Selling Products by Machine: a User-Sensitive Adversarial Training method for Short Title Generation in Mobile E-Commerce

Manyi Wang, Tao Zhang, Qijin Chen, Chengfu Huo, Weijun Ren
Alibaba Group
{manyi.wmy,guyan.zt,qijin.cqj,chengfu.huocf,afei}@alibaba-inc.com

ABSTRACT

In E-commerce portals, merchants don't have the ability to write various titles for different customers, so they usually tend to write lengthy product titles to get most buyers' attention. It is a crucial task to extract relevant keywords for displaying on the limited screen of mobile phones to fetch customers. Previous studies mainly focus on content-based methods, however, lack of user features may result in the generated titles containing rich product information while ignoring the user requirement. In this paper, we propose a Personalized Pointer Generative Adversarial Network (PPGAN) to generate personalized user-sensitive short titles. Even though in our sparse dataset the user-clicked data is limited due to the low Click-Through-Rate (CTR), our model could encourage the discriminator to identify the high-quality short titles from the user-unclicked data by employing an unsupervised information-theoretic assignment strategy. An extensive set of experiments on a large-scale E-commerce dataset indicates the advantage of the proposed method for generation qualities. Finally, we deploy our model into a real-word online E-commerce environment and boost the performance of CTR compared with state of art baseline methods.

CCS CONCEPTS

• **Information systems** → **Summarization**; **Personalization**.

KEYWORDS

Title summarization, Wasserstein GAN, User-sensitive

ACM Reference Format:
Manyi Wang, Tao Zhang, Qijin Chen, Chengfu Huo, Weijun Ren. 2020. Selling Products by Machine: a User-Sensitive Adversarial Training method for Short Title Generation in Mobile E-Commerce. In *2nd Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD'), August 24, 2020, San Diego, California, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

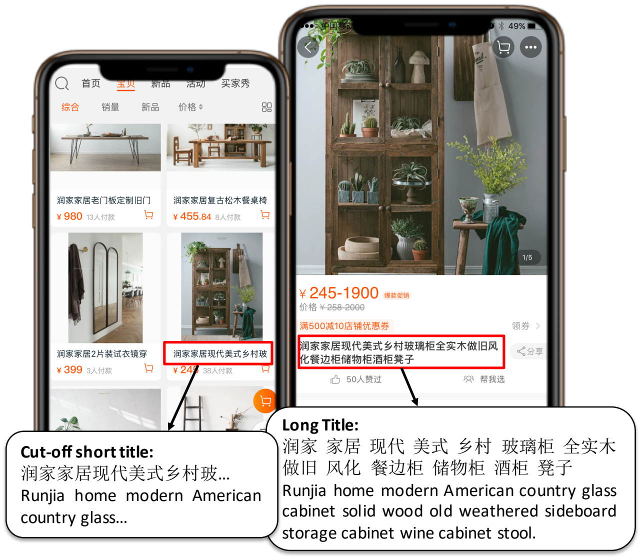
Cut-off short title:
润家家居现代美式乡村玻...
Runjia home modern American country glass...
Long Title:
润家 家居 现代 美式 乡村 玻璃柜 全实木 做旧 风化 餐边柜 储物柜 酒柜 凳子
Runjia home modern American country glass cabinet solid wood old weathered sideboard storage cabinet wine cabinet stool.
¥245-1900
润家家居现代美式乡村玻璃柜全实木做旧风化餐边柜储物柜酒柜凳子
润家家居现代美式乡村玻

Figure 1: A product with original long titles and cut-off short titles.

1 INTRODUCTION

Nowadays, the mobile Internet has become the leading platform for E-commerce. More and more online shopping transactions are completed on mobile phones instead of PCs. This trend demands many E-commerce giants such as Amazon, eBay, and Taobao to put more effort to improve the user experience of their mobile Apps. Usually, the merchants write product titles on their own. For more likely to be retrieved to the user by the search engine, such titles are often tedious, over-informative, and difficult to read. As shown in Figure 1 - Right Side, the product title consists of more than 30 Chinese words, serious hurting users' browsing experience. It is acceptable to display on PCs, but cannot be fully displayed on mobile phones due to the screen size limit, as shown in Figure 1 - Left Side. E-commerce websites should attract their customers with the product titles that will actually assist them to find the right product. Under the circumstances, generating a short and personalized product title for each customer is an essential and practical research problem in Mobile E-commerce.

This problem is related to text summarization, aiming to generate short summaries from long documents or sentences. Existing methods can be categorized mainly into two perspectives: extractive [2, 20] and abstractive [24, 28, 32]. Extractive summarization methods produce summaries by concatenating several sentences or words found directly in the original texts, whereas abstractive summarization methods generate a text summary based on the original sentence, which can generate text beyond the original input text.

The application scenario of our task is product title summarization in the field of E-commerce.Comparing with the conventional sentence summarization, an extra and essential constraint is that merchants usually do not want the generated short titles mingled with the words, not in their original titles. Moreover, short titles with any incorrect information may affect the sales or click-through rate of the product, which is unacceptable for sellers and may bring severe impacts. For example, generating the wrong brand "Apple" in the short title of the product "Microsoft". Thus, we adopt the extractive methods for our work.

Existing research of short product title generation [9, 25] mainly treat short title generation as a sentence compression task following attention-based extractive mechanism. In the real world, it is challenging to generate concise and non-redundant short titles from lengthy and verbose original titles. Another recent state-of-the-art work [31] introduces additional information from the products' visual image and attribute tags. This work adopts Generative Adversarial Network [10] (GAN) from the image domain [5, 13] to generate better short product titles. Methods above only consider the feature of the product itself but ignore the behavior of the target audience. However, words that directly impact each customer's engagement and conversation must be different.

