Answer Identification from Product Reviews for User Questions by Multi-Task Attentive Networks

Long Chen,[1] Ziyu Guan,[1*] Wei Zhao,[2] Wanqing Zhao,[1] Xiaopeng Wang,[1] Zhou Zhao,[3] Huan Sun[4]
[1]Northwest University, Xi'an, China
[2]Xidian University, Xi'an, China
[3] Zhejiang University, Hang Zhou, China
[4]Ohio State University, Columbus, USA
[1]{longchen@stumail., ziyuguan@, zhaowq@, wangxiaopeng@stumail.}nwu.edu.cn
[2]ywzhao@mail.xidian.edu.cn, [3]zhaozhou@zju.edu.cn, [4]sun.397@osu.edu

Abstract

Online Shopping has become a part of our daily routine, but it still cannot offer intuitive experience as store shopping. Nowadays, most e-commerce Websites offer a Question Answering (QA) system that allows users to consult other users who have purchased the product. However, users still need to wait patiently for others' replies. In this paper, we investigate how to provide a quick response to the asker by plausible answer identification from product reviews. By analyzing the similarity and discrepancy between explicit answers and reviews that can be answers, a novel multi-task deep learning method with carefully designed attention mechanisms is developed. The method can well exploit large amounts of user generated QA data and a few manually labeled review data to address the problem. Experiments on data collected from Amazon demonstrate its effectiveness and superiority over competitive baselines.

Introduction

Today, people become used to online shopping for convenience. However, such kind of virtual services cannot give customers the same intuitive experience as store shopping in which they can directly interact with the real products and salesclerks. This makes the shopping process in online services less efficient and less trustworthy. To this end, most e-commerce Websites now offer a Question Answering (QA) system that allows users to consult other users who have purchased the product. Despite its helpfulness, users still need to wait patiently for others' replies. On the other hand, customer reviews are an invaluable source of information where the user question may happen to be addressed therein. Nevertheless, it is unrealistic for a user to browse through the massive customer reviews for answers. It would greatly enhance user experience that we identify possible answers automatically from reviews and present them to users instantly.

Product-Related Question Answering (PRQA) in e-commerce Websites has received attention. Traditional PRQA works were focused on generating answers from reviews by unsupervised heuristic methods (Li et al. 2009; Moghaddam and Ester 2011; Yu, Zha, and Chua 2012), which cannot well handle sophisticated human language. Recently, the accumulation of QA data in the QA systems of e-commerce Websites has motivated research about supervised PRQA methods (McAuley and Yang 2016; Wan and McAuley 2016). However, these methods just used reviews as supporting data for answer prediction. They do not try to identify answers from reviews. To our knowledge, we are the first to address PRQA by detecting answers from reviews based on supervision on user generated QA data.

Table 1: Questions (Q) with answers (A) and positively labeled review sentences (R) for two products (P) in Amazon.

P	*Emerson EM510 Stereo Wireless Headset - Bluetooth Headset - Retail Packaging - Black*
Q:	How long does the **battery last**?
A:	Around 8+ hours - I use it and approx one hour a day M-F and only charge it every other week at the earliest.
R:	I work long hours and the **battery last** 10 to 12 hours.
P	*Cheerwing Syma X5SW-V3 FPV Explorers2 2.4Ghz 4CH 6-Axis Gyro RC Headless Quadcopter Drone UFO with HD Wifi Camera (White)*
Q:	**What happens** when drone flies **out of range** of controller?
A:	It will just fall so keep it at a low level if it goes too far.
R:	After the drone went **out of range** and didn't stop and was lost in the woods, I contacted the company and they are issuing a new one to us.

However, the distributions of explicit answers and review contents that can address the corresponding questions are not quite the same. The key difference is that, when explicitly answering a question, people tend to omit keywords about the *focus* of the question. By "focus", we mean the specific topic the asker is concerned with. This is intuitive, since repeating the focus of the question is usually unnecessary. Table 1 shows two examples. Words related to the focus are in bold face. The focus of the two questions are "battery

*Corresponding author

life" and "behavior of the UAV when it is out of range of the controller", respectively. Unlike explicit answers, positive review sentences (i.e. sentences that can be answers) often contain focus-related words. Therefore, it would not be a good idea to simply train answer identification models for reviews by supervision on user generated QA data. On the other hand, since the question focuses can be diverse, it is relatively hard to find positive Question-Review sentence (QR) pairs. In our labeled QR dataset, the percentage of positive QR pairs is below 10%. This means it is hard to get enough QR training data to capture the complex mapping relation between them. Fortunately, positive review sentences do share similar patterns (called *answer pattern*) with explicit answers, e.g. boxed contents in Table 1. The large-scale QA data accumulated in e-commerce Websites could be useful for capturing the mapping relation between questions and positive review sentences.

