Multi-Source Pointer Network for Product Title Summarization

Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang
Alibaba Group
Beijing, China
{ofey.sf,jiangpeng.jp,hansel.shx,changhua.pch,santong.oww,yongshu.wxb}@alibaba-inc.com

ABSTRACT

In this paper, we study the product title summarization problem in E-commerce applications for display on mobile devices. Comparing with conventional sentence summarization, product title summarization has some extra and essential constraints. For example, factual detail errors or loss of the key information are intolerable for E-commerce applications. Therefore, we abstract two more constraints for product title summarization: (i) do not introduce irrelevant information; (ii) retain the key information (*e.g.*, brand name and commodity name). To address these issues, we propose a novel multi-source pointer network by adding a new *knowledge encoder* for pointer network. The first constraint is handled by *pointer mechanism*, generating the short title by copying words from the source title. For the second constraint, we restore the key information by copying words from the knowledge encoder with the help of the soft gating mechanism. For evaluation, we build a large collection of real-world product titles along with human-written short titles. Experimental results demonstrate that our model significantly outperforms the other baselines.Finally, online deployment of our proposed model has yielded a significant business impact, as measured by the click-through rate.

CCS CONCEPTS

• **Information systems** → **Summarization**; *Information retrieval*;
• **Artificial intelligence** → **Natural language processing**;

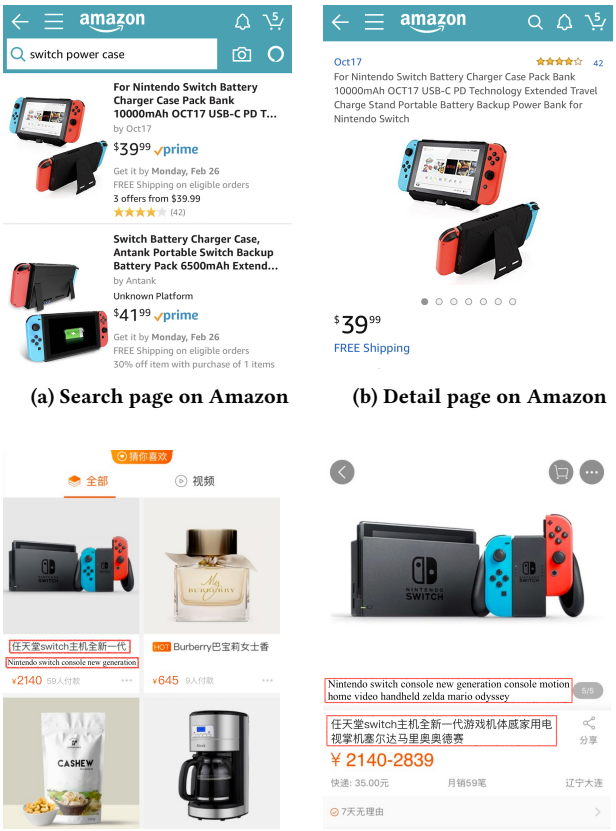
KEYWORDS

Title Summarization; Pointer Network; Extractive Summarization

ACM Reference Format:
Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. 2018. Multi-Source Pointer Network for Product Title Summarization. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271722

1 INTRODUCTION

Nowadays, more and more online transactions are made on mobile phones instead of on PCs. However, some user interfaces in E-commerce applications are not optimized for mobile phones. For screenshots in Figure 1a and 1c, the product titles cannot be fully displayed on two popular E-commerce apps. In such case, user must go further into the detail page to see the full title of the product. This really hurts users' browsing experience. These redundant and lengthy product titles on the E-commerce platforms (especially, customer to customer(C2C) websites) are often produced by the sellers for the sake of Search Engine Optimization(SEO). Although it is okay to display these lengthy titles on a PC's web browser, they are not suitable for displaying on the small screen of a mobile phone. Furthermore, products with a short and informative title may often better attract users' attention and receive more clicks[1] [42]. Thus, generating a short and informative title for each product is an important and practical research problem in E-commerce.



(a) Search page on Amazon (b) Detail page on Amazon

(c) Recommendation page on Taobao (d) Detail page on Taobao

Figure 1: The product titles cannot display completely in corresponding pages on Amazon and Taobao iOS apps.

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271722

[1] https://sellercentral.amazon.com/forums/message.jspa?messageID=2921001

We formalize this problem as product title summarization, a specific form of sentence summarization in the field of E-commerce. However, comparing with the conventional sentence summarization [7, 37], it has some extra and essential constraints. For example, it is not a big problem to encounter incorrect factual details in the news summary system. However, factual errors are intolerable for buyers and sellers on the E-commerce platforms. Based on these considerations, we abstract two more stringent constraints for product title summarization: (i) **Do not introduce irrelevant information**; (ii) **Retain the key information** (*e.g.*, brand name and commodity name). These constraints are essential and explicit for product title summarization, since both the sellers and the consumers will be dissatisfied if we break any of them.

Firstly, sellers usually do not want the generated short titles mingled with the words not in their original titles. This is because the words in original titles are usually carefully selected by the sellers and helpful to click-through rate [4]. Furthermore, it is totally unacceptable if the short title contains any incorrect information, *e.g.*, generating a wrong brand "*sony*" in the short title for the product "*Nintendo Switch*". Although neural abstractive summarization methods have achieved great success in news or wiki-like articles, they still frequently generate incorrect factual details in the summaries [38]. This is why we do not employ neural abstractive methods in this work.

Secondly, the generated short title should retain the key information in the original title. For example, it would be very confusing for customers if we lost the brand "*Incase*" or commodity name "*Sleeve*" in the short title "*Incase ICON Sleeve for MacBook*". Moreover, this is also unacceptable for the sellers on E-commerce platforms.

In this paper, we propose a novel model named *Multi-Source Pointer Network* (MS-Pointer) to explicitly model these two constraints. Specifically, for the first constraint, we model product title summarization as an extractive summarization problem using the Pointer Network [41], generating the short title by copying the words from the source title based on the attention mechanism [1]. That is to say, all the words in the compressed title are selected from the original title.

However, the pointer network cannot guarantee the generated short title containing the key information from the source title. To tackle this issue, we extend the pointer network in a data driven way. Specifically, we introduce a new *knowledge encoder* for pointer network to encode the key information about the product. At decoding time, MS-Pointer learns to copy different information from the corresponding encoders with the help of the soft gating mechanism. In short, MS-Pointer can learn to decode the key information (*e.g.*, brand name and commodity name) from the knowledge encoder in a data driven way.

For evaluation, we construct a large dataset containing 411,267 product titles with corresponding human-written short titles from Taobao.com. We compare our model with several abstractive and extractive baselines using both automatic and manual evaluations. The results demonstrate that our model significantly outperforms several strong baselines. In particular, on the brand retention test, MS-Pointer can correctly preserve more than 99% brand names. Finally, deployment of our MS-Pointer model on Taobao mobile app has yielded a significant business impact, as measured by the click-through rate.