Inspired by these, in this paper we propose a novel Personalized Pointer Generative Adversarial Network, abbreviated as PPGAN, to generate better short product titles that users will appreciate. Our model contains a generator and two discriminators. The generator employs two encoders to encode the product title and the user feature. It purposes to cheat the authentic discriminator to believe all generated samples are all real user clicked product titles. On the other hand, due to the low Click-Through-Rate (CTR), the dataset is sparse, which means the number of clicked samples is much less than that of unclicked samples. It is also too strict to label all unclicked short titles as negative samples, since the unclicked user may merely not interest in the product itself. Hence, another CTR discriminator tries to distinguish the high-quality short title from user-unclicked data and employ them to train generator continuously. Through the adversarial training between the generator and the discriminator, the generator can trick the discriminator successfully, generate short titles closed to the real ultimately and solve the problem of sparsity of dataset.

To the best of our knowledge, this work is the first that attempts to apply the user behavior feature to generate personalized title summarization. PPGAN's source code and datasets are available at https://github.com/Automanmm/PPGAN.

Our contributions can be summarized as follows:

- We propose an adversarial generation model fusion with CTR to automatically generate product short titles by using Wasserstein GAN [1] (WGAN). We focus on mining users' interest points and generate personalized and user-preferred short titles.
- By employing an unsupervised information-theoretic assignment strategy, our model could encourage the discriminator to identify the high-quality short titles from the user-unclicked data to train the generator, which addresses the problem of sparse data caused by low CTR.
- Our solution adopts a one-step Pointer Networks (Ptr-Net) to generate product short titles that solve the time-consuming problem of traditional multi-step Pointer Networks. PPGAN allows higher network throughput, which is required for online personalized title display marketing.

Extensive experiments on a large-scale real-world dataset with A/B testing show that PPGAN outperforms other baselines. The proposed approaches have been deployed in Alibaba online system to serve millions of customers per day, contributing significant improvement to the business.

2 RELATED WORK

Our work is related to text summarization tasks. In terms of text summarization, existing methods can be categorized into extractive and abstractive methods. Extractive summarization methods produce a text summary by extracting and concatenating several words directly from the original sentence. Abstractive summarization methods generate a text summary based on the original sentence, which can generate text beyond the original input text and usually generate more readable and coherent summaries. Traditional extractive methods heavily rely on human-engineered features, mainly including binary classifiers [15], Markov models [4], graphic model [7, 21] and integer linear programming (ILP) [30]. Recently, RNN-based approaches gradually become the main research direction in text summarization tasks. [11] proposes COPYNET, which use the copying mechanism to extract subsequences from the input sequence and put them at proper places in the output sequence. [24] provides a new approach for abstractive text summarization by extending the pointer network to decide whether to generate a token from the predefined vocabulary or the source sequence. For short product titles generation task, most of the methods are following extractive mechanism. [25] proposes a multi-source pointer network by adding a new knowledge encoder for pointer network. [9] considers rich semantic features of long product titles. [29] designs a multi-task model and uses user searching log data as an additional task to facilitate keywords extraction from original long titles. [31] introduces additional information from the products' visual image and attribute tags and adopts GAN to generate short titles. These studies all merely consider the characteristics of the commodity side but ignore the information of the user side. Our work aims to generate short product titles that users will click by combining product information with the user feature. Also, motivated by [6], we employ an unsupervised information-theoretic assignment strategy to identify the high-quality short titles from the user-unclicked data to train the generator better.

3 BACKGROUND

3.1 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 c of the source sequence L and decoder transforms the context vector c to the target sequence S. Given a training pair (L, S):

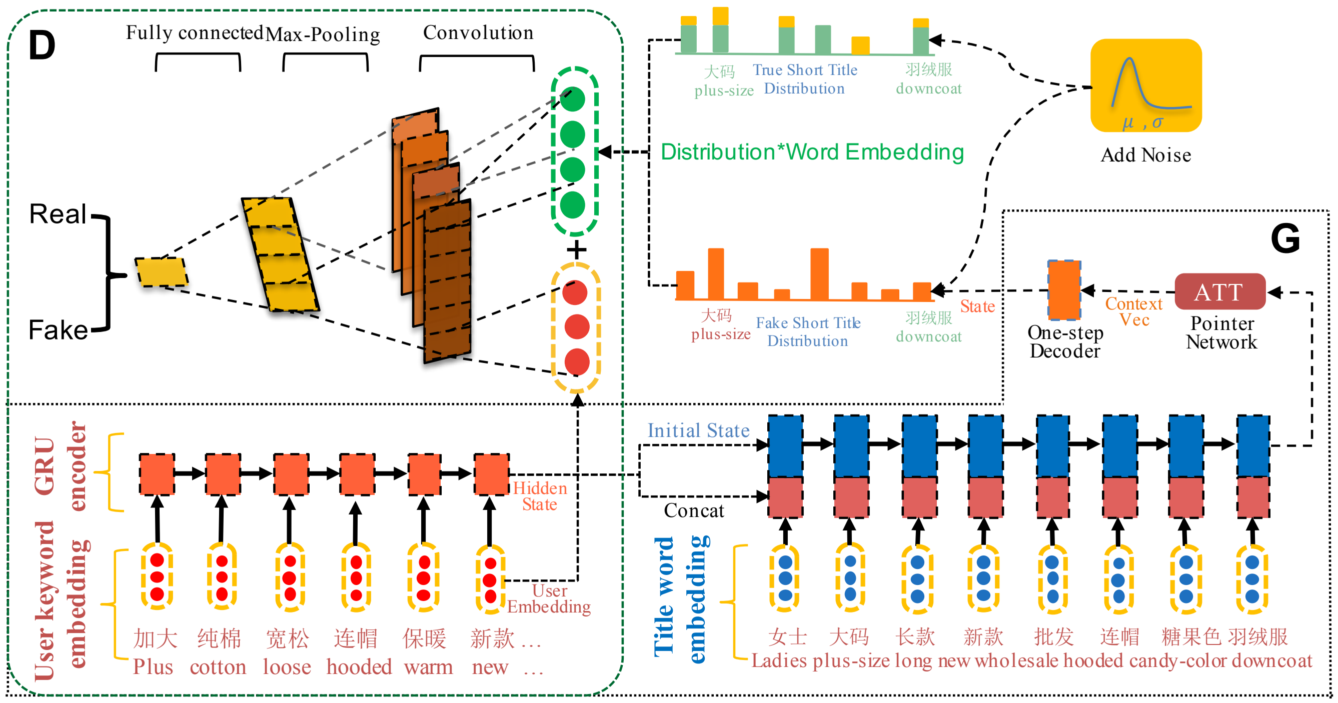

Figure 2: Architecture of Personalized Pointer Generative Adversarial Networks.

