Automatic Generation of Pattern-controlled Product Description in E-commerce

Tao Zhang
Alibaba Group
Hangzhou, China
guyan.zt@aliibaba-inc.com

Jin Zhang
Alibaba Group
Hangzhou, China
zj146372@aliibaba-inc.com

Chengfu Huo
Alibaba Group
Hangzhou, China
chengfu.huocf@aliibaba-inc.com

Weijun Ren
Alibaba Group
Hangzhou, China
afei@aliibaba-inc.com

ABSTRACT

Nowadays, online shoppers have paid more and more attention to detailed product descriptions, since a well-written description is a huge factor in making online sales. However, for a website with billions of product data like Alibaba, the writing efficiency of human copywriters cannot match the growth rate of new products. To address this issue, we propose a novel pointer-generator neural network to generate product description. In particular, coordinate encoders and a pattern-controlled decoder are utilized to improve generation quality with an attention mechanism. The coordinate encoders equipped with a Transformer and a gated convolutional unit is introduced to learn the source input representations. In the decoding phase, a pattern controlled decoder is proposed to control the output description pattern (such as category, length, and style) to ensure the quality of the description. For evaluation, we build a substantial collection of real-world products along with human-written descriptions. An extensive set of experiments with both human annotated data demonstrate the advantage of the proposed method for generation qualities. Finally, an online deployment shows significant benefits of our model in a real online shopping scenario, as measured by the click-through rate.

KEYWORDS

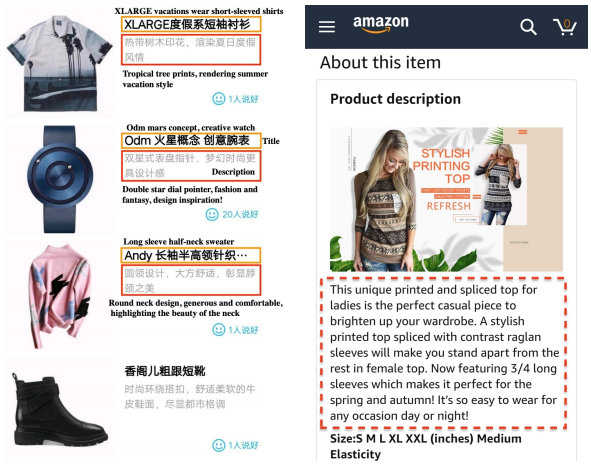
Natural Language Generation; Pattern-controlled; Product description;

ACM Reference Format:
Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019. Automatic Generation of Pattern-controlled Product Description in E-commerce. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313407

1 INTRODUCTION

The importance of content marketing for E-commerce has been significantly growing. Content marketing is one of the most effective ways for E-commerce websites to increase consumer engagement, namely because the customer chooses only to interact with the content that interests them, as opposed to traditional marketing. Major E-commerce sites, like Taobao, are gradually switching from trading platforms to content-based platforms since from users perspective the first is content consumption, followed by product consumption. On the Alibaba site, for example, there are many product oriented recommendations feeds flow, including product images, short videos, and recommendation reasons. However, the generation and maintenance of these content-based products require massive human costs. To reduce labor costs, product descriptions are expected to be automatically generated by the machine.

Generating product description is a difficult technical task of how to apply the state of art Natural Language Generation (NLG) technology to the field of E-commerce. Most human copywriters use product titles, attributes, and images, as well as the their own E-commerce industry experience to create product discriptions. Meanwhile, the writing techniques of copywriters can make a tremendous difference in the quality of generated product descriptions. Thus, it is a hugely difficult task to replace human being to



(a) Recommendation page on taobao (b) Detail page on Amazon

Figure 1: Product descriptions are widely used in E-commerce website Taobao and Amazon

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313407

generate product description for a machine. Product title and properties are unstructured datasets which are cluttered and disordered. The machine needs to analyze the product title to capture essential product information. Due to the diversity of E-commerce platform, it is necessary to control the pattern of description writing. For example, in the home page on mobile devices, the length of product description should be as short as possible without losing essential information owing to the limited display space on mobile phones. Another example is controlling style in the product description. In sales & promotion, we expect to add marketing-styled information into product description.

Generating product description can be viewed as a special kind of natural language generation. In NLG communities, machine translation and summarization are similar tasks to text generation. But the difference is that the length of the product description is not strictly limited to be approximately equal to or less than that of the source text, which is a controlled algorithm parameter in this task. Besides, compared with the traditional summarization task, this project has more restrains: **1)** Do not display any irrelevant product information. The objectivity and accuracy of product description shall be ensured; **2)** Try to focus on essential product information, like product categories and brands; **3)** Try to control the pattern of writing, including length and style. For consumers and sellers, wrong information in the product description is disallowed. An E-commerce study conducted by NNgroup[1] found that 20% of overall task failures when the user failed to complete a purchase could be attributed to incomplete or unclear product information. There are many successful abstract generation methods for news and wiki, like pointer-generator [36], but lots of wrong information can be found from generator mode. When generating long description based on the product title and property, we need to pay attention to the consistency between description and product. Moreover, the sequence to sequence (seq2seq) method prefers to generate safer text. When generating long text, seq2seq method will produce a large number of duplicated sub-sequences which seriously affects the quality of the final text. Thus, how to grasp essential product information more accurately and generate the corresponding product descriptions is particularly important. In addition, many similar products have similar key information at the Alibaba site. As a result, the description produced by this vanilla attention-based method are also similar, which is terrible for recommending similar products. For this reason, the diversity of product description needs to be considered as well.

