Stick to Facts: Towards Fidelity-oriented Product Description Generation

Zhangming Chan^{1,2,*,†}, Xiuying Chen^{1,2,*}, Yongliang Wang³, Juntao Li^{1,2},

Zhiqiang Zhang³, **Kun Gai**³, **Dongyan Zhao**^{1,2} and **Rui Yan**^{1,2,‡}

¹ Center for Data Science, AAIS, Peking University

² Wangxuan Institute of Computer Technology, Peking University

³ Alibaba Group, Beijing

{zhangming.chan,xy-chen,ljt,zhaody,ruiyan}@pku.edu.cn
{yongliang.wyl,zhang.zhiqiang,jingshi.gk}@alibaba-inc.com

Abstract

Different from other text generation tasks, in product description generation, it is of vital importance to generate faithful descriptions that stick to the product attribute information. However, little attention has been paid to this problem. To bridge this gap we propose a model named Fidelity-oriented Product Description Generator (FPDG). FPDG takes the entity label of each word into account, since the product attribute information is always conveyed by entity words. Specifically, we first propose a Recurrent Neural Network (RNN) decoder based on the Entity-labelguided Long Short-Term Memory (ELSTM) cell, taking both the embedding and the entity label of each word as input. Second, we establish a keyword memory that stores the entity labels as keys and keywords as values, and FPDG will attend to keywords through attending to their entity labels. Experiments conducted a large-scale real-world product description dataset show that our model achieves the state-of-the-art performance in terms of both traditional generation metrics as well as human evaluations. Specifically, FPDG increases the fidelity of the generated descriptions by 25%.

1 Introduction

The effectiveness of automatic text generation has been proved in various natural language processing applications, such as neural machine translation (Luong et al., 2015; Zhou et al., 2017), dialogue generation (Tao et al., 2018a; Hu et al., 2019), and abstractive text summarization (Chen et al.; Gao et al., 2019). One such task, product description generation has also attracted considerable attention (Lipton et al., 2015; Wang et al.,

Input	high-waist; straight; jeans; blue; ZARA
Bad output	This UNIQLO low-waist jeans are suitable for the curvature of the waistline and will not slip when worn for a long time. The straight version and black color together makes you even slimmer.
Good output	This <u>blue jeans</u> look personalized and chic, making you more distinctive. With a <u>high-waist</u> design, the legs look even more slender, and the whole person looks taller. The <u>straight</u> design hides the fat in the legs and is visually slimmer.

Table 1: Example of product description generation. The text with underline demonstrates faithful description, and text with deleteline demonstrates wrong description.

2017; Zhang et al., 2019a). An accurate and attractive description not only helps customers make an informed decision but also improves the likelihood of purchase. Concretely, the product description generation task takes several keywords describing the attributes of a product as input, and then outputs fluent and attractive sentences that highlight the feature of this product.

There is one intrinsic difference between product description generation and other generation tasks, such as story or poem generation (Li et al., 2018; Yao et al., 2019); the generated description has to be faithful to the product attributes. An example case is shown in Table 1, where the good product description matches the input information, while the bad description mistakes the brand and the style of the jeans. Our preliminary study reveals that 48% of the outputs from a state-ofthe-art sequence-to-sequence system suffer from this problem. In the real-world e-commerce product description generation system, generating text with unfaithful information is unacceptable. A wrong brand name will damage the interests of advertisers, while a wrong product category might break the law by misleading consumers. Any kind of unfaithful description will bring huge economic

^{*}Equal contribution. Ordering is decided by a coin flip.

[†]Contribution during internship at Alibaba Group.

[‡] Corresponding author.

losses to online platforms.

Though of great importance, no attention has been paid to this problem. Existing product description generators include (Chen et al., 2019; Zhang et al., 2019b), where they focus on generating personalized descriptions and patterncontrolled descriptions, respectively.