Question focuses are different from "aspects" in opinion mining (Liu 2012) or "topics" in social network analysis (Li et al. 2016; Wang et al. 2013) since they can be complicated and at different granularities (see examples in Table 1). This renders traditional aspect/topic extraction techniques inapplicable here. Moreover, keyword mapping is not a good solution either since expressions in reviews are very diverse and context dependent. In this work, we devise a novel multi-task deep learning method which can well exploit both user generated QA data and manually labeled QR pairs to train an end-to-end deep model for answer identification in review data. We treat the QR binary prediction problem (can answer or not) as the main task, with QA binary prediction as an auxiliary task. The two tasks are abbreviated as *QR task* and *QA task* respectively. The model consists of three sub-networks named Q-subnet, A-subnet and R-subnet respectively. The three sub-networks all employ Bidirectional Gated Recurrent Unit (Bi-GRU) (Cho et al. 2014) to extract hidden features of word sequences, based on which proper attention techniques are devised to generate high-level embeddings of texts. Specifically, we put a bi-directional attention (bi-attention for short) module (Seo et al. 2016) between Q-subnet and A-subnet to learn the mapping relation between question focuses and answer patterns; a self-attention module (Lin et al. 2017) is used to extract focus-related content from review sentences in R-subnet. For the QA task, the QA pair is fed to Q-subnet and A-subnet, and the output embeddings are fused for prediction. In the QR task, the question is also fed to Q-subnet, while the review sentence is inputted to both A-subnet and R-subnet, in order to extract answer patterns and focus-related content in it respectively. The outputs are fused for the final prediction. The two tasks share Q-subnet and A-subnet, so that useful information (i.e. the mapping between question focuses and answer patterns) can be transferred from the auxiliary task to the main task. We further use regularization to better help the self-attention module capture focus-related content in reviews. The proposed multi-task deep learning method (and the overall network architecture) is dubbed *QAR-net*.

The contributions of this work are: (1) we propose to address the PRQA problem by identifying plausible answers in reviews and develop a novel multi-task deep learning method, QAR-net, which can well leverage both large-scale user generated QA data and manually labeled QR data to achieve this goal. (2) we construct a manually labeled QR dataset from the Amazon dataset (McAuley and Yang 2016) which has 449 positive instances and 7,017 negative instances for model training and evaluation. (3) we perform thorough experiments based on the publicly available Amazon dataset (McAuley and Yang 2016) to show the effectiveness of QAR-net[1] and its superiority over baseline methods.

Related Work

Questing Answering. Our problem falls into the research area of Questing Answering (QA). Different QA variant problems have been studied, including Knowledge Base QA (KBQA) (Bordes, Chopra, and Weston 2014), Community QA (CQA) (Nakov et al. 2017), answer selection (dos Santos et al. 2016) and Reading Comprehension QA (RCQA) (Xiong, Zhong, and Socher 2016). Our problem is obviously different from KBQA where a knowledge base is required. CQA is concerned with finding answers for a user submitted question from answers (called comments) of existing questions. Similarly, answer selection tries to rank a set of candidate answers for a given question so that the correct answers are ranked the highest. RCQA aims to answer questions by finding answer spans in a passage or document. The latter three problems are more similar to ours. However, the key difference is that they only have one distribution for the answer side, while in our context we need to deal with two distributions (i.e. explicit answers and reviews).