This paper puts forward an attention based and pattern controlled neural network to solve the above three restrains. For the first restrain, we refer to the pointer-generator mechanism which used to solve the problem of out of vocabulary (OOV) words, like the brand names and commodity names in our task. Besides, we introduce coordinate encoders equipped with transformer[41] and Gated Convolutional Neural Network (Gated-CNN)[15] to learn the dependencies and representations among the words of product titles and properties on the input side which aims to improve the keywords capture ability when generating the text. The experiment proves that these two methods can improve the category and band accuracy of the generated text. Regarding the second problem, we use a name entity recognition tool called Term Weight to extract keywords in text, such as style, material, marketing information, and others. These keywords are fed into the decoding phase to control the output style. The third constraint is handled by adding length vector for decoding phase. The traditional method of length control is to limit the maximum search depth of decoding by truncation, which is so violent that severely impacts user's experience. In this paper, a length vector is introduced to restore the length information of each time step. We automatically label the length of descriptions in the training phase and made model learn length information. Then, the length of the generated text can be restrained by setting customized length vector when decoding.

To the best of our knowledge, this work is the first attempt to apply the neural language generation technology to product description generation. Our contributions are three-fold:

1. We put forward a product description generation technology by multi-task learning, which is far advanced than traditional generation method by seq2seq model;
2. We introduce a pointer-generator architecture with coordinate encoders (Transformer and Gated CNN) to extract the source features, which can capture the local correlations and ensure the accuracy and diversity of output description generation;
3. We intervene in the pattern of the text, such as the length and the style of the product description, to reduce model development period when switching the shopping scenarios and enhance the category accuracy.

2 BACKGROUND

Given the input sequence of product title, our goal is to generate the product description. We discuss this description generation problem in the sequence-to-sequence problem setting. Suppose that the input sequence $x = (x_0, x_1, x_2 \ldots x_M)$ and the output sequence $y = (y_0, y_1, y_2 \ldots y_N)$.

2.1 Sequence to Sequence Model

Recently, sequence-to-sequence (seq2seq) model has been widely used in machine translation and summarization system. The seq2seq model usually contains two Recurrent Neural Network (RNN) components, an encoder and a decoder. The encoder gets the context vector of the source sequence x and the decoder transforms the context vector c to the target sequence, given x.

$$\boldsymbol{h}_t = f(\boldsymbol{h}_t, \boldsymbol{w}_t)$$

$$\boldsymbol{c} = g(\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N)$$

where h_t is the RNN hidden state at time t and w_t is the word embedding of word t. f is the activation function of RNN unit. In practice , gating mechanisms such as LSTM [19] and GRU [8] has been shown to exhibit better performance than vanilla RNN. The context vector C is also called the attention vector and is calculated using an attention mechanism [1] as a weighted sum of annotations of the encoder states.

$$C_j = \sum_{i=1}^{n} \alpha_{ji} h_i$$

[1] https://www.nngroup.com/reports/ecommerce-user-experience/

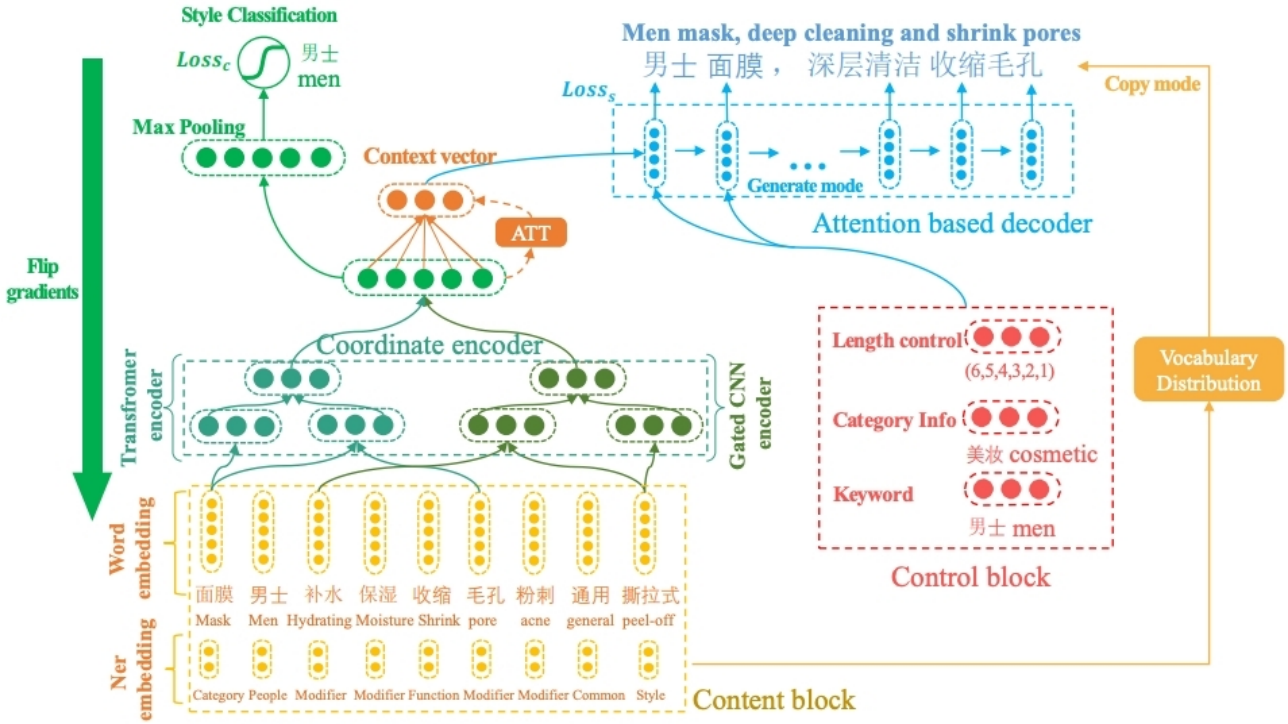


Figure 2: Architecture of Pattern Controlled Pointer-Generater Network.