2 RELATED WORK

In this section, we will briefly review the two lines of related works, *i.e.*, sentence summarization and pointer mechanism.

2.1 Sentence Summarization

Generally, sentence summarization methods can be classified into two categories: abstractive methods and extractive methods.

Abstractive models generate the condensed sentence in a bottom-up way, *i.e.*, creating a summary from scratch based on understanding the source text. They build an internal semantic representation for the source text and then use natural language generation techniques to create a summary. The task of abstractive sentence summarization was formalized around the DUC-2003 and DUC-2004 competitions [33]. Earlier studies mainly focused on syntactic transduction [8, 32] and phrase-based statistical machine translation approach [3, 9, 46]. Inspired by the success of neural machine translation [1, 39], Rush et al. [37] use convolutional models to encode the source, and a attentive feed-forward neural network to generate the summary. Recently, Chopra et al. [7] extended [37] with a attentive recurrent decoder. Further, Nallapati et al. [31] proposed an RNN encoder-decoder architecture for summarization.

Comparing with abstractive methods, extractive models are more related to our work. They assemble a summary by selecting a subset of important words from the original text. Traditional approaches to this task focused on word deletion using rule-based [10, 49] or statistical methods [14, 15, 23, 26, 45]. More recently, deep learning models have also been applied to extractive sentence summarization. Cheng and Lapata [5] used RNN based encoder-decoder to learn a word extractor for extractive document summarization. Filippova et al. [13] built a competitive sentence compression system via making the word deletion decision based on seqence-to-sequence framework. However, it is very difficult for the models based on deletion to deal with word reordering in the summarizations [21].

2.2 Pointer Mechanism

Pointer mechanism was first introduced by Vinyals et al. [41] to solve the problem of generating a sequence whose target dictionary varies depending on the input sequence. It uses attention mechanism as a pointer to select elements from the input sequence as output. This allows it to generate previously unseen tokens.

Since being proposed, pointer mechanism has drawn more and more attention in text summarization [28, 31, 36, 38], machine translation [18], and dialogue generation [12, 17], as it provides a potential solution for rare and out of vocabulary (OOV) words. In addition, it has also been shown to be helpful for geometric problems [41], question answering [22, 43, 44], code generation [25], and language modeling [27]. It is also referred as *copying mechanism* in text generation [12, 17, 19]. The key ideas of these works are very similar, extending pointer network in a soft [17, 38] or hard [18, 31] way to decide whether to generate a token from the predefined dictionary or from the input sequence.

Other related work

Another related work is constrained sentence generation in dialogue systems [30, 47]. Mou et al. [30] and Yao et al. [48] only leveraged a single cue word in responses generation. Xing et al.

[47] used topic modeling to guide responses generation in conversation system. However, none of these works meet the constraints of the task we studied in this paper. Perhaps, the closest work to ours is [42]. Wang et al. [42] proposed a multi-task approach for product title compression using user search log data. However, their work does not consider the second constraint we discussed in the introduction.

3 BACKGROUND

In this section, we first review the sequence-to-sequence models which have been widely adopted for sentence summarization, and then introduce the pointer network for extractive summarization.

3.1 Sequence-to-Sequence(seq2seq) Model

Recently, there has been a surge of work proposing to build the text summarization system within a seq2seq framework. These models are usually composed of two RNNs, an encoder and a decoder. The encoder maps the original text to a vector and the decoder transforms the vector to a summary.

Formally, denote the input text $\mathcal{S} = (w_1, w_2, \ldots, w_N)$ as a sequence of N words, and the output sequence $\mathcal{Y} = (y_1, y_2, \ldots, y_M)$ as a M words sequence. In the seq2seq framework, the source sequence $\mathcal{S}$ is converted into a fixed length vector $\boldsymbol{c}$ by the RNN encoder,

$$\begin{aligned} \boldsymbol{h}_t &= f(\boldsymbol{h}_{t-1}, \boldsymbol{w}_t) \\ \boldsymbol{c} &= g\big(\{\boldsymbol{h}_1, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}\big) \end{aligned}$$

where $\boldsymbol{w}_t$ is the word embedding of w_t, $\boldsymbol{h}_t$ is the RNN hidden state for word w_t at step t, f is the dynamics function of RNN unit, $\boldsymbol{c}$ is the so-called context vector, and g is a function to summarize the hidden states $\boldsymbol{h}_t$ (*e.g.*, a typical instance of g is choosing the last state). In practice, gated RNN alternatives such as LSTM [20] or GRU [6] often perform much better than vanilla ones. Thus, in this work, we implement f using LSTM [20] which is parameterized as:

$$\begin{aligned} \boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \\ \boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \\ \boldsymbol{z}_t &= \boldsymbol{f}_t \odot \boldsymbol{z}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z) \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{z}_t) \end{aligned}$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, $\boldsymbol{i}$, $\boldsymbol{f}$, and $\boldsymbol{o}$ are respectively the *input gate*, *forget gate*, *output gate*, $\boldsymbol{z}_t$ is the information stored in memory cells, all of which are the same size as the hidden vector $\boldsymbol{h}_t$, and all the non-linear operations are computed element-wise. The subscripts for weight matrix and bias terms have the obvious meaning. For example $\boldsymbol{W}_f$ is the forget gate matrix, $\boldsymbol{b}_f$ is the bias term for the forget gate etc.

The decoder is to generate the target sequence based on the context vector $\boldsymbol{c}$ through the following dynamics process:

$$\begin{aligned} \boldsymbol{d}_t &= f(\boldsymbol{y}_{t-1}, \boldsymbol{d}_{t-1}, \boldsymbol{c}) \\ p(y_t = w|\mathcal{S}, y_{<t}) &= \phi(\boldsymbol{d}_t) \end{aligned}$$

where $\boldsymbol{d}_t$ is the hidden state of the decoder at time step t, y_t is the predicted target symbol at t through function ϕ, $\boldsymbol{y}_{t-1}$ is the word embedding of y_{t-1}, $y_{<t}$ denotes the history $(y_1, y_2, \ldots, y_{t-1})$. In primitive decoder models, context vector $\boldsymbol{c}$ is the same for generating all the output words. In practice, the attention mechanism [1] is usually adopted to dynamically change the context vector in order to pay attention to different parts of the input sequence at each step of the output generation.

3.2 Pointer Network for Summarization

Unlike vanilla seq2seq models, pointer network [41] uses the attention mechanism [1] as a pointer to select tokens from the input as output rather than picking tokens from a predefined vocabulary. This distinct characteristic makes pointer network very suitable for extractive summarization.