Researcher also tried to attack the PRQA problem. Li *et al.* (Li et al. 2009) constructed a random walk model to rank review sentences by considering questions' topics and sentiments. Moghaddam and Ester (Moghaddam and Ester 2011) proposed a solution for PRQA which considered aspect information and performed ad-hoc analysis of questions and reviews. Yu *et al.* (Yu, Zha, and Chua 2012) proposed a similar solution based on hierarchical organization of reviews according to aspect granularity. The above methods are all based on heuristics and hand-designed rules, due to lack of training data. It is difficult to model sophisticated human language by heuristics. Recent work for PRQA has begun to use user generated QA data for model training. McAuley and Yang (McAuley and Yang 2016) developed a mixtures-of-experts framework called Moqa to address PRQA. Later, Moqa was extended to handle questions with multiple answers (Wan and McAuley 2016). These works are different to ours in that they predict the answer for a question from candidate answers ("Yes" or "No" for binary questions; ranking answers before non-answers for open-ended questions), by using review sentences as supporting "experts", while we identify plausible answers from reviews *without candidate answers*. Hence, Moqa is not applicable to our setting.

Attentive Learning in NLP Tasks. Attentive neural networks have achieved great success in many NLP tasks, e.g. machine translation (Vaswani et al. 2017), textual entailment

[1]Code & dataset: https://github.com/chen89/PRQA-MultitaskAttentiveNet.

recognition (Rocktäschel et al. 2015) and QA (dos Santos et al. 2016; Wang et al. 2017; Xiong, Zhong, and Socher 2016). Word-by-word attention is often used in machine translation (Vaswani et al. 2017) and textual entailment (Rocktäschel et al. 2015) since word-by-word alignment features are important for them. In RCQA, additional Recurrent Neural Networks (RNNs) are usually built upon the word-by-word (co)-attention layer to compute a question-aware representation of passages (Wang et al. 2017; Xiong, Zhong, and Socher 2016). This is helpful for processing long text sequences. In answer selection, the bi-attention pooling technique (dos Santos et al. 2016) has achieved state-of-art performance. Since we do not deal with long texts and word-by-word alignment is not our concern either, we choose to use bi-attention pooling to extract salient mapping features between questions and answers. Moreover, a self-attention module (Lin et al. 2017) together with regularization is developed to capture focus-related content in reviews.

Multi-task Learning. Multi-task learning is deemed to be an effective learning paradigm for boosting the performance of tasks with insufficient training instances by related tasks with abundant training data. The key idea is to share common "knowledge" between related tasks. This is usually done by sharing model parameters (hard or soft). Readers can refer to (Ruder 2017) for a recent survey. Hard-sharing is done through common model structures. In the NLP field, Bonadiman *et al.* divided the CQA problem into three subtasks and developed a partially hard-sharing deep model (Bonadiman, Uva, and Moschitti 2017). The model requires all the inputs for the three subtasks to be present, which limits its applicability. Soft-sharing constructs separate networks for different tasks and uses regularization (e.g. L2 norm (Duong et al. 2015), trace norm (Yang and Hospedales 2016)) to force different networks to be similar. In this work, we develop a partially hard-sharing network which explicitly tries to transfer the mapping relations between question focuses and answer patterns from the auxiliary task to the main task.

The Method

This section presents QAR-net. We first formally define the problem and notations. Then we describe the model architecture in details. Finally, we discuss how to train QAR-net.

Problem Formulation

We are given a set of labeled QR pairs $\mathcal{S}^{qr} = \{(Q, R, y^{QR})\}$, where Q is a user submitted question, R is a review sentence and $y^{QR} \in \{0, 1\}$ is the binary label indicating whether R can answer Q. In this work we take sentences as the basic unit of reviews. The reason is that, when writing a review users usually comment on multiple aspects of the product. Treating whole reviews as the basic unit is not reasonable, since the focus of Q usually belongs to one aspect. One possible solution is to segment reviews according to aspect analysis of sentences. However, this would introduce additional issues if the aspect detection method is not accurate enough[2].

Besides, we also have an automatically labeled set of QA pairs $\mathcal{S}^{qa} = \{(Q, A, y^{QA})\}$ obtained from user generated QA data, where $y^{QA} \in \{0, 1\}$ is defined similarly as above. The problem is to train a classification model from $\mathcal{S}^{qr}$ and $\mathcal{S}^{qa}$ for the QR task, so that for a new pair $(Q^{\text{new}}, R^{\text{new}})$ the model can accurately predict whether R^{new} can answer Q^{new}. Note that unlike the answer selection problem (dos Santos et al. 2016), we formulate the problem as a classification problem rather than a ranking problem. For answer selection it is usually assumed that at least one answer exists in the candidates, while in our context there could be no answer at all. Setting a "yes/no" threshold for ranking scores is difficult.