Where a_{ji} are attention weights corresponding to each encoder hidden state output hi obtained as follows:

$$\alpha_{ji} = \frac{\exp(z_i)}{\sum_{k=1}^{n} \exp(z_k)}$$

Activations $z_k = a(d_{j-1}, h_k)$ are calculated by using a context function such as the dot product between the decoder state d_{j-1} and the encoder hidden states h_k. The decoder is also an RNN component, which is used to predict a target sequence,

$$\boldsymbol{d}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{d}_{t-1}, \boldsymbol{c})$$

$$p(y_t = w | y_{[0:t-1]}; x) = g(y_{t-1}, d_{t-1}, c)$$

where d_t is the decoder hidden state of time step t, y_t is the predicted output word symbol at t.

2.2 Copy mechanism model

The copy mechanism model [36] is a pointer-generator network, which is a hybrid between seq2seq network and pointer network [42]. This model allows both copying words via pointing and generating words from a fixed vocabulary which can be viewed as a balance between extractive and abstractive approaches. This model obeys the sequence-to-sequence architecture. Given the attention distribution a_t and the context vector $\boldsymbol{C}$, we can calculate the generation probability $p_{gen} \in [0, 1]$ as:

$$p_{gen} = f(C_t, s_t, y_t)$$

where f is a sigmoid function, s_t is decoder state and y_t is decoder input. In addition, p_{gen} is the soft switch to choose between generating a word from vocabulary or copying a word from source sentence according to the attention distribution a_t. The copy mechanism calculates the probability distribution over the extended vocabulary which consists of the vocabulary and all words from the source sentences:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t$$

Besides, the $P_{vocab}(w)$ is zero if w is an out-of-vocabulary (OOV) words. Similarly, a_i^t is zero if w not appear in the source sentence. In this way, the copying mechanism has the ability to deal with the issues of OOV words by repeating themselves which improves accuracy.

3 PATTERN CONTROLLED P-G NETWORKS

Although pointer-generator network gets good performance in practice especially in the field of summarization, it still works not so well in product description generation especially when generating a long description. The copy mechanism prefers to repeat source words with higher value of attention distribution which results in higher repetition of generated descriptions. Also, copy mechanism can't recognize the key information such as brand names and categories.

To deal with this issue, we proposed a Pattern-Controled Pointer-Generator Network (PGPCN) as shown in Figure 2. Basically, we use

the seq2seq model as the main building block. However, unlike the traditional RNN-based sequence model, we instead use coordinate encoders with attention mechanism to generate the context vector. Also, a style classification is added to encourage the encoder to attend to style-decorrelated tokens by gradients reverse. Finally, we use the features in the control module to generate outputs in decoders.

3.1 Content Block

Given the product title and property, we use a specialized Named Entity Recognition (NER) tool[2] for E-commerce to label entities in product title. In total there are 36 types of entities, which are of common interest in E-commerce scenario, such as "色"(color), "格"(style) and "修" (modifier). For example, in segmented product title "包Nike 品牌的色" (Nike red sweatpants with free shipping), "包"(Free shipping) is labeled as Marketing Service, "Nike" is labeled as Brand, "色"(Red) is labeled as Color and "" (Sweatpants) is labeled as Category. then we look up an embedding matrix E to get the word embedding $E = (e_0, e_1, e_2 \ldots e_D)$ and word NER embedding Ner to get the named entity recognition(NER) embeddings $Ner = (ner_0, ner_1, ner_2 \ldots ner_D)$. D denotes the dimension of embeddings and N denotes the vocabulary size. The input word vector $Xi = E_i + Ner_i$. By combining the NER information into the word vector, the model can easily recognize the important source words while striving to preserve them in the outputs at decode time to aid reproduction of factual product details.

3.2 Coordinate Encoder

To enhance the accuracy and diversity of the generated description, we fed the source token vectors of the product title and product properties into coordinate encoders (Transformers [41] feature attention networks with multiple attention heads and convolutional layers equipped with gated linear units [15]). Different from the traditional RNN encoder, the Transformer encoder and Gated-CNN encoder are parallel structures which can speed up training time. In recent work, Chen et al.[40] find that hybrid architectures with a Transformer encoder and an RNN decoder can outperform a pure Transformer model. They speculate that the Transformer encoder is better at encoding or extracting features than the RNN encoder. Thus, the Transformer encoder can improve the output accuracy of our model. CNNs are hierarchical networks, in that convolutional layers capture local correlations. The n-gram features captured by CNN encoder can enhance the diversity of the output description. Besides, our model is equipped with Gated Linear Units and residual connections [10].

For Transformer encoder, we compute the hidden state h_i^l as:

$$\overrightarrow{h}_i^l = \overrightarrow{h}_i^{l-1} + f(\text{self} - \text{attention}(\overrightarrow{h}_i^{l-1}))$$

Scaled dot-product attention is adopted as the basic attention function in the Transformer, which describes as:

$$\text{attention}(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})$$

where f represents a feed forward network with ReLU as the activation function and layer normalization.

As for convolutional encoder, the local context size depends on the size of kernel and number of layers. To keep the output length same as the input, the CNN model adds padding at each layer. Also, we add residual connections from the input of each convolution to the output of the block.

$$\overleftarrow{h}_i^l = \overleftarrow{h}_i^{l-1} + f_{cnn}(W^l[\overleftarrow{h}_{i-\lfloor k/2 \rfloor}^{l-1}, \ldots, \overleftarrow{h}_{i+\lfloor k/2 \rfloor}^{l-1}] + b^l])$$

where a convolutional kernel $k \in \mathbb{R}^{d \times q \times d}$ is applied to each possible window of q words. ConvS2S choose a Gated Linear Units (GLU) [10] which can be viewed as a gated variation of ReLUs.

$$f([A, B]) = A \otimes \sigma(B)$$

where A, B are the outputs of two stacked convolutional networks, $\otimes$ is the point-wise multiplication. The gates $\sigma(B)$ control which inputs A of the current context is relevant. At last, the encoder outputs are the average of Transformer outputs and Gated CNN outputs $h^l = \text{avg}(\overrightarrow{h^l}, \overleftarrow{h^l})$.

