Meaningful Answer Generation of E-Commerce Question-Answering

SHEN GAO* and XIUYING CHEN*, Peking University
ZHAOCHUN REN, Shandong University
DONGYAN ZHAO, Peking University
RUI YAN, Renmin University of China and Peking University

In e-commerce portals, generating answers for product-related questions has become a crucial task. In this article, we focus on the task of *product-aware answer generation*, which learns to generate an accurate and complete answer from large-scale unlabeled e-commerce reviews and product attributes.

However, *safe answer problems* (i.e., neural models tend to generate meaningless and universal answers) pose significant challenges to text generation tasks, and e-commerce question-answering task is no exception. To generate more meaningful answers, in this article, we propose a novel generative neural model, called the *Meaningful Product Answer Generator* (*MPAG*), which alleviates the safe answer problem by taking product reviews, product attributes, and a prototype answer into consideration. Product reviews and product attributes are used to provide meaningful content, while the prototype answer can yield a more diverse answer pattern. To this end, we propose a novel answer generator with a review reasoning module and a prototype answer reader. Our key idea is to obtain the correct question-aware information from a large-scale collection of reviews and learn how to write a coherent and meaningful answer from an existing prototype answer. To be more specific, we propose a read-and-write memory consisting of selective writing units to conduct *reasoning among these reviews*. We then employ a prototype reader consisting of comprehensive matching to extract the *answer skeleton* from the prototype answer. Finally, we propose an answer editor to generate the final answer by taking the question and the above parts as input. Conducted on a real-world dataset collected from an e-commerce platform, extensive experimental results show that our model achieves state-of-the-art performance in terms of both automatic metrics and human evaluations. Human evaluation also demonstrates that our model can consistently generate specific and proper answers.

CCS Concepts: • **Information systems** → *Question answering*;

Additional Key Words and Phrases: Question-answering, e-commerce, product-aware answer generation

ACM Transactions on Information Systems, Vol. 39, No. 2, Article 18. Publication date: February 2021.

18

1   INTRODUCTION

In recent years, the explosive popularity of *question-answering* (QA) is revitalizing the task of *reading comprehension* with promising results [53, 69]. Unlike traditional knowledge-based QA methods that require a structured knowledge graph as the input and output resource description framework (RDF) triples [31], most of the reading comprehension approaches read context passages and extract text spans from input text as answers [51, 69].

E-commerce is 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 proper 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) 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 has become more and more helpful in this scenario. In this paper, the task on which we focus is the *product-aware answer generation*. Our goal is to respond product-aware questions automatically given a large number of reviews and attributes of a specific product. Figure 1 shows an example of product-aware answer generation task. Unlike either a "yes/no" binary classification task [39] or a review ranking task [42], product-aware answer generation provides a natural-sounding sentence as an answer.

The definition of our task is similar to the reading comprehension [47, 78] which reads some paragraphs and then answers the question by extracting text spans as the response. The knowledge source of the reading comprehension task always comes from formal documents, like news articles and Wikipedia. However, product reviews from e-commerce websites are informal and noisy, whereas in reading comprehension the given context passages are usually in a formal style. Generally, using existing reading comprehension approaches to tackle the product-aware answer generation confronts three challenges:

(1) Some of the review texts are 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".

To overcome these drawbacks, we propose the *product-aware answer generator* (PAAG) in our early work [22], 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
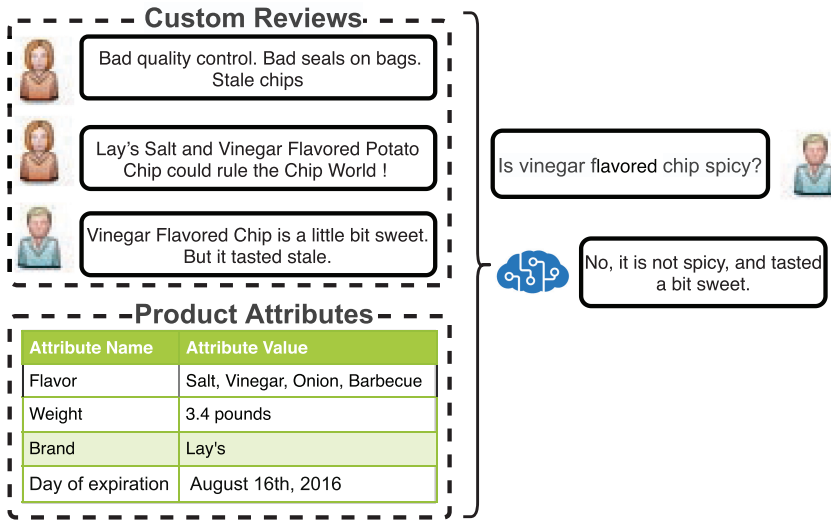
Fig. 1. Example of giving an answer for a product-related question by sourcing from custom reviews and product attributes.

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 for optimizing parameters. To demonstrate the effectiveness of our proposed model, we collected a large-scale e-commerce question answering dataset from one of the largest online shopping websites, JD.com. Experimental results conducted on our proposed dataset demonstrate that the PAAG model achieves significant improvement over other baselines, including the state-of-the-art reading comprehension model. Our experiments verify that adversarial learning is capable of significantly improving the denoising and facts-extracting capacity of PAAG.

Although using adversarial learning techniques can reduce the probability of generating safe answers to a certain extent, the model still has a high probability of generating safe answers since the model lacks answer patterns that can be referenced. This is known as the *safe answer problem*, which is a commonly faced problem in text generation tasks [37, 73]. Moreover, to answer some complex questions, the QA system needs the reasoning ability that could make inference from the product reviews.

Motivated by this observation, in this work, we take it one step further and improve our previously proposed PAAG framework that addresses the safe answer problem in e-commerce question-answering. To be more specific, we solve the problem by introducing a large-scale collection of reviews and a prototype question-answer pair and employing the memory network to incorporate the reasoning result into the answer generation process. Existing works [9, 23] only employ a limited number of reviews (less than 10) and are thus inclined to generate a biased and inaccurate answer. To avoid this, we target learning accurate and appropriate content information from massive amounts of product reviews and product attributes. However, discovering and utilizing information from large quantities of reviews is highly challenging. Sometimes the answer-generation model needs to do reasoning among the reviews to obtain the final facts that are necessary for generation answers. Then, to learn a more diverse and interesting answer pattern, the prototype answer can be of great help. The prototype answer gives a natural language pattern of a similar
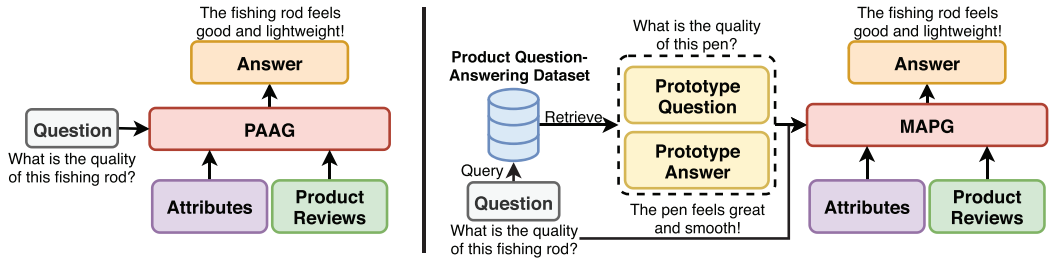
Fig. 2. Illustration of the dataflow of two product-related question-answering model. The PAAG model (left) only uses the question, product attributes, and custom reviews as input. To overcome the safe answer problem, MAPG (right) incorporates the retrieved prototype question-and-answer pair additionally, which can give a natural answer pattern and boost the performance of answer generation.

question, and it can be referred to by the answer-generation model. In fact, existing approaches [28, 72] in the dialog generation field have proven the usefulness of incorporating prototype response for improving performance, which retrieves a prototype text and then post-edits it as the final dialog response. Nevertheless, combining prototype text and reasoning results with the question-answering task has yet to be explored.

In this article, we propose a novel answer-generation model, named Meaningful Product Answer Generator (MPAG), for e-commerce question answering. Figure 2 illustrates the framework of MPAG, which takes the question and three additional information sources as input: product reviews, product attributes, and a prototype question with an answer. First, for product reviews, MPAG uses a simple but efficient clustering method, K-means, to aggregate similar reviews to the same cluster, so as to better utilize review information. Then, we employ Convolutional Neural Network (CNN) to encode these reviews. To reason about these reviews, we propose a write-read memory architecture that selectively writes review information to the memory, and then reads out corresponding information related to the question. Next, MPAG employs a key-value memory network to encode product attributes. To tackle the safe answer problem, we retrieve a prototype answer from the dataset and employ a prototype reader to learn the answer skeleton. Specifically, we use the question to retrieve the most similar question from the dataset as the prototype question and use the answer to this prototype question as prototype answer. Finally, we propose an answer editor to incorporate the answer skeleton with the reasoning result and product attributes, and then generate the new answer. Experiments conducted on a public large-scale benchmark dataset demonstrate that MPAG achieves significant improvement over the state-of-the-art baselines. Experiments also verify the effectiveness of each module in MPAG as well as its explanation ability.

