar representation vector. FastText can find a word vector even a word is misspelled in the given context. FastText has a high computation cost because creates n-grams of each word in the given context, but has the advantage of creating more vectors to represent a word.

6.1.3 Global Vectors for Word Representation (GloVe)

GloVe has the same logic with Word2vec model. GloVe measures contextual word co-occurrence of words that are related or analogous which based on the approach that a word co-occur with another word in the same context frequently if they have meaningful relationship (Pennington, Socher, & Manning, 2014). GloVe considers whole co-occurrence counts globally at a time, which creates its difference from Word2vec. Word2vec tries to get co-occurrence of two words in one window at each time, but GloVe looks at how often these words occur together by using inner product of the words. This calculation makes training time short because the whole network is not updated each time. GloVe also can work better on small corpus for that reason. GloVe can observe a semantic relationship between words in a context.

6.2 ROUGE Performance of Default embedding on D_{temp} and D_{rich}

For our first set of experiments, we used random weights to represent the input datasets. Table 6.1 shows the ROUGE performance of four different models for varying network size and for varying the choice of optimizer on both datasets separately. The results in Table 6.1 indicate that increasing the size of the network did not improve predictions on D_{temp}. On the other hand, ROUGE-2 scores of four models on D_{temp} are almost half of the ROUGE-1 scores indicate that the models were able to generate half as accurate bigrams as the unigrams.

Varying the optimizers, increased ROUGE-1 score by 0.3% for models with RNN size as 128, but decreased ROUGE-1 score by 1.2% for models with RNN size as 512. We recommend working with moderate hyperparameters on simple datasets such as D_{temp} for practical reasons. Increasing the size of RNN did not produce significant improvement on ROUGE scores while increasing the time complexity of the model.

The ROUGE scores on D_{rich} indicate that moderate hyperparameters are not sufficient on more complex datasets. As shown in Table 6.1, ROUGE scores on D_{rich} are lower compared to D_{temp}. Since D_{temp} contains repetitive template-based sentences e.g. “this is great place to learn”, “start learning at your own pace”; the model was able to learn the neural connections and generated useful creatives. In order to improve performance on D_{rich}, we enriched the data size with more content around similar topics.

We compared the ground truth creatives with the creatives generated by our models in Table 6.2. Although the predictions on D_{rich} were sub-optimal in terms of their ROUGE scores, the model generated concise creatives. Surprisingly, the model was able to learn ad creative action phrases such as “learn how to”, “learn the basics of”, and “the complete guide to”, which are usually preferred by an advertiser in order to attract user’s attention.

The performance of models with size 128 and 512 approximated the ground truth creatives on D_{temp}. The models learned the structure of ground truth creatives, 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. Therefore, generating the first sentence is more difficult than generating the generic ones. Though the last two sentences are different for each summary, the models predicted the first sentence in the ground truth creative correctly as depicted in Table 6.2. Models with RNN sizes of 128 and 512 had similar ROUGE performance on D_{temp},

	Model Parameters	ROUGE-1	ROUGE-2	ROUGE-L
D_{temp}	$\|RNN\|$ = 128, opt = sgd	43.002	22.336	42.133
	$\|RNN\|$ = 512, opt = sgd	**45.858**	**25.527**	**45.190**
	$\|RNN\|$ = 128, opt = adam	43.355	21.306	42.697
	$\|RNN\|$ = 512, opt = adam	44.604	23.875	43.772
D_{rich}	$\|RNN\|$ = 128, opt = sgd	14.950	2.464	13.738
	$\|RNN\|$ = 512, opt = sgd	14.076	1.837	12.817
	$\|RNN\|$ = 128, opt = adam	13.547	2.850	12.859
	$\|RNN\|$ = 512, opt = adam	**20.635**	**4.693**	**18.984**

Table 6.1: The effect of varying model size and optimizer on ROUGE scores. The experiments are conducted on D_{temp} and D_{rich}. $|RNN|$ represents the size of the network while *opt* corresponds to the choice of optimizer used.

and they generated similar creatives qualitatively as well.

6.3 ROUGE performance of Word2Vec, FastText and GloVe Embedding on D_{rich}

We created Word2Vec, FastText, and Glove embedding on D_{rich} and trained a model with parameters that performed best on D_{rich} as indicated in Table 6.1. Table 6.3 reports ROUGE scores of the generated creatives by using three embedding models. A model with the GloVe embedding performed best ROUGE-1, ROUGE-2, and ROUGE-L scores and generated creatives with better quality. Outcomes of this experiment lead us to use GloVe as word embedding for the representation of data. Our next experiments were performed to create improvement in performance of GloVe embedding model.

