Product-Aware Answer Generation
in E-Commerce Question-Answering

Shen Gao*
ICST, Peking University
shengao@pku.edu.cn

Zhaochun Ren
JD.com
renzhaochun@jd.com

Yihong Zhao
JD.com
ericzhao@jd.com

Dongyan Zhao
ICST, Peking University
zhaody@pku.edu.cn

Dawei Yin
JD.com
yindawei@acm.org

Rui Yan†
ICST, Peking University
ruiyan@pku.edu.cn

ABSTRACT

In e-commerce portals, generating answers for product-related questions has become a crucial task. In this paper, we propose the task of *product-aware answer generation*, which tends to generate an accurate and complete answer from large-scale unlabeled e-commerce reviews and product attributes. Unlike existing question-answering problems, answer generation in e-commerce confronts three main challenges: (1) Reviews are informal and noisy; (2) joint modeling of reviews and key-value product attributes is challenging; (3) traditional methods easily generate meaningless answers. To tackle above challenges, we propose an adversarial learning based model, named PAAG, which is composed of three components: a question-aware review representation module, a key-value memory network encoding attributes, and a recurrent neural network as a sequence generator. Specifically, we employ a convolutional discriminator to distinguish whether our generated answer matches the facts. To extract the salience part of reviews, an attention-based review reader is proposed to capture the most relevant words given the question. Conducted on a large-scale real-world e-commerce dataset, our extensive experiments verify the effectiveness of each module in our proposed model. Moreover, our experiments show that our model achieves the state-of-the-art performance in terms of both automatic metrics and human evaluations.

CCS CONCEPTS

• **Information systems** → *Question answering*;

KEYWORDS

Question answering, e-commerce, product-aware answer generation

*Work performed during an internship at JD.com.
†This author is the corresponding author

1 INTRODUCTION

In recent years, the explosive popularity of *question-answering* (QA) is revitalizing the task of *reading comprehension* with promising results [25, 34]. Unlike traditional knowledge-based QA methods that require a structured knowledge graph as the input and output resource description framework (RDF) triples [10], most of reading comprehension approaches read context passages and extract text spans from input text as answers [23, 34].

E-commerce are playing an increasingly important role in our daily life. As a convenience of users, more and more e-commerce portals provide community question-answering services that allow users to pose product-aware questions to other consumers who purchased the same product before. Unfortunately, many product-aware questions lack of proposer answers. Under the circumstances, users have to read the product's reviews to find the answer by themselves. Given product attributes and reviews, an answer is manually generated following a cascade procedure: (1) a user skims reviews and finds relevant sentences; (2) she/he extracts useful semantic units; (3) and the user jointly combines these semantic units with attributes, and writes a proper answer. However, the information overload phenomenon makes this procedure an energy-draining process to pursue an answer from a rapidly increasing number of reviews. Consequently, automatic product-aware question-answering become more and more helpful in this scenario. The task on which we focus is the *product-aware answer generation* given reviews and product attributes. Our goal is to respond product-aware questions automatically given a large amount of reviews and attributes of a specific product. Unlike either a "yes/no" binary classification task [13] or a review ranking task [17], product-aware answer generation provides a natural-sounding sentence as an answer.

The definition of our task is similar as the reading comprehension. However, most of existing reading comprehension solutions only extract text spans from contextual passages [37]. Since the target of product-aware answer generation is to generate a natural-sounding answer instead of text spans, most of reading comprehension methods and datasets (*e.g.,* SQuAD [20]) are not applicable. As far as we know, only few of reading comprehension approaches

aim to generate a natural-sounding answers from extraction results [16, 29]. With a promising performance on MS-MARCO [18], *S-Net* framework proposed by Tan et al. [29] focuses on synthesizing answers from extraction results. However, S-Net requires a large amount of labeling data for extracting text spans, which is still unrealistic given a huge number of reviews. Moreover, product reviews from e-commerce website are informal and noisy, whereas in reading comprehension the given context passages are usually in a formal style. Generally, existing reading comprehension approaches confront three challenges when addressing product-aware question answering: (1) Review text is irrelevant and noisy. (2) It's extremely expensive to label large amounts of explicit text spans from real-world e-commerce platforms. (3) Traditional loss function calculation in reading comprehension tends to generate meaningless answers such as "I don't know".

In this paper, we propose the *product-aware answer generator* (PAAG), a product related question answering model which incorporates customer reviews with product attributes. Specifically, at the beginning we employ an attention mechanism to model interactions between a question and reviews. Simultaneously, we employ a key-value memory network to store the product attributes and extract the relevance values according to the question. Eventually, we propose a recurrent neural network (RNN) based decoder, which combines product-aware review representation and attributes to generate the answer. More importantly, to tackle the problem of meaningless answers, we propose an adversarial learning mechanism in the loss calculation for optimizing parameters. Conducted on a large-scale real-world e-commerce dataset, we evaluate the performance of PAAG using extensive experiments. Experimental results demonstrate that the PAAG model achieves significant improvement over other baselines, including the state-of-the-art reading comprehension model. Furthermore, we also examine the effectiveness of each module in PAAG. Our experiments verify that adversarial learning is capable to significantly improve the denoising and facts extracting capacity of PAAG.

To sum up, our contributions can be summarized as follows:

- We propose a product-aware answer generation task.
- To tackle this task, we propose an end-to-end learning method to extract fact that is helpful for answering questions from reviews and attributes and then generate answer text.
- Due to the review is in an informal style with noise, we propose an attention based review reader and use the Wasserstein distance based adversarial learning method to learn to denoise the review text. The discriminator can also give an additional training signal for generating more consistence answer.
- Experiments conducted on a large-scale real-world dataset show that our PAAG method outperforms all baselines, include the state-of-the-art model in terms of all metrics. The effectiveness of each module in PAAG is also demonstrated in our experiments.