This work is a substantial extension of our previous work reported at WSDM 2019 [23]. The extension in this article includes a novel memory network and a prototype editing-based answer generator, a proposal of a new framework for answering the product-related questions in e-commerce portals which can generate more meaningful answers than the previous method. Specifically, the contributions of this work include the following:

- We come up with a meaningful answer generator model in the e-commerce question-answering task.
- We propose a review reasoning module to reason about a large number of reviews.
- We employ a prototype editing-based answer generator to generate answers by revising a given prototype answer and incorporating the reasoning results.

- Experiments conducted on a public large-scale benchmark dataset show that our model outperforms all baselines, including state-of-the-art models. Experiments also verify the effectiveness of each module in MPAG, as well as its interpretability in answer generation.

The rest of the article is organized as follows: We introduce related work in Section 2. We formulate our research problem in Section 3. We introduce our extended method which incorporates answer prototype and a novel memory network in Section 4. Then, Section 5 details our experimental setup and Section 6 presents the experimental results. Finally, Section 7 concludes this article.

2 RELATED WORK

In this section, we detail related work on product-aware question-answering, reasoning in question-answering, reading comprehension, text generation methods, and prototype editing, respectively.

2.1 Product-Aware Question-Answering

Studies on product reviews include [33, 66, 71]. It is a long-standing issue in information retrieval. Most of the existing strategies for product-aware question-answering aim at extracting relevant sentences from input reviews to answer the given question [39, 79, 80]. For example, Yu et al. [79] proposed an opinion-based question-answering framework, which organizes reviews into a hierarchical structure and retrieves a review sentence as the answer. With the development of knowledge graphs, reviews have been considered as external knowledge [39] to predict the answer. However, it can only return simple answers, such as "yes" or "no". Unfortunately, more often than not there is no proper review that can be used as an answer. Gupta et al. [27] proposed a review-based question answering dataset. However, with this dataset, a high BLEU score (with a 78.56 BLEU-1 score) can be achieved by just randomly selecting a review as the answer. Thus, it is not suitable for generative question answering tasks, which have gained particular interest in recent years due to the emergence of neural networks. For example, Gao et al. [23] proposed an adversarial learning-based model which combines product attributes and review information to generate an answer for a given question. Chen et al. [9] proposed a convolutional text generation model which uses review snippets to guide the decoding attention. However, these generation-based approaches are not sufficiently robust against the safe answer problem, and they also lack reasoning ability.

2.2 Reasoning in Question-Answering

Question-answering has long been a task used to assess a model's ability to understand and reason about language. Large-scale datasets such SQuAD [48] have encouraged the development of many advanced, high-performing attention-based neural models. The ability of reasoning is an important research ingredient in question-answering [55, 61, 70]. Weston et al. [70] released a dataset, named bAbI, to specifically focus on multi-step reasoning by requiring models to reason using disjoint pieces of information. For this task, iteratively updating the query representation with context information has also been shown to effectively emulate multi-step reasoning. Kumar et al. [34] proposed a dynamic memory network where questions trigger an iterative attention process to condition the model's attention on inputs and the result of previous iterations. Xiong et al. [74] proposed several improvements to memory and input modules and apply them to visual question-answering. Instead of extractive fact-finding QA, Bauer et al. [4] focused on a multi-hop generative task, which requires the model to reason, gather, and synthesize disjoint pieces of information within the context to generate an answer. Apart from multi-hop based reasoning, reasoning with a memory write-read mechanism has also been considered [12, 25, 36, 64]. For instance, Graves et al. [25] proposed a machine learning model, namely differentiable neural computer, which consists

of a neural network that can read from and write to an external memory matrix, analogous to the random-access memory in a conventional computer. Subsequently, Le et al. [36] proposed a modified version aiming to balance between maximizing memorization and forgetting via overwriting mechanisms.

Inspired by the above solutions, we also apply the write-read mechanism in this article. In contrast with DNC, we come up with a novel reasoning module, which is simpler and more efficient for e-commerce question-answering.

2.3 Reading Comprehension

Given a question and relevant passages, reading comprehension extracts a text span from passages as an answer [47]. Recently, based on a widely applied dataset, i.e., SQuAD [47], many approaches have been proposed [7, 14, 32, 40, 56]. Seo et al. [51] use bi-directional attention flow mechanism to obtain a query-aware passage representation. Wang et al. [69] 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 [78] achieves the state-of-the-art performance in reading comprehension. Cui et al. [13] place another attention mechanism over the document-level attention and induces "attended attention" for final answer predictions. As mentioned above, most of the effective methods contain question-aware passage representation for generating a better answer. This mechanism makes the models focus on the important part of passage according to the question. Following these previous studies, our method models the reviews of product with a question-aware mechanism.

2.4 Text Generation Methods

In recent years, sequence-to-sequence (seq2seq) [57] based neural networks have been proved effective in generating a fluent sentence. The seq2seq model is originally proposed for machine translation and later adapted to various natural language generation tasks, such as text summarization [10, 17, 18, 21, 24, 38, 45, 65, 67] and dialogue generation [6, 19, 20, 60, 77, 81, 82]. Rush et al. [49] apply the seq2seq mechanism with attention model to text summarization field. Then See et al. [50] add copy mechanism and coverage loss to generate summarization without out-of-vocabulary and redundancy words. Tao et al. [59] propose a multi-head attention mechanism to capture multiple semantic aspects of the query and generate a more informative response. Yao et al. [77] propose to use the content introducing method to solve the problem of generating meaningless response. Wang et al. [68] use three channels for widening and deepening the topics of interest and try to make the conversational model chat more turns.

Different from vanilla seq2seq models, our model utilizes not only the information in input sequence but also much external knowledge from user reviews and product attributes to generate the answer that matches the facts. Similar to our e-commerce question answering task, several tasks input data in key-value structure instead of a sequence. In order to utilize these data when generating text, key-value memory network (KVMN) [2, 75] is purposed to store this type of data. He et al. [31] incorporate copying and retrieving knowledge from the knowledge base stored in KVMN to generate natural answers within an encoder-decoder framework. In detail, they retrieve some relative facts and store them in a KVMN fashion, then use an attention mechanism to attend the facts and fuse them into context vector. Tu et al. [62] use a KVMN to store the translation history which gives model the opportunity to take advantage of document-level information instead of translate sentences in an isolation way. In view of the superior performance of storing structure data in neural models, we employ the key-value memory network in our model to store the attributes data of product.

Table 1. Glossary

Symbol	Description
X^q	a question sentence
X^r	a set of reviews sentences
A	a set of product attribute key-value pairs
a_t^k, a_t^v	the tth attribute key and value in A
$Y, \hat{Y}, \tilde{Y}$	ground truth answer, generated answer, prototype answer
$x_{n,t}^r$	tth word in nth review
$x_t^q, \hat{y}_t, \tilde{y}_t$	tth word in corresponding sentence
N	number of reviews in a cluster
K	number of clusters
C_k	kth review cluster
T_r^n	length of nth review
T_a	number of attributes
T_q, T_y	length of corresponding word sequence

2.5 Prototype Editing

The safe answer problem has been widely explored in recent years [29, 37, 46]. Among the methods that aim to solve this challenge, prototype editing has proven to be one of the most effective. Guu et al. [28] were the first to propose the prototype editing model, where a prototype sentence is sampled from the training data and edited into a new sentence. Subsequently, Wu et al. [72] proposed a new paradigm for response generation, which first retrieves a prototype response from a pre-defined index and then edits it according to the differences between the prototype context and current context. Different from this soft attention method, Cai et al. [5] proposed a hard-editing skeleton-based model to promote the coherence of the generated stories. Specifically, a skeleton is generated by revising the retrieved responses; then, a generative model uses both the generated skeleton and the original query to generate a response. Cao et al. [8] applied this prototype editing method to the task of summarization, where they employed existing summaries as soft templates to generate a summary.

While previous prototype-based methods have achieved much success in various areas, none have incorporated reviews or product attributes into their generation, limiting their ability to produce appropriate and accurate answers. Thus, our proposed method is the first attempt to apply the prototype editing method to question-answering, taking advantage of reasoning results and product attributes to generate an answer. The differences of technical design between our model and previous prototype-based methods lie in that these methods directly use the attention mechanism [8, 72] to obtain the edit vector, which ignore the relationships between the prototype answer and prototype question. And this relationship can help our model to identify which part in the prototype has a low correlation with the prototype question, and that part will be used as the answer prototype.