Model Architecture

The model architecture is depicted in Figure 1. Horizontally, the low level part consists of three subnetworks: A-subnet, Q-subnet and R-subnet. Q-subnet and A-subnet are shared between the QR task (main task) and the QA task (auxiliary task), in order to extract common patterns of the two tasks, i.e. focus from questions and answer patterns from answers/review sentences. R-subnet is exclusive for the QR task which is used to extract focus-related content from review sentences. Intuitively, R-subnet provides complementary information to A-subnet for matching questions and review sentences. Vertically, QAR-net can be divided into four steps: text embedding, attentive pooling, fusion and output. In the following, we detail these steps.

Text Encoding. We abstract questions, answers and reviews as sequences of words, i.e. $Q =< w_t^Q >_{t=1}^n$, $A =< w_t^A >_{t=1}^m$ and $R =< w_t^R >_{t=1}^l$. Since the operations in this step are shared among Q, A and R, we omit the superscripts temporally and simply consider a sequence $< w_t >$[3]. Each word w_t in the sequence is first converted to its respective word-level embedding vector $\mathbf{e}_t$ and character-level embedding vector $\mathbf{c}_t$. For word-level embeddings, we use pre-trained GloVe embeddings (Pennington, Socher, and Manning 2014). Similar to (Wang et al. 2017), the character-level embedding vector $\mathbf{c}_t$ is from the final hidden states of a bi-directional RNN (bi-RNN) applied to w_t's characters and trained with the model. Character-level embeddings can help handle out-of-vocabulary words. We concatenate $\mathbf{e}_t$ and $\mathbf{c}_t$ to represent w_t and then feed the word sequence to a bi-GRU network to generate low level representation $\mathbf{h}_t$ for each word. Compared to other RNN models such as Long Short-Term Memory (LSTM), GRU is computational efficient and can achieve competitive performance (Wang et al. 2017). The computation of bi-GRUs can be formulated as:

$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{GRU}(\overrightarrow{\mathbf{h}_{t-1}}, \mathbf{e}_t \oplus \mathbf{c}_t). \quad \overrightarrow{\mathbf{h}_t} \in \mathbb{R}^u \tag{1}$$

$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{GRU}(\overleftarrow{\mathbf{h}_{t+1}}, \mathbf{e}_t \oplus \mathbf{c}_t). \quad \overleftarrow{\mathbf{h}_t} \in \mathbb{R}^u \tag{2}$$

[2]We do try to merge adjacent sentences if we are sure that they belong to the same aspect. We still call them sentences for clarity.

[3]However, we use different Bi-GRUs for Q, A and R to capture their unique characteristics.

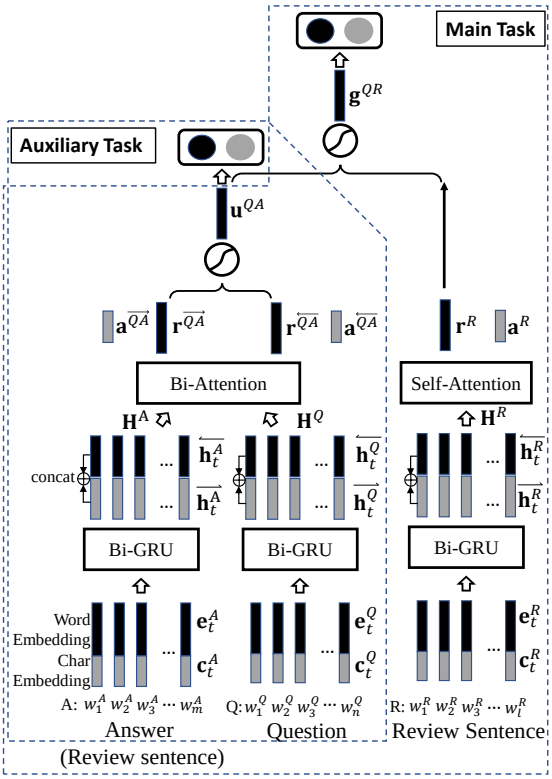

Figure 1: The architecture of QAR-net.

where operator $\oplus$ denotes the concatenation of vectors, $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ represent forward and backward hidden states of bi-GRU respectively. We then concatenate $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ for each word w_t to get the complete hidden state $\mathbf{h}_t \in \mathbb{R}^{2u}$. The hidden states provide contextualized representations of words. We store $\mathbf{h}_t$'s of a sequence as column vectors of a matrix. For Q, A and R, they are $\mathbf{H}^Q \in \mathbf{R}^{2u \times n}$, $\mathbf{H}^A \in \mathbf{R}^{2u \times m}$ and $\mathbf{H}^R \in \mathbf{R}^{2u \times l}$ respectively.