2 RELATED WORK

We detail related work on product-aware question-answering, reading comprehension, and sequence-to-sequence architecture.

Product-aware question answering. In recent years, product-aware question answering has received several attention. Most of existing strategies aim at extracting relevant sentences from input

text to answer the given question. Yu *et al.* [39] propose a framework for opinion QA, which first organizes reviews into a hierarchy structure and retrieves review sentence as the answer. Yu *et al.* [40] propose an answer prediction model by incorporating an aspect analytic model to learn latent aspect-specific review representation for predicting the answer. External knowledge has been considered with the development of knowledge graphs. McAuley *et al.* [13] propose a method using reviews as knowledge to predict the answer, where they classify answers into two types, binary answers (i.e. "yes" or "no") and open-ended answers. Incorporating review information, recent studies employ ranking strategies to optimize an answer from candidate answers [17, 38]. Meanwhile, product-aware question retrieval and ranking has also been studied. Cui *et al.* [4] propose a system which combines questions with RDF triples. Yu *et al.* [38] propose a model which retrieves the most similar queries from candidate QA pairs, and uses corresponding answer as the final result.

However, all above task settings differ from our task. Unlike above approaches, our method is aimed to generate an answer from scratch, based on both reviews and product attributes.

Reading Comprehension. Given a question and relevant passages, reading comprehension extracts a text span from passages as an answer [20]. Recently, based on a widely applied dataset, i.e., SQuAD [20], many approaches have been proposed. Seo *et al.* [23] use bi-directional attention flow mechanism to obtain a query-aware passage representation. Wang *et al.* [34] propose a model to match the question with passage using gated attention-based recurrent networks to obtain the question-aware passage representation. Consisting exclusively of convolution and self-attention, QANet [37] achieves the state-of-the-art performance in reading comprehension. As mentioned above, most of the effective methods contain question-aware passage representation for generating a better answer. This mechanism make the models focus on the important part of passage according to the question. Following these previous work, our method models the reviews of product with a question aware mechanism.

Sequence-to-sequence architecture. In recent years, sequence-to-sequence (seq2seq) based neural networks have been proved effective in generating a fluent sentence. The seq2seq model [28] is originally proposed for machine translation and later adapted to various natural language generation tasks, such as text summarization and dialogue generation [31, 36]. Rush *et al.* [21] apply the seq2seq mechanism with attention model to text summarization field. Then See *et al.* [22] add copy mechanism and coverage loss to generate summarization without out-of-vocabulary and redundancy words. The seq2seq architecture has also been broadly used in dialogue system. Tao *et al.* [30] propose a multi-head attention mechanism to capture multiple semantic aspects of the query and generate a more informative response. Different from seq2seq models, our model utilizes not only the information in input sequence but also many external knowledge from user reviews and product attributes to generate the answer that matches the facts. Unlike traditional seq2seq model, there are several tasks which input data is in key-value structure instead of a sequence. In order to utilize these data when generating text, key-value memory network (KVMN) is purposed to store this type of data. He *et al.* [10] incorporate copying and retrieving knowledge from knowledge base stored in KVMN

to generate natural answers within an encoder-decoder framework. Tu *et al.* [32] use a KVMN to store the translate history which gives model the opportunity to take advantage of document-level information instead of translate sentences in an isolation way. We will use the KVMN architecture in our model to store and retrieve the product attributes data.

3 PROBLEM FORMULATION

Before introducing our answer generation task for product-aware question, we introduce our notation and key concepts.

At the beginning, for a product, we assume there is a question $X^q = \{x_1^q, x_2^q, \ldots, x_{T_q}^q\}$, T_r reviews $X^r = \{x_1^r, x_2^r, \ldots, x_{T_r}^r\}$ and T_a key-value pairs of attributes $A = \{(a_1^k, a_1^v), (a_2^k, a_2^v), \ldots, (a_{T_a}^k, a_{T_a}^v)\}$, where a_i^k is the name of i-th attribute and a_i^v is the attribute content. In our task, we assume that each attribute, both key a_i^k and value a_i^v are represented as a single word. Given a question X^q, an answer generator reads the reviews X^r and attributes A, then generates an answer $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T_y}\}$. The goal is to generate an answer $\hat{Y}$ that is not only grammatically correct but also consistent with product attributes and opinions in the reviews. Essentially, the generator tries to optimize the parameters to maximize the probability $P(Y|X^q, X^r, A) = \prod_{t=1}^{T_y} P(y_t|X^q, X^r, A)$ where $Y = \{y_1, y_2, \ldots, y_{T_y}\}$ is the ground truth answer.

4 PAAG MODEL

4.1 Overview

In this section, we propose our product-aware answer generator model, abbreviated as PAAG. The overview of PAAG is shown in Figure 1. PAAG can be split into two main parts: answer generator and consistency discriminator. We start by detailing the answer generator which generates an answer according to the reviews and attributes. We then describe the consistency discriminator which distinguishes whether the generated answer matches the facts given by reviews and attributes.

• *Answer generator.* (1) **Review reader:** (See Section 4.2) In this part, we encode the review text into vector representations. By matching the relevance of the given question, we signify important semantic units of reviews. (2) **Attributes encoder:** (See Section 4.3) Our model stores the product attributes information into a key-value memory network. For each key-value pair, a correlation score between a key and the question is aggregated into the value. (3) **Facts decoder:** (See Section 4.4) To generate the answer, we use the RNN-based decoder which fuses the facts extracted from reviews and attributes when generating words.