3 PROBLEM FORMULATION

Before detailing our answer generation model, we first introduce our notations listed in Table 1.

For a product, there is a question $X^q = \{x_1^q, x_2^q, \ldots, x_{T_q}^q\}$, along with reviews $X^r = \{x_1^r, x_2^r, \ldots, x_{T_r}^r\}$, where T_r represents the number of reviews, x_t^q is the tth word in question and answer and x_t^r is the tth review. We assume there exists T_a key-value pairs of product 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 ith attribute and a_i^v is the attribute content. Both key a_i^k and value a_i^v include one word. Since our newly proposed model is a prototype-based method, a prototype question $\tilde{X}^q = \{\tilde{x}_1^q, \tilde{x}_2^q, \ldots, \tilde{x}_{T_q}^q\}$ and prototype answer $\tilde{Y} = \{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{T_y}\}$, where T_q and T_y is the number of words in prototype question and answer, is also attached. The goal of our task is to generate an answer $\hat{Y}$ that is in accordance with product attributes A and information mentioned in reviews X^r.

We formulate the MPAG as follows: Given a question X^q, MPAG first reads reviews X^r and attributes A, then generates an answer $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T_y}\}$ via editing the prototype answer $\tilde{Y}$. That is, the generator maximizes the probability $P(Y|X^q, X^r, A) = \prod_{t=1}^{T_y} P(y_t|X^q, X^r, A, \tilde{X}^q, \tilde{Y})$, where $Y = \{y_1, y_2, \ldots, y_{T_y}\}$ is the ground truth answer.

4 MPAG MODEL

Although our previous proposed PAAG model employs an adversarial learning strategy to encourage the model to generate meaningful answers, and that training method punishes the model when generating answers which do not include the correct product facts, the PAAG model still tends to generate safe answers, like "Ask the custom service" or "I don't know". In this article, we propose to explicitly introduce a natural answer pattern to the answer-generation model, which can be used when generating new answers. Moreover, to extract accurate information from the reviews to form the answer, we propose a novel memory architecture to reason from the reviews. Then, we jointly incorporate the answer pattern and the reasoning results in the final answer generation process. We argue that these extensions will increase the performance of generating more accurate answers for product-related questions.

Although we use the same attribute encoder (key-value-based memory network) and question encoder (RNN-based text encoder) with the PAAG model, there are three significant differences in our MPAG model compared with PAAG:

(1) The PAAG only leverages a few reviews and uses a simple attention-based interaction module with the question. And in this article, our model uses many reviews as input and employs a clustering method and reasoning module to extract useful information from these reviews.
(2) We incorporate prototype question-answer pair to facilitate the answer-generation process, which is not used in PAAG.
(3) We propose to use the editing gate to fuse the information from the answer skeleton and the reasoning result dynamically when generating the answer.

Specifically, we show the comparison between PAAG and MAPG in Table 2.

4.1 Overview

In this section, we introduce our meaningful product answer generator model in detail. The overview of MPAG is shown in Figure 3 and can be split into five modules: (1) *Review clustering* (see Section 4.2). We employ the K-means algorithm to aggregate reviews into clusters; (2) *Review reasoning module* (see Section 4.3). We employ a write-read memory reasoning framework to reason among all reviews to learn useful information from the reviews according to the question; (3) *Attributes Encoder* (see Section 4.4). We encode and extract attributes related to the question by key-value memory network; (4) *Prototype Reader* (see Section 4.5). We use a recurrent network to model the prototype context and extract prototype answer skeleton that can be reused; (5) *Answer*

Table 2. Comparision between PAAG and MAPG

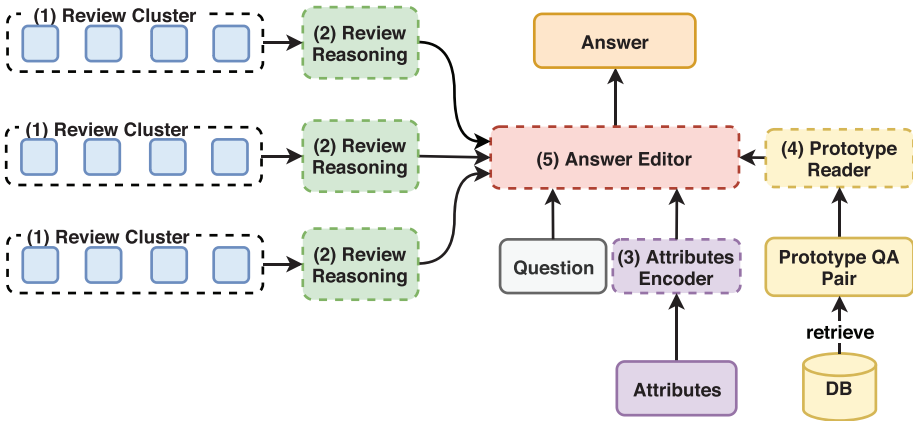|	PAAG	MAPG
Question Encoder	RNN	RNN
Review Encoder	RNN	SRU
Review Clustering	-	K-means
Review Reasoning	-	Read-Write Memory
Product Attribute Encoding	Key-Value Memory	Key-Value Memory
Propotype Reader	-	Answer Skeleton Extractor
Decoder	RNN	Editing Gating
Training Strategy	Adversarial Training	NLL

Fig. 3. Overview of MPAG. We divide our model into five parts: (1) *Review clustering module* aggregates the review into K clusters by using the K-means clustering method on the bag-of-word (BOW) vector of all the reviews; (2) *Review reasoning module* uses a read-write memory to reason over the reviews in each cluster and produce the reasoning result; (3) *Attributes encoder* learns a question-aware attribute representation; (4) *Prototype reader* generates an answer skeleton from prototype answer to enhance the diversity of sentence pattern; and (5) *Answer editor* fuses the result from previous stages by an editing gate and generates an answer.

Editor (see Section 4.6). Eventually, we employ an RNN-based decoder to generate the answer incorporating prototype skeleton, reasoning result, and attribute representation.

4.2 Review Clustering

Since our model takes a large number of reviews focusing on different aspects as input, processing them together will confuse the model and make it hard to learn useful information from them. Thus, we employ a clustering step to aggregate similar reviews into the same cluster. To begin with, we use the BOW vector to represent each review sentence. We then employ the K-means algorithm to aggregate these reviews into K clusters. In each cluster, if the number of reviews is less than N, we append the empty reviews to the cluster to pad the cluster. Conversely, if the number of reviews is larger than N, we drop some reviews to keep each cluster at exactly N reviews. These clustering reviews can be denoted as $X^r = \{C_1, \ldots, C_K\}$, where C_k denotes the kth review cluster which contains N reviews.
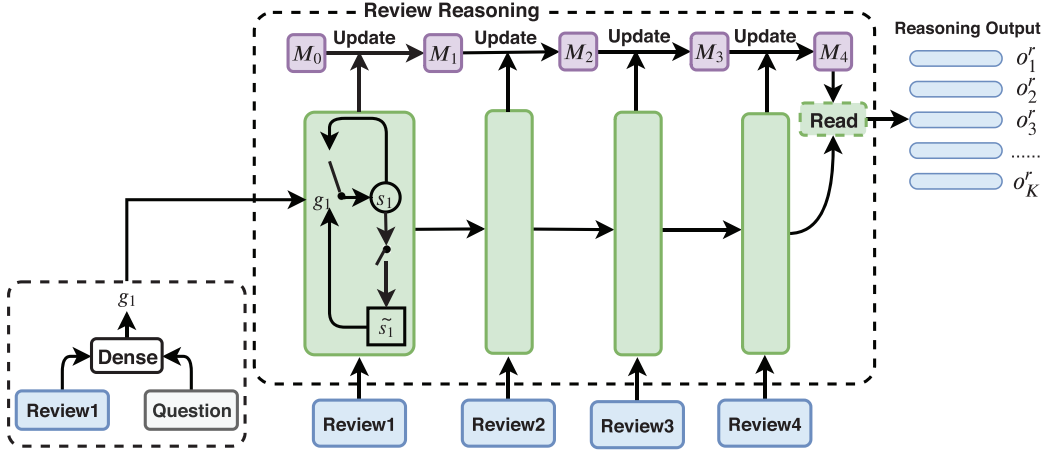
Fig. 4. Overview of the review reasoning module in MPAG. In the beginning, the selective recurrent unit is applied on each review; then retrieved information from each review is written to memory; finally, a multi-head memory reading mechanism is employed to read the memory.

4.3 Review Reasoning Module

For each review cluster, we employ a reasoning module to conduct reasoning among the reviews in each cluster separately. Figure 4 illustrates the whole process. In this section, we omit the subscript k of cluster index for simplicity.

