Towards Knowledge-Based Personalized Product Description Generation in E-commerce

Qibin Chen[1]*, Junyang Lin[2,3]*, Yichang Zhang[3], Hongxia Yang[3]†, Jingren Zhou[3], Jie Tang[1]†
[1] Department of Computer Science and Technology, Tsinghua University
[2] School of Foreign Languages, Peking University
[3] Alibaba Group
chen-qb15@mails.tsinghua.edu.cn,linjunyang@pku.edu.cn
{yichang.zyc,yang.yhx,jingren.zhou}@alibaba-inc.com
jietang@tsinghua.edu.cn

ABSTRACT

Quality product descriptions are critical for providing competitive customer experience in an e-commerce platform. An accurate and attractive description not only helps customers make an informed decision but also improves the likelihood of purchase. However, crafting a successful product description is tedious and highly time-consuming. Due to its importance, automating the product description generation has attracted considerable interest from both research and industrial communities. Existing methods mainly use templates or statistical methods, and their performance could be rather limited. In this paper, we explore a new way to generate personalized product descriptions by combining the power of neural networks and knowledge base. Specifically, we propose a **K**n**O**wledge **B**ased p**E**rsonalized (or *KOBE*) product description generation model in the context of e-commerce. In *KOBE*, we extend the encoder-decoder framework, the Transformer, to a sequence modeling formulation using self-attention. In order to make the description both informative and personalized, *KOBE* considers a variety of important factors during text generation, including product aspects, user categories, and knowledge base. Experiments on real-world datasets demonstrate that the proposed method outperforms the baseline on various metrics.[1] *KOBE* can achieve an improvement of 9.7% over state-of-the-arts in terms of BLEU. We also present several case studies as the anecdotal evidence to further prove the effectiveness of the proposed approach. The framework has been deployed in Taobao,[2] the largest online e-commerce platform in China.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies →** *Natural language generation.*

[1]The **TaoDescribe** dataset and our code is publicly available at https://github.com/qibinc/KOBE.
[2]https://www.taobao.com/

KEYWORDS

Product Description Generation; Controllable Text Generation; Personalization; Knowledge Base

ACM Reference Format:
Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, Jie Tang. 2019. Towards Knowledge-Based Personalized Product Description Generation in E-commerce. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330725

1 INTRODUCTION

In e-commerce, product recommendation which aims at surfacing to users the right content at the right time [5] plays an important role in providing a superior shopping experience to customers. One of the biggest challenges is to timely understand customers' intentions, help them find what they are looking for, and provide valuable assistance during their entire shopping process. Different from physical stores where salespeople could have a face-to-face conversation with customers, online stores in e-commerce heavily rely on textual product descriptions to provide crucial product information and, eventually, convince customers to buy the recommended products. However, until now, most of the product descriptions in the online shopping platforms are still created manually, which is tedious, time-consuming, and less effective. On the one hand, textual descriptions should contain relevant knowledge and accurate product information to help customers make an informed decision. On the other hand, such descriptions should be personalized, based on customers' preferences, and promote their interests. In this paper, we focus on automating the product description generation for online product recommendation. Specifically, the system can intelligently generate an accurate and attractive description for a specific product based on the product information and user preferences.

Recently, there has been increasing attention on automatic product description generation. Most existing methods are based on templates and traditional statistical frameworks [17, 39]. Although applicable, these methods have some inherent drawbacks that they are limited in several respects. They place a restriction on the phrasal expression and discourse structure, and they cannot generate personalized and informative descriptions. Moreover, they do

*These authors contributed equally to this work.
†Corresponding Authors.

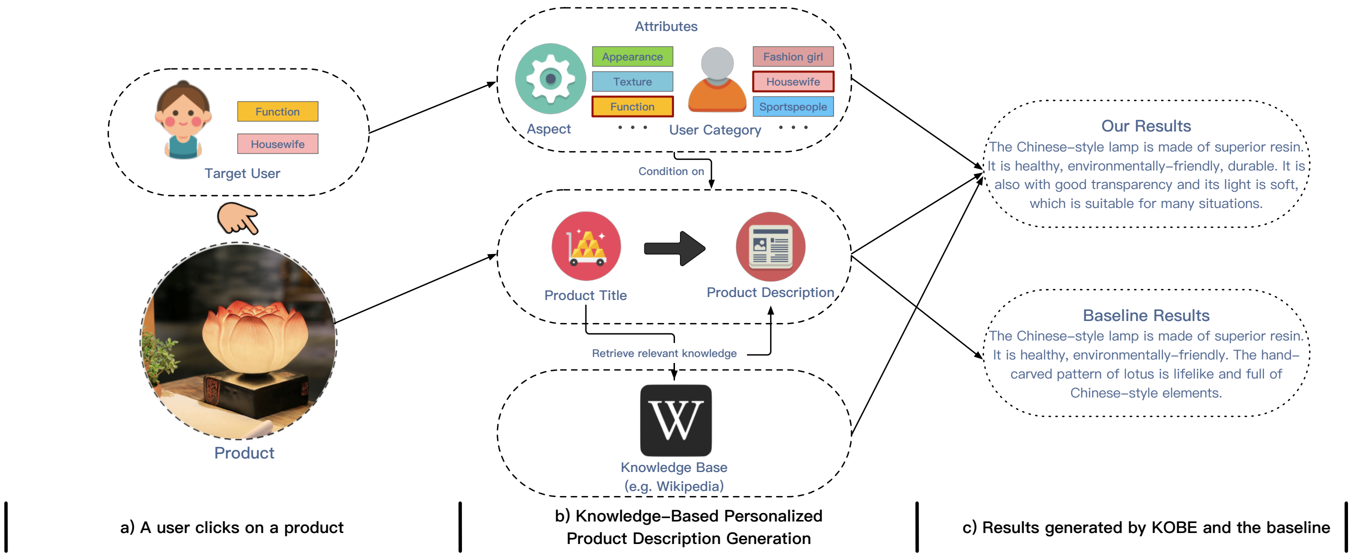
Attributes
Appearance
Texture
Function
Fashion girl
Housewife
Sportspeople
Aspect
User Category
Function
Housewife
Target User
Condition on
Our Results
The Chinese–style lamp is made of superior resin. It is healthy, environmentally–friendly, durable. It is also with good transparency and its light is soft, which is suitable for many situations.
Product Title
Product Description
Retrieve relevant knowledge
Baseline Results
The Chinese–style lamp is made of superior resin. It is healthy, environmentally–friendly. The hand–carved pattern of lotus is lifelike and full of Chinese–style elements.
Product
Knowledge Base (e.g. Wikipedia)
a) A user clicks on a product
b) Knowledge–Based Personalized Product Description Generation
c) Results generated by KOBE and the baseline

Figure 1: A motivating example of knowledge-based personalized product description generation. (a) A user clicks on a product, which is a Chinese-style resin lamp. The user focuses on the "function" product aspect and belongs to the "housewife" category. (b) The goal of *KOBE* is to generate a product description, given 1) the product title, 2) the desired product aspect and user category and 3) the relevant knowledge retrieved from an existing knowledge base. (c) The first and second box displays the descriptions generated by our model and the baseline respectively. In this example, *KOBE* focus on the specified "function" product aspect and considers the user category "housewife". It also incorporates the knowledge "resin is transparent". In the meanwhile, the result produced by the baseline describes the product's appearance and style, which is less in this user's interest and might be more appropriate for younger user groups instead. The baseline result doesn't reflect external knowledge either.

not take any information about users or external knowledge into consideration.