3.3 Attention based Decoder

Repetition is a severe problem for language generation task, especially for generating long text. We use the coverage model to solve this problem. A coverage vector $\boldsymbol{c}$ is maintained to keep track of which source words have been translated('covered') in the past decoding time steps.

$$c^t = \sum_{t'=0}^{t-1} a^{t'}$$

c^0 is initialized as a zero vector denotes that no source words have been translated. Intuitively, coverage vector is used as an additional input to the attention mechanism, which reminds the attention mechanism about its previous decisions.

$$e_i^t = v^T \tanh(W_h \boldsymbol{h}_i + W_s \boldsymbol{s}_t + w_c \boldsymbol{c}_i^t + \boldsymbol{b}_{attn})$$

The attention distribution is calculated as:

$$a^t = \text{softmax}(e^t)$$

Next, we use the attention distribution to produce a context vector $\boldsymbol{C}$ which is a weighted sum of the encoder hidden states:

$$C_t = \sum_i a_i^t h_i$$

3.4 Control Block

Although the end-to-end network described above can works very well, it still generate wrong informations in practice. Alse, the pattern of generated descriptions are uncontrollable which is inflexible for online deployment. To address this issue, we introduced a control block to intervene in the pattern of description in decoding phase.

As shown in Figure 2, there are three components in the control module: style control, length control, and category control. We use the style keywords to represent the text style, described as $\boldsymbol{k}$ which share the embeddings with input words:

$$k_i = E_{k_i} + Ner_{k_i}$$

[2]https://www.alibabacloud.com/zh/product/machine-learning

Also, we look up an embedding matrix E_{cate} to get the category embeddings $E_{cate_i} \in \mathbb{R}^{d^c \times N^c}$ where d^c represents the dimension of category embeddings and N^c represents the maximum size of category. As for length control, a vector $\boldsymbol{L}$ is maintained to represents the distance between each word i to the end $\langle EOS \rangle$. Suppose the length of output text is n, $\boldsymbol{L}$ can be described as:

$$\boldsymbol{L} = (L_0, L_1, ..., L_n)$$

where $L_i = distance(i, \langle EOS \rangle)$ is the distance from position i to the end. At the decoding phase, we initialize L_0 with the value of target length. In addition, the L_i will decrease one at each time step before it fed into the model. When L_i equals 0, it means next word is the end of the sentence $\langle EOS \rangle$ and prediction will be terminated. The length parameter tells the model how much longer until the ending of the decoder. Besides, we look up an embedding matrix $E_L \in \mathbb{R}^{d^l \times N^l}$ to get the length embeddings before we feed it into the encoder. Here, d^l denotes the dimension of the length embeddings and N^l denotes the maximum length of target text.

We follow the architecture of pointer-generator network, so the generation probability p_{gen} for timestep t is calculated from the context vector C_t, the decoder states S_t , the decoder input x_t and the pattern control parameter E_{Li}, E_{Cate_i}, k_i:

$$p_{gen} = \sigma(w_i^T \boldsymbol{C}_t + w_s^T \boldsymbol{S}_t + w_x^T \boldsymbol{x}_t + w_L^T \boldsymbol{E}_{\boldsymbol{L}t} + w_{cate}^T \boldsymbol{E}_{\boldsymbol{cate}t} + w_k^T \boldsymbol{k}_t + \boldsymbol{b})$$

3.5 Classification Block

As shown in Figure 2, we feed the encoder vector into a style classification network. We draw from a line of research in the style of [5, 13, 20, 33]. This method involves passing gradients through a gradient reversal layer, which multiplies gradients by a negative constant, i.e. -1, as they propagate back through the network. Intuitively, this encourages parameters to update away from the optimization objective.

If $Loss_s$ and $Loss_c$ are the losses from two different prediction task of our network, where s represents description generation task, and c represents classification task. Thus, the loss function of our multi-task framework becomes $Loss = Loss_s + Loss_c$. The classification loss $Loss_c$ can be described as

$$Loss_c = \mathbb{E}_{h^i \sim \mathcal{H}} \left[-\log p_c(k_j \mid h^i) \right]$$

where k_j is the style keyword represents the style of product description i as discussed in 3.4. However, when backpropagating from each prediction network to the encoder, we reverse the gradients of the networks that are predicting style. This means that the prediction networks still learn to predict style, but the encoder is forced to learn style-invariant representations which are not useful to these downstream tasks. We hope that such representations encourage the encoder to attend to style-decorrelated tokens. In this way, we can easily control the style from a encoder vector without style features. Then we can use the style parameter in control block to generate the text in a specified style.

Table 1: The statistics of the pair dataset.

Train dataset size	3,516,805
Test dataset size	50,000
Avg. length of original titles	12
Avg. length of product properties	6
Avg. length of product description	25
Max. length of product description	50

Table 2: Parameter settings of our system.

Dim. of word embeddings	128
Dim. of ner embeddings	8
Num. of encoder/decoder hidden units	512
Adam optimizer	$lr = 1e-4$
Beam size	5
Minibatch size	128
Product category size	52
Vocabulary size	50000

4 EXPERIMENTS

4.1 DataSet

Our proposed method mainly requires two kinds of data: extended product description data and product title data[3]. These data are collected from **taobao**[4] a popular E-commerce website in China, since there is no public benchmark dataset for our task yet. We leverage the product description data from a product recommendation channel and product detail page. Displayed long descriptions are all written by experienced copywriters. We exclude the products whose descriptions are shorter than 8 Chinese characters and longer than 50 Chinese characters, since the product descriptions should be long enough to cover the essentials but short enough to keep it interesting. Moreover, product titles and property data are crawled from the detail page.

These two kinds of datasets are merged and organized as pairs, i.e.,(product title, product property and product description). Therefore, the dataset can be represented as $\langle T, D \rangle$, where T means the products' titles and the products' properties, and D means the products' descriptions. The details of our dataset are shown in Table 1.