• *Consistency discriminator.* (See Section 4.5) Existing approaches easily generate a grammatically correct answer but conflicts to the facts. In order to produce factual answer, we use a discriminator to determine whether the generated sentence matches the facts. By employing the earth-mover distance to optimize our network, we use the result of discriminator as a training signal to encourage our model to produce a better answer.

4.2 Review reader

At the beginning, we use an embedding matrix e to map one-hot representation of each word in the question X^q, reviews X^r, and attributes A to a high-dimensional vector space. We denote $e(x)$ as the embedding representation of word x. From these embedding representations, we employ a bi-directional recurrent neural network (Bi-RNN) to model the temporal interactions between words:

$$h_t^q = \text{Bi-RNN}_q(e(x_t^q), h_{t-1}^q), \tag{1}$$

$$h_{i,t}^r = \text{Bi-RNN}_r(e(x_{i,t}^r), h_{i,t-1}^r). \tag{2}$$

where h_t^q and $h_{i,t}^r$ denote the hidden state of t-th step in Bi-RNN for question X^q and i-th review in X^r respectively. We denote the final hidden state $h_{T_q}^q$ of Bi-RNN$_q$ as the vector representation of the question X^q. Following [11, 23], we choose the long short-term memory (LSTM) as a cell of Bi-RNN.

For producing a fixed size vector representation of reviews, an intuitive method is to conduct an average-pooling strategy on all the hidden states of each review, which neglects the question-oriented salient part of the review. Accordingly, we propose a gated attention-based method to incorporate the focus point of question into the review representation. Furthermore, we add an additional gate which learns the relevance between question X^q and review X^r via a soft-alignment, so we have:

$$s_{i,j}^k = v^{\mathsf{T}} \tanh(W_q h_k^q + W_r h_{i,j}^r), \tag{3}$$

$$s_{i,j} = \max(s_{i,j}^1, s_{i,j}^2, \ldots, s_{i,j}^{T_q}), \tag{4}$$

$$\alpha_{i,j} = \exp(s_{i,j}) / \sum_{t=1}^{T_r^i} \exp(s_{i,t}), \tag{5}$$

where W_q, W_r, v are all trainable parameters. $\alpha_{i,j} \in \mathbb{R}$ refers to the importance score of the j-th word in the i-th review given X^q. Thereafter, we apply the attention-pooling operation on each review hidden state $h_{i,\cdot}^r$ to produce the question-aware review representation c_i^r, shown in Equation 6:

$$c_i^r = \sum_{t=1}^{T_r^i} \alpha_{i,t} h_{i,t}^r, \tag{6}$$

Given an answer generation procedure, not all the reviews are useful to answer the question due to the informal style problem. Thus if we directly calculate the arithmetic mean vector of all reviews' representations, we can not capture salient passages. To tackle this problem, a **gated fusion** method is utilized here to sum up all the review representations. We first calculate the relevance between each review representation c_i^r and question representation $h_{T_q}^q$ via a bi-linear layer shown in Equation 7:

$$u_i = c_i^r W_f h_{T_q}^q, \tag{7}$$

$$u_i' = \exp(u_i) / \sum_{t=1}^{T_r} \exp(u_t), \tag{8}$$

where W_f is a trainable parameter. Afterwards, we use a softmax function to simulate the relevance score u_i, *i.e.,* u_i', shown in Equation 8. Then we use u_i' as the weight of the i-th review to do weighted-average on review representation c_i^r over all reviews, so we have:

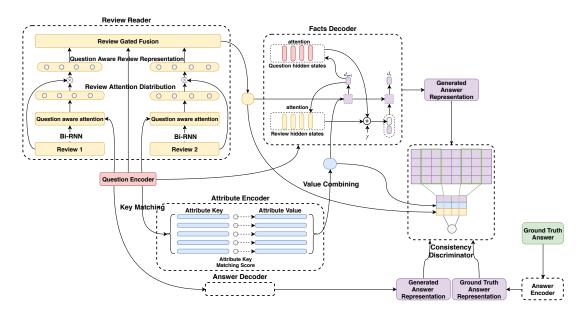$$c^r = \sum_{i=1}^{T_r} u_i' c_i^r. \tag{9}$$

Figure 1: Overview of PAAG. We divide PAAG into four parts: (1) *Review reader* reads the review to extract relevant semantic parts. (2) *Attribute encoder* encodes the attribute key-value pairs using key-value memory network. (3) *Facts decoder* generates the final answer according to the facts learned by the two modules introduced before. (4) *Consistency discriminator* distinguishes whether the generated answer matches the extracted facts, and we also use the result of discriminator as another training signal.

4.3 Attributes encoder

The attributes of a product can be seen as structured knowledge data in our task. As key-value memory network (KVMN) is shown effective in structured data utilization [10, 14, 32], in our work we employ KVMN to store product attributes for generating answers. Correspondingly, we store the word embedding of each attribute's key and value in the KVMN. The *read* operation in our KVMN is divided into two steps: *key matching* and *value combination*.

Key matching The goal of key matching is to calculate the relevance between each attribute and the given question. Given question X^q, for the i-th attribute $a_i = (a_i^k, a_i^v) \in A$, we calculate the probability of a_i over X^q, i.e., $P(a_i|X^q)$, as the *matching score*. To this end, we exploit the question representation $h_{T_q}^q$ to calculate the probability distribution:

$$P(a_i|X^q) = \frac{\exp(h_{T_q}^q W_a e(a_i^k))}{\sum_{t=1}^{T_a} \exp(h_{T_q}^q W_a e(a_t^k))}, \quad (10)$$

Since question representation $h_{T_q}^q$ and attribute key representation $e(a_i^k)$ are not in the same semantic space, following [15, 27], we use a trainable key matching parameter W_a to transform these representations into a same space.