Owing to the success of neural networks in natural language processing, we propose to apply neural methods and sequence-to-sequence (Seq2Seq) learning [35] to product description generation. The Seq2Seq model has achieved tremendous success in natural language generation, including neural machine translation [2, 35, 40] and abstractive summarization [27, 30]. Furthermore, as mentioned above, product descriptions should be personalized and informative. We therefore propose a **K**n**O**wledge **B**ased p**E**rsonalized (or *KOBE*) product description generation model for online recommendation. We extend the effective encoder-decoder framework, the self-attentive Transformer, to product description generation. Besides, two main components of *KOBE*, Attribute Fusion and Knowledge Incorporation, effectively introduce the product attributes and incorporate the relevant external knowledge. Figure 1 shows an example of knowledge-based personalized product description generation. We also showcase how *KOBE* is deployed in the section *Guess You Like*, the front page of Mobile Taobao.

To provide a large dataset for the automatic product description generation, we construct and make publicly available a new dataset of large volume, collected from Taobao. The dataset contains 2,129,187 pairs of product basic information and descriptions created by shop owners from November 2013 to December 2018. Each instance in the dataset is labeled with attributes. For example, the user category attribute specifies the category of interest for potential readers. We also retrieve the relevant knowledge about the products from a large-scale knowledge base in Chinese, CN-DBpedia [41]. Models trained on the dataset are able to learn how to generate descriptions based on titles, attributes and relevant knowledge.

Our contributions are illustrated below:

- We propose to extend neural methods and sequence-to-sequence learning to product description generation, and we implement our model based on the self-attentive Transformer framework.
- To achieve effects in recommendation, we focus on the generation of personalized and informative product descriptions and propose a novel model *KOBE* that uses product attributes and external knowledge to improve the quality of generation.
- We construct a new dataset of large volume, **TaoDescribe**, for product description generation. The dataset contains basic information of products and the corresponding descriptions. The data are collected from the real-world data in Taobao and we contribute this dataset to the community to nourish further development (refer to the Appendix).
- The evaluation and the analyses in the experiments demonstrate that our method outperforms the baseline model, and it is able to generate personalized and informative product descriptions.

Organization The rest of the paper is organized as follows: Section 2 formulates the knowledge-based personalized product description generation problem. Section 3 introduces the proposed *KOBE* framework in detail. In Section 4, we conduct extensive experiments and case studies. Finally, Section 5 summarizes related work and Section 6 concludes this work.

2 PROBLEM FORMULATION

In this section, we formulate the problem of automatic product description generation. The objective of the task is to build a system that can generate product description automatically based on the input text. In the basic version of the problem, we take the product title as the input. Given the product title represented as an input sequence of words $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}$, the objective of the system is to generate the description $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \mathcal{Y}$, a sequence of words describing the product. Our ultimate goal is to obtain a personalized and informative description, so we introduce attributes and knowledge and further provide an improved definition.

Definition 2.1. **Attribute** Each product title is annotated with multiple attribute values. There are l attribute sets $\mathcal{A}_1, \ldots, \mathcal{A}_l$. For each attribute type, each product has its attribute value $a_i \in \mathcal{A}_i$. To generalize, it can be presented as $\boldsymbol{a} = (a_1, \ldots, a_l) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_l$. In our context, we have two attribute sets, where $\mathcal{A}_1$ represents the aspects of the products, such as quality and appearance, and $\mathcal{A}_2$ represents the user categories, which reflect the features of interested users. $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$ represents the number of aspects and the number of user categories.

Definition 2.2. **Knowledge** We consider knowledge as an extensive collection of raw text entries $\mathcal{W}$ (e.g., Wikipedia, CN-DBpedia) indexed by named entities in $\mathcal{V}$. Each entry consists of a named entity $v \in \mathcal{V}$ as the key and knowledge $\boldsymbol{w} \in \mathcal{W}$ as the value, which is a sequence of words $\boldsymbol{w} = (w_1, w_2, \ldots, w_u)$.

Target problem: Given the above definitions, our problem can be formally defined as generating a personalized and informative product description $\boldsymbol{y}$, based on the product title $\boldsymbol{x}$, the attributes $\boldsymbol{a}$ as well as the related knowledge $\boldsymbol{w}$.

3 *KOBE*: PROPOSED MODEL FRAMEWORK

In this section, we first review our basic framework Transformer [37]. We then propose a novel model called *KOBE*. Besides the Transformer framework, *KOBE* consists of two other modules: Attribute Fusion and Knowledge Incorporation. Attribute Fusion is responsible for integrating the attributes, including product aspects and corresponding user categories, with the title representation; and Knowledge Incorporation is responsible for incorporating the relevant knowledge retrieved from the knowledge base. These two modules respectively contribute to the generation of personalized and informative product description. As shown in Figure 2, our proposed neural network model consists of a product title and attribute encoder, a knowledge encoder, a bi-attention layer, a description decoder, and an output layer. In the following, we first introduce the basic framework of our model, namely Transformer, and then explain how we realize the personalization through the integration of attributes and how we improve the informativeness of the generation through the incorporation of knowledge in *KOBE*.

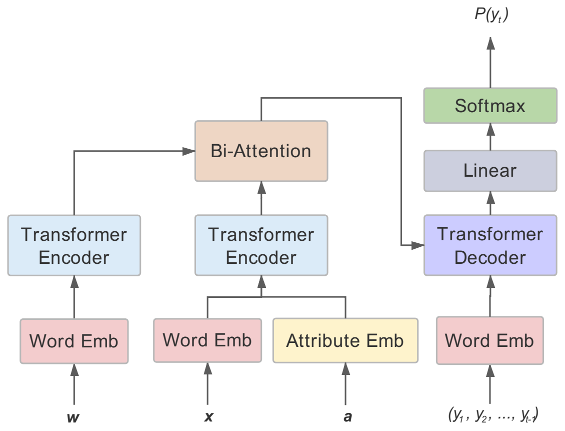

Figure 2: *KOBE* model architecture. We use six-layer transformer encoders for encoding the title $\boldsymbol{x}$ and the knowledge $\boldsymbol{w}$. The resulting representations $\boldsymbol{h}$ and $\boldsymbol{u}$ are then combined through bi-attention. We use a two-layer transformer decoder for decoding.