For question X^q, we use an embedding function e to map one-hot representation of each word $x_t^q \in X^q$ into a high-dimensional vector space, $e(x_t^q)$. (The words in reviews X^r, attributes A, prototype question $\tilde{X}^q$, and answer $\tilde{Y}$ are also embedded in this way.) We then employ a *bi-directional recurrent neural network* (Bi-RNN) to model the temporal interactions between words in X^q, so we have:

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

where h_t^q denotes the hidden state of tth step in Bi-RNN for question X^q. We use the final hidden state $h_{T_q}^q$ of Bi-RNN$_q$ to represent the whole question sentence X^q. We here choose the long short-term memory (LSTM) as the cell of Bi-RNN. One can also replace LSTM with similar algorithms such as Gated Recurrent Unit (GRU) [11]. We leave the study for future work.

To extract the semantic features from each review, we first employ an CNN with a max-pooling operation, then apply a Selective Reading Unit (SRU)-based RNN to obtain final representation for each review. To begin with, a list of kernels with different width are used in the CNN operation, and their outputs are concatenated together, denoted as $h_{n,t}^r$ in Equation (2). These different kernels capture different n-grams features. A max-pooling operation is then conducted to extract the most salient feature from the output of CNN, shown in Equation (3):

$$h_{n,t}^r = \text{CNN}\left(e\left(x_{n,t}^r\right)\right), \tag{2}$$

$$h_n^r = \text{max-pool}\left(\left\{h_{n,1}^r, \ldots, h_{n,T_r^n}^r\right\}\right), \tag{3}$$

where $x_{n,t}^r$ denotes the tth word in nth review, T_r^n is the length of nth review, and h_n^r is the vector representation of nth review.

Since we need to identify the salient review to give the answer of current question, the relationship between review and question should be considered when generating the representation for

each review. Inspired by GRU [11], to further study the interactions between reviews, we establish an RNN made up of SRUs. First, in Equation (4), we fuse the representation of question and review together. Then we conduct a dense layer on the fusion representation n_i (see Equation (5)) and normalize the update gate g_i over N steps using softmax function (see Equation (6)). For ith step (review), SRU calculates an update gate g_i, which is decided by question and review together, as shown in Equation (6):

$$n_i = \left[h_i^r \times h_{T_q}^q ; h_i^r ; h_{T_q}^q \right], \tag{4}$$

$$z_i = W_2 \tanh(W_1 n_i + b_1) + b_2, \tag{5}$$

$$g_i = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)}, \tag{6}$$

where $\times$ denotes the element-wise multiplication. Note that the review with a high g_i value means that the information of this review should be mostly retained, and will play a more important role when generating the answer. Thus, the update gate g_i can also be seen as the *salience weight* of ith review.

Unlike GRU, SRU incorporates the question representation $h_{T_q}^q$ into the calculation of update gate which can help the model to identify which review has more contribute on answering current question. Then update gate g_i is used in updating the hidden state s_i^r, shown in Equation (9):

$$q_i = \sigma\left(W_q h_i^r + U_q s_{i-1}^r + b_q\right), \tag{7}$$

$$\tilde{s}_i^r = \tanh\left(W_s h_i^r + q_i \times U_s s_{i-1}^r + b_s\right), \tag{8}$$

$$s_i^r = g_i \times \tilde{s}_i^r + (1 - g_i) \times s_{i-1}^r. \tag{9}$$

We use the hidden state of the ith step s_i^r as the final representation of ith review.

Now we have a more comprehensive representation s_i^r for each review. Next, we focus on conducting reasoning among these reviews, and we propose a review reasoning memory network with a write-read mechanism [25]. As preparation, we initialize an empty memory matrix $M_0 \in \mathbb{R}^{S \times H}$ with S memory slots. Each slot is a H-dimension vector which is set as a representation of learned information and reviews with the same information will be written into the same slot. We use the notion m_j^0 to denotes the jth slot in memory M_0.

4.3.1 Writing to Memory. In this section, we describe the memory writing process that writes each review representation into memory one-by-one, from s_1^r to s_N^r, and the memory is updated from M_0 to M_N simultaneously. In each step, MPAG reads a review representation s_i^r and writes it to the memory matrix M_i. To update the memory matrix, this module calculates three components: write weight for each slot, a write content vector, and an erase vector.

We now detail our writing process. To begin with, we calculate a *write weight* for i-th review, ensuring that similar reviews will have similar write weights. The module first calculates a write key $\kappa_i^w \in \mathbb{R}^H$ using a dense layer applied on the input review representation s_i^r:

$$\kappa_i^w = \text{Dense}(s_i^r), \tag{10}$$

where the write key κ_i^w contains the information learned from the ith review. We obtain the *write weight* $\pi_{i,j}^w \in \mathbb{R}$ for the jth memory slot m_j^i by calculating the similarity between κ_i^w and memory slot m_j^i:

$$\pi_{i,j}^w = \frac{\exp(\mathcal{S}(\kappa_i^w, m_j^i))}{\sum_{l=1}^{S} \exp(\mathcal{S}(\kappa_i^w, m_l^i))}, \tag{11}$$

where $\mathcal{S}$ is a similarity function that measures the relation between the write key and memory slot:

$$\mathcal{S}(x, y) = \frac{x \cdot y}{|x||y|}. \tag{12}$$

In this way, reviews with similar semantic meanings tend to have similar write weights for each memory slot.

Next, we prepare the content which should be written into the memory, and we use *write content vector* $v_i \in \mathbb{R}^H$ to store this content:

$$v_i = \text{Dense}(s_i^r), \tag{13}$$

where $v_i \in \mathbb{R}^H$ represents the information of current input s_i^r that should be written into memory. Then, an *erase vector* $E_i \in \mathbb{R}^H$ is produced to decide which dimension of the memory is useless and should be erased, as shown in Equation (14):

$$E_i = \sigma\left(\text{Dense}(s_i^r)\right), \tag{14}$$

where σ denotes the sigmoid function.

To update the memory slot, we should first write the write content into the slot controlled by the write weight and then erase the useless information from the memory slot. By combining write weight $\pi_{i,j}^w \in \mathbb{R}$, write content $v_i \in \mathbb{R}^H$, and erase vector $E_i \in \mathbb{R}^H$, we update a memory slot $m_j^i \in \mathbb{R}^H$ as follows:

$$m_j^i = m_j^{i-1} \times \left(1 - \pi_{i,j}^w E_i^\mathsf{T}\right) + \pi_{i,j}^w v_i^\mathsf{T}, \tag{15}$$

where $1 \in \mathbb{R}^H$ is an H matrix of ones. After N memory writing steps, memory matrix M_N stores all information collected from this review cluster.

4.3.2 Reading Memory. After we write all the review information into the memory slots, we need to read the reasoning result from the memory to conduct answer generation. Similar to the procedure of writing memory, we calculate a read key to decide the read weight on each slot. Inspired by Vaswani et al. [63], instead of performing one single read function, we find it beneficial to use various read keys to address different slots in memory, i.e., multi-head reading mechanism. We employ a dense layer to project the s_N^r T times to obtain the *read keys* $\{\kappa_1^r, \ldots, \kappa_t^r, \ldots, \kappa_T^r\}$, and we use the tth read key $\kappa_t^r \in \mathbb{R}^H$ as an example to illustrate the process:

$$\kappa_t^r = \text{Dense}_t(s_N^r). \tag{16}$$

On each of these read keys, we apply the similarity function $\mathcal{S}$, yielding tth *read weight* $\pi_{t,j}^r \in \mathbb{R}$ for jth memory slot, shown in Equation (17):

$$\pi_{t,j}^r = \frac{\exp(\mathcal{S}(\kappa_t^r, m_j^N))}{\sum_{l=1}^S \exp(\mathcal{S}(\kappa_t^r, m_l^N))}. \tag{17}$$

In this way, multi-head addressing allows the model to address suitable read location from different read key representation subspaces in different positions. Finally, these read weights π^r are used to produce a weighted sum of memory slots, as shown in Equation (18):

$$r_t = \sum_{j=1}^S m_j^N \pi_{t,j}^r, \tag{18}$$

$$o^r = W_v s_N^r + W_r[r_1 \oplus \cdots \oplus r_T], \tag{19}$$

where $\oplus$ denotes the concatenation between vectors, $r_t \in \mathbb{R}^H$ is the readout vector of the tth read head and $o^r \in \mathbb{R}^H$ is the output of this memory reasoning module for current review cluster. Hence,

the output of the reasoning module o^r can be seen as the representation of reasoning result in this cluster of reviews with respect to the question. Recall that we have aggregated the reviews into K clusters, we now again add the cluster index k in following sections and use notion o_k^r to represent the reasoning result in kth cluster.

4.4 Attributes Encoder

KVMN is shown effective in structured data utilization [31, 41, 62]. Inspired by this, in MPAG we employ an KVMN to infer representations of the structured knowledge, i.e., product attributes. Embedding of attribute's key is regarded as the key in KVMN and embedding of the attribute's value is used as the value.