4.2 Comparison Methods

To verify the effectiveness of our proposed model, we implement rich description generation methods. We compare our method with popular neural machine generation architectures, including:

- Vanilla sequence-to-sequence (**Seq2Seq**) is a basic encoder-decoder model based on LSTM unit and attention mechanism [1].
- Pointer-Generator (**Ptr-Gen**) [36] is a hybrid neural network model combing Seq2Seq-Gen with pointer network . Besides copying words from the input, it can also generate words from a predefined vocabulary.

[3]The dataset and code will be released soon.
[4]https://www.taobao.com

- Convolutional-based NMT (**ConvS2S**) [15] are hierarchical networks which is equipped with Gated Linear Units (GLU) and residual connections.
- Transformer-based NMT (**Trans**) [41] rely heavily on self-attention networks. Each token is connected to any other token in the same sentence directly via self- attention. Transformers feature attention networks with multiple attention heads which is more fine-grained, compared to conventional 1-head attention mechanisms.

4.3 Experiment Setup

For all experiments, we use the Xavier scheme [16] for parameter initialization, where weights are initialized using a normal distribution $\mathbf{W}_{i,j} \sim \mathcal{N}(0, \sigma)$, $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$; n_{in} and n_{out} are numbers of the input and output units of the network. We train all models using Adam with leaning rate $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e-8$. We use 128-dimensional word embeddings, 8-dimensional word ner embeddings and 512-dimendional hidden states for LSTM units in both encoder and decoder. For bidirectional implementations, we linearly transform the forward hidden states and backward hidden states into 512-dimensional states.

We implement all the models in Tensorflow1.4[5]. For Ptr-Gen[6] we modify the I/O of the code to fit our dataset. For ConvS2S and Transformer[7], we use an open-source Python library OpenNMT. All the models are trained on a single Tesla P100 GPU with a batch size of 128. At train time, we use maximum encoding step 10 and maximum decoding step size 50. All training instances are sorted by the target sequence length and partitioned into mini-batches. The shorter sequences are padded to have the same length as the longest sequence in the batch. All batches are shuffled at the beginning of each epoch. At test time, we use beam search with beam size 5 and maximum decoding step size 20.

For the dataset preprocessing, all models are implemented based on Chinese characters. We use a Chinese word tokenizer and a pre-trained Ner tool for E-commerce to segment and label entities. Finally, our vocabulary contains 50k words according to the word frequency and 36 word entities. More detailed network parameters are presented in Table 2.

4.4 Automatic Evaluation

To evaluate the content prediction performance, we use some standard metrics (eg., BLEU [32], ROUGE [26], METEOR [2]). BLEU metrics is widely used for text generation tasks, such as machine translation, summarization. The metric analyzes co-occurrences lexical n-grams units between the candidate and references. For BLEU metric, we report BLEU-1, BLEU-2 here. The ROUGE metric measures the description quality by counting the overlapping units (e.g., n-grams) between the generated candidate and reference descriptions. Following the common practice, we report the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. In addition, the METEOR metric was introduced to address several weaknesses in BLEU. It is calculated based on an explicit alignment between the unigrams in the candidate and references. In this work, the METEOR scores are evaluated in exact word match mode. We obtain the BLEU and METEOR scores using the nlg-eval11[8] packages, and ROUGE scores using python-rouge12[9] package.

The results of the last group indicate that the control block is helpful for description generation. Pattern-Controled Pointer Generator Network (PGPCN) achieves higher BLEU, ROUGE and METEOR scores on test dataset than other methods. PGPCN is a hybrid architecture with a Transformer encoder and an RNN decoder. Transformer encoder is proved to be better at encoding and extracting features than the RNN encoder. Also, we use the copy mechanism and control block to improve the quality of the output description. We did a comparative experiment that only used word embeddings as encoder input without NER information as shown in the last two group of Table 3. The results show that encoder with multi-data input including word embeddings and NER embeddings helps the model to get better performance. By using the NER information, the model can easily recognize the important source words while striving to preserve them in the outputs at decode time. This is consistent with our expectation as discussed in Section 3.1.

Also, we calculate the differences of metric score in different length, here we compare Chinese character size of length 10 with length 20. We use LENGTH-GAP represents the average retention rate by dividing the metric score of length 10 into score of length 20. The results show that our PGPCN method can reserve more information in short length than the other method. This is because our model takes the length as a model parameter during training step, it can memorizes the distance to the end $\langle EOS \rangle$, so it will compress key information in advance. However, the model without length control can only get fixed length by truncating which will result in loss of information.

4.5 Key Information Retention Test

For product description, it is particularly important to retain some key information (i.e., band names, commodity names) which is an essential requirement to attract people. This information helps users understand the product and accurate the descriptions of products. The evaluation metric for this task is the error rate of the brand names (B-ACC.) and commodity names (C-ACC.) in the generated product description. If the key information is missing or getting wrong in the product description, this prediction will be marked as an error. We also test the model on the sampled online data from a real-world E-commerce website.

The results of each model on test sets are shown in Table 4, seq2seq achieves the worst results in accuracy task due to the simpleness of the model architecture. Also, we find that models with pointer-generator mechanism can get higher accuracy than regular generator model, since the pointer mechanism can copy the relevant words to the output text directly. The conventional NMT model like ConvS2S and Transformer perform poor in Accuracy metric. Both of them are stuck in the source words, while some words in product title are ambiguous due to the similarity of category between E-commerce product.

[5]https://www.tensorflow.org
[6]https://github.com/abisee/pointer-generator
[7]https://github.com/OpenNMT/OpenNMT-py
[8]https://github.com/Maluuba/nlg-eval
[9]https://github.com/tagucci/pythonrouge

Table 3: BLEU, ROUGE (F1), and METEOR scores on the test set. Baselines on the top group are seq2seq. Bold scores are the best overall.