3.1 An Encoder-decoder Framework

Transformer is based on vanilla feed-forward neural networks and attention mechanism [37]. In our implementations, the Transformer has demonstrated improved performances over the conventional frameworks, such as the RNN-based Seq2Seq. Thus we focus on the details of Transformer in the following description.

Encoder Generally, the encoder takes in the input text, and encodes it to a series of hidden representations. Specifically, the Transformer encoder takes in a sequence of words $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and sends it through an embedding layer to obtain the word embedding representations $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$. Besides the embedding layer, positional encoding is applied in order to represent positional information. We follow Vaswani et al. [37] to use sinusoidal positional encoding. Then the word representations are transformed by the encoding layers to deep contextualized representations $\boldsymbol{h} = (h_1, h_2, \ldots, h_n)$.

On top of the embedding layer, the encoder is stacked with 6 identical layers, following Vaswani et al. [37]. Take the first layer as an example. Inside the layer, the input representations are first transformed by multi-head self-attention. The attention score computation generates a distribution over the values $\mathbf{V}$ with the queries $\mathbf{Q}$ and keys $\mathbf{K}$, and the mechanism obtains an expectation of $\mathbf{V}$ following the distribution. The computation can be described below:

$$\mathbf{C} = \alpha \mathbf{V}, \tag{1}$$

$$\alpha = \text{softmax}(f(\mathbf{Q}, \mathbf{K})), \tag{2}$$

where $\mathbf{C}$ refers to the output of attention, α refers to the attention distribution, and f refers to the function for attention scores.

As for self-attention, we first introduce the implementation of the uni-head attention and then the multi-head version. For the uni-head self-attention, the queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ are the linear transformation of the input context $\boldsymbol{e}$. To be specific, $\mathbf{Q} = W_Q \boldsymbol{e}$, $\mathbf{K} = W_K \boldsymbol{e}$ and $\mathbf{V} = W_V \boldsymbol{e}$, where W_Q, W_K and W_V are weights. The representation of uni-head attention $\mathbf{C}_{self}$ can be presented as below:

$$\mathbf{C}_{self} = \text{softmax}\left(\frac{\mathbf{QK^T}}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3}$$

where d_k refers to the size of each input representation $\boldsymbol{e}$.

Instead of using uni-head attention, our baseline model is equipped with multi-head attention, where we concatenate the attention representation of each head $\mathbf{C}^{(i)}_{self}$, $i \in \{1, \ldots, c\}$, where c is the number of heads. Specifically, for each head, we use the same computation for uni-head attention to generate the attention representation of the i-th head $\mathbf{C}^{(i)}_{self}$. In the next step, the output representations are sent into the Point-Wise Feed-Forward Neural Network (FFN), whose computation can be described in the following:

$$FFN(z) = W_2(ReLU(W_1 z + b_1)) + b_2 \tag{4}$$

where z refers to the input of FFN.

Decoder Similar to the encoder, the decoder is stacked with 2 decoding layers containing multi-head self-attention and FFN. Furthermore, the decoder of Transformer performs attention on the source-side context, which is the final representation of the encoder $\mathbf{h}$. Here, the attention is named "context attention" for the distinction from "self-attention". Context attention is still the aforementioned multi-head attention. The query $\mathbf{Q}$ is the linear transformation of the decoder state, and the keys $\mathbf{K}$ and values $\mathbf{V}$ and the linear transformation of the source-side context.

Training The training of Transformer is based on the idea of maximum likelihood estimation. The objective is to generate a sequence to approximate the *target* based on the input sequence. Formally, we have:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{m} P\left(y_t | y_1, y_2, \ldots y_{t-1}, \boldsymbol{x}\right), \tag{5}$$

where $\boldsymbol{x}$ refers to the source sequence, $\boldsymbol{y}$ refers to the target sequence, and t refers to the decoding time step.

To put into a nutshell, our proposed model is based on the self-attentive Transformer, which essentially is an encoder-decoder framework. We utilize the advantage of self-attention in our task to generate product descriptions. In Section 3.2 and 3.3, we will describe how our proposed method enables personalization and informativeness for product description generation.

3.2 Attribute Fusion

One limitation of the baseline encoder-decoder framework is that it often gives general and vague descriptions which are often boring and useless. In addition, it does not consider the preferences of the users and usually "speak" in a monotonous flavor. To alleviate these problems, we not only consider product titles and descriptions but also intake "flavored" specific attributes, such as aspects and user categories. We hope to generate different "flavored" descriptions targeting at different groups of people, for example, formal style for office people, fashion styles for adolescents, and so forth. Following Ficler and Goldberg [7], Sennrich et al. [31], we obtain these attributes from two sources: (1) the description using a heuristic and (2) user feedback associated with the description.

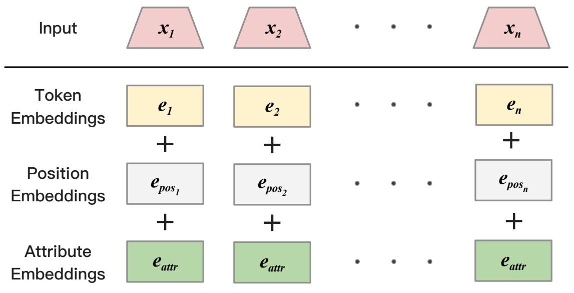

Figure 3: The input embeddings of the conditioned model are the sum of the title token embeddings, the position embeddings and the attribute embeddings.

Aspects A product description may contain information about different aspects of the product. Through a preliminary study on the large dataset (Cf. Section 4 for the detail of the dataset), we found that a successful description usually has a "focus" on one aspect of the product. For example, for a lamp, the focused aspect might be "appearance" with words like "beautiful" and "delicate" describing its external features. Based on our statistics of the dataset, we empirically select $|\mathcal{A}_1|$ aspects, e.g., "appearance", "function".

Due to the massive scale of our data set, precisely labeling the aspect of each description in the dataset is not possible. We thus extract the aspect from the description using a heuristic method based on semantic similarity. Each description $\boldsymbol{y} \in \mathcal{Y}$ is labeled with an aspect a_1, based on the semantic similarity between the description and the aspects. Details about the choice of the aspects and labeling methods are introduced in the additional pages (Cf. Appendix A).

User Categories Besides the different aspects of each description, we also try to discover the group of users who would be interested in a description. This is important to further design personalized description for target users. In our e-commerce setting, each user is labeled with "interest tags", which derive from his/her basic information, e.g., browsing history and shopping records. In our context, we consider the user's most important tag as his/her category and each user is assigned to one of the $|\mathcal{A}_2|$ categories.

In the ideal conditions, we have users' implicit feedback (click, dwell time) data for each description $\boldsymbol{y}$ in the dataset. Each description can then be assigned to a user category, according to the major group of users that have clicked or stayed long enough on this description. Note that soft assignment to user categories is also feasible in our framework. We find that hard assignment is simple and effective enough and thus we use hard assignment in our experiments and online systems.