We first calculate the relevance between each attribute key and the given question. Given question X^q, for the ith attribute $a_i = (a_i^k, a_i^v) \in A$, their matching function λ is calculated as:

$$\lambda(a_i, X^q) = \frac{\exp(h_{T_q}^q W_a e(a_i^k))}{\sum_{t=1}^{T_a} \exp(h_{T_q}^q W_a e(a_t^k))}, \tag{20}$$

where $h_{T_q}^q$ is question representation and W_a is the parameter of linear transform which converts $h_{T_q}^q$ and $e(a_i^k)$ into the same space.

Then, we use these matching scores to produce a weighted sum of all attribute values since an attribute with a high matching score is more related to the question, and thus should take a larger proportion in the overall attribute representation:

$$o^a = \sum_{i=1}^{T_a} \lambda(a_i, X^q) e(a_i^v), \tag{21}$$

where o^a is the output of KVMN and will be used to guide the answer generation.

4.5 Prototype Reader

To tackle the "universal answer" problem, in this article, we employ the prototype editing method to generate the answer by editing the prototype instead of generating answer from scratch. As introduced in Section 1, the prototype question-answer pair is retrieved according to its similarity to the current question. A prototype answer $\tilde{X}^q$ and a prototype question $\tilde{Y}$ are given to our prototype reader to assist the generation process. Prototype reader in MPAG learns to extract the *answer skeleton*, i.e., template words in the prototype answer that are not highly related to the prototype question, to be reused in generating the new answer to increase the diversity of sentence pattern. We first employ Bi-RNN to model the temporal interactions between words in prototype question $\tilde{X}^q$ and answer $\tilde{Y}$:

$$\tilde{h}_t^q = \text{Bi-RNN}_q\left(e\left(\tilde{x}_t^q\right), \tilde{h}_{t-1}^q\right), \tag{22}$$

$$\tilde{h}_t^a = \text{Bi-RNN}_a\left(e\left(\tilde{y}_t\right), \tilde{h}_{t-1}^a\right), \tag{23}$$

where $\tilde{h}_t^q$ and $\tilde{h}_t^a$ denotes the hidden state of tth step in Bi-RNN for tth word in $\tilde{X}^q$ and $\tilde{Y}$, respectively.

We then employ an attention mechanism to analyze the dependency between $\tilde{h}_t^q$ and $\tilde{h}_t^a$ to learn answer skeleton. Then the dependency will be used to extract the answer skeleton from prototype answer which is not highly related to the prototype question. The attention is derived from a shared similarity matrix $D \in \mathbb{R}^{T_q \times T_y}$, which is calculated by each word of prototype question $\tilde{X}^q$ and prototype answer $\tilde{Y}$. D_{ij} here indicates the similarity between the ith question word $\tilde{x}_i^q$ in the
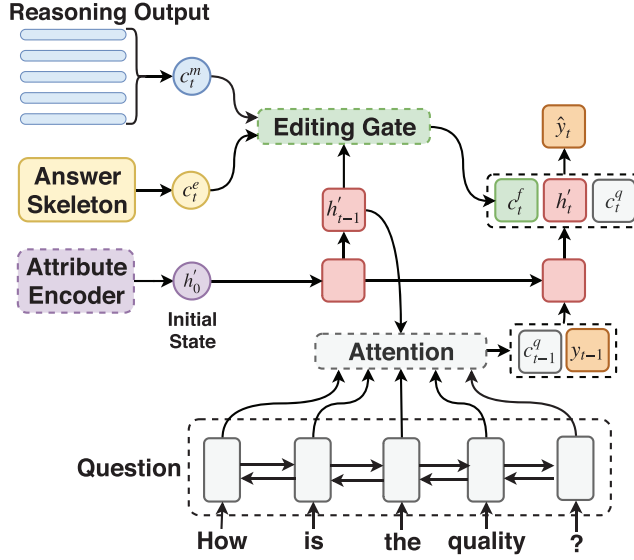
Fig. 5. An overview of the answer editor. In this module, we incorporate the reasoning result, answer skeleton, and product attribute into generating the answer.

question and the jth answer word $\tilde{y}_j$ in the answer and is computed as:

$$D_{ij} = \alpha\left(\tilde{h}_i^q, \tilde{h}_j^a\right),$$
$$\alpha(x, y) = w^{\mathsf{T}}[x \oplus y \oplus (x \times y)], \tag{24}$$

where α is a trainable scalar function that encodes the similarity between two input vectors. We let $d_t = \text{mean}(D_{:t}) \in \mathbb{R}$ represent the attention weight on the tth prototype answer word by prototype question words, and multiply with the corresponding prototype answer hidden state $\tilde{h}_t^a$, resulting in an answer skeleton, $\hat{h}_t^a$. In this way, the module assigns high importance weights to the words which can be reused in a new answer.

4.6 Answer Editor

To generate a diverse and meaningful answer, we propose an RNN-based decoder which incorporates outputs of reasoning result, answer skeleton, question, and attributes. Figure 5 illustrate the framework of the answer decoder. To force the decoder focus on the current question and product, we first apply a linear transform on the concatenation of question vector representation $h_{T_q}^q$ and attributes o^a, and then use this vector h_0' as the initial state of the LSTM shown in Equation (25). The t-th decoding step is shown in Equation (26):

$$h_0' = W_e \left[h_{T_q}^q \oplus o^a \right] + b_e, \tag{25}$$

$$h_t' = \text{LSTM}\left(h_{t-1}', [c_{t-1}^q \oplus e(y_{t-1})] \right), \tag{26}$$

where W_e, b_e are trainable parameters, h_t' is the hidden state of tth decoding step, c_{t-1}^q is the context vector produced by the standard attention mechanism [3] over the hidden states $h_{:}^q$ of question. To dynamically fuse the answer skeleton and reasoning result into the generation process, we come up with an editing context vector c_t^e and a memory context vector c_t^m.

Editing context vector c_t^e is used to dynamically collect useful prototype information from the answer skeleton according to current decoding state. We first show how editing context vector c_t^e is calculated from the answer skeleton, as shown in Equations (27)–(29):

$$\delta_{it} = \frac{\exp(f(\hat{h}_i^a, h_t'))}{\sum_j^{T_y} \exp(f(\hat{h}_j^a, h_t'))}, \tag{27}$$

$$f\left(\hat{h}_i^a, h_t'\right) = h_t' W_f \hat{h}_i^a, \tag{28}$$

$$c_t^e = \sum_j^{T_y} \delta_{jt} \hat{h}_j^a, \tag{29}$$

where $\hat{h}_i^a$ is the ith hidden state in the answer skeleton, f is a bi-linear matching function which models the relationship between the current decoding state h_t' and each hidden state $\hat{h}_i^a$ of answer skeleton.

As for the memory context vector, remember in Section 4.3, there are K reasoning results for each cluster $\{o_1^r, \dots, o_K^r\}$. We thus employ a dynamic-fused method to produce a context vector of all the reasoning results:

$$\epsilon_{kt} = \frac{\exp(f(o_k^r, h_t'))}{\sum_j^K \exp(f(o_j^r, h_t'))}, \tag{30}$$

$$c_t^m = \sum_j^K \epsilon_{jt} o_j^r, \tag{31}$$

where f is the same function used in Equation (28) with different trainable parameters.

In each decoding step, we use an *editing gate* to decide which information should be used in generating current word between prototype words and the reasoning result. And the editing gate is used as an threshold to adjust the proportion of editing context vector and memory context vector. Now we show how to combine the edit context vector c_t^e with the memory context vector c_t^m by an *editing gate* γ_t. $\gamma_t \in \mathbb{R}$ is determined by the decoder state h_t' and is used to decide the importance of edit and memory context vectors at tth decoding step, shown in Equation (32). And then we mix the editing context vector and the memory context vector together using an editing gate as shown in Equation (33):

$$\gamma_t = \sigma\left(Dense(h_t')\right), \tag{32}$$

$$c_t^f = \left[\gamma_t c_t^m \oplus (1 - \gamma_t) c_t^e\right], \tag{33}$$

where σ denotes the sigmoid function, and c_t^f dynamically mixes the information of memory and edit context vector.

Intuitively, to generate the answer word, there are three parts of information sources should be incorporated into generation process: prototype and reasoning result, current question and decoding state. Finally, we concatenate c_t^f with question context vector c_t^q and decoder hidden state h_t' and apply a full connection layer on these vectors. Then, we predict the output word distribution P_v over all the words:

$$h_t^o = W_o\left[h_t' \oplus c_t^f \oplus c_t^q\right] + b_o, \tag{34}$$

$$P_v = \text{softmax}\left(W_v h_t^o + b_v\right). \tag{35}$$

where W_o, W_v, b_o, b_v are all trainable parameters. Our objective function is the negative log likelihood of the target word y_t, shown in Equation (36):

$$\mathcal{L} = - \sum_{t=1}^{T_y} \log P_v(y_t). \tag{36}$$

We employ the gradient descent method to update all parameters to minimize this loss function.