Formally, given an input sequence $\mathcal{S}$=$(w_1, w_2, \ldots, w_N)$ of N words, pointer network uses an LSTM as encoder to produce a sequences of *encoder hidden states* $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N)$. At each step t, the decoder (a single-layer LSTM) produces the *decoder hidden state* $\boldsymbol{d}_t$ using the word embedding of the previous word y_{t-1} and the last step decode state $\boldsymbol{d}_{t-1}$. Then, the *attention distribution* $\boldsymbol{a}^{(t)}$ is calculated as in [1]:

$$\begin{aligned} u_{ti} &= \boldsymbol{v}^\top \tanh\left(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_d \boldsymbol{d}_t + \boldsymbol{b}_{\text{attn}}\right) \\ \boldsymbol{a}_t &= \text{softmax}(\boldsymbol{u}_t) \end{aligned}$$

where softmax normalizes the vector $\boldsymbol{u}_t$ to be an distribution over the input position; $\boldsymbol{v}, \boldsymbol{W}_h, \boldsymbol{W}_d$, and bias term $\boldsymbol{b}_{\text{attn}}$ are learnable parameters.

Considering a word w may appear multiple times in the input sequence $\mathcal{S}$, we define the *output distribution* of word w by summing probability mass from all corresponding parts of the attention distribution, as in [38]:

$$p(y_t = w|\mathcal{S}, y_{<t}) = \sum_{i:w_i=w} a_{ti}$$

Finally, the training loss for step t is defined as the negative log likelihood of the target word w_t^* at that step:

$$\mathcal{L}_t = -\log p(y_t = w_t^*|\mathcal{S}, y_{<t})$$

4 MULTI-SOURCE POINTER NETWORK

Although pointer network works very well in practice, it still loses the brand name or commodity name of product from time to time. We aim to endue the pointer network with the capacity retaining such key information in the generated short title.

To achieve this, in addition to the encoder for the source title, we introduce a new *knowledge encoder*. It encodes the brand name and commodity name using an LSTM, just like what we have done with the source title. At test time, the decoder can generate a short title by copying words from not only the title encoder but also the knowledge encoder. In this way, the model can learn to decode the key information from the knowledge encoder in a data driven way. The architecture of MS-Pointer model is shown in Figure 2.

As Figure 2 shows, MS-Pointer combines the original title ("*Nintendo switch console* $\cdots$") and background knowledge (brand name "*Nintendo*" and commodity name "*console*") to produce the short

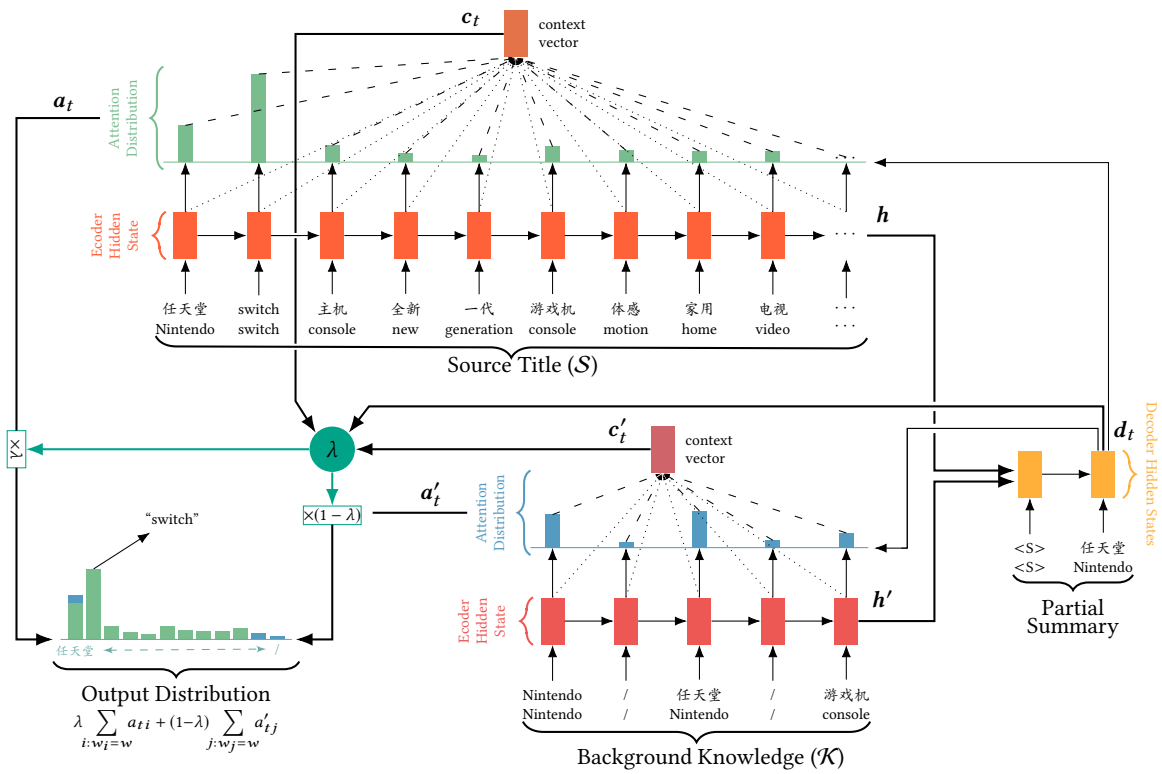


Figure 2: Multi-source pointer network (MS-Pointer) with two encoders[2]. The most distinctive characteristic of MS-Pointer is that it can *copy* words from multiple encoders. At each decoding time step, a soft gating weight $\lambda \in [0, 1]$ is calculated, which weights the probability of copying words from the source title, versus copying words from the background knowledge. The final output distribution (from which we make prediction) is weighted sum of attention distribution $\boldsymbol{a}_t$ and $\boldsymbol{a}'_t$.

title about the product "*Nintendo switch*". Here, we simply concatenate the brand name and commodity name of the product as its background knowledge, using a separator "/"[3].

Formally, for a product with source title $\mathcal{S} = (w_1, w_2, \ldots, w_N)$ and background knowledge $\mathcal{K} = (k_1, k_2, \ldots, k_M)$, we use LSTM to produce series of hidden states $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N)$ and $(\boldsymbol{h}'_1, \boldsymbol{h}'_2, \ldots, \boldsymbol{h}'_M)$, respectively. Next, we transform the final hidden states $\boldsymbol{h}_N$ and $\boldsymbol{h}'_M$ into the initial state $\boldsymbol{d}_0$ of the decoder using rectified layer [16]:

$$\boldsymbol{d}_0 = \mathrm{ReLU}\big(\boldsymbol{W}_f \cdot [\boldsymbol{h}_N, \boldsymbol{h}'_M]\big)$$

where $\mathrm{ReLU} = \max(0, x)$, and $\boldsymbol{W}_f$ is learnable parameters.