		BLEU-1	BLEU-2	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	Info-Retention
Seq2Seq	Length 10	6.13	2.49	11.05	3.43	10.80	9.98	0.68
	Length 20	11.95	4.71	17.15	3.56	16.17	12.82	
Ptr-Gen	Length 10	7.29	3.68	12.94	5.68	12.91	13.70	0.72
	length 20	12.70	5.81	18.05	7.60	17.62	15.07	
ConvS2S	Length 10	7.54	3.71	13.98	7.78	13.17	14.81	0.71
	Length 20	13.41	6.22	20.62	8.34	20.21	17.61	
Trans	Length 10	7.97	3.73	13.39	7.76	13.36	13.95	0.71
	Length 20	13.71	6.58	20.09	8.35	19.91	16.59	
PGPCN$_{noNER}$	Length 10	7.43	3.12	12.70	7.17	12.43	13.15	0.75
	Length 20	12.09	5.55	17.73	7.22	17.32	14.87	
PGPCN	Length 10	**9.38**	**4.32**	**15.68**	**10.01**	**15.61**	**15.35**	**0.76**
	Length 20	**15.56**	**6.92**	**22.00**	**10.07**	**21.86**	**16.94**	

Table 4: Results of key info retention experiment. Bold scores are the best.

	Length	C-ACC.	B-ACC.	Repete RATE
Seq2Seq	10	3.72	4.98	7.76
	20	3.46	4.97	
Ptr-Gen	10	1.33	1.09	6.62
	20	1.15	1.03	
ConvS2S	10	2.58	2.57	**4.94**
	20	2.34	2.53	
Trans	10	1.57	2.96	6.94
	20	1.65	2.54	
PGPCN$_{noCB}$	10	1.27	0.90	5.17
	20	1.06	0.73	
PGPCN	10	**0.30**	**0.22**	5.36
	20	**0.28**	**0.21**	

Table 5: Manual evaluation results. Bold scores are the best. The improvements from baselines to PGPCN is statisti- cally signicant according to two-tailed t-test (p < 0.05).

Model	Completeness.	Consistency.	Readable.	Attract.
Seq2Seq	76.32%	62.90%	1.13	1.24
Ptr-Gen	93.13%	90.13%	3.52	2.92
ConvS2S	83.37%	80.23%	4.27	3.45
Trans	87.29%	82.95	4.53	3.67
PGPCN$_{nC}$	**94.51**%	91.07%	4.55	3.55
PGPCN	93.38%	**95.73**%	**4.60**	**3.96**

Moreover, the significant improvements on Accuracy metrics demonstrate that our method PGPCN can better recognize the commodity names and brand names after the intervention of the Control Block (CB). We also calculated the word repetition rate (Repete RATE.) of all description generated from each model. The results show that ConvS2S model performed best and got the lowest repetition rate. This is because the ConvS2S model has a robust capability to learn compressed expressions and address sentences with variable lengths. Also, the convolutional units can extract the N-gram features which is helpful to understand the text. As a result, our method has lower repetitive rate score than other non-convolutional methods.

4.6 Manual Evaluation

Like the previous work ([12], [39]), we also conduct manual evaluation on the generated product descriptions. As such, evaluation by human judges of the quality of generated text is the best measure of our methods' quality. We randomly sampled 500 products from our test datasets. Three experienced E-commerce website users were asked to measure the quality of the product descriptions. Three perspectives are considered during manual evaluation process: (1) **Completeness**, is the key information properly kept in the product description? (2) **Consistency**, is the generated description consistent with the given styles? (3)**Readability** (Readable), how fluent, grammatical the product description is? (4) **Attractiveness** (Attract.), how attractive the product description is?

Essential information retention is an extra and crucial requirement for product description generation. We use a very strict criterion for this property, the generated product description will be assessed as one only if it correctly retains both brand name and commodity name, otherwise 0. To make the annotation results more consistent and reliable, we excluded instances with divergent ratings (i.e., their variance is greater than the threshold). The readability and attractiveness are assessed with a score from 1 to 5 (i.e., 1 for worst and 5 for best).

The average results are shown in Table 5. The results indicate that our PGPCN method outperforms the conventional neural language generation method. The effects on Completeness show that all models built with copy mechanism can get a particularly good mark, since the copy mechanism can better retain the keywords from the inputs. The significant improvements to Attract metric demonstrate that our PGPCN can learn style information from the keywords parameter from the control block. Also, comparing with the PGPCN with no classification ($PGPCN_{nC}$), the PGPCN with classification can get a higher score on Consistency metrics. It mainly because the classification block can filter the irrelevant style by the gradient reverse. Considering all three metrics, our PGPCN produces more readable and more informative descriptions which show the advantages of classification and control block.

4.7 Online A/B Testing

This subsection presents the results of online evaluation in the product recommendation page scenario of an E-commerce mobile app with a standard A/B testing configuration. We deploy it in a real word product during E-commerce website sales day[10]. The A/B testing lasted for a half day, since millions of people visit our website during that sales day and we have got adequate users for testing. For online deployment, we generate fixed length description for about five hundred thousand sales product using PGPCN model. Moreover, we use different keywords to intervene in the description pattern, such as "新品" (new product) for new product scenario recommendation and "促" (promotion) for promotional products sales scenario. The baseline method uses several template description that only changing the product name. The A/B testing system split online users equally into two groups and direct them into two separate buckets respectively (about 1 million user views (UV) for each bucket). Then for users in the A buckets (i.e., the baseline group), the display product descriptions are generated by using a predefined template. While for users in B buckets (i.e., the experimental group), the display product descriptions are generated by the PGPCN method. We adopt the Click-Though Rate (CTR) to measure the performance of our description. In our recommendation scenario, product descriptions are displayed on the front page which is a crucial deciding factor in determining whether to click the product or skip to another. It can be obtained as:

$$CTR = \frac{product\_click}{product\_PV}$$

where product_click is the click times of the product, product_PV is the page view times of the product. We calculated the overall CTR for all products on sales day (from 20180920 12:00 to 20180920 22:00). We found that the performance in the experimental bucket is significantly better than that in the baseline bucket. PGPCN improved the CTR by 5.21% over the baseline. The results of the CTR improvement indicates that the users are more likely to click the products with the PGPCN generated descriptions.