Value combination As the relevance between question X^q and attribute a_i, the matching score $P(a_i|X^q)$ can help to capture the most relevant attribute for generating a correct answer. Therefore, as shown in Equation 11, the attribute encoder reads the information m from KVMN via summing over the stored attribute values, and

guide the follow-up answer generation, so we have:

$$m = \sum_{i=1}^{T_a} P(a_i|X^q) e(a_i^v). \quad (11)$$

4.4 Facts decoder

PAAG generates an answer based on a set of facts extracted from reviews and attributes. Same as our encoder settings, we set LSTM as a cell in our RNN-based facts decoder. We concatenate the question, review and attribute representations and apply a linear transform, then use this vector as the initial state d_0; at every decoding step, we feed a context vector g_t into RNN cell. At t-th decoding step, context vector g_t summarizes the input question and review, and we will show the detail of producing g_t at follows. The procedure of t-th decoding step is shown in Equation 13. We use the notion $[\cdot; \cdot]$ as the concatenation of two vectors.

$$d_0 = W_e \left[m; h_{T_q}^q; c^r \right] + b_e, \quad (12)$$

$$d_t = \text{LSTM}\left(d_{t-1}, [g_{t-1}; e(y_{t-1})]\right), \quad (13)$$

where W_e, b_e are the trainable parameters, d_t is the hidden state of t-th decoding step.

Similar with the seq2seq with attention mechanism, we use the hidden state of previous step d_{t-1} to attend the question hidden states and review hidden states to get the context vector g_t of current decoding step. The algorithm of attending reviews hidden states is same as attending question hidden states, so we use h_i^* to

represent the hidden state where $*$ can be r or q.

$$\beta'_{i,t} = z^\mathsf{T} \tanh \left(W_s h_i^* + W_d d_t \right), \tag{14}$$

$$\beta_{i,t} = \exp \left(\beta'_{i,t} \right) / \sum_{j=1}^{T_q} \exp \left(\beta'_{j,t} \right), \tag{15}$$

$$g_t^* = \sum_{i=1}^{T} \beta_{i,t} h_i^*, \tag{16}$$

where W_s, W_d are all trainable parameters. After two attention procedures of question and review finished, we concatenate context vector g_t^r and g_t^q with a balanced gate γ which is determined by decoder state d_t:

$$\gamma = \sigma \left(W_g d_t + b_g \right), \quad g_t = \left[\gamma g_t^r; (1-\gamma) g_t^q \right], \tag{17}$$

The context vector g_t, which can be seen as a representation of reading from the question and reviews, is concatenated with the decoder state d_t and then fed into a linear transformation layer to conduct the generated word distribution P_v over the vocabulary.

$$d_t^o = (W_o[d_t; g_t] + b_o), \tag{18}$$

$$P_v = \text{softmax} \left(W_v d_t^o + b_v \right), \tag{19}$$

At the t-th decoding step, we set the loss as the negative log likelihood of the target word y_t:

$$loss_g = -1/T_y \sum_{t=1}^{T_y} \log P_v(y_t). \tag{20}$$

In order to handle the out-of-vocabulary (OOV) problem, we equip the pointer network [7, 22, 33] with our decoder, which makes our decoder capable to copy words from question. The procedure of pointer network is the same as the model proposed by See *et al.* [22], and we omit this procedure in our paper due to the limited space.

Up to now, we can use $loss_g$ to compute gradients for all the parameters in answer generator and use gradient descent method to update these parameters. But the correctness constraint given by cross entropy loss $loss_g$ is not enough. So we need a classifier to judge whether the generated answer is consistent with the facts. In this way, we use this classification result to guide the answer generator to produce more consistent answers.

4.5 Consistency discriminator

To generate sentences which are more consistent with the facts, we add a discriminator to provide additional training signals for the answer generator. We propose a convolutional neural network (CNN) based classifier as discriminator. The goal of this classifier is to distinguish whether a sentence is consistent with the given facts. So we can use the confidence of classifying a sentence as a training signal to encourage the answer generator to produce a better answer. We use the answer generated by the facts decoder presented in Section 4.4 as the negative sample for classification, and use the representation of ground truth answer as the positive sample.

As for giving a positive sample for discriminator, we use an RNN to encode the ground truth answer into a vector representation.

$$\tilde{d}_t^g = LSTM(y_t, \tilde{d}_t^g) \tag{21}$$

Since the ground truth is encoded by another RNN which is different from the decoder RNN of d_t^o, we use a linear transformation to

transform the high-dimensional space of $\tilde{d}_t^g$ to the same space as d_t^o in.

$$d_t^g = W_z * \tilde{d}_t^g + b_z \tag{22}$$

where W_z and b_z are all trainable parameters.

For training the discriminator ability of capturing whether an answer is consistent with the facts, we construct another negative samples to train the discriminator. We present an answer decoder (shown at the bottom of Figure 1), which employs the same decoding mechanism as the facts decoder but no fact is attached during decoding. Specifically, we use the hidden state d_t^o shown in Equation 18 as the representation of each word in the generated sentence by the facts decoder. Similarly, the decoder RNN without feeding facts representation g_t generates hidden states d_t^f. We use d_t^f as the representation of generated answer without facts support.