6.4 ROUGE performance of GloVe Embedding on D_{rich}

GloVe considers how often the words in a context co-occur by using the inner product of word frequencies. GloVe can find a semantic relationship between words because idea of being in a same context indicates a having similar meaning.

We created GloVe embedding on D_{rich} with several various vector dimensions,

Source Document for D_{temp}
the complete ruby on rails developer course. dissecting ruby on rails 5 - become a professional developer. 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 Summary (Ad Creative)
online ruby on rails. quality videos by domain experts. why wait? learn ruby on rails now.
Auto-generated Summary by a model with $\|RNN\|$=128, opt=sgd
online ruby on rails. this is a great place to learn. improve your ruby on rails skills.
Auto-generated Summary by a model with $\|RNN\|$=512, opt=sgd
online ruby on rails. the best price for the best value. learn at your own pace. start now!

(a) The samples were obtained on D_{temp}.

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 to compromise your target .
Ground-truth Summary (Ad Creative)
hacking for beginners . perform your first hack in the next 2 hours !
Auto-generated Summary by a model with $\|RNN\|$=512, opt=sgd
the complete ethical hacking course for beginners . learn the basics of ethical hacking , ethical hacking , and more !

(b) The samples were obtained on D_{rich}.

Table 6.2: The side by side comparison of ground truth creatives with auto-generated creatives. The results were obtained on datasets D_{temp} and D_{rich} on models of varying size $\{128, 512\}$.

minimum word occurrence number, and window size. Table 6.4 reports ROUGE scores of the generated creatives under these various parameter settings. Using the GloVe embedding increased ROUGE-1, and ROUGE-L scores by 3% and improved the quality of generated creatives.

6.4.1 Assessing the Quality of Auto-Generated Creatives

In previous sections, we presented ROUGE performance of generated creatives which relies on the approach of word overlapping. To measure how an auto-generated

Dataset is $D_{\textbf{rich}}$	**Vector Parameters**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
$\|RNN\|$ = 512, opt = adam	Word2Vec dim = 300, min(w_c) = 1, iter = 10, $\|W\|$ = 5	20.0	4.8	18.0
	FastText dim = 300, min(w_c) = 1, iter = 10, $\|W\|$ = 5	19.0	4.5	18.0
	GloVe dim = 300, min(w_c) = 1, iter = 10, $\|W\|$ = 5	**22.481**	**5.311**	**21.0**
	random weight embedding	20.635	4.693	18.984

Table 6.3: The effect of using various word embeddings on prediction performance quantified by ROUGE scores. The dataset used is D_{rich}. The size of the network is 512 ($|RNN| = 512$). For Word2Vec, FastText,and GloVe, we experimented with window size $|W| = \{5\}$ and minimum word frequency threshold of min(w_c) = 1.

Dataset is $D_{\textbf{rich}}$	**GloVe Parameters**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
$\|RNN\|$ = 512, opt = adam	dim = 300, min(w_c) = 1, iter = 10, $\|W\|$ = 5	22.481	5.311	21.0
	dim = 300, min(w_c) = 1, iter = 15, $\|W\|$ = 15	**23.143**	**5.438**	**21.352**

Table 6.4: The effect of using GloVe word embedding on prediction performance quantified by ROUGE scores. The dataset used is D_{rich}. The size of the network is 512 ($|RNN| = 512$). For GloVe, we experimented with varying window size $|W| = \{5, 15\}$ and minimum word frequency threshold of min(w_c) = 1.

summary is close to a human-generated summary, a human should review and compare the summaries with objectivity.

For qualitative assessment, auto-generated creatives were rated using a scale of three grade labels by two human judges. The goal is to quantify the relevance of generated ad creatives with respect to ground truth ad creatives. As shown in Table 6.5, 32% of the ad creatives were judged as not-relevant while 55% were judged as relevant.

A single human judge may not provide a reliable assessment of the quality of the generated creatives due to subjectivity. In order to address subjectivity, we considered the agreement between two judges. We used Cohen's kappa in order to measure the

		2^{nd} **Judge**			
		relevant	not-relevant	not-sure	total
1^{st} **Judge**	relevant	247	8	14	269
	not-relevant	5	136	17	158
	not-sure	16	12	26	54
	total	268	156	57	481

Table 6.5: Two judges assessed the quality of auto-generated creatives from D_{rich}. The judges used three labels as relevant, not-relevant, and not-sure during their assessment.

inter-rater reliability from the judgments shown in Table 6.5 (Cohen, 1960).

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{6.1}$$

where

- $P(A)$ denotes maximum-likelihood estimate of the probability of agreement between the two judges:

  $(247 + 136 + 26)/481 = 0.85$.