Attentive Pooling. As shown in Table 1, not all words are equally important for QA/QR matching. The purpose of the attentive pooling step is to extract salient patterns from questions, answers and reviews, so that more efficient and accurate matching can be performed in the subsequent steps. To extract the mapping patterns between Q and A (R), we set a bi-attention module (dos Santos et al. 2016) between Q-subnet and A-subnet. We also put a self-attention module (Lin et al. 2017) in R-subset, in order to distill focus-related keywords in R.

Figure 2 shows an illustration of the bi-attention mechanism. It takes $\mathbf{H}^Q$ and $\mathbf{H}^A$ as input and outputs the fixed-length vector representations, $\mathbf{r}^{\overleftarrow{QA}}$ and $\mathbf{r}^{\overrightarrow{QA}}$, for Q and A respectively. Arrows here denote the direction of the attention. The first step is to compute an affinity matrix $\mathbf{G}$ as follows

$$\mathbf{G} = \tanh((\mathbf{H}^A)^T \mathbf{U} \mathbf{H}^Q). \quad \mathbf{G} \in \mathbb{R}^{m \times n} \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{2u \times 2u}$ is a parameter matrix. Eq. (3) intrinsically provides a soft mapping between words in Q and A in

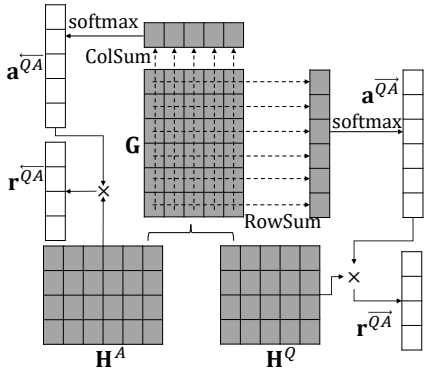

Figure 2: Illustration of the bi-attention mechanism.

terms of the obtained hidden vectors. The (i, j)-th entry of $\mathbf{G}$ reflects the similarity between $\mathbf{h}_i^A$ and $\mathbf{h}_j^Q$. $\mathbf{U}$ accounts for the correlation among the hidden dimensions. We then apply row-wise summation and column-wise summation on $\mathbf{G}$ to obtain score vectors $\mathrm{RowSum}(\mathbf{G})$ and $\mathrm{ColSum}(\mathbf{G})$, which are in turn used to calculate attention vectors $\mathbf{a}^{\overrightarrow{QA}}$ and $\mathbf{a}^{\overleftarrow{QA}}$:

$$\mathbf{a}^{\overrightarrow{QA}} = \mathrm{softmax}(\mathrm{RowSum}(\mathbf{G})). \quad \mathbf{a}^{\overrightarrow{QA}} \in \mathbb{R}^m \tag{4}$$

$$\mathbf{a}^{\overleftarrow{QA}} = \mathrm{softmax}(\mathrm{ColSum}(\mathbf{G})). \quad \mathbf{a}^{\overleftarrow{QA}} \in \mathbb{R}^n \tag{5}$$

Note we use sum pooling here rather than max-pooling (dos Santos et al. 2016). The reason is that in PRQA many questions are subjective ones and a few focus keywords may correspond to a number of answer-related keywords. Sum pooling could better reflect the overall importance of words. Finally, the attention-weighted representations of Q and A are calculated as:

$$\mathbf{r}^{\overrightarrow{QA}} = \mathbf{H}^A \mathbf{a}^{\overrightarrow{QA}}. \quad \mathbf{r}^{\overrightarrow{QA}} \in \mathbb{R}^{2u} \tag{6}$$

$$\mathbf{r}^{\overleftarrow{QA}} = \mathbf{H}^Q \mathbf{a}^{\overleftarrow{QA}}. \quad \mathbf{r}^{\overleftarrow{QA}} \in \mathbb{R}^{2u} \tag{7}$$