Then a two-dimensional convolutional layer convolves the hidden states d_t^* with multiple convolutional kernels of different widths. Each kernel corresponds a linguistic feature detector which extracts a specific pattern of multi-grained n-grams [12]. A convolutional filter W_c maps hidden states in the receptive field to a single feature. As we slide the filter across the whole sentence, we obtain a sequence of new features $n = [n_1, n_2, \dots, n_l]$, shown in Equation 23:

$$n_t^* = \text{relu} \left(d_t^* \otimes W_c + b_c \right), \tag{23}$$

where W_c, b_c are all trainable parameters and $\otimes$ denotes the convolution operation. For each convolutional filter, the max-pooling layer takes the maximal value among the generated convolutional features n^f, n^o and n^g respectively, resulting in a fixed-size vector N^f, N^o and N^g. Then we obtain the classification result $D(d_t^*) \in \mathbb{R}$ through an interaction between N^* and the facts, i.e., attribute representation m and review representation c^r. So we have:

$$D(d_t^*) = W_h \text{relu} \left(N^* + m + c^r \right) + b_h, \tag{24}$$

where W_h, b_h are all trainable parameters. Here we apply the Vanilla generative adversarial network (GAN) with a sigmoid function on the $D(d_t^*)$ to produce the classification probability and tries to minimize the Jensen-Shannon divergence between real and generated data distribution.

However, as vanilla GAN often leads to gradient vanishing as the discriminator saturates [8], which makes the discriminator can not give the correct training signal. Inspired by previous work [2, 8], we tackle this problem by minimize the earth-mover (also called Wasserstein-1) distance $W(\mathbb{P}_r, \mathbb{P}_g)$ instead of Jensen-Shannon divergence. Informally, given a distribution $\mathbb{P}_r$ of ground truth answer and a distribution $\mathbb{P}_g$ of facts-based answer or answer without facts. Then we minimize the cost of transporting mass from $\mathbb{P}_r$ to $\mathbb{P}_g$. The discriminator $D \in \mathcal{D}$ is a *1-Lipschitz* function, where $\mathcal{D}$ is the set of *1-Lipschitz* functions [8].

In order to meet the *Lipschitz* constraint of discriminator D, we use an alternative way to enforce the *Lipschitz* constraint. We add a gradient norm of the output of D with respect to its input, which is simply sampled uniformly along a straight line between points sampled from the ground truth representation d_t^g and the facts based output d_t^o. Then our objective function is shown in Equation 26.

$$d'_t = \epsilon d_t^o + (1 - \epsilon)d_t^g, \tag{25}$$

$$loss_d = \frac{1}{T_y} \sum_t^{T_y} D(d_t^f) + D(d_t^o) - D(d_t^g) + \lambda \left(\left\| \nabla_{d'_t} D(d'_t) \right\|_2 - 1 \right)^2. \tag{26}$$

where $\epsilon \sim U[0, 1]$ is a random number and λ is a coefficient of gradient penalty term. Then we can use the optimization methods to update the parameters of discriminator use the loss function $loss_d$. Meanwhile, we add the $-D(d_t^o)$ to the previous defined $loss_g$ in Equation 20 to encourage the answer generator produce better result.

5 EXPERIMENTAL SETUP

5.1 Research questions

We list four research questions that guide the remainder of the paper: **RQ1**: What is the overall performance of PAAG? Does it outperform state-of-the-art baselines? **RQ2**: What is the effect of each module in PAAG? Does the discriminator give a useful training signal to the answer generation module? **RQ3**: Is PAAG capable to extract useful information from noisy reviews? **RQ4**: What is the performance of PAAG at different data domain?

5.2 Dataset

We collect a large-scale dataset from a real-world e-commerce website, including question-answering pairs, reviews, and product attributes. This dataset is available at https://github.com/gsh199449/productqa. On this website, users can post a question about the product. Most questions are asking for an experience of user who has already bought the product. In the collected data, each QA pair is associated with the reviews and attributes of the corresponding product. We remove all QA pairs without any relevant review and split the whole dataset into training and testing set. In total, our dataset contains cover 469,953 products and 38 product categories. The average length of question is 9.03 words and ground truth answer is 10.3 words. The average number of attribute is 9.0 key-value pairs. There are 78.74% of training samples have more than 10 relevant reviews and 75.33% of training samples have more than 5 attributes.

5.3 Evaluation metrics

To evaluate our proposed method, we employ BLEU [19] to measure the quality of generated sentence by computing overlapping lexical units (*e.g.,* unigram, bigram) with the reference sentence. We also consider three embedding-based metrics [6] (including Embedding Average, Embedding Greedy and Embedding Extreme) to evaluate our model, following several recent studies on text generation [24, 30, 35]. These three metrics compute the semantic similarity between the generated and reference answer according to the word embedding.

Since automatic evaluation metrics may not always consistent with human perception [26], we use human evaluation in our experiment. Three annotators are invited to judge the quality of 100 randomly sampled answer generated by different models. These annotators are all well-educated Ph.D. students and they are all native

Table 1: Ablation models for comparison.

Acronym	Gloss
RAGF	**R**eview reader + **A**ttributes encoder + **G**ated **F**usion
RAGFD	RAGF + consistency **D**iscriminator
RAGFWD	RAGF + **W**asserstein consistency **D**iscriminator
PAAG	RAGFWD + Gradient Penalty

speakers. Two of them have the background of NLP/summarization and another annotator does not major in computer science. We show human annotators a question, several reviews and attributes of the product along with answers generated from each model.

Statistical significance of observed differences between the performance of two runs are tested using a two-tailed paired t-test and is denoted using ▲(or ▼) for strong significance for $\alpha = 0.01$.

5.4 Comparisons

In order to prove the effectiveness of each module in PAAG, we conduct some ablation models shown in Table 1.