[10]https://www.1688.com

4.8 Case Study

In this subsection, we show three realistic cases of the description generated by different methods. Generally, models without pointer-generator mechanism method achieve worse performance as shown in Table 6. The generator models are more likely to generate some irrelevant information such as Korea in the first example generate by Ptr-Gen. Also, we find that ConvS2S and Transformer prefer to generate some safe words, which may get a higher score at beam search step. However, this words can't help to improve the quality of the product description.

From the second case, we also found that the model without Control Block could not recognize the product category precisely. The Transformer model regard table lamp as robot toy which results in poor user experience. Consider the brand name of two examples, it is obvious that the model with pointer mechanism can recognize the brand information and copy it to the output. Besides, all method can produce fluent and grammatic descriptions due to the encoder-decoder sequential learning method.

We also give an example of how style keywords affect the output description as shown in Table 7. We use different keywords as the input of control parameter. Apparently, the results show that the keywords can control the description generation in different style. This is quite impressive, if we consider the user feature, we can generate personalized product description for different users according to their preference

5 RELATED WORK

Automatic text generation from structured data (Data-to-Text) is an import task in Natural Language Generation (NLG). Its primary goal is to produce readable text in natural language, from a non-linguistic data input. In NLG communities, machine translation[23, 38] and summarization [9, 44], which are two similar tasks to description generation. Machine translation aims for complete transduction of the sentence to be translated. Summarization condenses one long article to several sentences to highly summarize the article. Differently, this form of NLG is typically taken to require addressing (at least) two separate challenges: content selection, the selection of an appropriate subset of the input sequence [3, 4], and surface realization, producing text according to a given grammar.[21, 29]. Traditionally, these two challenges have been modularized and handled separately by generation systems.

Previous researches exploit NLG task in various domains. For example, Reiter et al. [34] generate weather forecasts from meteorological records and Chen and Mooney [6] generate sportscasting text from robot soccer events in RoboCup dataset. Christina Sauper et al. [35] investigate an approach for creating a multi-paragraph overview article. Jinpeng Wang et al. [43] present a statistical framework that generates product description from product attributes. A hierarchical semi-Markov method is proposed [25] which first associates the text sequences to corresponding records and then generates corresponding descriptions from these records. These works use a content selection way by selecting relevant material from source data and organizing them which is grammatically and syntactically. While this method is robust and portable across domains, output sequence often suffers from coverage and coherence problem.

Table 6: Examples of generated product description by different method. “␣” denotes visible whitespace in Chinese context.

Product Title	LIM's␣蓝山␣风味␣咖啡豆 LIM's Blue Mountain Flavored Coffee Beans
Seq2Seq	这␣款␣咖啡␣是␣以␣优质␣的␣咖啡豆␣为␣原料␣，␣经过␣深度␣烘焙␣制作␣而␣成␣，␣带␣有␣微微␣的␣玫红色␣，␣非常␣美妙␣。 This coffee is made from high-quality coffee beans and deeply baked. It has a slight rose red color and is very beautiful.
Ptr-Gen	LIM's␣蓝山␣风味␣咖啡豆␣，␣韩国␣原装进口␣，␣选用␣优质␣咖啡豆␣为␣原料␣，␣浓浓␣的␣，␣口感␣公主风␣柔滑␣。 LIM's Blue Mountain coffee beans, imported from South Korea, made from high-quality coffee beans, taste smooth ,like a princess.
ConvS2S	␣蓝山␣风味␣咖啡豆␣，␣精选␣优质␣咖啡豆␣，␣经过␣烘焙␣，␣烘焙␣出␣更多␣的␣咖啡␣。 Blue Mountain flavored coffee beans, selected high quality coffee beans, blew for more.
Trans	␣蓝山␣风味␣咖啡豆␣，␣由␣阿拉比␣卡␣咖啡豆␣为␣原料␣，␣经␣烘焙␣后␣精心␣烘焙␣而␣成␣，␣散␣发出␣浓郁␣的␣坚果␣香气 Blue Mountain flavored coffee beans, from Arabica, medium roast, giving off a rich nutty aroma.
PGPCN	LIM's␣蓝山␣风味␣咖啡豆␣，␣精选␣优质␣咖啡豆␣，␣经过␣精心␣烘焙␣制作␣，␣口感␣清新␣，␣味道␣香浓 LIM's Blue Mountain flavored coffee beans, carefully selected and fresh roasted, enjoy the rich, yet smooth and full-bodied taste.
Product Title	耐尔金␣创意␣时尚␣复古␣机器人␣卧室␣床头␣台灯 Nilkin Creative Fashion Vintage Robot Bedroom Table Lamp
Seq2Seq	这␣是␣一␣款␣创意␣十足␣的␣台灯␣，␣采用␣优质␣树脂␣材料␣精心␣制作␣而␣成␣，␣造型␣可爱␣，␣充满␣童趣␣。 This is a creative table lamp, crafted from high-quality resin materials, cute and full of childlikeness.
Ptr-Gen	这␣是␣一␣款␣非常␣有␣创意␣的␣台灯␣，␣阅读灯␣与␣清爽型␣的␣完美␣结合␣，␣简约␣而␣不␣简单␣。 This is a very creative table lamp, the perfect combination of reading light and refreshing style, more than just simple.
ConvS2S	树脂␣材质␣机器人␣，␣触摸␣可调光␣，␣光线␣易␣透过␣智能␣时尚␣。 Resin-based robot, touch dimmable, smart and stylish, light is easy to pass.
Trans	床头␣灯␣机器人␣玩具，␣采用␣优质␣铁艺␣材质␣，全新␣的␣设计␣，␣安全␣环保␣。 Bedlamp robot toy, made of high quality wrought iron material, new design, safe and environmentally.
PGPCN	耐尔金␣卧室␣台灯␣，␣机器人␣造型␣，␣创意␣时尚␣，␣个性␣有趣␣，␣装饰␣效果␣极␣佳␣。 Nelkin bedroom robotic table lamp, creative design, seems vivid and full of personality, adds a neat decorative touch to the room.