- The pooled marginals for relevant, not-relevant, and not-sure suggestions are:

  $P(\text{relevant}) = (269 + 268)/(962) = 0.558$,

  $P(\text{non-relevant}) = (158 + 156)/(962) = 0.326$ and

  $P(\text{not-sure}) = (54 + 57)/(962) = 0.115$.

- $P(E)$ denotes maximum-likelihood estimate of the probability of agreement due to chance:

  $P(\text{relevant})^2 + P(\text{non-relevant})^2 + P(\text{not-sure})^2 = 0.43$.

- The kappa statistic is computed using $P(A)$, and $P(E)$ as $(0.85 - 0.43)/(1 - 0.43)$, which is equal to 0.73.

A kappa score between 0.4 and 0.6 indicates moderate agreement and score between 0.6 and 0.8 indicates satisfactory agreement (Altman, 1990). According to kappa we obtained, we can state that the quality of the auto-generated ad creatives is good.

In general, we observed that the model performed well for courses that are represented well in the training dataset while it performed poorly for courses that lack sufficient representation. For the source sample, "understand the SAT structure and be able to plan out an effective study schedule in a short time", the model generated the following ad creative: "the complete guide to the ultimate guide. a step by step guide". The model could not capture any specific word due to poor context present in the source data.

6.4.2 Quality of Auto-generated Ad Creatives on Hand Crafted Datasets

We hand crafted two new datasets from D_{rich}. Ad titles in the dataset were formatted according to Google's creative standards, which has tight restrictions on the length of each ad component. Each component must have maximum characters due to its limits:

"Headline 1 (30) | Headline 2 (30) | Headline 3 (30) . Description 1 (90) . Description 2 (90)".

Furthermore, new creatives were generated by a domain expert. In the dataset named ${D^*}_{\text{rich}}$, all ads are in Google's format and every word is pre-processed to remove redundancy and to improve writing. Also, contents of landing pages are cleaned from punctuations except comma and period sign.

In another dataset named ${D^+}_{\text{rich}}$, original forms of ad titles were added to the content of landing pages as source documents. With that approach, contents of landing pages are completely was taken as a source document.

The same newly organized target ad creatives were used in ${D^*}_{\text{rich}}$, and ${D^+}_{\text{rich}}$. Both datasets contain 3732, 458, and 467 entries for training, validation, and test respectively. Tables 6.6 and 6.7 show sample documents/ads.

6.4.2.a Performance of LSTM models on ${D^*}_{\text{rich}}$ and ${D^+}_{\text{rich}}$

Since the GloVe embedding achieved the highest ROUGE scores, we used GloVe with its best parameter settings on the newly created datasets. 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 at every 250 steps, the resulting model was saved.

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 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 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 .
Source Document #3	**Target Document #3**
master the most important tools in lightroom . take your photos from flat to fabulous using lightroom's develop module . develop a workflow that will save you time import select best edit export . organise your photos using lightroom's library module in a way that will makes sense to you . learn how to approach editing any subject matter portraits , cityscapes / travel , interior covered in the course . how to follow along step by step with all the raw files provided . learn how to set up and use export presets to save time . learn how to set up and use import presets matter to save time .	adobe lightroom cc | the ultimate guide for beginners | edit your photos in adobe lightroom cc . how to organize and edit your photos like a pro by using lightroom classic cc .

Table 6.6: The source and target document samples from D^*_{rich}, which contain ad creatives conforming to the Google's ad standards. Headlines are separated with $pipe$ $(|)$ sign; first dot $(.)$ sign indicates the end of headlines. Each sentence between $dots$ is a description based on Google's Ad standards. Every punctuation mark is written out separately from nearby words in order to compute word-frequencies correctly.

We used validation perplexity in order to identify the intermediate model to use for generating ad creatives. The model with the lowest perplexity was chosen. Figure 6.4 gives the validation perplexity of the first 6 models on both datasets. The perplexity started at 20% on D^+_{rich}, and decreased gradually till a point, beyond which perplexity

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 and financial models	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 .
Source Document #3	**Target Document #3**
complete a short tai chi routine . theory of use and different techniques . practical techniques for joints and regions of body . tai chi for absolute beginners part 1 . a step by step guide to get you started learning tai chi : improve your balance and find inner calm .	tai chi for absolute beginners | theory of different techniques . a step by step guide to get you started learning tai chi : improve your balance and find inner calm .

Table 6.7: The source and target documents samples from D^{+}_{rich}, which contains unformatted creatives (in addition to landing page) in source documents to provide more context. Ad creatives conform to Google's Ad standards in target documents. Headlines are separated with $pipe$ $(|)$ sign, first dot $(.)$ sign indicates the end of headlines. Each sentence between $dots$ is a description based on Google's Ad standards. Every punctuation mark is written out separately from nearby words in order to compute word-frequencies correctly.

started to increase. This behavior is acceptable and indicates model reliability. On the other hand for D^{*}_{rich}, the perplexity started at 105%, and the lowest value was obtained in the second saved model at 500^{th} step. Then on, its validation perplexity grew exponentially.