5 EXPERIMENTAL SETUP

5.1 Research Questions

We list four research questions that guide the remainder of this article:

(1) **RQ1** (See Section 6.1): What is the overall performance of MPAG? Does it outperform state-of-the-art baselines?
(2) **RQ2** (See Section 6.2): What is the effect of the review clustering in MPAG?
(3) **RQ3** (See Section 6.3): Does the saliency score (calculated in Equation (6)) explain why the generated answer holds the corresponding opinion?
(4) **RQ4** (See Section 6.4): Can the answer editor in MPAG learn a useful answer skeleton?

5.2 Dataset

We conduct experiments on a large-scale real-world product aware question-answering dataset proposed by Gao et al. [23]. This dataset is collected from JD.com, one of the largest e-commerce websites in China. On this website, users can post a question about the product. Most questions are asking for experience of the user who has already bought the product. This dataset is available at https://github.com/gsh199449/productqa. It includes question-answering pairs, a large number of reviews, and product attributes. Most questions in the dataset are about personal user experience. In this article, the only difference from [23] is that, rather than using BM25 to select a small number of review for each question, we retrieve up to 100 relevant reviews to obtain more information. We also follow the retrieval method proposed by Wu et al. [72] to collect a prototype question-answer pair for each question. We remove all QA pairs without any relevant review and split the whole dataset into training and testing sets. In total, our dataset contains cover 469,953 products from 38 categories. We use all the training dataset as our retrieve database. The average review and attribute numbers of a product are 59.1 and 9.0, respectively. The average lengths of a question and ground truth answer are 9.03 and 10.3 words, respectively.

5.3 Evaluation Metrics

To evaluate the methods, we employ BLEU [44] to measure the lexical unit overlapping (e.g., unigram, bigram) between the generated answer and ground truth. Following [23, 52, 59, 76], we also use embedding-based metrics [16] (including Embedding Average, Embedding Greedy and Embedding Extreme) to compute their semantic similarity. Besides, to quantitatively evaluate the safe answer problem, we use the distinct metric [37], which evaluates the diversity of the generated answers by calculating the number of distinct unigrams and bigrams.

Since only using automatic evaluation metrics can be misleading [54], we also conduct human evaluation. Three annotators are invited to judge the quality of 100 randomly sampled answers generated by different models. These annotators are all well-educated Ph.D. students and are all native speakers. Two of them have a background of NLP while another annotator does not major in computer science. The statistical significance of two runs is tested using a two-tailed paired t-test and is denoted using ▲(or ▼) for strong significance for $\alpha = 0.01$.

Table 3. Ablation Models for Comparison

Acronym	Glossary
MPAG-P	w/o **P**rototype answer
MPAG-M	w/o **M**emory module
MPAG-K	w/o **K**-means Clustering
DNC	Replace Review Reasoning with DNC
SDNC	Replace SRU with LSTM

5.4 Comparisons

To prove the effectiveness of each module, we conduct ablation studies as shown in Table 3. We remove each key module in our proposed model, and then form three baseline methods MPAG-P, MPAG-M, and MPAG-K. Due to the fact that our review reasoning module takes inspiration from DNC [25], we also use the original DNC network to replace our review reasoning module (shown in Section 4.3) as a baseline method, named as DNC. To examine the effectiveness of SRU compared with LSTM cell [25], we replace SRU with the LSTM cell, named as SDNC.

Apart from the ablation study, we also compare our model with the following baselines:

(1) 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 the ranking list as the answer.

(2) 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 the question.

(3) S2SA is the Sequence-to-Sequence (seq2seq) framework [57] which has been proposed for language generation task. We use the seq2seq framework which is equipped with the attention mechanism [3] and copy mechanism [26] as a baseline method. Attention mechanism [3] has been proposed to tackle the alignment between the input sequence and the generated sequence. Copy mechanism [26] has been widely used in the text generation task to tackle the OOV problem, which can copy some words from the input sequence when generating new text. The input sequence is a question and the ground truth output sequence is the answer.

(4) S2SAR is 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.

(5) SNet [58] 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 [43], 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.

(6) QS is the query-based summarization model proposed by Hasselqvist et al. [30]. Accordingly, we use product reviews as an original passage and answer as a summary.

(7) PAAG is the product-aware answer generation model proposed in our previous work [23].

(8) Proto is the prototype editing response generation model in dialog generation task proposed by [72].

Table 4. Automatic Evaluation Comparison with Baselines

	BLEU	BLEU1	BLEU2	BLEU3	BLEU4
Text generation methods					
S2SA	1.62	15.48	3.14	0.83	0.17
S2SAR	1.75	15.17	3.22	0.91	0.21
RAGE	0.22	8.58	0.72	0.05	0.01
SNet	0.96	13.70	2.54	0.40	0.06
QS	1.68	15.50	2.95	0.83	0.21
Proto	2.83	21.80	5.36	1.33	0.41
Re^{3}Sum	2.83	22.03	5.62	1.50	0.34
PAAG	2.02	16.22	3.57	1.03	0.28
MPAG	**3.96**▲	**24.25**▲	**6.68**▲	**2.09**▲	**0.73**▲
Sentence extraction methods					
BM25	0.41	6.96	0.71	0.13	0.04
TF-IDF	0.25	5.55	0.51	0.08	0.02

(9) Re^{3}Sum is the text summarization model [8], which retrieves summaries and conducts template aware summary generation.

(10) RAGE is a review-driven e-commerce question answering model using convolutional sequence generation [9].

5.5   Implementation Details

All parameters in our model are randomly initialized. The number of K-means cluster is set to 3. The maximum number of reviews in each cluster is 20. The RNN-based networks have 512 hidden units and the dimension of a word embedding is 256. We limit the length of the question and answer sentence to 25 words and the review sentence to 30 words. The beam search algorithm is employed with a beam width of 4. We use Adagrad [15] to update the parameters with a learning rate of 0.1 and training batch size of 64. Our model is implemented via TensorFlow [1] framework and trained on an NVIDIA GTX 1080 Ti GPU.

6   EXPERIMENTAL RESULT

6.1   Overall Performance

At the beginning, we addressed research question **RQ1**. In Table 4, the significant differences are with respect to PAAG (row with shaded background). In these experimental results, we see that PAAG achieves a 111% increment over the state-of-the-art question answering baseline SNet in terms of BLEU, which demonstrates the effectiveness of using adversarial training method and incorporating product attributes. For our newly proposed model MPAG, we can see that MPAG achieves a 93.18% increase over the state-of-the-art baseline PAAG in terms of BLEU, and the improvements are all significant (with p-value < 0.05). As for the prototype-based baselines Proto and Re^{3}Sum, they all outperform our previously proposed model PAAG. This suggests that introducing prototype answer and employing the novel memory network can help the model to generate better answers. Despite the prototype-based methods obtain the help from the prototype question-answer pairs, these methods can not beat our proposed MPAG, since they fail to fully utilize the prototype answer due to their lack of reasoning ability.

We also employ the embedding metric [16] as another automatic evaluation metric which goes beyond simple N-gram matches. From the experiment in Table 5, we find that MAPG achieves

Table 5. 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
MAPG	**0.526868**▲	**139.3584**▲	**0.432037**▲
Sentence extraction methods			
BM25	0.325946	76.814465	0.172976
TF-IDF	0.308293	85.020442	0.155390

Table 6. Human Evaluation Comparison with Main Baseline

	Fluency	Consistency	Correctness
PAAG	2.75	2.52	69.0%
MPAG	**2.86**▲	**2.78**▲	**91.0%**▲

a 24.20% increase over the state-of-the-art baseline PAAG in terms of Average, and the improvements are all significant (with p-value < 0.05). The embedding-based metric measures the semantic matches between the generated answer and ground-truth answer, and this suggests that our newly proposed model can generate answers with high semantic consistency with the ground-truth answer.

For human evaluation, annotators rate each generated answer according to two objectives: (1) *Consistency.* Is the meaning of the answer consistent with the question? (2) *Fluency.* Is the generated answer well-written? The rating score ranges from 1 to 3, with 3 being the best. The results are shown in Table 6, and MPAG outperforms PAAG by 4.0% and 10.3% in terms of fluency and consistency. The paired student t-test demonstrates the significance of the above results. The kappa statistics are 0.56 and 0.53 for fluency and consistency respectively, which indicates moderate agreement between annotators.[1]

6.2 Effect of Clustering

In this section, we address the research question **RQ2**. To verify the effectiveness of the review clustering, we randomly aggregate the reviews into three clusters instead of using the K-Means algorithm, and feed the review clusters to the model named MPAG−K. The automatic evaluation results in Table 7 show that the performance of MPAG−K decreases by 4.75% compared with MPAG, in terms of BLEU1. This demonstrates the necessity of clustering reviews into the corresponding aspect.