$$\boldsymbol{p}(S|L;\theta) = \prod_{i=1}^{m} p(s_i|s_1, s_2, ..., s_{i-1}, L; \theta)$$

where s_i is represented as the word embedding of word s_i. In practice， the activation function of RNN unit such as LSTM [12] and GRU [3] 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 as a weighted sum of annotations of the encoder states.

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

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

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

The parameters of the model are learnt by maximizing the conditional probabilities for the training set,i.e.

$$\boldsymbol{\theta} = argmax_{\theta} \sum_{L,S} \log p(S|L;\theta)$$

3.2 Pointer Networks

Different from the vanilla seq2seq models, pointer network use the attention mechanism as a pointer to select tokens from the input sequence as output rather than picking tokens from a predefined vocabulary. Thus, this makes Ptr-Net more suitable for extractive summarization task.

$$\boldsymbol{p}(Y|S;\theta) = \frac{\exp(z_i)}{\sum_{k=1}^{n}\exp(z_k)}$$

It is a softmax process, which normalizes the vector z_i (of length n) to be an output distribution over the dictionary of inputs.

4 PERSONALIZED POINTER GENERATIVE ADVERSARIAL NETWORKS

In this section, we provide a detail description of the proposed PPGAN. The overview of PPGAN is shown in Figure 2. PPGAN can be split into two parts: short title generator and two different discriminators, authentic discriminator and CTR discriminator. We use the result of discriminator as a training signal to encourage our model to produce high-quality and personalized short titles.

- Short title generator. (1) Title encoder: In this part, we encode the source title into vector representations. By using the pointer mechanism, we extractive the essential units of source title. (2) User feature encoder: To enhance the personality of the proposed model, we feed the user attributes into the user feature encoder. We use a statistic word set to represent user attributes. (3) Pointer decoder: To generate the short title, we use a one-step decoder which focuses on the user concerned data extracted from the source title.
- Discriminator. Existing methods only consider the content quality of the generated short title but exclude user preference. To enhance the CTR and accuracy of the generated short title, we use two discriminators to determine whether

the user likes the generated short title. (1) Authentic discriminator; (2) CTR discriminator. The former tries to generate short titles close to real data and the latter distinguishes the user liked short title from user-unclicked data.

Notations. P denotes the word distribution of output data. θ denotes the parameters of the generative modules G_θ. Φ denotes the parameters of the discriminator modules D_Φ. T means the original long title sequence, S means the short title sequence and U means the user feature sequence.

4.1 E-commerce word embeddings

Given the product title and user attributes, we use a specialized Named Entity Recognition (NER) tool[1] for E-commerce to label entities in product titles. In total there are 36 types of entities, which are of common interest in an E-commerce scenario, such as "颜色" (color), "风格" (style) and "品类" (category). For example, in segmented product title "包邮 Nike 品牌的红色运动裤" (Nike red sweatpants with free shipping), "包邮 " (Free shipping) is labeled as Marketing Service, "Nike" is labeled as Brand, "红色" (Red) is labeled as Color and "运动裤" (Sweatpants) is labeled as Category.

For each word t_i in the original long title, we look up a pre-trained embedding matrix $E \in \mathbb{R}^{d^w \times |V|}$ to initialize the word embedding e_i^t and the named entity recognition (NER) embedding ner_i^t, where d^w is the dimension of pre-trained word embedding and V is a fixed-sized vocabulary. We define w^t as the long title word embedding sequence, which is the concatenation of the i-th word's word embedding e_i^t and NER embedding ner_i^t in a sentence:

$$w_i^t = [e_i^t, ner_i^t] \tag{1}$$

For user feature sequence U, we can get its corresponding embedding sequence w^u similarly:

$$w_j^u = [e_j^u, ner_j^u] \tag{2}$$

Thus, $w_i^t, w_j^u \in \mathbb{R}^{2*d^w}$. By combining the NER information into the word vector, the model can easily figure out the important source words while striving to preserve them in the outputs.

4.2 Personalized Generator

The generator task is fed with product titles and semantic user features and generates corresponding short titles, named as title compression. In particular, the compressed titles are generated in an extractive manner where the output dictionary is composed of the words from the input sequence. We implement the generator with one-step Pointer Networks (Ptr-Net), which replaces traditional multi-step Ptr-Net with only containing one time step in the decoding phase and extracting the top k largest probability words in the step. Thus one-step decoding can solve the repetitive extraction problem of multi-step decoding.

Firstly, we use a statistical approach to compute user semantic feature based on the CTR log. For each user clicked product $item_i$, we collect the product sequence $(item_{i-1}, item_{i-2}...item_{i-10})$ that user have clicked before $item_i$. So our user features are dynamic. We rank the user sensitive words U in titles by word frequency in an increasing order to represent user feature. According to the word frequency, we select the top 10 to form user feature sequence $U = (u_1, u_2...u_{10})$.