In this paper, we address the fidelity problem in generation tasks by developing a model named Fidelity-oriented Product Description Generator (FPDG), which takes keywords about the product attributes as inputs. Since product attribute information is always conveyed by entity words (up to 89.72% in input keywords), FPDG generates faithful product descriptions by taking into account the entity label of each word. Specifically, first, we propose an Entity-label-guided Long Short-Term Memory (ELSTM) as the cell in the decoder RNN, which takes the entity label of an input word, as well as the word itself as input. Then, we establish a keyword memory storing the input keywords with their entity labels. In each decoding step, the current hidden state of the ELSTM focuses on proper words in this keyword memory in regard to their entity categories. We also collect a largescale real-world product description dataset from one of the largest e-commerce platforms in China. Extensive experiments conducted on this dataset show that FPDG outperforms the state-of-the-art baselines in terms of traditional generation metrics and human evaluations. Specifically, FPDG greatly improves the fidelity of generated description by 24.61%.

To our best knowledge, we are the first to explore the fidelity problem of product description generation. Besides, to tackle this problem, we propose an ELSTM and a keyword memory to incorporate entity label information, so as to generate more accurate descriptions.

2 Related Work

We detail related work on text generation, entityrelated generation, and product description.

Text generation. Recently, sequence-tosequence (Seq2Seq) neural network models have been widely used in NLG approaches. Their effectiveness has been demonstrated in a variety of text generation tasks, such as neural machine translation (Luong et al., 2015; Bahdanau et al., 2014; Wu et al., 2016), abstractive text summarization (See et al., 2017a; Hsu et al., 2018; Chen and Bansal, 2018), dialogue generation (Tao et al., 2018a; Xing et al., 2017), etc. Along another line, there are also works based on an attention mechanism. Vaswani et al. (2017) proposed a Transformer architecture that utilizes the self-attention mechanism and has achieved state-of-the-art results in neural machine translation. Since then, the attention mechanism has been used in a variety of tasks (Devlin et al., 2018; Fan et al., 2018; Zhou et al., 2018).

Entity-related generation. Named entity recognition (NER) is a fundamental component in language understanding and reasoning (Greenberg et al., 2018; Katiyar and Cardie, 2018). In (Ji et al., 2017), they proved that adding entity related information can improve the performance of language modeling. Building upon this work, in (Clark et al., 2018), they combined entity context with previous-sentence context, and demonstrated the importance of the latter in coherence test. Another line of related work generates recipes using neural networks to track and update entity representations (Bosselut et al., 2018). Different from the above works, we utilize entity labels as supplementary information to assist decoding in the text generation task.

Product descriptions. Quality product descriptions are critical for providing a competitive customer experience in an e-commerce platform. Due to its importance, automatically generating the product description has attracted considerable interests. Initial works include (Wang et al., 2017), which incorporates statistical methods with the template to generate product descriptions. With the development of neural networks, (Chen et al., 2019) explored a new way to generate personalized product descriptions by combining the power of neural networks and a knowledge base. (Zhang et al., 2019b) proposed a pointer-generator neural network to generate product description whose patterns are controlled.

In real-world product description generation application, however, the most important prerequisite is the fidelity of generated text, and to the best of our knowledge, no research has been conducted on this so far.

3 Problem Formulation

FPDG takes a list of keywords $X = \{x_1, ..., x_{T_X}\}$ as inputs, where T_X is the number of keywords. These keywords are all about the important at-



Figure 1: Overview of FPDG. Green denotes an entity label and purple denotes a word. We divide our model into two components: (1) The *Keyword Encoder* stores the word and its entity label in the token memory, and uses Self-Attention Modules (SAMs) to encode words and entity labels; (2) The *Entity-based Generator* generates product description based on the token memory and SAM encoders.

tributes of the product. The goal of FPDG is to generate a product description $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_{T_{\hat{Y}}}\}$ that is not only grammatically correct but also consistent with the input information, such as the brand name and the fashion style. Essentially, FPDG tries to optimize the parameters to maximize the probability $P(Y|X) = \prod_{t=1}^{T_Y} P(y_t|X)$, where $Y = \{y_1, ..., y_{T_Y}\}$ is the ground truth answer.

4 Model

In this section, we introduce our *Fidelity-oriented Product Description Generator* (FPDG) model in detail. An overview of FPDG is shown in Figure 1 and can be split into two modules:

(1) *Keyword Encoder* (See \S 4.1): We first use a key-value memory to store the entity-label as key and the corresponding word as value. To better learn the interaction between words, we employ two self-attention modules from Transformer (Vaswani et al., 2017) to model the input keywords and their entity-labels separately.

(2) Entity-based Generator (See \S 4.2): We propose a recurrent decoder based on Entity-label-guided LSTM (ELSTM) to generate the product description.