For title encoder and knowledge encoder, we compute the attention distribution as follows:

$$\begin{aligned} u_{ti} &= \boldsymbol{v}^\top \tanh\big(\boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{W}_d \boldsymbol{d}_t + \boldsymbol{b}_{\text{attn}}\big) \\ u'_{tj} &= \boldsymbol{v}'^\top \tanh\big(\boldsymbol{W}'_h \boldsymbol{h}'_j + \boldsymbol{W}'_d \boldsymbol{d}_t + \boldsymbol{b}'_{\text{attn}}\big) \\ \boldsymbol{a}_t &= \mathrm{softmax}(\boldsymbol{u}_t), \quad \boldsymbol{a}'_t = \mathrm{softmax}(\boldsymbol{u}'_t) \end{aligned}$$

where $\boldsymbol{a}_t$ is attention distribution for title encoder, $\boldsymbol{a}'_t$ is attention distribution for knowledge encoder, $\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{W}_h, \boldsymbol{W}'_h, \boldsymbol{W}_d, \boldsymbol{W}'_d, \boldsymbol{b}_{\text{attn}}$, and $\boldsymbol{b}'_{\text{attn}}$ are parameters to be learned. $\boldsymbol{d}_t$ is decoder hidden state at time step t, computed by:

$$\boldsymbol{d}_t = f(\boldsymbol{d}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{c}'_{t-1})$$

where $\boldsymbol{d}_{t-1}$ is decoder state at step $t-1$, $\boldsymbol{y}_{t-1}$ is the input of the decoder at step t (the embedding of predicted target word[4] y_{t-1} at $t-1$), f is a nonlinear function. Here, we use LSTM as f. $\boldsymbol{c}_{t-1}$ and $\boldsymbol{c}'_{t-1}$ are context vectors for title encoder and knowledge encoder respectively, computed as:

$$\boldsymbol{c}_t = \sum_i a_{ti} \boldsymbol{h}_i, \qquad \boldsymbol{c}'_t = \sum_i a'_{ti} \boldsymbol{h}'_i$$

where, a_{ti} is the weight of $\boldsymbol{a}_t$ at position i, and a'_{ti} is the weight of $\boldsymbol{a}'_t$ at position i.

Output Distribution

As shown in Figure 2, in decoding, MS-Pointer tries to retain the key information with the help of the knowledge encoder. Specifically, it learns to generate the brand name and the commodity name by picking words from the knowledge encoder. To this end, we introduce a soft gating weight λ to combine the attention distribution

[2]It is noteworthy that we use Chinese words here for convenience of presentation. In fact, our model is built on Chinese characters instead of Chinese words.

[3]"/" is also a separator between multi-language versions of the brane name, *e.g.*, *Nintendo*/任天堂.

[4]During training, this is the embedding of the previous word in the reference summary. At test time, it is the embedding of the previous word emitted by the decoder.

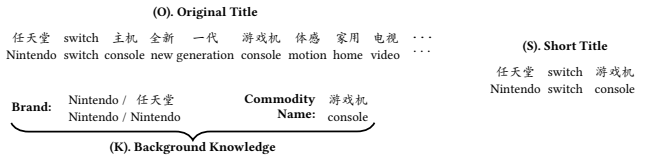


Figure 3: An example for the dataset.

$\boldsymbol{a}_t$ and $\boldsymbol{a}'_t$ as the final output distribution:

$$p(y_t{=}w|\mathcal{S},\mathcal{K},y_{<t}){=}\lambda\sum_{i:w_i=w}a_{ti}+(1{-}\lambda)\sum_{j:w_j=w}a'_{tj}$$

Thus, the model can learn to copy words from different encoders by adjusting the gating weight λ. Obviously, λ should be able to automatically adjust according to the decode state $\boldsymbol{d}_t$, the decode input $\boldsymbol{y}_{t-1}$, the source title's context vector $\boldsymbol{c}_t$, and the background knowledge's context vector $\boldsymbol{c}'_t$. In this paper, we define it using sigmoid function:

$$\lambda=\sigma\big(\boldsymbol{w}_d^\top\boldsymbol{d}_t+\boldsymbol{w}_y^\top\boldsymbol{y}_{t-1}+\boldsymbol{w}_c^\top\boldsymbol{c}_t+\boldsymbol{w}_{c'}^\top\boldsymbol{c}'_t\big)$$

where vector $\boldsymbol{w}_d$, $\boldsymbol{w}_y$, $\boldsymbol{w}_c$, and $\boldsymbol{w}_{c'}$ are parameters to be learned, and $\sigma(x)=1/(1+\exp(-x))$.

Here, the gating weight λ works like a classifier to tell the decoder to extract different information from corresponding encoders. At the first few steps, MS-Pointer usually produces a small λ. In this way, our model can easily copy the brand name (*e.g.*, *Nintendo*) from the knowledge encoder. After that, λ will become larger to push the model to copy other *modifier* information (*e.g.*, *motion* or *video*) from the title encoder. At last, λ will become smaller again, so the knowledge encoder can help to decode the commodity information.

Softmax is another feasible and general way to calculate the weights for each decoder. In our test, there is no visible difference between sigmoid and softmax. In this paper, we choose sigmoid as the case for easy explanation. We also try to define the output distribution as the sum of all encoders' attention distribution, *i.e.*, set $\lambda=0.5$ constantly. However, it works very poorly. As Figure 2 shown, the knowledge input is usually much shorter than the source title. As a result, the decoder often generate the brand repeatedly, like "*NintendoNintendo*. . .", due to the higher probability of words in knowledge encoder.

Finally, we define the training loss as the the negative log likelihood of the target sequence:

$$\mathcal{L}=\frac{1}{T}\sum_{t=0}^{T}-\log p(y_t{=}w_t^*|\mathcal{S},\mathcal{K},y_{<t})$$

where w_t^* is the target word at step t, T is the length of the target sequence.

5 EXPERIMENTS

5.1 Dataset Construction

For evaluation purposes, we build a new product title summarization dataset[5] from Taobao.com, since there is no public benchmark dataset for our task yet. Our proposed model requires two parts of data: (i) original product titles and their corresponding short titles; (ii) background knowledge about the products, *i.e.*, brand name and commodity name.

[5]http://ofey.me/data/pts

Table 1: The statistics of the data set. All lengths are counted by Chinese characters, and English word is counted as Chinese character.

Dataset size	411,267
Number of category	94
Avg. length of original titles	25.42
Avg. length of short titles	7.77
Avg. length of background knowledge	5.91

In terms of <original title, short title> pairs, we crawl the human-generated pairs from a product recommendation channel of the website. The product titles and their corresponding short titles are manually written by professional editors (the corresponding short titles are rewritten in an extractive manner), thus suitable to be viewed as gold-standard for our task. In terms of background knowledge, we collect these information from the corresponding fields in the database for each product. Thus, the data set can be represented as $<O,K,S>$, where O means products' original titles, K means the background knowledge about the products, and S represents the human-written short titles. A triplet example is presented in Figure 3.

We exclude the products whose short titles are longer than 10 Chinese characters since only 10 Chinese characters can be displayed in one line on mobile phones due to the screen size limitation. Eventually, we get a dataset with 411,267 pairs in 94 categories. Table 1 provides the detailed statistics about this dataset. Finally, we randomly stratified split the dataset into a training set (80%, 329,248 pairs), a validation set (10%, 41,031 pairs), and a test set (10%, 40,988 pairs) by preserving the percentage of samples for each category.