Table 6.8 shows the ROUGE, and BLEU scores obtained on both datasets. As expected, when we used the raw ad titles in the source side and human-generated ad creatives in the target side (D^{+}_{rich}), we obtained the highest ROUGE, and BLEU

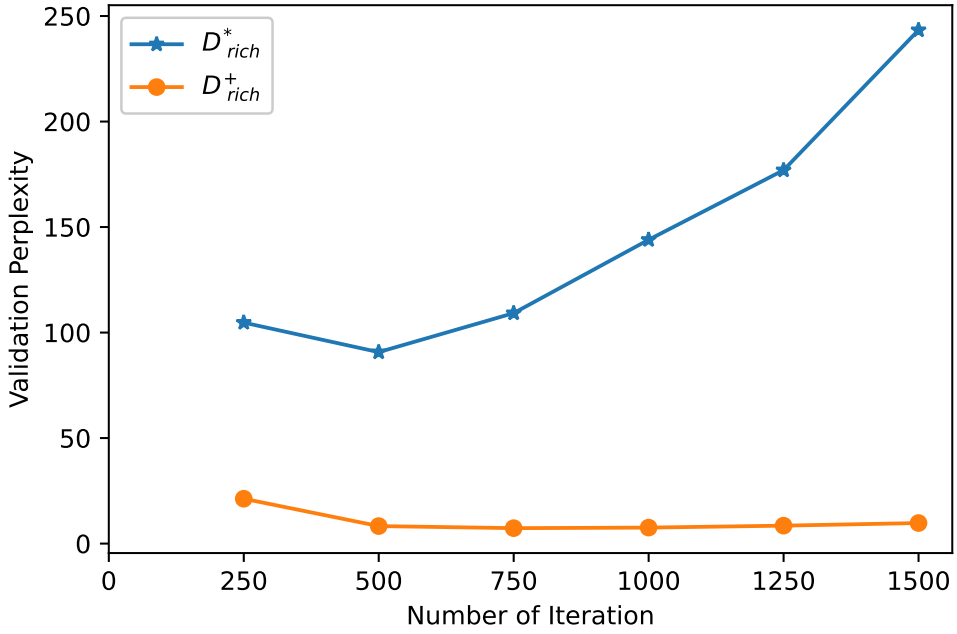

Figure 6.4: Validation Perplexity of the first 6 models on D^*_{rich} and D^+_{rich}. We made use of validation perplexity scores in order to decide on which model to use for generating ad creatives without overfitting.

scores. A ROUGE score of 40 is deemed as good performance for a summarization task; our model obtained 62.72 in ROUGE-1, and 61.91 in ROUGE-L.

	Model Parameters	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**BLEU**
$D^*_{\textbf{rich}}$	$\|RNN\|$ = 512, opt = adam, LR = 0.001	27.684	8.091	26.121	6.03
$D^+_{\textbf{rich}}$	$\|RNN\|$ = 512, opt = adam, LR = 0.001,	**62.72**	**48.077**	**61.91**	**41.48**

Table 6.8: ROUGE and BLEU scores of our LSTM model with network size $|RNN| = 512$, using *adam* as the optimizer with a learning rate 0.001 on D^*_{rich} and D^+_{rich}.

The ad creatives generated by the selected model are shown in Table 6.9 along with the ground truth creatives. Newly generated creatives are similar to the real ones. In general, all generated creatives had the right format according to the standard and matched closely with the context found in the source.

We experimented with variations of our model and measured ROUGE, and BLEU scores in each case. The results are summarized in Table 6.10. Decreasing the learning rate decreased the quality of generated creatives. In general, a learning

Source Document #1 for D^+_{rich}
learn and understand the key sql . learn how to use oracle json sql . learn the key json sql functions . learn how oracle handles bad json data with the error clause . learn about json indexes in comparison to relational indexes . discover techniques to analyze and maintain json sql code . learning oracle sql json . learn how to design and develop databases using oracle sql developer
Ground Truth Summary (Ad Creative)
learning oracle sql json | oracle sql json course . learn how to design and develop databases using oracle sql developer .
Auto-generated Summary by a model with $\|RNN\|$=512, opt=adam
learning oracle sql json . learn how to design and develop databases using oracle sql developer .

(a) The first set of samples obtained on D^+_{rich}.