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

Then we employ a GRU network to encode the user feature. We feed the user feature embedding sequence w^u into the user-side GRU encoder and compute the hidden state.

$$h_j^u = g(h_{j-1}^u, w_j^u) \tag{3}$$

where h_j^u is the hidden state for word w_j^u at step j, g is the dynamics function of GRU unit. To combine the user feature and product feature, we concatenate the title word embeddings w_i^t of $item_i$ with the final hidden state $h^u \in \mathbb{R}^{d^h}$ of user encoder. Then we feed the concatenated result $\tilde{x}_i = [w_i^t, h^u]$ into the product-side GRU encoder.

$$h_i^s = g(h_{i-1}^s, \tilde{x}_i) \tag{4}$$

Unlike the Ptr-Nets [27], our proposed method only use a one-step decoder to get the output distribution $P_g = (p_1^g, p_2^g ... p_n^g)$. We pick the top k largest probability words to compose the short title. Since we have already placed the category word to the end of the source title, there is no need to adjust the sequence of output short title. We obtain the output distribution P^g using the attention mechanism:

$$z_i = v^T \tanh(w_1 h_i^s + w_2 d_0) \quad i \in (1, ..., n) \tag{5}$$

where v,w_1,w_2 are learnable parameters of the output model and d_0 means we decode only in the first time step. Then we use softmax to normalize the vector z to generate the output distribution over the input sequence.

$$p_i^g = softmax(z_i) \tag{6}$$

4.3 Discriminator

The discriminator model D is a binary classifier which takes an input of word distribution over source title and distinguishes whether it is sampled from user clicked human-generated (real) data P_r or machine-generated (fake) data $P_g = G(T, U)$. To describe the input of the real human-generated data, we denote $P_r = (p_1^r, p_2^r, ...p_n^r)$ as the input distribution, where n is the length of source long title and p_i^r represents the probability of word i may be chosen as the word of the short title. As for the short title $S = (s_1, s_2, ..., s_m)$, m represents the length of short title.

$$p_i^r = \begin{cases} 1/m & if \quad (s_i \in T) \\ 0 & if \quad (s_i \notin T) \end{cases} \tag{7}$$

where $1/m$ is the normalized one hot result of word chosen probability. Consequently, the dimension of real title distribution and fake title distribution are aligned. In particular, the discriminator can easily "notice" that human-generated distribution is a matrix of numbers of the form $1/m$ and 0. Thus, by adding noises to the real and fake distributions, the forced overlaps can stabilize the training process, where obtaining an optimal discriminator is no longer problematic.

$$\tilde{\mathcal{P}} = \mathcal{P} + \mathcal{N}(0, \sigma) \tag{8}$$

where $\mathcal{N}(0, \sigma)$ means the normal distribution with mean 0 and variance σ . $P \in (P^g, P^r)$. Based on the generated and real short titles' distribution $\tilde{P} \in (\tilde{P}^g, \tilde{P}^r)$ and the long title embedding sequence w^t, we can obtain the "short title" embedding representation $w^s \in (w^{sg}, w^{sr})$ by multiplying them.

$$w^s = w^t \times \tilde{\mathcal{P}} \tag{9}$$

The discriminator is a convolutional neural network that receives the matrix w^{sg} and w^{sr}. The authentic discriminator tries to generate short titles close to real data while the CTR discriminator distinguishes the high-quality short title from user-unclicked data. In order to encode the user feature to evaluate the textual fitness between the user and short title. We concatenate the "short title" embedding $w^s \in (w^{sg}, w^{sr})$ with the user feature embedding w^u.

$$\mathcal{X} = [w^s, w^u] \tag{10}$$

Then we input $\mathcal{X} \in (\mathcal{X}^g, \mathcal{X}^r)$ into the textCNN [14] to make a distinguish.

Let $w \in \mathbb{R}^{d^w}$ be the convolutional filter, d^w represents the dimension of the word embeddings and h represents the filter sliding size. We use n filters ($w = w_1, w_2, ..., w_n$) and the convolutional operation can be expressed as follows:

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{11}$$

where f is a non-linear function(e.g, relu) and c_i used to compose the feature map. Like the textCNN method, we apply a max-pooling operation over the feature map. These features are passed to a fully connected layer to generate the outputs.

4.4 Adversarial Training Loss

The generator G tries to generate a short title distribution close to the real for a target user U, while the discriminator tries to maximize the distance of real user clicked short title and machine-generated short title. For the first, we employ a large set of original long titles(T) and human-generated short title(S) to pre-train the generator, which can quickly update our generator model and help to accelerate the convergence of following confrontation model. Especially, we use $\mathcal{L}_2$ loss function as pre-training loss $\mathcal{L}_{preG_\theta}$.

$$\mathcal{L}_{preG_\theta} = \sum_{i=1}^{n} (\tilde{p}_i - p_i^r)^2 \tag{12}$$

Due to sparsity of the dataset, there is not enough user click action data to cover all the products. So we only encode the source long title feature for the generator at pre-training step and don't encode user features.

In the formal training phase, we fed the user clicked title pair data $[\mathcal{T}^+, \mathcal{U}^+]$ into our model to the generator G. The loss of G on WGAN can be formulated as follows:

$$\mathcal{L}_{G_\theta} = \sum_{i=1}^{n} (\tilde{p}_i - p_i^r)^2 - \beta \mathbb{E}_{\tilde{p} \sim \mathcal{P}_g} [D(\tilde{p})] \tag{13}$$

where β is a hyper-parameter to balance the two-part loss. Then, the real user clicked data $[\mathcal{S}_r^+, \mathcal{U}^+]$ and machine-generated data $[\mathcal{S}_g^+, \mathcal{U}^+]$ are gathered to train the authentic discriminator $authD_\Phi$. The loss of $authD_\Phi$ on WGAN is:

$$\mathcal{L}_{authD_\Phi} = \mathbb{E}_{t,u \sim \mathcal{T}^+, \mathcal{U}^+} [D(G(t, u))] - \mathbb{E}_{\tilde{p} \sim \mathcal{P}_r^+} [D(\tilde{p})] \tag{14}$$