5.2 Baselines

To verify the effectiveness of our proposed model, we compare it with two classes of baselines.

The abstractive methods including:

- Vanilla sequence-to-sequence (**Seq2Seq-Gen**) is a basic encoder-decoder model based on LSTM unit [20] and attention mechanism [1].
- Pointer-Generator (**Ptr-Gen**) [38] is a hybrid model combing Seq2Seq-Gen with pointer network. Besides copying words from the input, Ptr-Gen can also generate words from the predefined vocabulary.

The extractive methods including:

- Truncation (**Trunc.**) is the simplest baseline for product title summarization, where the words are kept in their original order until the limit is reached. It is the practical solution in most E-commerce applications (*e.g.*, Amazon, eBay, and Taobao).
- **TextRank** [29] is a keyword extraction framework. It builds an automatic summary by extracting keywords or sentences from the text according to their scores computed by an algorithm similar to PageRank.
- **Seq2Seq-Del** is introduced by Filippova et al. [13] to compress the sentence by deletion in a seq2seq framewaork.

Different from Seq2Seq-Gen generating words for summarization at each step, Seq2Seq-Del predicts the binary label (*i.e.*, delete or retain) for each words in the original title at each decode steps[6].

- **LSTM-Del** is a standard sequence labeling system. It can be seen as a simplified version of Seq2Seq-Del since LSTM-Del performs the binary labeling based on the encoder's outputs.
- Pointer network (**Ptr-Net**) [41] is an extractive summarization baseline as we introduced in Section 3.2.
- **Ptr-Concat** is the pointer network with concatenated input (*i.e.*, concatenating the background knowledge with the source title). We implement this baseline to better evaluate the effectiveness of our proposed model.

For model x built on RNN architecture, we implement two versions based on unidirectional and bidirectional LSTM unit, indicated as x_{uni} and x_{bi}, respectively. Generally, we omit the subscript to refer to the model, not a specific implementation.

5.3 Experiment Settings

We implement all the models in `Tensorflow`[7] except Trunc. and TextRank. For TextRank, we adopt the implementation in an open-source Python library `SnowNLP`[8]. For Ptr-Gen[9], we modify the IO of the code released by the authors to fit our dataset. For all RNN encoder-decoder models, we use 128-dimensional word embeddings and 256-dimensional 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 256-dimensional states.

For all experiments, the word embeddings are initialized using normal distribution $\mathcal{N}(0, 10^{-8})$ and learned from scratch. Other learnable parameters are initialized in the range $[-0.02, 0.02]$ uniformly. We train all the models using Adagrad [11] with learning rate 0.15 and an initial accumulator value of 0.1. The gradient is clipped [35] when its ℓ_2 norm exceeds a threshold of 2. We do not use any form of regularization in all experiments. All the models are trained on a single Tesla M40 GPU with a batch size of 128. The validation set is used to implement early stopping and tune the hyperparameters. At test time, we decode the short titles using beam search with beam size 4 and maximum decoding step size[10] 11. For Seq2Seq-Del and LSTM-Del, we keep all the words with positive predicted label (*e.g.*, retain).

In this work, we try to minimize the preprocessing on the dataset. All models are implemented based on Chinese characters for the following considerations: (i) avoid the effect of Chinese word segmentation error; (ii) easily control the output length; (iii) models based on Chinese characters perform better. We only ignore the numbers that do not appear in the products' brand name. The product model (*e.g.*, *PS4*, *DDR4*, or *XXL*), the numbers in brand (*e.g.*, *7.Up* or *5.11*) or punctuations in brand (*e.g.*, *Coca-Cola*, *J.crew*, or *Kiehl's*) are all kept during training and testing. During training, we keep the tokens occurring greater than 2 times, resulting in a vocabulary of 47,110 tokens (4260 Chinese characters and 42,850 other tokens). For models based on pointer mechanism, we add the term <*EOS*> to the end of the source title, so they can terminate at the decoding time.

5.4 Automatic Evaluation

To automatically evaluate the performance of different models, we leverage three standard metrics: BLEU [34], ROUGE [24], and METEOR [2]. The BLEU metric is originally designed for machine translation by analyzing the co-occurrences of n-grams between the candidate and the references. For BLEU metric, we report BLEU-1, BLEU-2, and BLEU-4 here. The ROUGE 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 F_1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. 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 match mode (rewarding only exact matches between words). We obtain the BLEU and METEOR scores using the `nlg-eval`[11] packages, and ROUGE scores using `pythonrouge`[12] package.

Table 2 presents the results on seven automatic metrics. As expected, Trunc. and TextRank perform the worst on all metrics. For Trunc., the reason is it has no ability to extract the informative words (*e.g.*, commodity name like "*console*" or "*handheld*") from the tail of the product title. For TextRank, this is because the algorithm cannot extract meaningful keyword in such short text. Deletion based summarization models (*i.e.*, Seq2Seq-Del and LSTM-Del) perform significantly better than TextRank and Trunc. with a large gap. However, they are much worse than other seq2seq models like Ptr-Net and MS-Pointer. This is mainly due to the reordering phenomenon in short titles. Take the first case in Table 6 as an example, the short title changes the order between "*MOD-X*" and "*Nintendo Switch*". In fact, we find that more than half of the data (65.39% in training set and 65.81% in test set) adjust the word order to produce more fluent short titles. Seq2Seq-Del outperforms LSTM-Del mainly because it can leverage more information from the entire title representation under encoder-decoder framework.

Comparing different groups, it is easy to see that extractive models based on pointer mechanism perform better than abstractive models, especially significantly better than Seq2Seq-Gen. This demonstrates that pointer mechanism is very suitable for extractive summarization. An interesting phenomenon is that Ptr-Gen performs very close to Ptr-Net. This is because, as a hybrid model combing Seq2Seq-Gen with Ptr-Net, Ptr-Gen can degenerated into Ptr-Net on an extractive dataset. However, Ptr-Gen still make some factual mistakes occasionally, such as generating wrong brand name, replacing an uncommon (but in-vocabulary) word with a more-common alternative.

The results of the last group indicate that the background knowledge is helpful for product title summarization. Moreover, the gaps between MS-Pointer and Ptr-Concat demonstrate that our proposed

[6]The decoder's input for Seq2Seq-Del is also the original title. This is different from the vanilla sequence-to-sequence models.
[7]https://www.tensorflow.org
[8]https://github.com/isnowfy/snownlp
[9]https://github.com/abisee/pointer-generator
[10]The lsat token is invisible token <*EOS*>. So the real max length of generated short title is 10, as in training setting.

[11]https://github.com/Maluuba/nlg-eval
[12]https://github.com/tagucci/pythonrouge

Table 2: BLEU, ROUGE (F_1), and METEOR scores on the test set. Baselines on the top group are abstractive, while those in the following two groups are extractive. Bold scores are the best overall.