To evaluate the performance of our dataset and the proposed framework, we compare our model with the following baselines: (1) **S2SA**: Sequence-to-sequence framework [28] has been proposed for language generation task. We use seq2seq framework which is equipped with attention mechanism [3] and copy mechanism [7] as baseline method. The input sequence is question and ground truth output sequence is the answer. (2) **S2SAR**: We implement a simple method which can incorporate the review information when generating the answer. Different from the S2SA, we use an RNN to read all the reviews and concatenate the final state of this RNN with encoder final state as the initial state of decoder RNN. (3) **SNet**: S-Net [29] is a two-stage state-of-the-art model which extracts some text spans from multiple documents context and synthesis the answer from those spans. Due to the difference between our dataset and MS-MARCO [18], our dataset does not have text span label ground truth for training the evidence extraction module. So we use the predicted extraction probability to do weighted sum the original review word embeddings, and use this representation as extracted evidence to feed into the answer generation module. (4) **QS**: We implement the query-based summarization model proposed by Hasselqvist *et al.* [9]. Accordingly, we use product reviews as original passage and answer as a summary. (5) **BM25**: BM25 is a bag-of-words retrieval function that ranks a set of reviews based on the question terms appearing in each review. We use the top review of ranking list as the answer. (6) **TF-IDF**: Term Frequency-Inverse Document Frequency is a numerical statistic that is intended to reflect how important a question word is to a review. We use this statistic to model the relevance between review and question and select the most similar review as the answer of question.

5.5 Implementation details

Without using pre-trained embeddings, we randomly initialize the network parameters at the beginning of our experiments. All the RNN networks have 512 hidden units and the dimension of word embedding is 256. To produce better answers, we use beam search with beam size 4. Adagrad [5] with learning rate 0.1 is used to optimize the parameters and batch size is 64. We implement our

model using TensorFlow [1] framework and train our model and all baseline models on NVIDIA Tesla P40 GPU.

6 EXPERIMENTAL RESULT

6.1 Overall performance

For research question **RQ1**, to demonstrate the effectiveness of PAAG, we examine the overall performance in term of BLEU, embedding metrics and human evaluation. Table 2 and Table 3 list performances of all comparisons in terms of two automatic evaluation metrics. Significant differences are with respect to SNet (row with shaded background). In these experimental results, we see that PAAG achieves a 111%, 8% and 62.73% increment over the state-of-the-art baseline SNet in terms of BLEU, embedding greedy and consistency score, respectively. In Table 3, we see that our PAAG outperforms all the baseline significantly in semantic distance with respect to the ground truth.

For human evaluation, we ask annotators to rate each generated answer according to two aspects: consistency and fluency. The rating score ranges from 1 to 3, and 3 is the best. We finally take the average across answers and annotators, as shown in Table 4. In Table 4, we can see that PAAG outperforms other baseline models in both sentence fluency and consistency with the facts. We calculate the variance score in Table 4, which shows that annotators agree with each other's judgments in most cases. Although the BLEU score of S2SAR is lower than the S2SA, the embedding score and human score for S2SAR are higher than S2SA. Regardless of few word overlapping between generated and ground answer, the human evaluation and results in terms of embedding metrics verify S2SAR outperforms S2SA. This observation demonstrates the effectiveness of incorporating review in answer generation.

To explore the difficulty of this task, we use a very intuitive method by adding the review information into decoder shown in S2SAR. Although there is a small increment of S2SAR with respect to S2SA in all metrics, we still find a noticeable gap between S2SAR and PAAG. This observation demonstrates that PAAG makes better use of review and attribute information than the simple method S2SAR. In view of the facts extracted from the review and attributes, we examine directly using the most similar review to question as the answer. More specifically, we evaluate the performance of the top of review ranking list which is ranked by text similarity algorithm such as BM25 and TF-IDF. From the result of three metrics, the performance of extractive methods is worth than all the generative methods. It is worth noting that since the answer generated by extractive methods is written by human, it have very high fluency scores. But these answers may not match the question, so the consistency score is very low. Consequently, using the most similar review to question as answer is not a better method than generating answers from scratch.

As our task definition and query based text summarization have some similarities in some way, we can see the reviews as original passage and answer as a query based summary. We also use the query-based text summarization algorithm [9] to generate answer. Similarly, we also employ a reading comprehension method SNet to tackle this task. Since query-based text summarization and reading comprehension models are not defined to tackle QA task in

Table 2: BLEU scores comparison between baselines.

	BLEU	BLEU1	BLEU2	BLEU3	BLEU4
Text generation methods					
S2SA	1.6186	15.4754	3.1437	0.8267	0.1706
S2SAR	1.7549	15.1708	3.2156	0.9078	0.2142
SNet	0.9550	13.7029	2.5374	0.4007	0.0597
QS	1.6848	15.4961	2.9508	0.8315	0.2119
PAAG	**2.0189**▲	**16.2232**▲	**3.5711**▲	**1.0290**▲	**0.2787**▲
Sentence extraction methods					
BM25	0.4125	6.9630	0.7097	0.1333	0.0439
TF-IDF	0.2548	5.5480	0.5127	0.0779	0.0190

Table 3: Embedding scores comparison between baselines.

	Average	Greedy	Extrema
Text generation methods			
S2SA	0.410013	98.653415	0.269461
S2SAR	0.419979	99.742679	0.278666
SNet	0.397162	95.791356	0.277781
QS	0.400291	93.255031	0.252164
PAAG	**0.424218**▲	**103.912364**▲	**0.288321**▲
Sentence extraction methods			
BM25	0.325946	76.814465	0.172976
TF-IDF	0.308293	85.020442	0.155390

Table 4: Consistency and fluency comparison by human evaluation.