Algorithm 1 Adversarial Training for PPGAN

```
Require: Long titles $\mathcal{T}$, Short titles $\mathcal{S}$, user features $\mathcal{U}$; pre-Generator
    ${}_{pre}G_\theta$, Generator $G_\theta$, Authentic discriminator $authD_\Phi$, CTR discrim-
    inator $ctrD_\Phi$
 1: Step 1: Pretrain ${}_{pre}G_\theta$ using $\mathcal{L}_{preG}$
 2: Step 2: Training $G_\theta$ and $D_{auth}$ iteratively
 3: for number of training epoches do
 4:     for $i = 1, \ldots D$-epoches do
 5:         Input user clicked data pair $[t, u, s] \sim [\mathcal{T}^+, \mathcal{U}^+, \mathcal{S}^+]$
 6:         Generate short title ditribution $\tilde{p}_g^+ \leftarrow G_\theta(t, u)$
 7:         Transform real short title ditribution by (8), (9)
 8:         Update $authD_\Phi$ parameters via RMSProp by (15)
 9:         $\Phi \leftarrow clip(\Phi, -c, c)$
10:     end for
11:     Input user clicked data pair $[t, u, s] \sim [\mathcal{T}^+, \mathcal{U}^+, \mathcal{S}^+]$
12:     Update $G_\theta$ parameters via RMSProp by (14)
13: end for
14: Step 3: Training $G_\theta$ and $ctrD_\Phi$ iteratively
15: Repeat 3-12, input user unclicked data $[\mathcal{T}^-, \mathcal{U}^-]$, change discriminator
    to $ctrD_\Phi$ and update parameters by (18)
```

The $authD_\Phi$ realizes 1-Lipschitz function in WGAN by clipping the weights and constraining them within a range.

$$\Phi \leftarrow clip(\Phi, -c, c) \tag{15}$$

Finally, we fed the unclick pair data $[\mathcal{T}^-, \mathcal{U}^-]$ into the generator. In fact, the user didn't click short title may not only be caused by the improper short title, but also the user didn't like this product itself. So we consider a more reasonable soft objective for the discriminator. We use a clustering objective for the discriminator to encourage a large margin between two classes. We follow an existing work [6] to maximize the information-theoretic margin between positively and negatively disguised samples. We assume there are N unlabeled points. When assigning these N points to 2 classes by a discriminator neural network $D(\cdot)$, the assignment confidence of the discriminator could be well characterized by the following additive conditional entropy $M_D(x)$.

$$\mathcal{M}_D(x) = -\frac{1}{N} \sum_i \{D(x_i) \log [D(x_i)] + [1 - D(x_i)] \log [1 - D(x_i)]\} \tag{16}$$

We combine the two losses mentioned above and form a final loss for CTR discriminator, which is formulated as:

$$L_{ctrD_\Phi} = \mathbb{E}_{t,u \sim \mathcal{T}^-, \mathcal{U}^-} [D(G(t, u))] - \mathbb{E}_{\tilde{p} \sim \mathcal{P}_r^-} [D(\tilde{p})] + \mathcal{M}_D(\tilde{p}_g^-) \tag{17}$$

5 TRAINING PROCEDURE

Step 1: we pre-train the generator G_θ given whole human-generated data $[\mathcal{T}, \mathcal{S}]$ without user feature, and then transfer the pre-trained model weight to the generator module.

Step 2: we train the generator to produce machine-generated data $[\mathcal{S}_g^+, \mathcal{U}^+]$, which is fed into the authentic discriminator togther with the real user-clicked data pair $[\mathcal{S}_r^+, \mathcal{U}^+]$.

Step 3: we set another CTR adversarial training by adding the information-theoretic margin to the discriminator loss using the user-unclicked data pair $[\mathcal{S}_r^-, \mathcal{U}^-]$.

We perform an adversarial training process based on the WGAN architecture, as shown in Algorithm 1.

6 EXPERIMENTS

6.1 Dataset

The dataset[2] used in our experiment is collected from a "有好货|Youhaohuo" channel of the well-known online shopping platform in China, **Taobao**[3]. What is different from ordinary Taobao products is that online merchants are required to submit a short title for each Youhaohuo product. This short title, written by humans, is readable and describes the key properties of the product. Furthermore, we filter the dataset so that the words in the short titles must always from the original titles. We can represent our dataset as $<O, S, U, L>$, where O means products' original titles, S means the human-written short titles, U represents the user characteristics of the product clicked by users, and L represents a binary label, i.e., L is labeled with 1 (user clicked) or 0 (user unclicked). Eventually, we get a dataset with $7.59M$ samples including $0.64M$ clicked and $6.95M$ unclicked, which shows our dataset is sparse. We randomly split the dataset into approximately 90% for training, 5% for validation, and 5% for testing. Table 1 provides detailed statistics about this dataset, and Figure 3 presents a Quad example.

Table 1: The statistics of the pair dataset.

Sum dataset size	7,587,532
Clicked dataset size	640,429
Unclicked dataset size	6,947,103
Number of products	248,502
Number of users	102,337
Avg. length of original titles	11.29
Length of short titles	5
Length of user features	10
Vocabulary size	70,000
Product category size	52

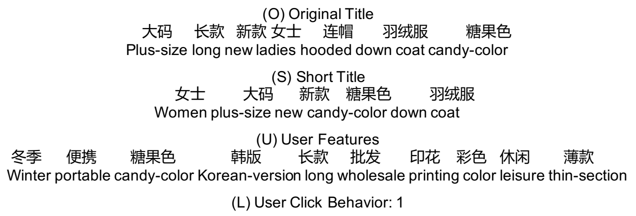
(O) Original Title
大码 长款 新款 女士 连帽 羽绒服 糖果色
Plus-size long new ladies hooded down coat candy-color
(S) Short Title
女士 大码 新款 糖果色 羽绒服
Women plus-size new candy-color down coat
(U) User Features
冬季 便携 糖果色 韩版 长款 批发 印花 彩色 休闲 薄款
Winter portable candy-color Korean-version long wholesale printing color leisure thin-section
(L) User Click Behavior: 1

Figure 3: An example for the dataset.