4.1 Keyword Encoder

In the product description dataset, most of the input keywords are entity words (up to 89.72%), and the description text should be faithful to these entity words. Hence, we incorporate entity label information to improve the accuracy of the generated text. In this section, we introduce how to embed input keywords with entity label information.

To begin with, we use an embedding matrix e to map a one-hot representation of each word in

 x_i into a high-dimensional vector space. Since our input keywords have no order information, we use the Self-Attention Module (SAM) from Transformer (Vaswani et al., 2017) to model the temporal interactions between the words, instead of RNN-based encoder. We use three fullyconnected layers to project $e(x_i)$ into three spaces, <u>i.e.</u>, the query $q_i = F_q(e(x_i))$, the key $k_i =$ $F_k(e(x_i))$ and the value $v_i = F_v(e(x_i))$. The attention module then takes q_i to attend to each k_i , and uses these attention distribution results $\alpha_{i,\cdot} \in$ \mathbb{R}^{T_X} as weights to obtain the weighted sum of v_i , as shown in Equation 2. Next, we add the original word representation $e(x_i)$ on β_i as the residential connection layer, as shown in Equation 3:

$$\alpha_{i,j} = \frac{\exp\left(q_i k_j\right)}{\sum_{n=1}^{T_X} \exp\left(q_i k_n\right)},\tag{1}$$

$$\beta_i = \sum_{j=1}^{T_X} \alpha_{i,j} v_j, \tag{2}$$

$$\hat{h}_i = e(x_i) + \beta_i, \tag{3}$$

where $\alpha_{i,j}$ denotes the attention weight of the *i*-th word on the *j*-th word. Finally, we apply a feed-forward layer on \hat{h}_i to obtain the final word representation h_i :

$$h_i = \max(0, h_i \cdot W_1 + b_1) \cdot W_2 + b_2,$$
 (4)

where W_1, W_2, b_1, b_2 are all trainable parameters. We refer to the above process as:

$$h_i = \text{SAM}(F_q(x_i), F_k(x_{\cdot}), F_v(x_{\cdot})).$$
 (5)

In the meantime, we leverage the e-commerceadapted AliNER¹ to label each word in the input such as "brand name" and "color". Non-entity words are labeled as "normal word". We denote c_i

¹https://ai.aliyun.com/nlp/ner



Figure 2: The structure of ELSTM, which is a hybrid of three LSTMs.

as the entity label for the *i*-th word. To better learn the interaction between these entity labels, we use a second SAM, taking c. as input, and obtain the final representation for c_i as m_i :

$$m_i = \text{SAM}(F_q(c_i), F_k(c_{\cdot}), F_v(c_{\cdot})).$$
 (6)

We also propose a key-value keyword memory that stores the label results as shown in Figure 1. The keys in the keyword memory are T_C representations of different entity-labels, where T_C is the number of entity categories. The values in the memory are self-attention representations of words that belong to each category. We denote the k-th entity-label in the keyword memory as c^k , and the i-th word belonging to this category as $e(x_i^k)$. This keyword memory will be incorporated into the generation process in §4.2.3.

4.2 Entity-based Generator

To incorporate entity label information into generation process, we propose a modified version of LSTM, named Entity-label-guided Long Short-Term Memory (ELSTM). We first introduce EL-STM and then introduce the RNN decoder based on ELSTM.

4.2.1 ELSTM

As shown in Figure 3, ELSTM consists of three LSTMs, two Word-LSTMs and one Entity-label-LSTM. The hidden states of the two Word-LSTMs are integrated together, forming w_t^h , while the hidden state of Entity-Label-LSTM is l_t^h . The word hidden state w_t^h attends to the SAM outputs to predict the next word, while the entity label hidden state l_t^h is used to attend to the keyword memory and predict entity label of next word. In other

words, ELSTM is a hybrid of three LSTMs with a deeper interaction between these cells. Overall, ELSTM takes two variables as input, the embedding of an input word $e(y_t)$ and its entity label m_t .