	Fluency		Consistency	
	mean	variance	mean	variance
Text generation methods				
S2SA	2.22	0.3	1.62	0.29
S2SAR	2.405	0.365	1.82	0.39
SNet	1.93	0.36	1.355	0.225
QS	2.335	0.285	1.725	0.355
PAAG	2.865▲	0.105	2.205▲	0.445
Sentence extraction methods				
BM25	2.70	0.24	1.45	0.29
TF-IDF	2.48	0.38	1.14	0.12

Table 5: BLEU scores of different ablation models.

	BLEU	BLEU1	BLEU2	BLEU3	BLEU4
RAGF	1.7931	15.7213	3.3705	0.9385	0.2079
RAGFD	1.8597	15.9021	3.4160	0.9409	0.2340
RAGFWD	1.9389	16.1755	**3.5986**	0.9865	0.2461
PAAG	**2.0189**	**16.2232**	3.5711	**1.0290**	**0.2787**

e-commerce scenario, it can not fully utilize the interactions between question, review, and attributes. These methods also lack of ability of denoising the reviews.

6.2 Ablation studies

Next we turn to research question **RQ2**. We conduct ablation tests on the usage of adversarial learning method. The BLEU score of each ablation model is shown in Table 5. In the method RAGFD, we use the vanilla GAN architecture which minimize the divergence. There is a slight increment from RAGF to RAGFD, which demonstrates the effectiveness of discriminator. From Table 5, we find that RAGFWD achieves a 4.3% improvement over RAGFD in terms of BLEU, and PAAG outperforms RAGFWD 4.1% in terms of BLEU. Accordingly, we conclude that the performance of PAAG
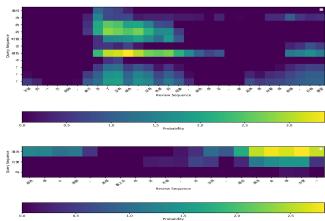
Figure 2: Visualizations of question-aware review attention map.

benefits from using Wasserstein distance based adversarial learning with gradient penalty. This approach can help our model to achieve a better performance than the model using the vanilla GAN architecture.

6.3 Denoising ability

To address **RQ3**, in this section we provide an analysis of the denoising ability of our model. According to Table 2 and Table 5, we observe RAGF achieves 2.1% improvement over SASAR, in terms of BLEU. Such observation demonstrates that question-aware review generation module gives the denoising ability to the model. To further investigate the effectiveness of extracting facts from reviews, we visualize two question-review attention maps, shown in Figure 2. Question of the left figure in Figure 2 is "Will the color fade when cleaning?" and the right is "Is it convenient to clean". The review of the left figure is "Good shopping experience. The pants were washed without discoloration and no color difference compared to the picture. It looks good, comfortable and cheap." and the right is "The color looks good and the texture is great. I haven't started it yet, but it's very easy to clean". In this figure, we can see that there is a very strong interaction between question word 清洗 (cleaning) and phrase in review 清洗会很方便 (very easy to clean). Concretely, these figures show that the question-review attention module can capture the salience semantic part in review according to the question.

In the most cases, the higher word overlap between question and review, the more useful the review is. To prove the ability of review gated fusion module shown in Equation 8, we use the BM25 algorithm to calculate the similarity between question and each review. Then we calculate the cosine distance between the salience score produced by review gated fusion module calculated and BM25 similarity score, shown in Figure 3. In order to demonstrate the denoising ability of adversarial learning method, we compare our full model PAAG with the baseline model RAGF, this experiment proves that the usage of WGAN can encourage our model to capture the salience review better.

6.4 Discussions

Finally, we address **RQ4**. Table 6 shows an example and its corresponding generated answers by different methods. We observe
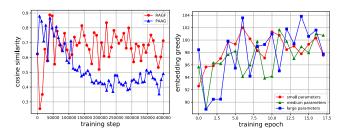

Figure 3: Similarity between Figure 4: Greedy embedding review gates and BM25 score. metric with training epoch.

Table 6: Examples of the generated natural answers by PAAG and other models.

reviews	衣服质量很好，就是我比较瘦，这个S 号对于我来说还是比较肥。很适合孕妇穿。(The quality of the clothes is very good. Because I am thin, the S size is still quite fat for me. It is suitable for pregnant women to wear.)
	颜色漂亮，宽松舒服，就是线头有点多，竟然还有口袋，方便实用！(The clothes are beautiful in color, comfortable to wear, and the thread is a bit more. This dress has a pocket and it is convenient and practical!)
	不轻的雪纺连衣裙，很有垂坠感，觉得夏天春天都适合穿，很衬肤色白呢，胸前会有透，不过不影响穿着(This chiffon dress feels heavy when worn. I think summer and spring are suitable for wearing. Putting on this dress will bring out my skin white. But this dress will be transparent on the chest, but it will not affect the wear.)
attributes	裙长: 中裙||裙型: A 字裙||材质: 涤纶||袖长: 七分袖||领型: 其它||袖型: 其它||上市时间: 2018 春季||版型: A 型
question	怀孕五个多月能穿吗(I have been pregnant for more than five months, can I wear it?)
reference	能穿到生都没问题的(You can wear it until your child is born)
S2SA	可以的，我儿子三个月就穿了(I can wear it, my son wore it when he was three months old.)
RAGF	可以啊，我的就是五个月(I can wear it, my pregnancy is five months.)
PAAG	可以的，我就是(I can wear it, I am pregnant.)

Table 7: Comparison of BLEU scores between different product categories.