6.2 Baselines

To verify the effectiveness of our proposed model, we implement two classes of state-of-art product title generation methods and compare our method with them. The abstractive methods including:

- Generative Adversarial Network (**GAN**) [18] which is a generative adversarial method for abstractive text summarization with only raw text as input.
- Pointer-Generator (**Ptr-Gen**) [24] which is a hybrid model combing seq2seq with pointer network. Besides copying words from the input, Ptr-Gen can also generate words from the predefined vocabulary.

The extractive methods including:

- Truncation-based Method (**Trunc.**). The usual solution is simply truncating the original long titles to adapt to the limitation in E-commerce.
- Pointer Networks (**Ptr-Nets**) [27] which is an attentive seq2seq extraction model by copying words from redundant long product titles.
- Agreement-based MTL (**Agree-MTL**) [29] which is a multi-task approach to improve product title compression with user searching log data.
- Feature-Enriched-Net (**FE-Net**) [9] which is a wide model combining three different categories of features (product title, TF-IDF and NER) to generate the short product titles.

6.3 Implementation Details.

We implement all the models in Tensorflow[4]. We use the Word2vec [22] CBOW model to pre-train word embeddings on our whole product titles data. The size of the pre-trained word embeddings is set to 50. For Out-Of-Vocabulary (OOV) words, embeddings are initialized as zero. For all encoder-decoder models in Generator, we use a three-layers GRU network with a 128-dimensional unit size. All product titles are padded to a maximum sentence length of 20. For Discriminator, all short titles and user features are encoded by convolution with filter windows of [2, 3, 4] followed by activate function leaky-relu, max-pooling and a full-connected layer.

All the models are trained on a single Tesla V100 GPU. We first pre-train the short title generative model given human-generated data via maximum likelihood estimation (MLE), and use Adam optimizer training for 10 epochs. After that, we perform adversarial training process on PPGAN (step 2, 3) with RMSProp optimizer for 500 epochs respectively. The number of D-steps is 10. The learning rates are initialized with 0.001 and the batch size is set to 512.

For the dataset pre-processing, 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, we create a vocabulary of 70K words according to the word frequency. Words appearing less than five times in the training set are replaced as <UNK>. Other parameters are empirically set as follows: $\alpha = 0.01$, $\sigma = 0.01$, $c = 0.01$ and $\beta = 0.5$.

6.4 Automatic Evaluation

To automatically evaluate the title generation performance of different models, we leverage four types of standard metrics:

BLEU [23] metrics is widely used for text generation tasks, such as machine translation, summarization. The metric analyzes co-occurrences of n-grams between the candidate and the references. For BLEU metric, we report BLEU-1, BLEU-2 here. **ROUGE** [17]

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

[4]https://www.tensorflow.org

Table 2: BLEU, ROUGE (F1), METEOR and Correlation Distance scores of various methods on the test set. Bold scores are the best overall.

Methods	BLEU-1	BLEU-2	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	Corr-Distance
GAN	69.37	58.92	68.90	51.21	68.39	36.24	21.49
Ptr-Gen	71.58	63.70	70.31	54.90	70.81	37.59	21.37
Trunc.	52.57	40.26	52.58	30.38	52.56	24.65	22.69
Ptr-Nets	69.61	61.14	68.95	55.10	67.74	36.76	21.31
Agree-MTL	72.04	63.92	71.39	57.92	70.35	37.54	19.56
FE-Net	73.26	64.80	72.53	57.84	71.26	38.82	21.43
PPGAN (step 2)	76.73	59.33	76.75	45.87	76.73	39.39	20.42
PPGAN (step 1+2)	**84.60**	**73.09**	**79.62**	**61.15**	**79.60**	**41.87**	18.64
PPGAN (full, step 1+2+3)	83.59	71.38	78.60	60.94	77.59	40.94	**16.28**
PPGAN (full, no ner)	78.41	67.02	77.90	55.28	70.79	39.86	17.05

metric measures the summary quality by counting the overlapping units (e.g., n-grams) between the generated summary and reference summaries. Following the common practice, we report the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. **METEOR** [16] 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 match mode (rewarding only exact matches between words). In addition, **Corr-Distance** evaluates the title relativity between the model generated short titles and the corresponding user features from semantics. We introduce the Correlation Distance metric and adopt the SimHash [19] and Hamming distance to measure it. The smaller the value, the greater the correlation between generated short titles and user features. We obtain the BLEU and METEOR scores using the nlg-eval[5] packages, and ROUGE scores using python-rouge[6] package.

The final results on the test set are shown in Table 2. We can find our method PPGAN outperforms both abstractive and extractive baselines on all metrics. Among several baselines, Trunc. perform the worst. For Trunc., it cannot extract the informative words from the tail of product titles, such as commodity names. When compared with GAN, PPGAN(full) achieves an improvement of 13.34%, 9.54%, 4.70% and 5.21 in BLEU(Avg.), ROUGE(Avg.), METEOR and Corr-Distance respectively. It verifies that adopting one-step Ptr-Net decoding and adding user features can help our model generate better short titles.

To explore the performance of the pre-training Generator and CTR Discriminator respectively, we conducted three comparative tests. The results show that model PPGAN(step 1+2) including pre-training performs better than PPGAN(step 2), with BLEU(Avg.) increased by 10.82%, ROUGE(Avg.) increased by 7.01%, METEOR increased by 2.48% and Correlation Distance decreased by 1.78. Besides, we note that PPGAN(full) performs best on Corr-Distance of 16.28%, which means it experts in generating personalized short titles. However, PPGAN(full) also has a slight decrease in BLEU, ROUGE and METEOR. It can be easily understood that the more personality short titles have, the fewer overlaps and co-occurrences they have compared with human-written short titles. These show the importance of training CTR Discriminator. Also, the result of PPGAN(full, no ner) shows that the ner information can significantly help improve the quality of generated short titles.

[5]https://github.com/Maluuba/nlg-eval
[6]https://github.com/tagucci/pythonrouge

Table 3: Key info retention test and time complexity analysis.