	BLEU-1	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Seq2Seq-Gen$_{uni}$	69.26	60.51	43.79	68.69	50.57	68.38	37.04
Seq2Seq-Gen$_{bi}$	69.87	61.34	44.71	68.87	51.09	68.56	37.43
Ptr-Gen$_{uni}$	72.70	64.54	48.87	71.91	55.37	71.63	38.48
Ptr-Gen$_{bi}$	73.13	65.15	50.04	72.80	56.32	72.55	38.86
Trunc.	44.31	36.26	23.58	45.94	30.16	44.35	26.57
TextRank	38.74	30.50	17.17	39.59	25.42	34.10	18.11
LSTM-Del$_{uni}$	63.67	52.28	33.52	65.23	43.61	64.35	33.87
LSTM-Del$_{bi}$	67.10	56.32	37.63	67.34	46.93	66.59	34.77
Seq2Seq-Del$_{uni}$	66.15	56.21	38.47	68.49	48.53	67.70	35.84
Seq2Seq-Del$_{bi}$	67.46	57.01	38.65	68.88	49.64	67.99	36.95
Ptr-Net$_{uni}$	73.88	65.81	50.73	73.40	56.93	73.13	39.10
Ptr-Net$_{bi}$	74.19	66.30	51.12	74.25	57.95	74.11	39.35
Ptr-Concat$_{uni}$	74.25	66.09	50.86	74.07	58.13	73.89	39.25
Ptr-Concat$_{bi}$	74.53	66.51	51.43	74.49	58.67	74.26	39.41
MS-Pointer$_{uni}$	75.11	67.17	52.55	75.15	59.62	74.96	39.91
MS-Pointer$_{bi}$	**75.57**	**67.72**	**53.06**	**75.69**	**60.29**	**75.45**	**40.25**

model is more flexible and effective in modeling such knowledge. Finally, the improvements from these baselines to MS-Pointer are statistically significant using a two-tailed t-test (p<0.05).

As shown in the table 2, comparing with unidirectional LSTM, the bidirectional LSTM only bring a very limited improvement for each model. This is largely because our input is very short and the decoder can only use unidirectional model. Considering the superiority of the bidirectional models, we only report the results of bidirectional models (omitting the subscript for convenience) in the following evaluations.

5.5 Brand Retention Test

Besides the standard metric for text generation, we also test whether the model retains the brand in the source title. Compared with normal sentence summarization task, retaining brand name is a particular and essential requirement for product title summarization. The evaluation metric for this task is the error rate of the brand names in the generated short titles. Besides the *offline* testset used before, we also build a new *online* testset by randomly sampling 140,166 product titles with brand names from Taobao.com.

Table 3 shows the results of each model on two testsets. As expected, TextRank achieves the worst results in this task because of its ignorance of semantics. Trunc. performes very well on this task because the brand name usually appears at the head of the title. Moreover, our MS-Pointer significantly outperforms the other baselines, especially on *online* testset. This demonstrates that introducing the knowledge encoder can help the pointer network to better retain the brand name. In addition, the results on the *online* testset also indicates that our MS-Pointer has a stronger generalization ability. The additional bad case analysis indicates that MS-Pointer's 2.89% error rate on *online* testset is largely due to the unknown (OOV) words. It is not easy for pointer mechanism to copy a right word when the input contains several OOV words,

Table 3: Results of brand retention experiment. Bold scores are the best

	offline	online
Seq2Seq-Gen	9.82%	28.55%
Ptr-Gen	2.85%	15.03%
Trunc.	2.42%	6.31%
TextRank	92.43%	93.08%
LSTM-Del	5.97%	22.31%
Seq2Seq-Del	4.71%	19.76%
Ptr-Net	2.54%	13.82%
Ptr-Concat	1.33%	6.48%
MS-Pointer	**0.13%**	**2.89%**

since all these OOV words share the same embedding for the symbol <*UNK*>. In practice, we reduce the error rate of MS-Pointer from 2.89% to **0.56**% on *online* testset by mapping each OOV word to unique embedding.

Besides the brand name, the commodity names are another key information for the products in the E-commerce platforms. However, it is very hard to automatically check the the short title generate the commodity name correctly or not, since the source title usually contains multiple commodity name while they are not necessary to be the same as in background knowledge, like examples in Table 6, So, we leave the test on commodity name in the manual evaluation in Section 5.6.

5.6 Manual Evaluation

Following the procedure in [13, 40], we conduct manual evaluation on 300 random samples from our testset. Three participants were asked to measure the quality of the short titles generated by each model from four perspectives: (i) **Key Information Retention (Accuracy)**, is the key information properly kept in the short title?

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

Model	Accuracy	Comm.	Readability	Info.
Seq2Seq-Gen	83.67%	91.33%	4.54	3.73
Ptr-Gen	91.03%	94.33%	4.71	4.09
Trunc.	29.67%	31.0%	2.67	2.31
TextRank	5.67%	33.33%	2.56	2.77
LSTM-Del	86.67%	92.33%	3.49	3.34
Seq2Seq-Del	89.33%	93.67%	3.52	3.46
Ptr-Net	92.0%	94.67%	4.79	4.21
Ptr-Concat	94.33%	95.33%	4.81	4.31
MS-Pointer	**97.33%**	**98.0%**	**4.87**	**4.55**
Human	100%	100%	4.93	4.51

(ii) **Commodity Name Retention (Comm.)**, is the commodity name correct? (iii) **Readability**, how fluent, grammatical the short title is? (iv) **Informativeness (Info.)**, how informative the short title is?

Comparing with conventional sentence summarization task, key information retention is an extra and essential requirement for product title summarization as we discussed in the introduction. We use a very strict criteria for this property, the generated short title will be assessed as 1 only if it correctly retains both brand name and commodity name, otherwise 0. In addition, we also test the commodity name retention precision solely as we explained in Section 5.5. The other two properties are assessed with a score from 1 (worst) to 5 (best).

The average results are presented in Table 4. Obviously, Trunc. and TextRank perform worst in this task as the same in automatic evaluations. The results on *Readability* metric show that all models built on Seq2Seq architecture (exclude Seq2Seq-Del) can generate very fluent titles. They also verify our previous claim that it is difficult to produce a fluent title for the models based on the deletion. The significant improvements on Accuracy metric demonstrate that our MS-Pointer can better retain the key information by the help of knowledge encoder. Considering all three metrics, our MS-Pointer produces more readable and more informative titles, which shows the advantage of introducing the knowledge encoder. Besides these baselines, we also conduct the experiments on human-written ground truths. The results show that MS-Pointer performs very close with human, ever better on Info. metric. This is because the editors often produce very concise titles while MS-Pointer may produces longer titles with more information.

5.7 Online A/B Testing

Previous experimental results have shown the superiority of our proposed MS-Pointer. In addition, we also deploy it in a real world application to test its practical performance. This subsection presents the results of online evaluation in a recommendation scenario of Taobao mobile app with a standard A/B testing configuration.