	PAAG		S2SA		S2SAR	
	BLEU1	BLEU2	BLEU1	BLEU2	BLEU1	BLEU2
Jewelry	**19.53**	**6.35**	17.65	4.26	18.74	4.69
Mattress	**18.89**	4.14	16.35	3.00	17.52	**5.57**
Clothing	18.18	**5.17**	**18.39**	4.98	18.36	4.68
Kitchenware	**18.00**	**4.31**	15.23	3.19	17.15	4.09
Power and Handtools	**16.34**	**3.98**	13.73	3.20	15.60	3.22
Skin Care	18.01	**4.57**	15.39	3.55	**18.33**	4.40
Gardening	13.67	**2.30**	11.86	1.52	**15.74**	**2.30**
Baby	**18.22**	**4.51**	16.95	3.71	17.27	3.75
Automotive Accessories	17.46	**3.43**	15.49	3.14	**17.86**	3.00
Gift	**19.25**	3.93	17.23	3.06	18.39	**4.24**

that S2SA only generates the answer which is fluent, but generated answers are contradictory to the facts. Due to there is no fact consistency constraint in RAGF, it will also face this problem when generating answers However, PAAG overcomes this shortcoming by using consistency constraint given by discriminator at training, and then produce the answer which is not only fluent but also consistent with the facts.

We evaluate performances of PAAG on different categories. Shown in Table 7, we see that our proposed model beats the other two baselines (S2SA and S2SAR), on majority of product categories in terms of BLEU score. To prove the significance of the above results, we also do the paired student t-test between our model and baseline methods, the p-value of S2SA is 0.0086 and S2SAR is 0.0100. From the t-test, we can see that the performance of our model is significantly higher than other baselines.

To investigate the robustness of parameter, we train our model in different parameter size and evaluate them by embedding metric shown in Figure 4. As the training progresses, the performance of each model is rising. However, the model with a large number of

parameters does not have a great advantage in the final performance of the model with a smaller parameters.

7 CONCLUSION

In this paper, we have proposed the task of product-aware answer generation, which aims to generate an answer for a product-aware question from product reviews and attributes. To address this task, we have proposed product-aware answer generator (PAAG): An attention-based question aware review reader is used to extract semantic units from reviews, and key-value memory network based attribute encoder is employed to fuse relevant attributes. In order to encourage the model to produce answers that match facts, we have employed an adversarial learning mechanism to give additional training signals for the answer generation. To tackle the shortcomings of vanilla GAN, we have applied the Wasserstein distance as value function in the training of consistency discriminator. In our experiments, we have demonstrated the effectiveness of PAAG and have found significant improvements over state-of-the-art baselines in terms of metric-based evaluations and human evaluations. Moreover, we have verified the effectiveness of each module in PAAG for improving product-aware answer generation.

Future work involves extending our model to multiple hop of memory network used as attribute encoder.

ACKNOWLEDGMENTS

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

[2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* abs/1701.07875 (2017).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[4] Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. SuperAgent: A Customer Service Chatbot for E-commerce Websites. In *ACL*.

[5] John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12 (2010), 2121–2159.

[6] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, Vol. 2.

[7] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*.

[8] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved Training of Wasserstein GANs. In *NIPS*.

[9] Johan Hasselqvist, Niklas Helmertz, and Mikael Kågebäck. 2017. Query-Based Abstractive Summarization Using Neural Networks. *CoRR* abs/1712.06100 (2017).

[10] Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning. In *ACL*.

[11] ZhenWang Jia, chen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. 2018. Joint Training of Candidate Extraction and Answer Selection for Reading Comprehension. In *ACL*.

[12] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL*.

[13] Julian McAuley and Alex Yang. 2016. Addressing Complex and Subjective Product-Related Queries with Customer Reviews. In *WWW (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 625–635. https://doi.org/10.1145/2872427.2883044

[14] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*. 1400–1409.

[15] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

[16] Rajarshee Mitra. 2017. An Abstractive approach to Question Answering. *arXiv preprint arXiv:1711.06238* (2017).

[17] Samaneh Moghaddam and Martin Ester. 2011. AQA: Aspect-based Opinion Question Answering. (2011), 89–96.

[18] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR* abs/1611.09268 (2016).

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

[20] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*.

[21] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*.

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

[23] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-directional attention flow for machine comprehension. In *ICLR*.

[24] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*.

[25] Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. 2017. Summarizing Answers in Non-Factoid Community Question-Answering. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 405–414. https://doi.org/10.1145/3018661.3018704

[26] Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating Evaluation Methods for Generation in the Presence of Variation. In *CICLing*.

[27] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS*.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

[29] Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-Net: From Answer Extraction to Answer Synthesis for Machine Reading Comprehension. In *AAAI*.

[30] Chongyang Tao, Shen Gao, Mingyue Shang, Wei Wu, Dongyan Zhao, and Rui Yan. 2018. Get The Point of My Utterance! Learning Towards Effective Responses with Multi-Head Attention Mechanism. In *IJCAI*. 4418–4424.

[31] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems. In *AAAI*.

[32] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to Remember Translation History with a Continuous Cache. *TACL* (2018).

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

[34] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*, Vol. 1. 189–198.

[35] Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural Response Generation via GAN with an Approximate Embedding Layer. In *EMNLP*.

[36] Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards Implicit Content-Introducing for Generative Short-Text Conversation Systems. In *EMNLP*.

[37] Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and Accurate Reading Comprehension by Combining Self-Attention and Convolution. In *ICLR*.

[38] Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling Domain Relationships for Transfer Learning on Retrieval-based Question Answering Systems in E-commerce. In *WSDM*.

[39] Jianxing Yu, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Answering opinion questions on products by exploiting hierarchical organization of consumer reviews. In *EMNLP*. 391–401.

[40] Qian Yu and Wai Lam. 2018. Aware Answer Prediction for Product-Related Questions Incorporating Aspects. In *WSDM*. ACM, 691–699.