6.3 Effect of Reasoning Module

Next, we turn to the research question **RQ3**. In addition to the ablation study in Table 7, where MPAG−M decreases by 9.63% compared with MPAG in terms of BLEU, we also compare reasoning module with DNC and SDNC. The fact that SDNC performs better than DNC demonstrates that vanilla DNC is not suitable for this scenario though it consists of complicated structures such as temporal

---

[1]Landis and Koch [35] 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 perfect agreement.

Table 7. Automatic Evaluation Comparison between Ablation Models

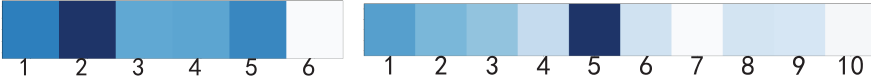|	BLEU	BLEU1	BLEU2	BLEU3	BLEU4
MPAG-P	2.13	19.47	4.14	1.03	0.25
MPAG-M	3.61	23.04	6.37	1.95	0.59
MPAG-K	3.54	23.15	6.35	1.82	0.58
DNC	3.41	22.69	6.10	1.72	0.57
SDNC	3.53	23.28	6.29	1.81	0.58

Fig. 6. Visualizations of salience weights over several reviews. The number is the review index and the review contents are listed in Section 6.3.1.

memory linkage and dynamic memory allocation. However, SRU further improves the BLEU score by 12.24%, compared with SDNC, which further verifies its superiority.

Note that, in Section 4.3, the SRU-based network learns to assign high weights to reviews that contain more useful information related to the question, and thus the saliency weight makes MPAG an explainable answer generator. Next we conduct two experiments to examine whether the saliency weight can faithfully reflect the importance of each review.

6.3.1   Visualization of Salience Weight. We first visualize the saliency weights of a review cluster in Figure 6 and determine whether it is in accordance with human intuition. The darker the color, the more important the corresponding review. For the top figure in Figure 6, the question is "鱼竿质量怎么样" (What is the quality of this fishing rod?). The second review is "质量很好, 手感不错, 轻巧" (The quality of the fishing rod is very good, feeling good, lightweight) and has the highest weight, while the sixth review is "物品已收到, 买一送一, 手竿比较轻, 比较软, 等有时间再去用用看质量如何" (Product received, buy one get one free. The handcuffs are light and soft. I will examine the quality when I have time to fish), which has the lowest weight. This is consistent with human intuition.

In the bottom of Figure 6, the question is "大家觉得这款剃的干净吗? 尤其是死角的地方" (Do you think this razor can shave cleanly? Especially the corners of the face). We can see that the fifth review has a higher weight than the seventh review. The fifth review is "送给老公的礼物, 非常好, 剃得很干净" (I use it as the gift for husband, very good, shaved very cleanly) and the seventh review is "剃须刀很漂亮, 用的也很顺手, 剃须功能棒棒的, 很喜欢这款" (The razor is very beautiful, easy to use, it is great, I like it very much). We can easily see that, the fifth review is more useful than the seventh with respect to the question.

6.3.2   Quantitative Usefulness Evaluation. It is intuitive to conduct a pairwise evaluation on whether our high weight reviews are as helpful as the rated useful ones selected by metrics based on word similarity such as BM25. Hence, we randomly select 100 data samples from two groups of reviews for comparison. Specifically, we choose the review with the highest weight score selected by MPAG (R_a) and BM25 (R_b), along with the question. The order of these two reviews is randomly shuffled and four choices are listed to annotators: (1) R_a is more useful than R_b; (2) R_b is more useful than R_a; (3) R_a and R_b are almost the same, both useful; and (4) R_a and R_b are almost the same, both useless.

The pairwise evaluation results are shown in Figure 7. For 49.5% of data samples, R_a is more useful than R_b. For 10.3% of data samples, the annotators think both reviews are useful and find
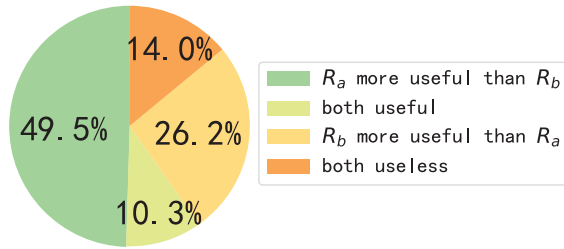
Fig. 7. Pairwise explanation annotation result.

Table 8. Diversity Evaluation Comparison with Baselines

	Distinct-1	Distinct-2	Distinct-3	Distinct-4
PAAG	0.0310	0.1129	0.2299	0.3495
Re³Sum	0.0291	0.1299	0.2826	0.4429
Proto	0.0273	0.1364	0.2946	0.4535
RAGE	0.0377	0.0476	0.1945	0.4439
MPAG	**0.0392**	**0.1902**	**0.3959**	**0.5763**

it hard to judge which is better. Finally, only 14% of data samples, the given explanation is useless. Therefore, we conclude that, in most cases, MPAG achieves good performance in providing explanations.

6.4 Effect of Answer Editor

Lastly, we address the research question **RQ4**. In Table 7, compared with MPAG, the performance of MPAG-P drops most by 24.61% in terms of BLEU1. This observation suggests that prototype question-answer pairs are helpful, and our model successfully learns how to utilize them. To examine whether the prototype method can alleviate the safe answer problem, we first evaluate by the distinct metric as shown in Table 8. MPAG outperforms the main baseline PAAG by 26.45% and 68.46% in terms of Distinct-1 and Distinct-2, respectively. Furthermore, human evaluation is also conducted to evaluate the proportion of safe answers in generation results. About 25% of the PAAG outputs are the safe answers, while our model produces only 20% safe answers, a great decrease. Nearly 95% of answers generated by our model are distinct, which makes our model far more practical than PAAG. Moreover, the corresponding kappa score for the inter-annotator agreement is 0.37. The above experiments demonstrate MPAG is indeed helpful for alleviating the safe answer problem.

Since the prototype context improves the performance of MPAG by such a large margin, it is possible that the prototype answer is already a good answer to the question, and MPAG directly copies the prototype answer as output. To examine whether the improvement is brought by the prototype answer or the prototype editing module, we calculate the BLEU scores between the prototype answer and the ground truth answer, and obtain 2.74, 18.63, 4.34, 1.21, 0.58 in terms of BLEU and BLEU1 to BLEU4, respectively. In contrast, the scores of MPAG are 3.96, 24.25, 6.68, 2.09, 0.73, i.e., higher than the scores obtained by the prototype answer, in all metrics. This means that the original prototype answer is not good enough and our answer editor module has an efficient revision ability.

To further investigate the editing module, we visualize the editing gates γ_t in Equation (33) and randomly pick one case, as shown in Figure 8. The question for this sample is "我, 一米六五身材, 体重140斤, 请问我能穿吗?" (My height is 1.65 meters, and my weight is 150 kg. Can I wear it?) and the prototype answer is "能" (It is ok). The generated answer is "能穿, 很舒服" (It is ok to wear, very comfortable). In Figure 8, we can see that the last word "舒服" (comfortable) has the

	能	穿	，		很	舒服
	OK	Wear			Very	Comfortable

Fig. 8. Visualization of editing gates. The darker the color is, the more information of this word comes from reasoning context vector.

Table 9. Examples of the Context and Answers

Reviews	很实用, 可以倒开水, 买了俩
	(Very practical, can be filled with boiled water, I bought two)
	不漏水, 用开水泡茶一样不漏
	(This cup does not leak water, and it does not leak either when you use boiling water to make tea.)
	装开水两次第二次杯底破了!
	(I used this cup to fill the water twice, and the bottom of the cup broke when I opened the water for the second time!)
	杯子很漂亮, 打开水也不烫手
	(The cup is very beautiful, and using this cup to put the boiling water will not burn your hands.)
	玻璃很薄, 买来后用开水烫了一下, 就放起来了
	(This glass is very thin. I bought it and washed it with boiling water. Then I put it away.)
Attributes	功能:带杯套, 花色: 无色 透明, 国产 / 进口:国产, 形状: 圆形, 数量: 1 个, 分类: 玻璃杯, 容量: 301-400ml
	(Function: glass cup with a cover, Color: colorless, transparent, Domestic / Import: Domestic, Shape: round, Quantity: 1, Category: Glass, Capacity: 301-400ml)
Prototype question	可不可以放猪血
	(Can it be used to put pig blood?)
Prototype answer	可以, 当然可以
	(Yes, of course.)
Question	可以 用 开水 泡茶 吗
	(Can I use this cup to put boiling water and make tea?)
Reference	可以
	(Yes, you can.)
PAAG	不 可以, 我 用 的是 开水
	(No, I put boiling water in it.)
MPAG	当然 可以, 我 就 是 盛开水
	(Yes, of course, I use this cup to put boiling water.)

highest editing weight, which is consistent with the fact that comfortable is a reasoning result from reviews. In contrast, the first word "能" (ok) has a low editing weight, because it can be directly copied from the prototype answer.