Table 7: Examples of generated product description from different style with length 9. “␣” denotes visible whitespace in Chinese context.

Product Title 羊毛呢␣毛领␣时尚␣女␣大衣␣大码␣厚␣连帽␣外套 Wool Collar Fashion Women's Coat Large Size Thick Hooded Coat	
KeyWord	Product Description
爆款 Top-selling	大家␣都␣在␣买␣的␣爆款␣连帽␣外套␣。 Top-selling hooded coat people were purchasing.
折扣 discount	年末␣折扣␣大␣作战␣，␣毛领␣大衣␣来␣帮你 Year-end discount battle, fur collar coat help you.
简约 simple	连帽␣外套␣，␣简约␣不␣单调␣的␣穿␣搭 Hooded jacket, simple and not monotonous
小公主 little princess	大毛领␣外套␣，␣做␣个␣温暖␣系␣的␣小公主 Big fur collar coat, to be a warm little princess

Recently, sequence-to-sequence (Seq2Seq) neural network model based on encoder-decoder are widely used in existing NLG approached.Remi Lebret [24] introduces a neural model for concept-to-text generation. Chen [7] propose a general neural networks model based on attentional sequence-to-sequence model for text generation. Tianyu Liu et al. [28] proposed a novel structure-aware seq2seq architecture which consists of a field-gating encoder and description generator with dual attention. Mei, Bansal, and Walter [30] also proposed a seq2seq model with an aligner between weather records and weather broadcast. The model used one-hot encoding to represent the weather records as they are relatively simple and highly structured.

Moreover, pointer networks have been widely explored in abstract generation task. Pointer networks [42] allows to produce output sequence consisting of elements from the input-side words and have been used for task like extractive summarization [17, 37], machine translation [18] and dialogue generation [11]. In addition, it has also been shown to be helpful for geometric problems [42], question answering [22, 45, 46], code generation [27], and language modeling [31]. This model uses the soft attention distribution by Bahdanau et al.[1] which allows the model to learn alignments between modularity. Pointer-generator [36] network provided a viable new approach for abstractive text summarization by extending pointer network in a soft or hard way to decide whether to generate a token from the predefined vocabulary or the source sequence.

In this paper, derived from the general pointer-generator network, we propose coordinate encoders (Transformer [41] encoder and gated convolutional encoder [14, 15] equipped with self attention mechanism) to learn the representation of source input data and a pattern controlled decoder generates sentences using an attention mechanism. To the best of our knowledge, we make the first attempt to develop a neural data-driven solution for product description generation. Besides this study is launched on a dataset

with up to two million product description data, which is much higher than the data scale of previous studies and make the results more convincing.

6 CONCLUSION AND FUTURE WORK

Inspired by the neural language generation theory and the importance of content marketing of E-commerce, we proposed a neural model for product description generation. However, different from the conventional task of language generation, product description generation has more constraints like repetition and accuracy, since it is a divergent task. We address this problem with a pattern-controlled neural model by leveraging pointer-generator mechanism paradigm. The proposed model uses coordinate encoders combining the Transformer features with the Gated CNN features. In addition, we use a control block to intervene in the pattern (ie., length, category, style) of output description. Extensive experiments demonstrate the validity of the proposed method and the superiority of our approach over four language generation baselines. Finally, online A/B testing shows the significant business impact of our model in a mobile E-commerce app.

For further improvements, an interesting direction would be using user profile features to generate product descriptions. We find out one product may have different selling point in different user's perspectives, so it is possible for us to generate personalized descriptions for different users according to the user profile and preference. Another direction is how to combine the picture features into our model, and we can get more product information to enrich the generated description.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[3] Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 331–338.

[4] Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039* (2004).

[5] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*. 137–144.

[6] David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*. ACM, 128–135.

[7] Shuang Chen. 2018. A General Model for Neural Text Generation from Structured Data. *E2E NLG Challenge System Descriptions* (2018).

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 93–98.

[10] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083* (2016).

[11] Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024* (2017).

[12] Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1481–1491.

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* 17, 1 (2016), 2096–2030.

[14] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344* (2016).

[15] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* (2017).

[16] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[17] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* (2016).

[18] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* (2016).

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Fredrik Johansson, Uri Shalit, and David Sontag. 2016. Learning representations for counterfactual inference. In *International Conference on Machine Learning*. 3020–3029.

[21] Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Vol. 3. Pearson London.

[22] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* (2016).

[23] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1700–1709.

[24] Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771* (2016).

[25] Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 91–99.

[26] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[27] Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744* (2016).

[28] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text Generation by Structure-aware Seq2seq Learning. *arXiv preprint arXiv:1711.09724* (2017).

[29] Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1611–1622.

[30] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838* (2015).

[31] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[33] Reid Pryzant, Young-Joo Chung, and Dan Jurafsky. 2017. Predicting Sales from the Language of Product Descriptions. In *Proceedings of the SIGIR 2017 Workshop on eCommerce (ECOM 17)*.

[34] Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167, 1-2 (2005), 137–169.

[35] Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 208–216.

[36] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* (2017).

[37] Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. 2018. Multi-Source Pointer Network for Product Title Summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 7–16.

[38] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[39] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1171–1181.

[40] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. *arXiv preprint arXiv:1808.08946* (2018).

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[42] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. 2692–2700.

[43] Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A Statistical Framework for Product Description Generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Vol. 2. 187–192.

[44] Jingang Wang, Junfeng Tian, Long Qiu, Sheng Li, Jun Lang, Luo Si, and Man Lan. 2018. A Multi-task Learning Approach for Improving Product Title Compression with User Search Log Data. *arXiv preprint arXiv:1801.01725* (2018).

[45] Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* (2016).

[46] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 189–198.