We find that user feedbacks on the product descriptions in our dataset are very sparse. This results in a lot of noise in the assigned user categories. To overcome this problem, we collect another set of texts $\mathcal{Z}$ in our e-commerce context. These texts are similar to product descriptions but have much more user feedback. Then we train a CNN-based [14] multi-class classifier on $\mathcal{Z}$ which takes the text as input and predicts the user category it belongs to. After that, we use the trained classifier M to label $\mathcal{Y}$ and obtain a user category attribute a_2 for each $\boldsymbol{y} \in \mathcal{Y}$. The collection of "interest tags", the dataset $\mathcal{Z}$, the architecture of the classifier and training procedure will be described with details in the additional pages (Cf. Appendix A).

Attribute Fusion Model In a conditioned encoder-decoder framework similar to [7, 36], we add an additional conditioning context, which is attribute $\boldsymbol{a}$ in our case. Then Equation 5 is changed to:

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{a}) = \prod_{t=1}^{m} P(y_t|y_1, y_2, \ldots y_{t-1}, \boldsymbol{x}, \boldsymbol{a}). \tag{6}$$

There are several ideas to condition the model on a context [18, 24, 31]. However, most of them are initially designed for LSTMs. For Transformer, we experimented with all these methods (Cf. Section 4). Our results show that it achieves the best performance by simply adding the attribute embedding to the title embeddings.

Specifically, we first embed the aspect attribute a_1 and user category attribute a_2 to obtain the attribute representation $\boldsymbol{e}_{a_1}$ and $\boldsymbol{e}_{a_2}$. We average the embedding of the two attributes to obtain a single representation $\boldsymbol{e}_{attr}$. Then, given the product title embeddings $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$, we add the attribute embedding to the title word embedding e_i at each timestamp i. The fused representation is illustrated in Figure 3. A similar idea has also been used in BERT [6].

In our experiments In Section 4, we will demonstrate that both the models with the aspects and user categories work effectively. In addition, the attribute fusion model is straightforward to generalize to other kinds of attributes.

3.3 Knowledge Incorporation

The basic product information is still far from enough for generating an interesting and informative product description, for the reason that in the real-world context, humans generate descriptions not only based on the product information, but also their commonsense knowledge in mind about the product. For example, when generating the description for a resin lamp, one may use his/her knowledge, such as the features of resin, to compose an informative description. Inspired by this, we propose that the generation should make a good combination of the basic information and the relevant external knowledge. We thus consider grounding the basic information of the specified products with the associated knowledge [10], following the procedure where people utilize their knowledge to describe the product.

To be more specific, we first retrieve the relevant knowledge with the product information from the knowledge base and then encode the knowledge, which is separated from the encoding of the product title. Before entering the phase of decoding, both encoded representations are integrated with our proposed techniques. Thus the decoding can be based on both the basic information and the relevant knowledge. We discuss our proposed method for knowledge incorporation, including knowledge retrieval, knowledge encoding as well as the combination, in the following.

Knowledge Retrieval Knowledge Retrieval is responsible for retrieving relevant knowledge for a product from the knowledge base $\mathcal{W}$. Specifically, we obtain the relevant knowledge for our input products from CN-DBpedia [41], which is a large-scale structural knowledge graph in Chinese. The dataset that we construct is in Chinese to be introduced in detail in Section 4.1, and CN-DBpedia perfectly matches the requirements of our purpose. As we assume that the relevant knowledge of each word in the product title should be relevant to the product itself, the words in the product title $\boldsymbol{x}$ are the search items for the relevant knowledge. Formally, given a product title $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, we match each word x_i to a named entity $v_i \in \mathcal{V}$, which should be a vertex in the knowledge graph. Then we retrieve the corresponding knowledge $\boldsymbol{w}_i$ from $\mathcal{W}$ based on the named entity v_i. Furthermore, as there are multiple knowledge concepts describing v_i in the knowledge base, we randomly sample 5 candidates as its matched knowledge for each product. We then concatenate the knowledge according to their original order (i.e., the order in which their corresponding named entity appears in the product title), separated by a special token <SEP>.

Knowledge Encoding and Combination The retrieved knowledge and the basic information are somehow different, describing different aspects of the product. A proper mechanism should be designed to combine the two pieces of information. We set a **Knowledge Encoder**, which is a self-attentive Transformer encoder. It is responsible for encoding the retrieved knowledge to a high-level representation $\boldsymbol{u}$. The obtained representation $\boldsymbol{u}$ is similar to the encoded representation $\boldsymbol{h}$ of the product title $\boldsymbol{x}$. We then apply the BiDAF (bidirectional attention flow [32]) to combine the two kinds of representations.

More specifically, we compute attentions in two directions: title-to-knowledge attention and knowledge-to-title attention. Both of these attentions are derived from a shared similarity matrix, $\mathrm{S} \in \mathbb{R}^{n\times u}$, between the contextualized representations $\boldsymbol{h} \in \mathbb{R}^{n\times d}$ of the title $\boldsymbol{x}$ and $\boldsymbol{u} \in \mathbb{R}^{u\times d}$ of the knowledge $\boldsymbol{w}$. S_{ij} indicates the similarity between the i-th title word and the j-th knowledge word. The similarity matrix is computed as:

$$\mathrm{S}_{ij} = \alpha\left(h_i, u_j\right) \in \mathbb{R}, \tag{7}$$

where α represents the function that encodes the similarity between the two input vectors. We choose $\alpha(h,u) = \mathbf{w}_{\mathrm{S}}^{\top}[h; u; h \circ u]$, where $\mathbf{w}_{\mathrm{S}} \in \mathbb{R}^{3d}$ is a trainable weight vector (d is the dimension of contextualized representation) , $\circ$ is element-wise multiplication, $[;]$ is vector concatenation across row, and implicit multiplication is matrix multiplication. Now we use $\mathbf{S}$ to obtain the attentions and the attended vectors in both directions.

Title-to-knowledge attention signifies which knowledge words are most relevant to each title word. Let $\mathbf{a}_i \in \mathbb{R}^u$ represent the attention weights on the knowledge words by the i-th title word, $\sum_j \mathbf{a}_{ij} = 1$ for all i. The attention weight is computed by $\mathbf{a}_i = \mathrm{softmax}\left(\mathrm{S}_{i:}\right)$, and subsequently each attended knowledge vector is $\tilde{u}_i = \sum_k \mathbf{a}_{ik} u_k$. Hence $\tilde{\boldsymbol{u}} \in \mathbb{R}^{n\times d}$ contains the attended knowledge vectors for the entire title.