Methods	B-Error-Rate	C-Error-Rate	Time Complexity
GAN	4.73%	14.60%	$O(nm + nk)$
Ptr-Gen	2.92%	10.32%	$O(nm)$
Trunc	2.51%	97.29%	$O(1)$
Ptr-Nets	2.47%	7.29%	$O(nm)$
Agree-MTL	**1.14%**	4.53%	$O(nm + nl)$
FE-Net	1.28%	**3.73%**	$O(n^2)$
PPGAN (step 2)	1.51%	3.04%	$O((n + u)(k + 1))$
PPGAN (step 1+2)	**0.24%**	**1.78%**	$O(n + (n + u)(k + 1))$
PPGAN (full)	0.28%	1.86%	$O(n + 2(n + u)(k + 1))$
PPGAN (full, no ner)	1.63%	3.44%	$O(n + 2(n + u)(k + 1))$

6.5 Key Information Retention Test

For short title generation, it is particularly important to retain critical information (i.e., band names, commodity names), which is more essential compared to other types of product descriptive words. The evaluation metric for this task is the error rate of the brand names (B-Error-Rate) and commodity names (C-Error-Rate) in the generated product short titles, which respectively represent the proportion of samples whose brand or commodity names are extracted in error or not extracted.

The results of each model on test sets are shown in Table 3. Trunc. performs well on the retention of brand names because the brand name usually appears at the head of the title. On the contrary, due to the commodity name often positioned at the tail of the title, Trunc. has bad performance on commodity names’ retention. Our PPGAN model significantly outperforms the other baselines. Compared with the best baseline result, PPGAN improves B-Error-Rate and C-Error-Rate by 0.90% and 1.95% respectively. It verifies that introducing the one-step Pointer Networks and GAN technique can help the model better retain key information in original product

Table 4: Manual evaluation results. Bold scores are the best. The improvements from baselines to PPGAN (full) is statistically signicant according to two-tailed t-test ($p < 0.05$).

Model	Comp.	Read.	Info.	Pers.
GAN	65.33%	4.16	3.93	3.29
Ptr-Gen	82.03%	4.51	4.25	3.42
Trunc.	32.67%	2.17	2.43	2.13
Ptr-Nets	83.43%	4.63	4.31	3.21
Agree-MTL	84.87%	4.67	4.33	3.47
FE-Net	85.14%	4.72	4.39	3.39
PPGAN (step 2)	84.93%	4.68	4.35	3.27
PPGAN (step1+2)	**87.67%**	**4.87**	4.46	3.53
PPGAN (full)	86.81%	4.80	**4.62**	**4.35**
Human	100%	4.93	4.51	3.94

titles. Also, we notice that PPGAN model with step 1(pre-training step) can effectively reduce the error rate since a large amount of corpus is involved to train the generator. As shown in the last two lines of Table 3, we did a comparative experiment that only used word embeddings as encoder's input without NER information. The results show that encoder with multi-data input including word embeddings and NER embeddings helps the model perform better. 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 4.1.

6.6 Time Complexity Analysis

The comparison of model time complexity is shown in Table 3, where n, u and m are the length of the long title, the user feature and the short title respectively, and k is the number of convolution kernels. Compared to most title compression baselines, although our method adds a discriminative training process, the time for the title generation process is greatly reduced due to the use of one-step decoding.

6.7 Manual Evaluation

Following the previous work [8, 26], we also conduct a manual evaluation on the generated product short titles. As such, evaluating the quality of generated short titles by human judges is a better measure of our methods' quality. We randomly selected 500 products and obtained the participants' real-time user features, then we leveraged our PPGAN (full) model to generate short titles as our test set. Three participants were asked to measure the quality of the short titles generated by each model from four perspectives: (1) **Completeness**, is the key information properly kept in the short title (including the brand and commodity name repeated or not)? (2) **Readable**, how fluent, grammatical the short title is? (3) **Informativeness**, how informative the short title is? (4) **Personalized**, does the generated short title match the user's preferences?

Comparing with conventional sentence summarization task, key information retention is an extra and essential requirement for product title summarization. We use a rigorous criterion for this

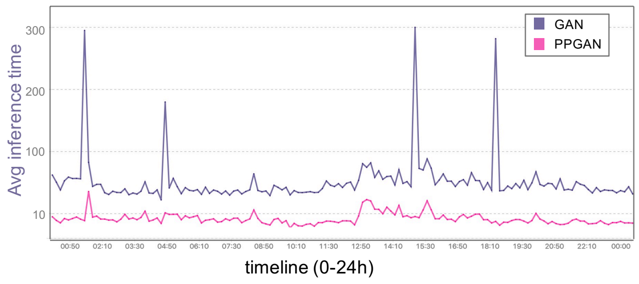

Figure 4: Averaged online inference time for GAN and PPGAN.

property. The generated short title will be assessed as 1 only if it correctly retains both the brand name and commodity name, otherwise 0. The Readability, Informativeness and Personalization are assessed with a score from 1 (worst) to 5 (best). Furthermore, it should be noted that to make the evaluation results more consistent and reliable, we excluded instances with divergent ratings (i.e., their variance is greater than the threshold).

The final average results are presented in Table 4. The results indicate that our PPGAN outperforms the other conventional sentence summarization methods. Obviously, Trunc. perform worst in this task as the same in automatic evaluations. Also, the abstractive method Ptr-Gen and the extractive method Ptr-Nets have a similar well performance. The significant improvements in Personalized metric demonstrate that our PPGAN can better consider the users' interest by the encoding of user features. Compared to human-written ground truths, the results show that our method with step 3 performs very close with human, even better on Personalized metric. That is largely because the CTR discriminator can distinguishes the user liked short title according to the user behavior information data.

6.8 Online A/B Testing

Previous experimental results have proven the advantages of our proposed approach, so we deploy it in a real-world online environment to test its practical performance. We perform A/B testing in the "发现好货|FindGoodProducts" scenario[7] (over 2 million daily active users). Online users are equally split into A/B/C buckets, respectively with GAN, FE-Net and our proposed PPGAN(full, step1+2+3). The online A/B testing lasted for one week. We adopt product Click Through Rate (CTR) to measure the performance of our generated short titles, which can be calculated as

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

where $product\_click$ is the clicking times of the product, $product\_PV$ is the number of times the product is shown.