The structure of each LSTM in ELSTM is the same as original LSTM, thus, is omitted here due to space limitations. Next, we introduce the interaction in ELSTM in detail. As shown in Figure 2, Word-LSTM₀ takes the $e(y_t)$ as input, and outputs the initial word hidden state $w_{t+1}^{h'}$:

$$w_{t+1}^{h'} = \text{Word-LSTM}_0(w_t^h, e(y_t)).$$
(7)

Next we calculate the entity label hidden state by Entity-Label-LSTM, taking both m_t and $w_{t+1}^{h'}$ as input:

$$l_{t+1}^{h'} = \text{Entity-Label-LSTM}(l_t^h, m_t + w_{t+1}^{h'}).$$
(8)

To further improve the accuracy of entity label prediction, we also apply another multi-layer projection, incorporating entity label context vector c_t^m (introduced in §16) to obtain polished entity hidden state l_{t+1}^h :

$$l_{t+1}^{h} = FC([l_{t+1}^{h'}, c_t^m]), \tag{9}$$

where [,] denotes the concatenation operation.

Finally, Word-LSTM₁ takes the entity-label hidden state l_{t+1}^h and word embedding $e(y_t)$ as input, and outputs predicted word hidden state that contains entity label information:

$$w_{t+1}^{h'_1} = \text{Word-LSTM}_1(w_t^h, e(y_t) + l_{t+1}^h).$$
 (10)

Note that it is possible that the predicted entitylabel hidden state is not accurate enough. Hence, we apply a gate fusion combining $w_{t+1}^{h'_1}$ with initial word hidden state $w_{t+1}^{h'}$ to ensure the word hidden state quality:

$$\gamma = \sigma(FC([w_{t+1}^{h'_1}, w_{t+1}^{h'_1}])), \quad (11)$$

$$w_{t+1}^{h} = \gamma w_{t+1}^{h'_{1}} + (1 - \gamma) w_{t+1}^{h'}.$$
 (12)

The fusion result is w_{t+1}^h , *i.e.*, the updated word hidden state.

In this way, the entity label information and word information are fully interacted in ELSTM cell, while the hidden states are updated.



Figure 3: An overview of the description generator.

4.2.2 ELSTM-based Attention Mechanism

Established on ELSTMs, we now have a new RNN decoder to generate descriptions, incorporating the two SAM encoders and the keyword memory. First, we apply an Bi-RNN to encode input keywords, and use its last hidden state as decoder initial hidden states, <u>i.e.</u>, w_0^h and l_0^h . The *t*-th decoding step is calculated as:

$$\begin{bmatrix} w_{t+1}^h \\ l_{t+1}^h \end{bmatrix} = \text{ELSTM}\left(\begin{bmatrix} w_t^h \\ l_t^h \end{bmatrix}, (e(y_t), m_t) \right).$$
(13)

Next, similar to the traditional attention mechanism from (Bahdanau et al., 2015), we summarize the input word representation $e(y_{.})$ and entity label representation $m_{.}$ into the word context vector c_t^w and entity label vector c_t^m , respectively. As for how to obtain c_t^w and c_t^m , we use the two hidden states in ELSTM, <u>i.e.</u>, w_t^h and l_t^h to attend to $h_{.}$ and $m_{.}$, respectively. Specifically, the entity label context vector c_t^m is calculated as:

$$\hat{\gamma}_{t,i} = W_a \tanh\left(W_b l_t^h + W_h h_i\right),\qquad(14)$$

$$\gamma_{t,i} = \exp\left(\gamma_{t,i}\right) / \sum_{j=1}^{I_X} \exp\left(\gamma_{t,j}\right), \qquad (15)$$

$$c_t^m = \sum_{i=1}^{T_X} \gamma_{t,i} m_i.$$
 (16)

The decoder state l_t^h is used to attend to each entity label representation m_i , resulting in the attention distribution $\gamma_t \in \mathbb{R}^{T_x}$, as shown in Equation 15. Then we use the attention distribution γ_t to obtain a weighted sum of the entity label representations m_{\cdot} as the word context vector c_t^m , shown in Equation 16. c_t^w is obtained in a similarity by using hidden state w_t^h attending to $e(y_{\cdot})$, and we omit the details for brevity. This two context vectors play different parts, and is introduced in §4.2.4.