Knowledge-to-title attention signifies which title words have the closest similarity to one of the knowledge words. We obtain the attention weights on the context words by $\mathbf{b} = \mathrm{softmax}\left(\max(\mathrm{S}_{i:})\right)$. Then the attended title vector is $\tilde{h} = \sum_k \mathbf{b}_k h_k$. This vector indicates the weighted sum of the most important words in the context with respect to the knowledge. $\tilde{h}$ is tiled n times across the column, thus giving $\tilde{\boldsymbol{h}} \in \mathbb{R}^{n\times d}$. Similar to [32], we use a simple concatenation $[\boldsymbol{h}; \tilde{\boldsymbol{u}}; \boldsymbol{h} \circ \tilde{\boldsymbol{u}}; \boldsymbol{h} \circ \tilde{\boldsymbol{h}}] \in \mathbb{R}^{4d\times T}$ to get the final combined representation.

4 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the performance of the proposed model. We first introduce the dataset, the compared baseline models and the evaluation metrics. We also demonstrate the experimental results in a series of evaluations and perform further analyses on the effectiveness of our approach in generating both personalized and informative descriptions. Details about the experiment setup and implementation will be described in the additional pages (Cf. Appendix A).

4.1 Dataset

In order to further the study of product description generation, considering that there is a lack of large-scale dataset for this task, we constructed a new dataset **TaoDescribe**, containing the basic information of the products, including the title, the aspect and the user category, as well as the product description. The data are collected from Taobao, a large-scale website for e-commerce in China.

The product information and the description are composed by the sellers and content producers on the website from November 2013 to December 2018. Each data instance is automatically annotated with its aspect and user category using the methods introduced in Section 3.2. Each data instance is also matched with knowledge retrieved from the knowledge base CN-DBpedia. This enables product description generation to make use of such information for personalization and knowledge incorporation. Overall, the dataset contains 2,129,187 $(\mathbf{x}, \mathbf{y}, \mathbf{a}, \mathbf{w})$ instances after preprocessing. More details about the dataset will be described in the additional pages (Cf. Appendix A).

4.2 Systems for Comparison

In this section, we introduce the baseline and choices for our model components.

Attribute Fusion We first investigate common choices of conditioning a sequence-to-sequence model on input attributes.

- **Baseline** Our baseline is a Transformer, with the product title as its source sequence and the description as its target sequence, without considering attributes or incorporating knowledge.
- **Attr-D** (**D**edicated) Before moving on to conditioned models, we train a dedicated model on each subset of the data following Ficler and Goldberg [7]. For example, for three aspects "appearance", "texture" and "function", we split the dataset into three subsets and train an unconditional model on each subset. We test them on the corresponding portions of the test set and average the scores of all testing instances. However, this method does not scale to multiple attributes or attributes with more possible values (e.g., user category) because there will be too many models to be trained and the dataset for each model will be tiny. Therefore, We train dedicated models for the aspects only.
- **Attr-S** (**S**ource token) Following Sennrich et al. [31], we can add attributes as special tokens to the source text (i.e., product title). In our experiments, we add one of <A-1>, <A-2>, ..., <A-$|\mathcal{A}_1|$> to represent the aspect and one of <U-1>, <U-2>, ... , <U-$|\mathcal{A}_2|$> to represent the user category.
- **Attr-T** (start of **T**arget sequence) Another technique to attribute fusion is to use an attribute-specific start token for the target sequence [24], instead of a shared token. For multiple attributes, we first embed each attribute value and then feed the averaged embedding as the start of sequence token embedding following Lample et al. [16].
- **Attr-A** (**A**dd) We simply add the attribute embeddings to the title embeddings at each timestep as introduced in Section 3.2. For multiple attributes, we embed them separately and average the attribute embeddings before adding to the title embeddings.

We conducted three experiments with each Attribute Fusion model: 1) only adding aspects, 2) only adding user categories, 3) adding both attributes. They are denoted as Attr-S (Aspect), Attr-S (User), Attr-S (Both).

Knowledge Incorporation We then compare the effect of incorporating knowledge using bi-attention.

- **Know-BiAttn** Know-BiAttn refers to the model equipped with bidirectional attention, which aims at integrating the title representation and knowledge representation as introduced in Section 3.3.

KOBE Our final model *KOBE* combines **Attr-A (Both)** and **Know-BiAttn**.

4.3 Evaluation Metrics

We evaluate our model on generation quality, diversity and attribute capturing ability.

BLEU We first verify that the introduction of the attributes indeed helps in achieving better results in terms of the BLEU score as a sanity check [28]. We compare the test BLEU score of our conditioned model to the unconditioned baseline trained on the same data.

Lexical Diversity A common problem in automatic text generation is that the system tends to generate safe answers without enough diversity [19, 20]. Thus we also evaluate the lexical diversity to observe whether our model is able to generate diversified texts. This is also a reflection of informativeness as human-generated texts are usually full of diversified contents. A low diversity score often means generated contents are general and vague, while higher diversity means the generated contents are more informative and interesting. We calculate the number of distinct n-grams produced on the test set as the measurement of the diversity of generated descriptions.

Attribute Capturing However, BLEU and lexical diversity are not sufficient for measuring how well the generated results correlate to the specified attributes. In particular, we are interested in evaluating how difficult it is to recover the input attributes from the descriptions generated by a model.

For aspects, it is straightforward to run the automatic tagging procedure on our generated descriptions; for user categories, we use the pretrained classifier to predict the user category of each generated description. For attribute k, the attribute capturing score is computed as the prediction accuracy:

$$\frac{1}{|\hat{Y}_T|} \sum_{(\hat{\boldsymbol{y}}, \boldsymbol{y}) \in (\hat{Y}_T, Y_T)} \mathbb{1}_{M(\hat{\boldsymbol{y}})=M(\boldsymbol{y})}, \tag{8}$$

where $M : \mathcal{Y} \rightarrow \mathcal{A}_2$ denotes the user category classifier and $\mathbb{1}_{M(\hat{\boldsymbol{y}})=M(\boldsymbol{y})}$ is the indicator function with value as 1 when $M(\hat{\boldsymbol{y}}) = M(\boldsymbol{y})$ and 0 otherwise.

4.4 Result Analysis

In this section, we conduct an analysis of our proposed model to evaluate the contribution of attributes and knowledge. We focus on a few issues illustrated below.

Does Attribute Fusion improve generation quality? Table 1 shows the BLEU scores of the models. As shown in Table 1, Attribute Fusion brings an improvement in BLEU score. Attr-A (Both) outperforms the baseline with an advantage of +0.7 BLEU (relatively 9.7%). This gain in BLEU score is notable, considering the attributes are categorical and contain little information highly relevant to the product.[3]

Do Attribute Fusion and Knowledge Incorporation improve generation diversity? Table 2 and Table 3 show the diversity scores of the models, including the ones conditioned on attributes and knowledge. Both aspect and user category improve the diversity significantly by 46.8%. Incorporating knowledge improves the diversity score by 56.4%. The improvement in diversity demonstrates that the descriptions generated by our model contain more content than those of the baseline.