For online deployment, we generate the short titles for about 60 million products using our MS-Pointer model with 50 Tesla P100 GPUs in about 10 hours. Due to the screen size limit, we restrict the length of the short title to 8–10 Chinese characters. Note that the baseline deployed online is the truncated two-line titles (about 20 Chinese characters[13]). The A/B testing system randomly split online users equally into two groups and direct them into two separate buckets (each bucket contains about 2 million daily active users) respectively. Then for users in the bucket A (*i.e.*, the baseline group), the titles they saw are the truncated two-line titles. While for users in the bucket B (*i.e.*, the experimental group), the displayed short titles are generated by our MS-Pointer model.

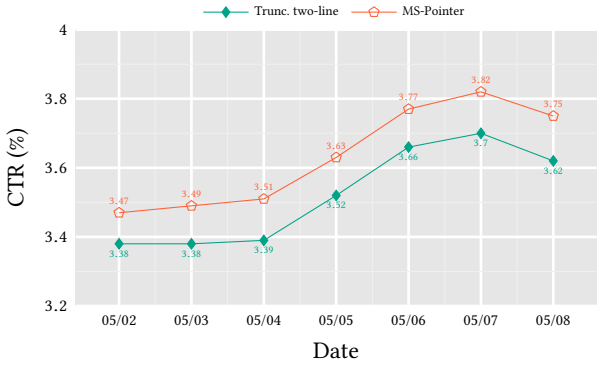


Figure 4: Online A/B Testing of CTR.

Table 5: CTR improvements under different categories

Clothing (Women)	Shoes (Women)	Beauty	Electronics
1.03%	2.71%	3.87%	13.26%
Clothing (Men)	**Shoes (Men)**	**Cell Phones**	**Computers**
5.84%	5.54%	9.75%	7.64%

This online A/B testing lasted for one week. All the settings of the two buckets are identical except the displayed titles. We adopt the Click-Through Rate (CTR) to measure the performance since the product titles often are a crucial decision factor in determining whether to click a product or skip to another. It is calculated as:

$$\text{CTR} = \frac{\#\text{click}}{\#\text{impression}}$$

where #click is the number of clicks on the product, #impression is the number of times the product is shown.

Figure 4 shows the results of overall CTR for all products in the two buckets in one week (from 05/02/2018 to 05/08/2018). It is obvious that the experimental bucket (*i.e.*, MS-Pointer) significantly outperforms the baseline bucket ($p < 0.05$). This clearly shows that the single-line short titles generated by MS-Pointer are more user-friendly and more likely to attract users to click on products.

In order to gain some intuition on how the these generated short titles affect the users, we also analysis the CTR improvements under different categories. Table 5 shows the overall CTR improvements under some typical categories in one week. This table reveals an interesting result that CTR improvements on categories like electronic devices are significantly higher than categories like clothing

[13]The full titles usually contain about 30 Chinese characters

Table 6: Examples of generated short titles. "␣" denotes visible whitespace in Chinese context.

Original Title	任天堂Switch␣游戏机专用背夹电池MOD-X真皮保护套 Nintendo Switch Console Dedicated Battery Case MOD-X Leather Case	美国␣曼哈顿Manhattan␣Portage␣邮差包␣单肩包挎包 US Manhattan Manhattan Portage Messenger Bag Shoulder Bag Satchel
Background Knowledge	Brand Name: MOD-X Commodity Name: 电池 // Battery	Brand Name: Manhattan Portage Commodity Name: 背提包 // Handbag and Knapsack
Ground Truth	MOD-X任天堂Switch背夹电池 // MOD-X Nintendo Switch Battery Case	Manhattan␣Portage␣邮差包 // Manhattan Portage Messenger Bag
Trunc.	任天堂Switch␣游戏机专用 // Nintendo Switch Console Dedicated	美国␣曼哈顿Manhattan␣Portage␣ // US Manhattan Manhattan Portage
TextRank	游戏机专用背夹电池MOD-X // Console Dedicated Battery Case MOD-X	Portage邮差包单 // Portage Messenger Bag single
Seq2Seq-Gen	任天堂NS␣switch主机 // Nintendo NS Switch Console	ORSLOW单肩斜挎包 // ORSLOW Shoulder Crossbody Bag
Seq2Seq-Del	任天堂游戏机保护套 // Nintendo Console Case	曼哈顿Manhattan␣Portage␣差包包 // Manhattan Manhattan Portage Bag Bag
LSTM-Del	任天堂游戏机 // Nintendo Console	曼哈顿Manhattan␣Portage包␣包包 // Manhattan Manhattan Portage Bag Bag
Ptr-Gen	任天堂switch游戏机 // Nintendo Switch Console	美国曼哈顿Manhattan␣邮差包 // US Manhattan Manhattan Messenger Bag
Ptr-Net	任天堂switch游戏机 // Nintendo Switch Console	曼哈顿单肩包 // Manhattan Shoulder Bag
Ptr-Concat	MOD-X任天堂Switch背夹电池 // MOD-X Nintendo Switch Battery Case	美国Manhattan␣Portage单肩包 // US Manhattan Portage Shoulder Bag
MS-Pointer	MOD-X任天堂Switch背夹电池 // MOD-X Nintendo Switch Battery Case	美国Manhattan␣Portage␣邮差包 // US Manhattan Portage Messenger Bag

and beauty, especially on clothing(women). This may be caused by the users' different behavioral pattern under different categories. When browsing the products like women's clothing or beauty, users usually pay more attention to the modifier words (*e.g.*, 刺绣/*embroidery*, 丝绸/*silk*, 高腰/*high-waist*, or 水润/*moisturizing*). However, ten Chinese characters are often difficult to include all the modifiers that attract the customers under these categories. While browsing the electronic devices or men's clothes, users usually do not care about these modifier words. A short and clear title is more attractive to users under these categories.

5.8 Case Study

To better understand what can be learned by each model, we show some typical examples they generated in Table 6. As expected, Trunc. and TextRank perform worst on these two cases. In terms of readability, all models based on encoder-decoder architecture can produce fluent and grammatical short titles. However, in terms of informativeness, MS-Pointer and Ptr-Concat perform much better than other baseline, thanks to introducing the background knowledge. Moreover, the short titles generated by our MS-Pointer are comparable with those ground truths.

With regard to informativeness, the difficult cases are those where brand name does not appear in the head of the title and where the baselines still try to pick it out from the head as they learned in the training data. For the right case in Table 6, Seq2Seq-Gen generates a wrong brand name "*ORSLOW*"; Ptr-Gen and Ptr-Net also fail to keep the brand intact. This type of error is unacceptable in the real-word applications. Nevertheless, our MS-Pointer surpasses Ptr-Concat with a more informative commodity name "*Messenger Bag*". The left case in Table 6 is more tough since it contains several brand name (任天堂/*Nintendo* and *MOD-X*) and commodity name (游戏机/*Console*,背夹电池/*Battery Case*, and 保护套/*Case*). Ptr-Net, Ptr-Gen, and deletion based methods all fail to generate the brand name "*MOD-X*" and commodity name "背夹电池/*Battery Case*". It is not easy to copy the right brand name and commodity name for models without the background knowledge.