Source Document #2 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 Summary (Ad Creative)
learn ruby programming | the easy way lite | application with ruby . learn to program in ruby , in a step by step easy to follow hands on course . run ruby interactively at the command prompt .
Auto-generated Summary 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 .

(b) The second set of samples obtained on D^+_{rich}.

Source Document #3 for D^+_{rich}
model game assets in blender . create uv maps and texture maps . use fbx to import blender models into unity . light your scenes in unity . create a test build in unity . create a game environment with blender and unity . use blender , photoshop , and unity to create your own game environments .
Ground Truth Summary (Ad Creative)
create a game environment | with blender and unity . use blender , photoshop , and unity to create your own game environments . light your scenes and create test in unity .
Auto-generated Summary by a model with $\|RNN\|$=512, opt=adam
create a game environment | with blender and unity . use blender , photoshop , and unity to create your own game environments .

(c) The third set of samples obtained on D^+_{rich}.

Table 6.9: Auto-generated creatives from D^+_{rich} against human-generated creatives. Headlines are separated with *pipe* $(|)$ sign and first *dot* $(.)$ marks the end of headlines. Each sentence between $dots$ are descriptions in Google's Ad standards.

rate of 0.001 is a suggested default value for "adam" optimizer and worked well on our datasets.

Working with a smaller network of 256 instead of 512 resulted in a 5% decrease in ROUGE scores. If the computational resources are not enough to train complex models, creating an LSTM network of size $|RNN| = 256$ may be admissible. The last column of Table 6.10 indicates the confidence of models. In each model, the validation accuracy was less than the train accuracy, which indicates that the selected model was not overfitting. Also in the selected model, the cross-entropy (loss over the number of words) was under one and the perplexity was under ten for validation, and training stages.

6.4.3 Kappa Score of Auto-Generated Creatives from D^{+}_{rich}

As shown in Table 6.11, $\%13$ of the ad creatives were judged as not-relevant while $\%77$ were judged as relevant.

For qualitative assessment of auto-generated creatives from D^{+}_{rich}, creatives were rated using a scale of three grade labels by different two human judges. The goal is observing quality of auto-generated creatives from human perspective to ensure high ROUGE scores.

From the judgments shown in Table 6.11 a Kappa score is calculated:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{6.2}$$

where

- $P(A)$ denotes maximum-likelihood estimate of the probability of agreement between the two judges:

  $(352 + 57 + 19)/467 = 0.92$.

- The pooled marginals for relevant, not-relevant, and not-sure suggestions are:

  $P(\text{relevant}) = (358 + 371)/(934) = 0.78$,

  $P(\text{non-relevant}) = (67 + 63)/(934) = 0.14$ and 3794

  $P(\text{not-sure}) = (46 + 29)/(934) = 0.08$.

- $P(E)$ denotes maximum-likelihood estimate of the probability of agreement due to chance:

  $P(\text{relevant})^2 + P(\text{non-relevant})^2 + P(\text{not-sure})^2 = 0.63$.

Model Parameters	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	Stats
$\|RNN\|$ = 512, opt = adam LR = 0.001 stepSize = 5000 modelStep = 750	**62.72**	**48.07**	**61.91**	**41.48**	val. perp. = 7.30 val. acc. = 71.79 train acc. = 84.86 train ppl. = 2.16 xent = 0.77
$\|RNN\|$ = 512, opt = adam LR = 0.0001 stepSize = 5000 modelStep = 4750	58.73	41.77	57.58	35.57	val. perp. = 11.33 val. acc. = 65.78 train acc. = 85.40 train ppl. = 2.21 xent = 0.79
$\|RNN\|$ = 256, opt = adam LR = 0.001 stepSize = 5000 modelStep = 1250	57.19	41.33	56.33	30.45	val. perp. = 13.28 val. acc. = 66.03 train acc. = 83.26 train ppl. = 2.76 xent = 1.02
$\|RNN\|$ = 256, opt = adam LR = 0.0001 stepSize = 2500 modelStep = 2500	32.02	11.23	30.38	7.19	val. perp. = 64.41 val. acc. = 37.43 train acc. = 39.56 train ppl. = 44.25 xent = 3.79
$\|RNN\|$ = 256, opt = adam LR = 0.0001 stepSize = 5000 modelStep = 5000	50.15	31.16	48.88	24.1	val. perp. = 20.92 val. acc. = 56.28 train acc. = 66.17 train ppl. = 7.29 xent = 1.99
$\|RNN\|$ = 256, opt = adam LR = 0.0001 stepSize = 10000 modelStep = 7000	57.57	41.14	56.44	34.77	val. perp. = 16.65 val. acc. = 63.21 train acc. = 81.46 train ppl. = 3.03 xent = 1.11