Online results show that compared with A and B bucket, our PPGAN method improves CTR by 5.18% and 3.53% in absolute value. The result of the CTR improvement indicates that users are more likely to click the products in which short titles are generated by PPGAN. It verifies the effectiveness of our method PPGAN. Also, we

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

Table 5: Examples of the generated product short titles by different methods.

Original Title	女士 大码 长款 新款 批发 连帽 糖果色 羽绒服 Ladies plus-size long new wholesale hooded candy-color downcoat
GAN	女士 加厚 长款 新款 糖果 Ladies thickened long new candy
Trunc	女士 大码 长款 新款 批发 Ladies plus-size long new wholesale
Ptr-Net	女士 长款 新款 连帽 羽绒服 Ladies long new hooded downcoat
User Feature	加大 纯棉 宽松 连帽 保暖 新款双人 帽子 韩版 Plus cotton loose hooded warm new double hat Korean-version
PPGAN	女士 大码 新款 连帽 羽绒服 Ladies Plus-size new hooded downcoat

compare the average online inference time for GAN and PPGAN as shown in Figure 4. PPGAN can significantly reduce the time complexity of online real-time inference and provides smooth flow characteristics.

6.9 Case Study

Table 5 shows a sample of product short titles generated by our PPGAN and baselines. From this table, we can note that except for our model, all other baseline methods are not personalized, that means short titles generated by these methods are the same for all users. In detail, there are some relevant words of "加大-plus", "连帽-hooded" and "帽子-hat" in the user features, and the personalized short title generated from PPGAN includes descriptive words "大码-plus-size" and "连帽-hooded". Becides, GAN and Trunc. perform worst on the case. For GAN, it is an abstractive method, which can generate words beyond the original title, such as "加厚-thickened" and "糖果-candy". For Trunc., it is very likely to generate unreadable short titles.

7 CONCLUSION

In this paper, we propose a user-sensitive adversarial training method to generate a user interest product short title in E-commerce. Different from conventional methods, we retain the most relevant product information to the user according to user behavior. In detail, we leverage the user clicked data to train a discriminator in a human-like view. Moreover, we use an unsupervised information-theoretic assignment strategy to encourage the discriminator to identify the high-quality short titles from the user-unclicked data, which solves the problem that the dataset is too sparse. Besides, we perform extensive experiments with realistic data from a popular Chinese E-commerce website. Experimental results demonstrate that our PPGAN model can generate personalized and fluent short titles and outperform other baselines. Finally, online A/B testing shows the significant business impact of our model in a mobile E-commerce app. Future works involve incorporating the knowledge graph into our model to enrich the content of short titles.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Leon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *ICML*. 214–223.

[2] Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. 2016. AttSum: Joint Learning of Focusing and Summarization with Neural Attention. In *COLING*. 547–556.

[3] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Fethi Bougares, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Computer Science* (2014).

[4] John M Conroy and Dianne P Oleary. 2001. Text summarization via hidden Markov models. In *SIGIR*. 406–407.

[5] Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In *ICCV*. 2989–2998.

[6] Yue Deng, Yilin Shen, and Hongxia Jin. 2017. Disguise Adversarial Networks for Click-through Rate Prediction. In *IJCAI*. 1589–1595.

[7] Gunes Erkan and Dragomir R Radev. 2004. LexPageRank: Prestige in Multi-Document Text Summarization.. In *EMNLP*. 365–371.

[8] Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence Compression by Deletion with LSTMs. In *Conference on Empirical Methods in Natural Language Processing*.

[9] Yu Gong, Xusheng Luo, Kenny Q Zhu, Wenwu Ou, Zhao Li, and Lu Duan. 2019. Automatic Generation of Chinese Short Product Titles for Mobile Display.. In *AAAI*. 9460–9465.

[10] Ian Goodfellow, Jean Pougetabadie, Mehdi Mirza, Bing Xu, David Wardefarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[11] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O K Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*. 1631–1640.

[12] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[13] Phillip Isola, Junyan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE*. 5967–5976.

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. 1746–1751.

[15] Julian M Kupiec, Jan O Pedersen, and Francine R Chen. 1995. A trainable document summarizer. In *SIGIR*. 68–73.

[16] Alon Lavie and Abhaya Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *workshop on statistical machine translation*. 228–231.

[17] Chinyew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL*. 74–81.

[18] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2017. Generative Adversarial Network for Abstractive Text Summarization.. In *AAAI*. 8109–8110.

[19] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. *the web conference* (2007), 141–150.

[20] Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *EMNLP*. 319–328.

[21] Rada Mihalcea. 2005. Language Independent Extractive Summarization. In *ACL*. 49–52.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[23] Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*. 311–318.

[24] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*. 1073–1083.

[25] Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. 2018. Multi-Source Pointer Network for Product Title Summarization. In *CIKM*. 7–16.

[26] Jiwei Tan, Xiaojun Wan, Jianguo Xiao, Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive Document Summarization with a Graph-Based Attentional Neural Model. In *Meeting of the Association for Computational Linguistics*.

[27] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*. 2692–2700.

[28] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. *automated software engineering* (2018), 397–407.

[29] 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. In *AAAI*. 451–458.

[30] Kristian Woodsend and Mirella Lapata. 2010. Automatic Generation of Story Highlights. In *ACL*. 565–574.

[31] Jianguo Zhang, Pengcheng Zou, Zhao Li, Yao Wan, Xiuming Pan, Yu Gong, and Philip S Yu. 2019. Multi-Modal Generative Adversarial Network for Short Product Title Generation in Mobile E-Commerce. In *ACL*. 64–72.

[32] Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019. Automatic Generation of Pattern-controlled Product Description in E-commerce. In *WWW*. 2355–2365.