We also visualize the gating weight λ in Figure 5 for each step of the decoder in these two cases. It is easy to see that our MS-Pointer produces very small λ for the brand name (*e.g.*, *MOD-X* and *Manhattan Portag*) to force the model to copy them from the knowledge encoder, and larger λ to force the model to copy the modifier from the source title encoder.

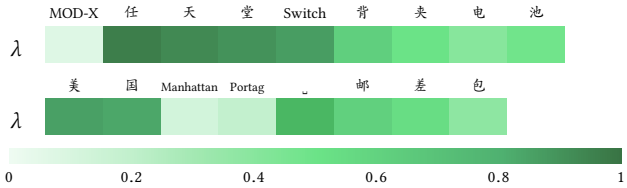


Figure 5: Heatmap of λ for the cases in Table 6.

6 CONCLUSION

In this paper, we study the product title summarization problem in E-commerce. In response to two particular constraints in this task, we propose a novel model MS-Pointer to generate the short titles by copying words from not only the source title but also the background knowledge. We perform extensive experiments with realistic data from a popular Chinese E-commerce website Taobao.com. Experimental results demonstrate that our model can generate informative and fluent short titles and significantly outperform other strong baselines. Finally, online A/B testing shows the significant business impact of our model in a real-world application.

Although introducing background knowledge makes our proposed model significantly outperforms the other baselines, the knowledge used in this paper is very elementary and limited. An interesting direction for the future work would be how to incorporate the knowledge graph into pointer network for product title summarization. Another direction is how to produce personalized short titles for different users, considering they may care about different properties about the products.

ACKNOWLEDGMENTS

We thank Siying Li, Yixing Fan, Liang Pang, Wen Chen, and Yanran Li for valuable comments and discussion. We also thank Abigail See for his kindness in sharing source code.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

[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*. Association for Computational Linguistics, Michigan, 65–72.

[3] Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline Generation Based on Statistical Translation. In *Proceedings of ACL*. Association for Computational Linguistics, 318–325.

[4] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop Clickbait: Detecting and preventing clickbaits in online news media. In *Proceedings of ASONAM*. 9–16.

[5] Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 484–494.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*. Association for Computational Linguistics, Doha, Qatar, 1724–1734.

[7] Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of NAACL*. Association for Computational Linguistics, San Diego, California, 93–98.

[8] Trevor Cohn and Mirella Lapata. 2008. Sentence Compression Beyond Word Deletion. In *Proceedings of COLING*. Manchester, UK, 137–144.

[9] Trevor Cohn and Mirella Lapata. 2013. An Abstractive Approach to Sentence Compression. *ACM Trans. Intell. Syst. Technol.* 4, 3, Article 41 (July 2013), 35 pages.

[10] Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*. 1–8.

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12 (July 2011), 2121–2159.

[12] Mihail Eric and Christopher Manning. 2017. A Copy-Augmented Sequence-to-Sequence Architecture Gives Good Performance on Task-Oriented Dialogue. In *Proceedings of EACL*. Association for Computational Linguistics, Valencia, Spain, 468–473.

[13] Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence Compression by Deletion with LSTMs. In *Proceedings of EMNLP*. Association for Computational Linguistics, Lisbon, Portugal, 360–368.

[14] Katja Filippova and Michael Strube. 2008. Dependency Tree Based Sentence Compression. In *Proceedings of INLG*. Association for Computational Linguistics, Salt Fork, Ohio, 25–32.

[15] Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of NAACL*. Association for Computational Linguistics, Los Angeles, California, 885–893.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of AISTAT*. PMLR, Fort Lauderdale, FL, USA, 315–323.

[17] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, 1631–1640.

[18] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, 140–149.

[19] Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning. In *Proceedings of ACL*. Association for Computational Linguistics, Vancouver, Canada, 199–208.

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[21] Hongyan Jing. 2002. Using Hidden Markov Modeling to Decompose Human-written Summaries. *Comput. Linguist.* 28, 4 (Dec. 2002), 527–543. https://doi.org/10.1162/089120102762671972

[22] Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text Understanding with the Attention Sum Reader Network. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, 908–918.

[23] Kevin Knight and Daniel Marcu. 2000. Statistics-Based Summarization - Step One: Sentence Compression. In *Proceedings of AAAI*. 703–710.

[24] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, 74–81.

[25] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent Predictor Networks for Code Generation. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, 599–609.

[26] Ryan McDonald. 2006. Discriminative Sentence Compression with Soft Syntactic Evidence. In *Proceedings of EACL*. 297–304.

[27] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *Proceedings of ICLR*.

[28] Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *Proceedings of EMNLP*. Association for Computational Linguistics, Austin, Texas, 319–328.

[29] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP 2004*. Association for Computational Linguistics, Barcelona, Spain, 404–411.

[30] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation. In *Proceedings of COLING*. Osaka, Japan, 3349–3358.

[31] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of CoNLL*. Berlin, Germany, 280–290.

[32] Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic Sentence Compression with a Character-based Metric: Tightening without Deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Association for Computational Linguistics, Portland, Oregon, 84–90.

[33] Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Inf. Process. Manage.* 43, 6 (Nov. 2007), 1506–1520. https://doi.org/10.1016/j.ipm.2007.01.019

[34] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*. PMLR, Atlanta, Georgia, USA, 1310–1318.

[36] Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *Proceedings of ICLR*.

[37] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP*. Association for Computational Linguistics, Lisbon, Portugal, 379–389.

[38] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of ACL*. Association for Computational Linguistics, Vancouver, Canada, 1073–1083.

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*. Curran Associates, Inc., 3104–3112.

[40] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive Document Summarization with a Graph-Based Attentional Neural Model. In *Proceedings of ACL*. Association for Computational Linguistics, Vancouver, Canada, 1171–1181.

[41] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Proceedings of NIPS*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 2692–2700.

[42] 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 *Proceedings of AAAI*.

[43] Shuohang Wang and Jing Jiang. 2017. Machine Comprehension Using Match-LSTM and Answer Pointer. In *Proceedings of ICLR*.

[44] 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 ACL*. Association for Computational Linguistics, Vancouver, Canada, 189–198.

[45] Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title Generation with Quasi-Synchronous Grammar. In *Proceedings of EMNLP*. Association for Computational Linguistics, Cambridge, MA, 513–523.

[46] Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence Simplification by Monolingual Machine Translation. In *Proceedings of ACL*. Association for Computational Linguistics, Jeju Island, Korea, 1015–1024.

[47] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic Aware Neural Response Generation. In *Proceedings of AAAI*. 3351–3357.

[48] Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards Implicit Content-Introducing for Generative Short-Text Conversation Systems. In *Proceedings of EMNLP*. Association for Computational Linguistics, Copenhagen, Denmark, 2190–2199.

[49] David M. Zajic, Bonnie J. Dorr, and Richard M. Schwartz. 2004. BBN/UMD at DUC-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop*. 112—-119.