Table 6.10: ROUGE and BLEU scores for varying the network size RNN in $\{256, 512\}$ and the learning rate in $\{10^{-3}, 10^{-4}\}$ on D^{+}_{rich} with $adam$ as optimizer; *val.* is for validation corresponding to the convergence of the training; *acc.* is accuracy, which indicates the percentage of the match in auto-generated creatives and human-generated creatives and calculated by $100 * \frac{\#\ of\ correct\ words}{\#\ of\ words}$; *xent* is cross-entropy is a loss function that measures the performance of a model and calculated by $\frac{loss}{\#\ of\ words}$, *perp. ppl.* are perplexity measures for how well a model predicts a sample and are calculated by $exp(min(xent, 100))$.

		2^{nd} **Judge**			
		relevant	not-relevant	not-sure	total
1^{st} **Judge**	relevant	352	1	18	371
	not-relevant	1	57	9	67
	not-sure	5	5	19	29
	total	358	63	46	467

Table 6.11: Two judges assessed the quality of auto-generated creatives from D^{+}_{rich}. The judges used three labels as relevant, not-relevant, and not-sure during their assessment.

- The kappa statistic is computed using $P(A)$, and $P(E)$ as $(0.92 - 0.63)/(1 - 0.63)$, which is equal to 0.8.

As we stated in section 6.4.1, a score between 0.6 and 0.8 indicates satisfactory agreement. According to kappa we obtained, we can state that the auto-generated ad creatives from D^{+}_{rich} have higher quality than the creatives obtained from D_{rich}. We increased the performance of the model by enriching the D_{rich}.

CHAPTER 7

CONCLUSION

Researches show that online advertising is one of the biggest sources of income for search engines. Google is the search engine which has the highest share in online marketing. Beside the profit of online marketing for search engines, companies make some part of their profit from e-commerce or online sales. Also, customers can reach a product which they intend to buy fast and easily by using the Internet without making an effort to look for a store. When a paid search advertising mechanism works properly, online marketing is an ecosystem that all parties win. It is important to create a well-worked campaign to provide satisfaction for each actor. This is the duty of an advertiser who is responsible for advertisement budget management, ad keywords management, and ad creative generation.

It is a time-consuming task for an advertiser to manage a campaign, especially when she is responsible for multiple campaigns. Also, the advertiser should respond as quickly as possible to feedback that a marketing tool of a search engine provides. We aim to create an automatic process for ad creative generation. A landing page of a product with full information to describe the selling item is a good source for an ad creative.

In this study, we formulate our ad creative generation task as a text summarization problem where ad creative is a summary of a given landing page for a product. In our problem, input size and output size are not fixed. When the size of data is not fixed, abstractive summarization methods works well.

We showed that our sequence to sequence model based on an encoder-decoder bidirectional recurrent neural network, performs well in generating ad creatives. Our base network has LSTM cells in the encoder, and the decoder states. We varied RNN size in our experiments to discover the performance of which model is the highest in terms of accuracy performance, resource allocation and computation cost.

In order to assess the validity of the proposed approach, we experimented on

four different datasets. The experiments on the template-based dataset D_{temp} have shown that our model generated good quality creatives with moderate hyper-parameters. On a content richer dataset D_{rich}, the model generated product-oriented ad creatives, but its complexity increased. To improve the performance of the model on D_{rich}, we used various word embedding models to represent the data. We showed that with same parameters in the creation of the model, GloVe word embedding model performed best. With GloVe used in vectorization, we observed a 3% increase in ROUGE-1, and ROUGE-L scores. The auto-generated creatives on D_{rich}, were validated by two judges with an inter-rater agreement score of 0.73.

Enriching D_{rich} with more information available in landing pages and reformatting ad creatives uplifted ROUGE scores and increased the quality of the generated creatives. ${D^*}_{\text{rich}}$ and ${D^+}_{\text{rich}}$ datasets include same creatives as target documents, but ${D^+}_{\text{rich}}$ dataset has richer content in landing pages as the source document. The ROUGE, and BLEU scores on ${D^+}_{\text{rich}}$ with whole landing pages as source and reformatted creatives according to Google's creative standards as target were 63, and 41 respectively. Visual inspection of generated creatives hints at potential for actual field tests. Auto-generated creatives from ${D^+}_{\text{rich}}$ conformed to Google's ad creative standards. The auto-generated ad creatives on ${D^+}_{\text{rich}}$, were validated by two other judges with an inter-rater agreement score of 0.79.

In a final word, our proposed model generates ad creatives conformed to Google's ad creative standards from landing pages. The generated creatives can be directly import to the related ad campaign.