Does our model capture characteristics of different attributes? Table 4 shows that the accuracy of the attribute classifier, whose computation of the accuracy is described in Equation 8. The low accuracy for the baseline indicates that it is highly possible that the description of the baseline does not match the interested user group. On the contrary, *KOBE* generates descriptions that better reflect the intended attributes. *KOBE* obtains a 13.4% absolute gain in the accuracy for the attribute “user category”, and a 12.4% gain in the accuracy for the attribute “aspect”. This means that our model targets the intended user groups more accurately and it is more focused on a specific aspect.

Ablation Study In Table 5, we report the results of an ablation study of *KOBE*. This ablation study is helpful for understanding the impacts of different components on the overall performance. The components are 1) knowledge, referring to Knowledge Incorporation; 2) user category, referring to the fusion of user category; 3) aspect, referring to the fusion of aspect. We remove one component successively from *KOBE* until all are removed. Removing knowledge increases the BLEU score from 7.6 to 8.2 but decreases the diversity from 15.1 to 11.1. The external knowledge that is not highly relevant to the product can harm the model's performance in BLEU. Yet it can significantly bring more contents and thus increase the diversity. Removing user category decreases BLEU sharply from 8.2 to 7.6, and still significantly decreases the diversity from 11.1 to 9.6. Removing aspect still causes a decrease in BLEU, from 7.6 to 7.2, and a decrease in diversity, from 9.6 to 7.8. These results demonstrate that the attributes do not negatively affect the model's performance in BLEU but make an important contribution to the diversity. In brief, the three components jointly bring a significant improvement to the informativeness of *KOBE*'s generation without sacrificing the performance in BLEU.

[3]Note that these attributes act as an oracle. This simulates the deployed setting where attributes are specified according to user preferences and categories.

Table 1: Test BLEU score with different methods of Attribute Fusion.

Model	Aspect	User Category	Both
Baseline	7.2	7.2	7.2
Attr-D	7.5	-	-
Attr-S	7.6	7.3	7.9
Attr-T	7.3	7.1	7.7
Attr-A	**7.6**	**7.5**	**8.2**

Table 2: Test n-gram diversity score with different methods of Attribte Fusion.

Model	n=3 ($\times 10^5$)	n=4 ($\times 10^5$)	n=5 ($\times 10^6$)
Baseline	2.4	7.8	2.0
Attr-S (Aspect)	2.8	9.7	2.5
Attr-T (Aspect)	2.8	9.6	2.5
Attr-A (Aspect)	2.8	9.6	2.5
Attr-S (User)	2.8	9.6	2.5
Attr-T (User)	2.6	9.1	2.4
Attr-A (User)	2.7	9.4	2.4
Attr-S (Both)	3.1	**11.1**	2.9
Attr-T (Both)	2.8	9.7	2.5
Attr-A (Both)	**3.3**	**11.1**	**3.0**

Table 3: Test n-gram diversity score improved by Knowledge Incorporation.

Model	n=3 ($\times 10^5$)	n=4 ($\times 10^5$)	n=5 ($\times 10^6$)
Baseline	2.4	7.8	2.0
Know-BiAttn	**3.1**	**12.1**	**3.3**

Table 4: Attribute capturing score with different methods of Attribute Fusion.

Model	Aspect (%)	User Category (%)
Baseline	63.0	71.9
Attr-S (Both)	74.0	**86.3**
Attr-T (Both)	72.8	83.3
Attr-A (Both)	**74.6**	86.0

Human Evaluation We conduct a human evaluation on a randomly sampled subset of the test set. Each instance contains an input product title, a description generated by the baseline and

Table 5: Model ablations on 3 model components.

Model	BLEU	Diversity (n=4) ($\times 10^6$)
KOBE	7.6	**15.1**
- knowledge	**8.2**	11.1
- user category	7.6	9.6
- aspect	7.2	7.8

Table 6: The gain from the proposed model *KOBE* over the baseline evaluated by pairwise human judgments.

Model	Fluency	Diversity	Overall Quality
Baseline	3.78	3.73	3.67
KOBE	**3.95**	**3.79**	**3.80**

a description generated by *KOBE*. The selected instances are distributed to human annotators with no prior knowledge about the systems of the descriptions. Following Li et al. [20], we require them to independently score the descriptions by three criteria, fluency, diversity and overall quality. The score of fluency reflects how fluent the description is. The score of diversity reflects how diversified the description is and whether it contains much competitive content. The score of overall quality reflects whether the description is reasonable, or to say, whether it is consistent with world knowledge. The score range is from 1 to 5. The higher the score is, the more consistent with the criterion the description is. The results are shown in Table 6. Consistent with the results of the automatic evaluation of diversity, *KOBE* outperforms the baseline by +0.06 in terms of diversity in human evaluation. However, the improvement in diversity does not sacrifice fluency. Instead, *KOBE* significantly outperforms the baseline by +0.17 in terms of fluency. As to the overall quality, *KOBE* still demonstrates a clear advantage over the baseline by +0.13, showing that our model is able to generate more reasonable descriptions with the introduction of external knowledge.

4.5 Case Studies

In this section, we perform case studies to observe how our proposed methods influence the generation so that the model can generate different contents based on different conditions.

Table 7 presents the generated examples by the models with different aspects. We compare the description generated by models conditioned on two aspects, "aspect" and "function". The product description in the left column contains the content about the appearance of the product, such as the description of the pattern of lotus, but the product description in the right column focuses more on the functions with some expressions about its functional features, such as "transparency", "light", etc.

We also observe that it can also generate different descriptions based on different user categories. It has the potential to generate texts of multiple styles. In Table 8, it can be found that the two user categories, representing two extremely different styles, can make a difference in the generation. With the category "housewife", the generated description focuses on the quality and safety of the product. But with the category "geek", it describes the same desk as a "computer" desk and focuses more on its functional features.

5 RELATED WORK

Neural networks have been widely used for automatic text generation. In this section, we briefly review related literature.

Text generation A basic model for text generation is the attention-based Seq2Seq model [2, 22, 35]. The attention-based Seq2Seq model has demonstrated to be effective in a number of tasks of text generation, including neural machine translation [2, 35, 40], abstractive text summarization [4, 27, 30], dialogue generation [3, 33, 38], etc. Generally speaking, the Seq2Seq model has become one of the most common frameworks for text generation.

While most of the Seq2Seq models are based on RNN, recently researchers have proposed frameworks based on CNN [8] and attention mechanism [37]. The Transformer has achieved state-of-the-art results in neural machine translation and rapidly become a popular framework for sequence-to-sequence learning due to its outstanding performance and high efficiency. [6] proposed BERT, a pretrained language model based on the Transformer, has achieved the state-of-the-art performances on 11 natural language processing tasks. Following the studies, we implement our methods upon the Transformer framework.