The self-attention mechanism on R-subnet is inspired by (Lin et al. 2017). In our case we do not use multiple attention vectors since we only want to extract focus-related content from R. The attention vector $\mathbf{a}^R$ and attention-weighted representation $\mathbf{r}^R$ are computed as follows:

$$\mathbf{a}^R = \mathrm{softmax}(\mathbf{v}^T \tanh(\mathbf{U}^r \mathbf{H}^R)). \quad \mathbf{a}^R \in \mathbb{R}^l \tag{8}$$

$$\mathbf{r}^R = \mathbf{H}^R \mathbf{a}^R. \quad \mathbf{r}^R \in \mathbb{R}^{2u} \tag{9}$$

where $\mathbf{U}^r \in \mathbb{R}^{k \times 2u}$ and $\mathbf{v} \in \mathbb{R}^k$ are parameters of the self-attention function with hyperparameter k. In order to encourage the self-attention module to extract focus-related keywords from R, we design in our objective function proper regularization terms which will be presented in the next subsection.

Fusion & Output. The purpose of fusion is to generate a uniform representation for the input QA (or QR) pair. We

take a strategy similar to (Rocktäschel et al. 2015) to perform fusion which has shown good performance for NLP tasks. We first fuse $\mathbf{r}^{\overrightarrow{QA}}$ and $\mathbf{r}^{\overleftarrow{QA}}$:

$$\mathbf{u}^{QA} = \tanh(\mathbf{W}^{\overrightarrow{qa}}\mathbf{r}^{\overrightarrow{QA}} + \mathbf{W}^{\overleftarrow{qa}}\mathbf{r}^{\overleftarrow{QA}}). \quad \mathbf{u}^{QA} \in \mathbb{R}^{2u} \tag{10}$$

where $\mathbf{W}^{\overrightarrow{qa}}$ and $\mathbf{W}^{\overleftarrow{qa}}$ are $2u \times 2u$ parameter matrices. For the QA task (auxiliary task), $\mathbf{u}^{QA}$ is used for label prediction

$$\hat{y}^{QA} = \sigma((\mathbf{w}^{qa})^T \mathbf{u}^{QA} + b^{qa}). \tag{11}$$

where $\sigma(\cdot)$ is the sigmoid function. For the QR task (main task), we further fuse $\mathbf{u}^{QA}$ with $\mathbf{r}^{R}$:

$$\mathbf{g}^{QR} = \tanh(\mathbf{W}^{qr}\mathbf{u}^{QR} + \mathbf{W}^{r}\mathbf{r}^{R}). \quad \mathbf{g}^{QR} \in \mathbb{R}^{2u} \tag{12}$$

Note that for the QR task $\mathbf{u}^{QA}$ is obtained based on R rather than A, i.e. $\mathbf{u}^{QR}$ in the above equation. The final output layer is similar to (11):

$$\hat{y}^{QR} = \sigma((\mathbf{w}^{qr})^T \mathbf{g}^{QR} + b^{qr}). \tag{13}$$

Model Training

The training objective functions for both QR and QA tasks are standard cross-entropy functions over the predictions $\{\hat{y}^{QR}\}$ and $\{\hat{y}^{QA}\}$ on training data

$$\mathcal{L}_{qr} = -\sum y^{QR}\log(\hat{y}^{QR}) + (1-y^{QR})\log(1-\hat{y}^{QR}) \tag{14}$$

$$\mathcal{L}_{qa} = -\sum y^{QA}\log(\hat{y}^{QA}) + (1-y^{QA})\log(1-\hat{y}^{QA}) \tag{15}$$

Additionally, for the main task we add regularization terms to encourage the self-attention module in R-subnet to capture focus-related content in reviews. Recall that review sentences are fed into both A-subset and R-subnet for attention extraction with different purpose. Hence, for a QR pair $(Q, R, y^{QR}) \in \mathcal{S}^{qr}$, we penalize the similarity between the two attention vectors:

$$\mathcal{R}_1 = \sum_{(Q,R,y^{QR})\in\mathcal{S}^{qr}} \|(\mathbf{a}^{\overrightarrow{QR}})^T \mathbf{a}^{R}\|^2 \tag{16}$$