4.2.3 Incorporating Keyword Memory

So far, we have finished calculating the context vector. Next, we describe how to incorporate the guidance from the keyword memory. We first use the entity label hidden state to attend to the entity label keys in the keyword memory by gumbel softmax (Jang et al., 2016) (Equation 17), and then use the attention weights to obtain the weighted sum of self-attention representation of values in the memory (Equation 19):

$$\pi_t^{k'} = \frac{\exp((l_t^h W_e c^k + p^k)/\tau)}{\sum_{j=1}^{T_c} \exp((w_t^h W_e c^j + p^k)/\tau)},$$

$$v_i^{k'} = \text{SAM}\left(F_q(e(x_i^k), F_k(e(x_\cdot^k), F_v(e(x_\cdot^k))), (18))\right),$$
(18)

$$= \sum_{k=1}^{T_C} \left(\pi_t^k \sum_{i=1}^{T_{ck}} (v_i^{k'}) \right), \tag{19}$$

where $p^k = -\log(-\log(g^k)), g^k \sim U(0, 1), \tau \in [0, 1]$ is the softmax temperature. T_{c_k} denotes the number of words in the input keywords that belong to the k-th entity category. We choose gumbel-softmax instead of regular softmax because a generated word can only belong to one entity category. In this way, FPDG first predicts the entity label of the predicted word and then uses o'_{t+1} to store the information of words that belong to this category.

4.2.4 Projection Layers

 o_{t+1}'

Next, using a fusion gate g_t , o'_{t+1} is combined with word hidden state c^w_{t+1} in a similar way in Equation 12 to obtain o_{t+1} . Finally, we obtain the final generation distribution P_v over vocabulary:

$$P_{t+1}^{v} = \text{softmax}\left(W_{v}[o_{t+1}, w_{t+1}^{h}] + b_{v}\right). \quad (20)$$

We concatenate the memory vector o_{t+1} , the word context vector c_{t+1}^w , and the output of the decoder ELSTM w_{t+1}^h as the input of the output projection layer.

Apart from predicting the next word, we also use the entity label hidden state l_t^h to predict the entity label of the next word as an auxiliary task. The distribution P_{t+1}^e over entity categories is calculated as:

$$P_{t+1}^e = \operatorname{softmax}(W_e[l_{t+1}^{h'}, c_{t+1}^m] + b_e) \qquad (21)$$

We use negative log-likelihood as loss function:

$$\mathcal{L} = -(\sum_{t=1}^{T_{\hat{Y}}} \log P_v(y_t) + \lambda \cdot \sum_{t=1}^{T_{\hat{Y}}} \log P_e(m_t)),$$
(22)

where λ is the weight of entity label prediction loss. The gradient descent method is employed to update all parameters and minimize this loss function.

5 Experimental Setup

5.1 Research Questions

We list four research questions that guide the experiments: **RQ1** (See § 6.1): What is the overall performance of FPDG? Are generated descriptions faithful? **RQ2** (See § 6.2): What is the effect of each module in FPDG? **RQ3** (See § 6.3): How does the keyword memory work in each decoding step? **RQ4** (See § 6.4): Can ELSTM successfully capture entity label information?

5.2 Dataset

We collect a large-scale real-world product description dataset from one of the largest ecommerce platforms in China². The inputs are keywords about the product attributes selected by sellers on the platform, and the product descriptions are written by professional experts that are good at marketing. Overall, there are 404,000 training samples, and 5,000 validation and test samples. On average, there are 10 keywords in the input, and 63 words in the product description. 89.72% inputs are entity words.

5.3 Evaluation Metrics

Following (Fu et al., 2017), we use the evaluation package of (Chen et al., 2015), which includes BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L. BLEU is a popular machine translation metric that analyzes the co-occurrences of n-grams between the candidate and reference sentences. METEOR is calculated by generating an alignment between the words in the candidate and reference sentences, with an aim of 1:1 correspondence. ROUGE-L is a metric designed to evaluate text summarization algorithms.

(Tao et al., 2018b) notes that only using the BLEU metric to evaluate text quality can be misleading. Therefore, we also evaluate our model by human evaluation. Three highly educated participants are asked to score 100 randomly sampled summaries generated by Pointer-Gen and FPDG. We chose Pointer-Gen since its performance is relatively higher than other baselines. The statistical significance of observed differences between the performance of two runs are tested using a twotailed paired t-test and is denoted using \blacktriangle (or \checkmark) for strong significance for $\alpha = 0.01$.