Product description generation As for product description generation, previous studies focused on statistical frameworks such as [39], which incorporates statistical methods with the template for the generation of product descriptions. Gerani et al. [9] also generates summarization of product reviews by applying a template-based NLG framework. Such methods are limited by the hand-crafted templates. To alleviate the limitation, researchers adopted deep learning models and introduce diverse conditions to the generation model. Lipton et al. [21] proposed a method to generate reviews based on the conditions of semantic information and sentiment with a language model, and Tang et al. [36] conditioned their generation on discrete and continuous information. Related literature can be also found in [1, 7, 11, 12, 18]. To the best of our knowledge, our research takes the first attempt to use neural methods and sequence-to-sequence learning for product description generation by considering personalization and informativeness.

6 CONCLUSION

In this paper, we study an interesting problem on how to automatically generate personalized product descriptions in an e-commerce platform. We propose a novel model based on the Transformer framework which incorporates multiple types of information efficiently, including product aspects, user categories, as well as the knowledge base. Our extensive experiments show that our method outperforms the baseline models through a series of evaluations, including automatic evaluation and human evaluation. We have successfully deployed the proposed framework onto Taobao, one of the world's largest online e-commerce platforms. A large volume dataset for product description generation, namely *TaoDescribe*, has been generated and annotated, which can be used as a benchmark dataset for future research in this field.

Table 7: Each pair of descriptions is generated by varying the aspect attribute while fixing the product name as input.

Varying description aspects *"appearance"*	*"function"*
The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly. The hand-carved pattern of lotus is lifelike and full of Chinese-style elements.	The Chinese-style lamp is made of superior resin. It is healthy, environmentally-friendly, durable. It is also with good transparency and its light is soft, which is suitable for many situations.

Table 8: Each pair of descriptions is generated by varying the user category attribute while fixing the product name as input.

Varying user categories *"housewife"*	*"geek"*
The Euro-style children's desk is made of superior wood, which is hard, solid and durable. It is covered with environmentally friendly oil paint, which is healthy and safe.	The Euro-style wooden computer desk is made of superior rubberwood, which is hard. It has a clear texture and solid structure and it is covered with environmentally friendly oil paint, which is healthy and safe. It has a multifunctional design for storage, which can meet your requirements for gathering.

ACKNOWLEDGMENTS

The work is supported by the NSFC for Distinguished Young Scholar (61825602), NSFC (61836013), and a research fund supported by Alibaba.

REFERENCES

[1] Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2018. Affective Neural Response Generation. In *ECIR'18*. Springer, 154–166.

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

[3] Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* (2016).

[4] Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *NAACL'16*. 93–98.

[5] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H. Chi. 2018. Q&R: A Two-Stage Approach Toward Interactive Recommendation. In *KDD'18*. 139–148.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Jessica Ficler and Yoav Goldberg. 2017. Controlling Linguistic Style Aspects in Neural Language Generation. In *Proceedings of the Workshop on Stylistic Variation*. 94–104.

[8] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML'17*. 1243–1252.

[9] Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1602–1613.

[10] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A Knowledge-Grounded Neural Conversation Model. (2017). arXiv:1702.01932 http://arxiv.org/abs/1702.01932

[11] Jonathan Herzig, Michal Shmueli-Scheuer, Tommy Sandbank, and David Konopnicki. 2017. Neural response generation for customer service based on personality traits. In *INLG'17*. 252–256.

[12] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955* (2017).

[13] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[14] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-Attribute Text Rewriting. In *ICLR'19*. https://openreview.net/forum?id=H1g2NhC5KQ

[17] Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *ACL'98*. Association for Computational Linguistics, 704–710.

[18] Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. In *ACL'16*, Vol. 1. 994–1003.

[19] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep Reinforcement Learning for Dialogue Generation. In *EMNLP'16*. 1192–1202.

[20] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial Learning for Neural Dialogue Generation. In *EMNLP'17*. 2157–2169.

[21] Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *arXiv preprint arXiv:1511.03683* (2015).

[22] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP'15*. 1412–1421.

[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[24] Paul Michel and Graham Neubig. 2018. Extreme Adaptation for Personalized Neural Machine Translation. *arXiv preprint arXiv:1805.01817* (2018).

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[26] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML'10*. 807–814.

[27] 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 *CoNLL'16*. 280–290.

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL'02*. 311–318.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[30] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP'15*. 379–389.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *ACL'16*. 35–40.

[32] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).

[33] Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017. Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation. In *AAAI'17*. 3288–3294.

[34] Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. *Thulac: An efficient lexical analyzer for chinese.* Technical Report. Tsinghua University.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS'14*. 3104–3112.

[36] Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900* (2016).
[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS'17*. 6000–6010.
[38] Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* (2015).
[39] Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A Statistical Framework for Product Description Generation. In *IJCNLP'17*, Vol. 2. 187–192.
[40] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).
[41] Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Yanghua Xiao. 2017. CN-DBpedia: A never-ending Chinese knowledge extraction system. In *IEA/AIE'19*. Springer, 428–438.

A APPENDIX

A.1 Details on the Experimental Setup and Hyperparameters

Hyper-parameter Configurations We use a six-layer Transformer Encoder for both the title encoder and the knowledge encoder and a two-layer Transformer Decoder for the description decoder. All input embedding dimensions and hidden sizes are set to $d_{model} = 512$. Following [37], we empirically set the size of the Transformer FFN inner representation size ($d_{ff} = 2048$) to four times of the embedding size ($d_{model} = 512$). We use ReLU [26] as the activation function for all models. During training, the batch size is set to 64. We use Adam optimizer [15] with the setting $\beta_1 = 0.9, \beta_2 = 0.998$ and $\epsilon = 1 \times 10^{-9}$. The learning rate is set to 1×10^{-4}. Gradient clipping is applied with range $[-1, 1]$. We also applied a dropout rate of 0.1 as regularization. At the inference stage, we use beam search with a beam size of 10.

Hardware Configuration The experiments are conducted on a Linux server equipped with an Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz, 512GB RAM and 8 NVIDIA V100-SXM2-16GB GPUs. Our full set of experiments took about 3 days with this multi-GPU setting. We expect that a consumer-grade single-GPU machine (e.g., with a Titan X GPU) could complete our main experiments, including *KOBE* and the baseline, in 4 days.

Software All models are implemented in PyTorch [29] version 0.4.1 and Python 3.6. We'll soon release our code for preprocessing the dataset and running all the experiments.

A.2 Additional Details on the Dataset

In this section, we provide some additional, relevant dataset details. Our code and the **TaoDescribe** dataset are available at **https://github.com/qibinc/KOBE**.

Data Preprocessing We observe that the samples are noisier if the product title or the product description is too long. For this reason, we discarded pairs with a title longer than 100 tokens or with a description longer than 150 tokens. We then substituted tokens that appeared less than 5 times in our dataset with the <UNK> token. Furthermore, a product title $\boldsymbol{x}$ in our dataset may be coupled with multiple descriptions $\boldsymbol{y}$ (e.g. covering different aspects, targeting different users, or written by different people). We split our dataset by products to keep product titles in the test set from being seen during training. The dataset is finally split into a training set with 2,114,034 instances, a validation set with 9,917 instances and a test set with 5,236 instances. The vocabulary size is 5,428 for product titles and 9,917 for descriptions. The average lengths of product titles and descriptions are 31.4 and 90.2 tokens, respectively.