CHAPTER 8

FUTURE WORK

As a future work, we are planning to incorporate more data sources and make use of generative adversarial networks (GANs) (Zhang, Gan, & Carin, 2016).

In a GAN, there are two leading roles such as discriminator and generator, and they can be co-trained to learn from generated summaries and ground-truth summaries together. In a GAN model, the generator creates fake data as closest as the real ones and the discriminator tries to catch which one is real or not. From the gradient between real and fake data, the generator updates its parameters and makes another data generation which creates data more close to real one from the previous step. The generation and discrimination process go on until the generated data is almost as good as the real one and it is almost not possible to differentiate generated summaries and ground truth summaries (Goodfellow et al., 2014).

Metaphorically, the generator behaves like a forger and creates fake products, and the discriminator acts like a detective to find out which products are real and which are fake by comparing the generated and real ones. After each comparison, the forger learns more information to make the fake products look more real, and the detective learns more information to distinguish between the real and the fake ones by seeing many more samples. Figure 8.1 illustrates a GAN model for an ad creative generation problem as a text summarization task.

Wang and Lee proposed a model by using a GAN structure for text summaries (Wang & Lee, 2018). Their model has 3 components such as a generator, a discriminator and a re-constructor. In generator and discriminator modules, they created an LSTM network. We used their model for our initial GAN experiments.

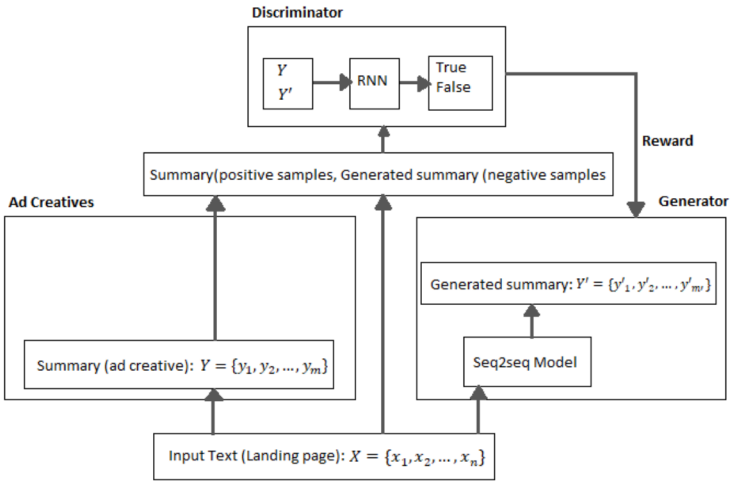

Figure 8.1: The figure illustrates a GAN model for ad creative generation task where X is a set of input sequences, Y is a set of sequences of human-generated creatives, and Y' is a set of output sequences which is created by a generator.

By using some shared parameters with our own LSTM model, GAN obtained promising ROUGE-1 score. Because we have no access to enough computation resources to train a more complicated GAN model, we are planning to increase our resources in the future.

References

Altman, D. G. (1990). Practical statistics for medical research. CRC press.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Berger, A. L., & Mittal, V. O. (2000). Ocelot: A system for summarizing web pages. In Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval (pp. 144–151).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Choi, Y., Fontoura, M., Gabrilovich, E., Josifovski, V., Mediano, M., & Pang, B. (2010). Using landing pages for sponsored search ad selection. In Proceedings of the 19th international conference on world wide web (pp. 251–260).

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1), 37–46.

Enberg, J. (2019). emarketer, what's shaping the digital ad market. Retrieved April 1, 2020, from `https://www.emarketer.com/content/global-digital-ad-spending-2019`

Fujita, A., Ikushima, K., Sato, S., Kamite, R., Ishiyama, K., & Tamachi, O. (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 (pp. 179–188).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672–2680).

Help, G. A. (2020). Clickthrough rate (ctr): Definition. Retrieved July 5, 2020, from `https://support.google.com/google-ads/answer/2615875?hl=en`

Joshi, A., & Motwani, R. (2006). Keyword generation for search engine advertising. In Sixth ieee international conference on data mining-workshops (icdmw'06) (pp. 490–496).

Karpathy, A., Johnson, J., & Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.

Kemp, S. (2020a). Digital 2020: April global statshot. Retrieved from https://datareportal.com/report/digital-2020-april-global-statshot

Kemp, S. (2020b). Digital 2020: Global digital overview. Retrieved from https://datareportal.com/report/digital-2020-global-digital-overview

Kim, L. (2020). Cost per click. Retrieved July 10, 2020, from https://www.wordstream.com/cost-per-click

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Klein, G., Kim, Y., Deng, Y., Senellart, J., & Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. arXiv preprint arXiv:1701.02810.