5.4 Comparison Methods

We first conduct an ablation study to prove the effectiveness of each module in FPDG. Model w/o ELSTM is implemented with only keys in the memory because there are no entity representations as values. Then, to evaluate the performance of our proposed dataset and model, we compare it with the following baselines:

(1) **Seq2Seq:** The sequence-to-sequence framework (Sutskever et al., 2014) is one of the initial works proposed for the language generation task.

(2) **Pointer-Gen**: A sequence-to-sequence framework with pointer and coverage mechanism proposed in (See et al., 2017b).

(3) **Conv-Seq2seq**: A model combining the convolutional neural networks and the sequence-to-sequence network (Gehring et al., 2017).

(4) **FTSum**: A faithful summarization model proposed in (Cao et al., 2018), which leverages open information extraction and dependency parse technologies to extract actual fact descriptions from the source text.

(5) **Transformer**: A network architecture that solely based on attention mechanisms, dispensing with recurrence and convolutions entirely (Vaswani et al., 2017).

(6) **PCPG**: A pattern-controlled product description generation model proposed in (Zhang et al., 2019b). We adapt the model for our scenario.

To verify whether the performance improvement is obtained by adding additional entity label inputs, we directly concatenate the word embedding with the entity embedding as input for these baselines, denoted as "with Entity Embedding" in Table 2.

5.5 Implementation Details

We implement our experiments in Pytorch³ on Tesla V100 GPUs⁴. The word and entity-label embedding dimensions are set to 256. The number of hidden units and the entity-label hidden size are also set to 256. All inputs were padded with zeros to a maximum keyword number of the batch. There are 36 categories of entity labels together.

³https://pytorch.org/

⁴Our model will be incorporated into Chuangyi-Taobao (https://chuangyi.taobao.com/pages/ smartPhrase).

²https://www.taobao.com/

Models	BLEU Metric				METEOR	ROUGE-L
	BLEU-1	BLEU-2	BLEU-3	BLEU-4		ROUGLE
Seq2seq	30.10	13.61	7.564	4.824	14.08	25.04
with Entity Embedding	30.46	13.77	7.722	5.012	14.09	24.96
Pointer-Gen	30.66	13.84	7.933	5.278	14.10	24.93
with Entity Embedding	30.92	13.92	7.988	5.219	14.14	24.85
Conv-Seq2seq	28.22	12.15	6.393	4.012	13.72	23.92
with Entity Embedding	28.23	12.22	6.401	4.012	13.75	24.03
FTSum	30.33	13.57	7.563	4.689	13.86	24.63
with Entity Embedding	30.47	13.61	7.611	4.937	13.97	24.82
Transformer	29.11	12.77	7.116	4.620	13.95	23.36
with Entity Embedding	29.37	12.64	6.852	4.315	13.88	23.73
PCPG(including Entity)	29.46	13.02	7.259	4.620	13.72	24.50
Our FPDG	32.26	14.79	8.472	5.629	15.17	25.27
w/o KW-MEM	31.85	14.36	8.134	5.351	15.01	25.09
w/o ELSTM	31.41	14.57	8.430	5.630	14.98	25.25

Table 2: RQ1: Comparison between baselines.

 λ is set to 0 in the first 500 steps, and 0.6 in the rest of the training process. We performed minibatch cross-entropy training with a batch size of 256 documents for 15 training epochs. we set the minimum encoding step to 15 and maximum decoding step size to 70. During decoding, we employ beam search with a beam size of 4 to generate a more fluent sentence. It took around 6 hrs on GPUs for training. After each epoch, we evaluated our model on the validation set and chose the best performing model for the test set. We use the Adam optimizer (Duchi et al., 2010) as our optimizing algorithm and the learning rate is 1e-3.

6 Experimental Results

6.1 Overall Performance

For research question **RQ1**, we examine the performance of our model and baselines in terms of BLEU, as shown in Table 2. Firstly, among all baselines, Pointer-Gen obtains the best performance, outperforming the worst baseline Conv-Seq2Seq by 2.44 in BLEU-1. Secondly, directly concatenating the entity label embedding with the word embedding does not bring much help, only leading to an improvement of 0.26 in BLEU-1 for the Pointer-Gen model. Finally, our model outperforms all baselines for all metrics, outperform-

	Fluency	Informativity	Fidelity
Pointer-Gen	2.23	1.84	1.91
FPDG	2.46 ▲	2.19 ▲	2.38▲

Table 3: RQ1: Human evaluation comparison withPointer-Gen baseline.

ing the strongest baseline Pointer-Gen, by 5.22%, 6.86%, 6.79%, and 6.65% in terms of BLEU-1, BLEU-2, BLEU-3, and BLEU-4, respectively.