Table 9: Examples of adjectives in different product aspects.

Aspect	Num.	Examples
appearance	754	contracted, elegant, monochrome
texture	235	soft, gentle, smooth, comfortable
function	265	household, convenient, automatic

Aspects Selection and Annotation Based on the statistics of the dataset and empirical support by domain experts, we set the number of product aspects $|\mathcal{A}_1|$ to three. We introduce the heuristic method used to extract the aspects from the description in detail and demonstrate its effectiveness as follows.

(1) We extract all the descriptions for the products in the dataset and run word2vec [25] on the description sentences to obtain the embedding for each word in the description vocabulary.
(2) For each adjective in the description vocabulary,[4] we compute the cosine distance between its embedding and the embedding of each of the $|\mathcal{A}_1|$ aspect words respectively and obtain similarity scores $\{s_1, s_2, \ldots, s_{|\mathcal{A}_1|}\}$.
(3) We discard the adjective if $\max\{s_1, s_2, \ldots, s_{|\mathcal{A}_1|}\} < \gamma \sum_{k=1}^{|\mathcal{A}_1|} s_k$,[5] which means the adjective cannot be categorized to any aspect. Each remaining adjective in the description vocabulary is then assigned to the aspect with the highest similarity score.
(4) We then automatically annotate each description $\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$ in the dataset with an aspect a_1, based on the semantic distance between the description and the set of adjectives belonging to the aspect.

We give examples and the number of obtained adjectives for the three aspects in Table 9. Furthermore, we visualize the word embeddings of adjectives in the three aspects in Figure 4.[6] As shown in the figure, adjectives belonging to different aspects are mostly well separated. This means different aspects can be distinguished in the semantic space and further supports our selection of the three aspects.

User Categories Collection In our platform, there are 41 predefined "interest tags" based on expert knowledge and statistics of user behaviors. These interest tags are mapped to "product categories". Each user is then soft assigned to these tags, based on his/her browsing, favoring and purchasing history on different product categories. Each description can then be assigned to a user category, according to the major group of users that have clicked or stayed long enough on this description. Ideally, the number of user categories should be the same as the number of interest tags. In practice, we find that user categories with a low appearing frequency can cause some noise during training, for there are too

[4] We use THULAC [34] for part-of-speech tagging.
[5] We empirically set γ to 0.8.
[6] 128 adjectives for each aspect are displayed for clear visualization.

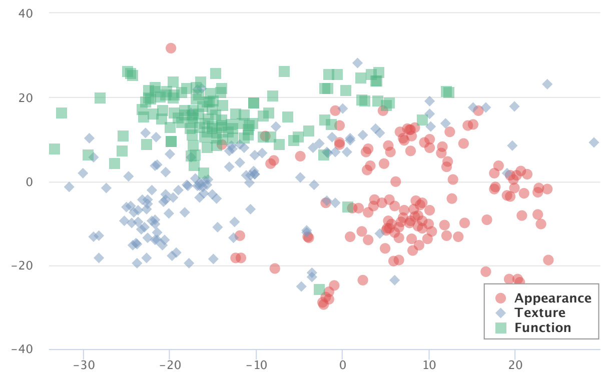

Figure 4: Word embeddings of adjectives in the three aspects projected onto a 2D plane by t-SNE [23]. Each color represents an aspect and each point represents an adjective in the description vocabulary.

Table 10: Examples of user categories.

User Category	Frequency
fashion girl	34,916
housewife	29,626
foodie	17,049
geek	15,882
...	...

few corresponding descriptions. To solve this problem, we replace user categories appearing less than 5 times with the <UNK> token. Finally, the number of obtained user categories $|\mathcal{A}_2|$ is 24. Table 10 lists examples of user categories and their frequency in $\mathcal{Z}$.

User Categories Annotation To overcome the problem that user feedbacks on $\mathcal{Y}$ are very sparse, we collect another dataset $\mathcal{Z}$ and train a user category classifier on $\mathcal{Z}$ to annotate $\mathcal{Y}$. In addition to $\mathcal{Y}$ which only contains single product descriptions, we make $\mathcal{Z}$ to incorporate descriptions of products and user feedbacks. We then discard descriptions with the number of user feedbacks (clicks) less than 100 and obtain 143,154 descriptions and their corresponding user categories. We find that this increase in data size significantly increases the generalization ability of the trained classifier, despite the slight difference between $\mathcal{Z}$ and $\mathcal{Y}$.

Now we introduce the architecture and training procedure of the user category classifier M. Due to the massive scale of our dataset, the classifier used to annotate the user category for the descriptions should be both effective and efficient. We applied fastText [13] as the text classifier but soon find that it is prone to overfitting. The performance heavily relies on carefully setting the number of training epochs to stop early. For this reason, we adopt convolutional neural networks (CNN) for sentence classification [14] as the user category classifier M. We clarify the architecture and configuration of M as follows.

Given the input sentence $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$, we first embed each word into d dimensions and obtain $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$, $\boldsymbol{e} \in \mathbb{R}^{h \times d}$. Then, a feature c_i is generated by applying a filter $\boldsymbol{w} \in \mathbb{R}^{h \times d}$ on a window of h words $\boldsymbol{e}_{i:i+h-1}$. A feature map $\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-h+1}]$ is then obtained by applying the filter $\boldsymbol{w}$ to each possible window of words in the sentence $\{\boldsymbol{e}_{1:h}, \boldsymbol{e}_{2:h+1}, \ldots, \boldsymbol{e}_{n-h+1:n}\}$. The feature $\hat{c}$ corresponding to this particular filter is finally obtained by a max-pooling operation over the feature map $\hat{c} = \max\{\boldsymbol{c}\}$. Finally, features computed by convolutional filters are projected to the user category space through a fully connected layer. The model is then trained by maximizing the likelihood $P(a|\boldsymbol{z}; \theta_M)$ of the correct user category a, where θ_M denotes the parameters of the classifier M.

For the classifier, we use word embedding size $d = 100$ and apply three types of filters with window sizes 3, 4 and 5, each with 100 filters. During training, the batch size is set to 64. We apply the cross entropy loss and an Adam optimizer with the default setting $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. The learning rate is set to 1×10^{-3}. We also applied a dropout rate of 0.5 to prevent overfitting. The testing classification accuracy for 24 user categories on $\mathcal{Z}$ is 81.0%. Finally, the classifier M trained on $\mathcal{Z}$ is used to annotate the user category for each description $y \in \mathcal{Y}$. We also provide $\mathcal{Z}$ along with the dataset for full reproducibility.