6.5 Case Study

We also show a case study in Table 9Table 9, which includes a question, representative reviews, prototype question-answer pair, and answers generated by different models. MPAG adapts the

prototype answer to the new context and generate the answer, which is a correct answer proved by user experience. In contrast, PAAG gives the answer "no" and then generates "I put boiling water in it", that makes up an inconsistent sentence confusing the readers. We can see that the generated answer of our MPAG can effectively generate reasonable and fluent answers.

7 CONCLUSION

In our previous work, we propose the task of generating answers of product-related questions using custom reviews and product attributes, and we also propose an adversarial learning method which employs an attention-based review reader to extract question-aware facts from reviews and attributes and finally generate an answer. Although the Wasserstein distance-based adversarial learning method is used to training the model which can reduce the probability of generating meaningless answers, the model still tends to generate safe answers. And the model lacks of reasoning ability when extracts facts from custom reviews, which is necessary for generating accurate answer.

Motivated by these observations, in this paper, we proposed the *Meaningful Product Answering Generator* (MPAG), which aims to generate a meaningful and diverse answer based on product attributes and reviews. Specifically, we employed a clustering algorithm to aggregate the reviews into several clusters, then we used a selective reading mechanism and read-write memory to encode these reviews so as to reason among them. We also used a key-value memory network to encode the product attributes. To alleviate the safe answer problem, we incorporated a prototype question-answer pair to extract answer skeletons. Finally, we combined all the intermediate results into an RNN-based decoder to generate the answer. Extensive experiments on a large-scale, real-world dataset showed that MPAG outperforms the state-of-the-art baselines and verified the effectiveness of each module in MPAG. Besides, pairwise experiments demonstrated that MPAG is able to provide a reasonable explanation why the generated answer holds such an opinion.

In future work, we are looking forward to introducing user profile features to the model to provide personalized services.

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] Ghodai Abdelrahman and Qing Wang. 2019. Knowledge tracing with sequential key-value memory networks. In *SIGIR*.

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

[4] Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *EMNLP*. 4220–4230.

[5] Deng Cai, Yan Wang, Victoria Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2018. Skeleton-to-response: Dialogue generation guided by retrieval memory. *arXiv preprint arXiv:1809.05296* (2018).

[6] Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *EMNLP*.

[7] Yu Cao, Meng Fang, Baosheng Yu, and Joey Tianyi Zhou. 2020. Unsupervised domain adaptation on reading comprehension. In *AAAI*.

[8] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*, Vol. 1. 152–161.

[9]   Shiqian Chen, Chenliang Li, Feng Ji, Wei Zhou, and Haiqing Chen. 2019. Review-driven answer generation for product-related questions in e-commerce. In *WSDM*.

[10] Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. 2019. Learning towards abstractive timeline summarization. In *IJCAI*.

[11] Junyoung Chung, Caglar Gülcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv* abs/1412.3555 (2014).

[12] Robert Csordas and Juergen Schmidhuber. 2019. Improving differentiable neural computers through memory masking, de-allocation, and link distribution sharpness control. In *the proceedings is ICLR 2019*.

[13] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neutral networks for reading comprehension. In *ACL*.

[14] Dheeru Dua, Sameer Singh, and Matt Gardner. 2020. Benefits of intermediate annotations in reading comprehension. In *ACL*.

[15] 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.

[16] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *NIPS*, Vol. 2.

[17] Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao, and Rui Yan. 2019. Learning towards abstractive summarization through prototype editing. In *EMNLP*.

[18] 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*.

[19] Shen Gao, Xiuying Chen, Chang Liu, Li Liu, Dongyan Zhao, Rui Yan, Shen Gao, Xiuying Chen, Chang Liu, Li Liu, Dongyan Zhao, and Rui Yan. 2020. Learning to respond with stickers: A framework of unifying multi-modality in multi-turn dialog. In *WWW*.

[20] Shen Gao, Xiuying Chen, Li Liu, Dongyan Zhao, and Rui Yan. 2020. Learning to respond with your favorite stickers: A framework of unifying multi-modality and user preference in multi-turn dialog. *ACM Trans. Inf. Syst.* (2020).

[21] Shen Gao, Xiuying Chen, Zhaochun Ren, Dongyan Zhao, and Rui Yan. 2020. From standard summarization to new tasks and beyond: Summarization with manifold information. In *IJCAI*.

[22] Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *WSDM*.

[23] Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in E-commerce question-answerings. In *WSDM*.

[24] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *EMNLP*.

[25] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538, 7626 (2016), 471.

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

[27] Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C. Lipton. 2019. AmazonQA: A review-based question answering task. *ArXiv* abs/1908.04364 (2019).

[28] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL* 6 (2018), 437–450.

[29] T. Hashimoto, K. Guu, Y. Oren, and P. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *NIPS*.

[30] Johan Hasselqvist, Niklas Helmertz, and Mikael Kågebäck. 2017. Query-based abstractive summarization using neural networks. *CoRR* abs/1712.06100 (2017).

[31] 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*.

[32] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2019. Read + Verify: Machine reading comprehension with unanswerable questions. In *AAAI*.

[33] Dongmin Hyun, Chanyoung Park, Min-Chul Yang, Ilhyeon Song, Jung-Tae Lee, and Hwanjo Yu. 2018. Review sentiment-guided scalable deep recommender system. In *SIGIR*. ACM, 965–968.

[34] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICLR*. 1378–1387.

[35] J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* (1977), 159–174.

[36] Hung Le, Truyen Tran, and Svetha Venkatesh. 2019. Learning to remember more with less memorization. *arXiv preprint arXiv:1901.01347* (2019).

[37] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*.

[38] Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *ACL*.

[39] 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.

[40] Todor Mihaylov and Anette Frank. 2019. Discourse-aware semantic self-attention for narrative reading comprehension. In *EMNLP*.

[41] 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.

[42] Samaneh Moghaddam and Martin Ester. 2011. AQA: Aspect-based opinion question answering. In *the proceedings is ICDM*. 89–96.

[43] 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).

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*.

[45] Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

[46] Lisong Qiu, Juntao Li, Wei Bi, Dongyan Zhao, and Rui Yan. 2019. Are training samples correlated? Learning to generate dialogue responses with multiple references. In ACL.

[47] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

[48] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[49] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

[50] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

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

[52] 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*.

[53] 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. DOI : https://doi.org/10.1145/3018661.3018704

[54] Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *CICLing*.

[55] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

[56] Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *NAACL*.

[57] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[58] 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*.

[59] 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.

[60] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

[61] Yi Tay, Shuohang Wang, Luu Anh Tuan, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narrativestsimple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *ACL*.

[62] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *TACL* (2018).

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

[64] Junbo Wang, Wei Wang, Yan Huang, Liang Wang, and Tieniu Tan. 2018. M3: Multimodal memory modelling for video captioning. In *CVPR*.

[65] Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *ACL*.

[66] Ke Wang and Xiaojun Wan. 2018. Sentiment analysis of peer review texts for scholarly papers. In *SIGIR*. ACM, 175–184.

[67] Wenbo Wang, Yang Gao, Heyan Huang, and Yuxiang Zhou. 2019. Concept pointer network for abstractive summarization. In *EMNLP*.

[68] Wenjie Wang, Minlie Huang, Xin-Shun Xu, Fumin Shen, and Liqiang Nie. 2018. Chat more: Deepening and widening the chatting topic via a deep model. In *SIGIR*. ACM.

[69] 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.

[70] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* (2015).

[71] Derek Wu and Hongning Wang. 2017. ReviewMiner: An aspect-based review analytics system. In *SIGIR*. ACM, 1285–1288.

[72] Yu Ping Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. *CoRR* abs/1806.07042 (2018).

[73] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, Vol. 17. 3351–3357.

[74] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*. 2397–2406.

[75] Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. In *NAACL*.

[76] 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*.

[77] Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *EMNLP*.

[78] 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*.

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

[80] Qian Yu and Wai Lam. 2018. Review-aware answer prediction for product-related questions incorporating aspects. In *WSDM*. ACM, 691–699.

[81] Hainan Zhang, Yanyan Lan, Liang Pang, Hongshen Chen, Zhuoye Ding, and Dawei Yin. 2020. Modeling topical relevance for multi-turn dialogue generation. In *IJCAI*.

[82] Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. 2020. Low-resource knowledge-grounded dialogue generation. In *ICLR*.