As for human evaluation, we ask three highly educated participants to rank generated summaries in terms of fluency, informativity, and fidelity. The rating score ranges from 1 to 3 with 3 being the best. The results are shown in Table 3, where FPDG outperforms Pointer-Gen by 10.31% and 19.02% in terms of fluency and informativity, and, specifically, FPDG greatly improves the fidelity value by 24.61%. We also conduct the paired student t-test between our model and Pointer-Gen, and the result demonstrates the significance of the above results. The kappa statistics are 0.35 and 0.49, respectively, which indicates fair and moderate agreement between annotators⁵.

 $^{^{5}}$ (Landis and Koch, 1977) characterize kappa values < 0 as no agreement, 0-0.20 as slight, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial, and 0.81-1 as almost



Figure 4: RQ3: Visualizations of entity-label-attention when generating the word on the left.

6.2 Ablation Study

Next, we turn to research question **RQ2**. We conduct ablation tests on the usage of the keyword memory and ELSTM, corresponding to FPDG w/o KW-MEM and ELSTM, respectively. The ROUGE score result is shown at the bottom of Table 2. Performances of all ablation models are worse than that of FPDG in terms of almost all metrics, which demonstrates the necessity of each module in FPDG. Specifically, ELSTM makes the greatest contribution to FPDG, improving the BLEU-1, BLEU-2 scores by 2.71% and 1.51%.

6.3 Analysis of Keyword Memory

We then address **RQ3** by analyzing the entitylabel-level attention on the keyword memory. Two representative cases are shown in Figure 4. The figure in the above is the attention map when generating the word "toryburch", and the bottom figure is when generating the word "printflower". The darker the color, the higher the attention. Due to limited space, we omit irrelevant entity categories. When generating "toryburch", which is a brand name, the entity-label-level attention pays most attention to the "Brand" entity label, and when generating "flower", which is a style element, it mostly pays attention to "Element". This example demonstrates the effectiveness of the entity-label-level attention.

6.4 Analysis of ELSTM

We now turn to **RQ4**; whether or not the ELSTM can capture entity label information. We examine this question by verifying whether ELSTM can predict the entity label of the generated word. The accuracy of the predicted entity label is calculated in a teacher-forcing style, i.e., each ELSTM takes

```
perfect agreement.
```

Input:正品; Prada;/;普拉达; 男包; 尼龙; 双肩背包; 男 士;休闲;旅行包;电脑包;登山包(authentic; Prada; man bag; leisure; travel bag; laptop bag; backpack) 这款来自prada的拉链双肩包。采用纯色设计, Reference 包面上的字母印花图案十分显眼,令人眼前一 亮,瞬间脱颖而出。开合双拉链处理,使用顺 滑,方便拿取物品。大容量的包包,为物品的 放置提供足够的空间。(This zippered shoulder bag from Prada. With a solid color design, the letter print on the surface of the bag is very conspicuous, making it instantly stand out. Open and close double zipper makes it easy to use take items smoothly. Large-capacity bag provides enough space for the placement of items.) 这款来自prada的翻盖双肩包。采用了翻盖的设 Pointer-Ger 计,具有很好的防盗效果,让你的出行更加安 全可靠。大容量的包型设计,满足日常的生活 所需,让你的出行更加方便快捷。(This flipover bag from Prada. The clamshell design has a good anti-theft effect, making your trip safer and more reliable. The large-capacity package design meets the needs of everyday life, making your travel more convenient and quick. 这款来自prada的纯色双肩包。纯色的设计 FPDG 给人一种大气又干练的感觉,打造时下流行 的简约风格。拉链的开合设计,划拉顺畅, 增加物品安全性。大容量的包型,既能容纳 生活物品,还让携带更加的便捷。(This solid color backpack from Prada. The solid color design gives people an atmosphere and a sense of exquisiteness, creating a simple style that is popular nowadays. The opening and closing design of the zipper smoothes the stroke and increases the safety of the item. Large-capacity package can accommodate living items and make it easier to carry.