Kryściński, W., Paulus, R., Xiong, C., & Socher, R. (2018). Improving abstraction in text summarization. arXiv preprint arXiv:1808.07913.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436.

Lee, M.-C., Gao, B., & Zhang, R. (2018). Rare query expansion through generative adversarial networks in search advertising. In Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining (pp. 500–508).

Lin, C.-Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/W04-1013

Manning, C. D., Raghavan, P., & Schutze, H. (2008). Introduction to information retrieval. Cambridge University Press. Retrieved from http://nlp.stanford.edu/IR-book/information-retrieval-book.html

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, 01). Efficient estimation of word representations in vector space. Proceedings of Workshop at ICLR, 2013.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., & Nogueira dos santos, C. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.

of Ecommerce Statistics, T. U. L. (2020). Conversion rate optimization (cro) statistics.

Retrieved July 10, 2020, from https://letstalkaboutmoney.com/ecommerce-statistics/

Olah, C. (2015). Understanding lstm networks.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 311–318).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp) (pp. 1532–1543).

Ravi, S., Broder, A., Gabrilovich, E., Josifovski, V., Pandey, S., & Pang, B. (2010). Automatic generation of bid phrases for online advertising. In Proceedings of the third acm international conference on web search and data mining (pp. 341–350).

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673–2681.

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929–1958.

Steinberger, J., & Ježek, K. (2012). Evaluation measures for text summarization. Computing and Informatics, 28(2), 251–275.

Sun, J.-T., Shen, D., Zeng, H.-J., Yang, Q., Lu, Y., & Chen, Z. (2005). Web-page summarization using clickthrough data. In Proceedings of the 28th annual international acm sigir conference on research and development in information retrieval (pp. 194–201).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104–3112).

Thomaidou, S., Liakopoulos, K., & Vazirgiannis, M. (2014). Toward an integrated framework for automated development and optimization of online advertising campaigns. Intelligent Data Analysis, 18(6), 1199–1227.

Wang, Y.-S., & Lee, H.-Y. (2018). Learning to encode text as human-readable summaries using generative adversarial networks. arXiv preprint arXiv:1810.02851.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... Dean, J.

(2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Yih, W.-t., Goodman, J., & Carvalho, V. R. (2006). Finding advertising keywords on web pages. In Proceedings of the 15th international conference on world wide web (pp. 213–222).

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. arXiv preprint arXiv:1409.2329.

Zhang, Y., Gan, Z., & Carin, L. (2016). Generating text via adversarial training. In Nips workshop on adversarial training (Vol. 21).

APPENDIX A

CV

Kevser Nur ÇOĞALMIŞ is a PhD student and research assistant in the Computer Engineering Department at Istanbul Sabahattin Zaim University. Her research interests are information retrieval, machine learning, text analysis and data analysis. Çoğalmış, received an MS in electrical and computer engineering from Istanbul Şehir University.

PROJECTS

- T2C2 Scalable Data Analytics, Türk Telekom, Project Manager: BULUT Ahmet, Research Assistants: ÇOĞALMIŞ Kevser Nur, BAKİROV Aslan, 01/03/2013 - 20/01/2014.
- Reklamcı ve Kullanıcı Geribildirimine Dayalı Reklam Kampanyası Kapsama Alanı Yönetimi, Tübitak Project, Project Manager: BULUT AHMET, Research Assistants: ÇOĞALMIŞ Kevser Nur, SAĞOĞLU Oğuzhan, 01/10/2013 - 01/10/2015.

PUBLICATIONS

- ÇOĞALMIŞ Kevser Nur, SAĞOĞLU Oğuzhan, BULUT Ahmet (2017). AdScope: Search Campaign Scoping Using Relevance Feedback. IEEE INTELLIGENT SYSTEMS, 32(3), 14-20, Doi: 10.1109/MIS.2017.47.
- BAKİROV Aslan, ÇOĞALMIŞ Kevser Nur, BULUT Ahmet (2016). Making Online Recommendations Made by Easy by Apache Spark. Informatika, 18(1), 3-6.
- BAKİROV Aslan, ÇOĞALMIŞ Kevser Nur, BULUT Ahmet (2016). Scalable sentiment analytics. Turkish Journal of Electrical Engineering and Computer Sciences, 1560-1570.

PROCEEDINGS

- ÇOĞALMIŞ Kevser Nur (2019). Traditional Advertising is Shifting to Influencer Advertising. 15th International Conference on Knowledge, Economy Management, 15, 39-44. (Extended Abstract Report/Presentation)
- BAKİROV Aslan, ÇOĞALMIŞ Kevser Nur, HABİBOV Javid, BULUT Ahmet (2016). Scaling Topic Modeling on Large Text Collections. EDCON EUROPE 2015 (Full Report/Presentation)