Table 4: Examples of the generated answers by Pointer-Gen and FPDG. The text with underline demonstrates faithful description, and text with deleteline demonstrates wrong description.

the ground truth entity label and word as input, and outputs the entity label of the next word. We employ recall at position k in n candidates $(R_n@k)$ as evaluation metrics. Over the whole test dataset, $R_1@36$ is 64.12%, $R_2@36$ is 80.86%, and $R_3@36$ is 94.02%, which means ELSTM can capture the entity label information to a great extent and guide the word generation.

We also show a case study in Table 4. The description generated by Pointer-Gen introduces the Prada bag as a "clamshell bag has a good antitheft effect", which is contrary to the fact. While our model generates the faithful description: "The opening and closing design of the zipper smoothes the stroke".

7 Conclusion and Future Work

In this paper, we explore the fidelity problem in product description generation. To tackle this challenge, based on the consideration that product attribute information is typically conveyed by entity words, we incorporate the entity label information of each word to enable the model to have a better understanding of the words and better focus on key information. Specifically, we propose an Entity-label-guided Long Short-Term Memory (ELSTM) and a token memory to store and capture the entity label information of each word. Our model outperforms state-of-the-art methods in terms of BLEU and human evaluations by a large margin. In the near future, we aim to fully prevent the generation of unfaithful descriptions and bring FPDG online.

Acknowledgments

We would like to thank the reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and NSFC No. 61672058). Rui Yan was sponsored by Alibaba Innovative Research (AIR) Grant. Rui Yan is the corresponding author.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. <u>arXiv preprint</u> arXiv:1409.0473.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In <u>ICLR</u>.
- Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. ICLR.
- Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In AAAI.
- Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. <u>arXiv preprint</u> arXiv:1903.12457.
- Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. <u>arXiv preprint</u> arXiv:1504.00325.

- Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. Learning towards abstractive timeline summarization. In IJCAI.
- Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In ACL, pages 675–686.
- Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In NAACL, pages 2250–2260.
- 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.
- John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. <u>JMLR</u>, 12:2121–2159.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In <u>ACL</u>, pages 889–898.
- Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. 2017. Aligning where to see and what to tell: image captioning with region-based attention and scene-specific contexts. <u>PAMI</u>, 39(12):2321– 2334.
- Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019. Abstractive text summarization by incorporating reader comments. In AAAI.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In <u>ICML</u>, pages 1243–1252. JMLR. org.
- Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In <u>EMNLP</u>, pages 2824–2829.
- Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In <u>ACL</u>, pages 132–141.
- Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. arXiv preprint arXiv:1905.13637.
- Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. <u>ICLR</u>.
- Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In <u>EMNLP</u>, pages 1830–1839.

- Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In <u>ACL</u>, pages 861– 871.
- J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. biometrics, pages 159–174.
- Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In <u>Proceedings of the 2018 Conference on Empirical</u> <u>Methods in Natural Language Processing</u>, pages <u>3890–3900</u>.
- 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.
- Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In ACL, volume 1, pages 11–19.
- Abigail See, Peter J Liu, and Christopher D Manning. 2017a. Get to the point: Summarization with pointer-generator networks. In <u>ACL</u>, pages 1073– 1083.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017b. Get to the point: Summarization with pointer-generator networks. pages 1073–1083. ACL.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In <u>NIPS</u>.
- Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018a. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In IJCAI, pages 4418–4424.
- Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018b. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In AAAI.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In NIPS, pages 5998–6008.
- Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A statistical framework for product description generation. In <u>IJCAI</u>, pages 187–192.
- 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. <u>arXiv preprint</u> arXiv:1609.08144.

- Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In <u>AAAI</u>.
- Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Planand-write: Towards better automatic storytelling. In <u>Proceedings of the AAAI Conference on Artificial</u> <u>Intelligence</u>, volume 33, pages 7378–7385.
- Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019a. Automatic generation of pattern-controlled product description in e-commerce. In <u>The World</u> Wide Web Conference, pages 2355–2365. ACM.
- Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019b. Automatic generation of pattern-controlled product description in e-commerce. In <u>WWW</u>, pages 2355–2365.
- Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. <u>arXiv preprint</u> arXiv:1704.06393.
- Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In <u>ACL</u>, pages 1118–1127.