Such a L_2 norm cost is shown to be better than Kullback-Leibler divergence for penalizing attention vectors (Lin et al. 2017). In this way, $\mathbf{a}^R$ is forced to capture keywords other than those related to answer patterns. Another regularization term is imposed on the attention-weighted representations of Q and R. Since for positive pairs we want both of them to capture the question focus, we penalize their Euclidean distance as follows

$$\mathcal{R}_2 = \sum_{(Q,R,y^{QR})\in\mathcal{S}^{qr}, y^{QR}=1} \|\mathbf{r}^{\overleftarrow{QR}} - \mathbf{r}^{R}\|_2 \tag{17}$$

In summary, the loss function for the main task then becomes

$$\mathcal{L} = \mathcal{L}_{qr} + \mu\mathcal{R}_1 + \eta\mathcal{R}_2, \tag{18}$$

where μ and η are regularization hyperparameters. QAR-net is trained with respect to both (15) and (18).

Training Strategy. Unlike the traditional Multi-task learning (MTL) setting, our two tasks do not share the same input, which means we cannot perform joint training. Hence, we adopt the "pre-training then fine-tuning" transfer learning scheme: the model is pre-trained on QA data and then we use labled QR pairs to fine-tune the whole model (R-subnet is randomly initialized). However, by preliminary experiments we find that it is not a good idea to sufficiently train QAR-net on QA data. This is because sufficient pre-training would fit A-subnet very well to the distribution of answers. Considering the much larger size of QA data, it is difficult to drag the model to fit the distribution of reviews in fine-tuning. Our problem setting is different from that of classic MTL problems such as MTL for multiple NLP tasks (Hashimoto et al. 2016), where tasks are equally important. In our problem, we only care about the main task. Therefore, we perform insufficient pre-training with the auxiliary task. In experiments, we will investigate how the pre-training degree affects the model performance for the main task.

Experiments

Dataset and Preprocessing

We collect QA pairs and reviews from randomly selected products of two domains, "Electronics" and "Cellphones & Accessories", in the Amazon dataset (McAuley and Yang 2016). For each question, the top-voted answer is used to form a positive QA pair, and we randomly select two answers from other products of the same domain to generate negative QA pairs. This results in totally 1,000,139 QA pairs. Regarding reviews, we use NLTK[4] to split them into sentences.

We then randomly select 10k questions from the two domains and combine them with review sentences for the respective products to generate QR pairs for labeling. As discussed earlier, positive QR pairs are rare. Hence, we perform pre-filtering to speedup labeling. Our assumption is that positive review sentences should at least overlap with the question in terms of aspect-related noun words (AN-words). We construct the set of AN-words for a domain according to the ratio of within-domain frequency to out-of-domain frequency (on a background text collection constructed from other unrelated domains) for each candidate noun word. An AN-word should have high within-domain frequency and low out-of-domain frequency. The threshold for finalizing the set of AN-words is decided by manual inspection. We filter out QR pairs that do not overlap in terms of AN-words. Note this also raises the difficulty of differentiating between answer and non-answer sentences since they all overlap with the question in terms of AN-words. After filtering, the total number of QR pairs is still large (nearly 0.5M), so we randomly select pairs for labeling (on the basis of questions to resist the power law effect of reviews). Each sampled QR pair is labeled by 3 trained student annotators. The annotators agree on 86% of all the sampled QR pairs. Disagreements are resolved by discussion. Finally, we get 449 positive QR pairs and 7,017 negative QR pairs. The final labeled set covers 5907 distinct questions. It is possible that a question corresponds to multiple positive/negative instances (QR pairs). We randomly split the QR dataset into training

[4] http://www.nltk.org/

set (4,000), validation set (977) and test set (2,489) with the same positive/negative proportion.

Implementation Details

We use Stochastic Gradient Decent (SGD) to train neural networks. We apply Adam (Kingma and Ba 2014) with initial learning rate set to 1e-6. The first and second momentum coefficients are set to 0.9 and 0.999 respectively (as suggested in (Kingma and Ba 2014)).We apply dropout (Srivastava et al. 2014) to the outputs of the Bi-GRU layer with dropout rate 0.5 to prevent overfitting. For fair comparison, the hyperparameters shared among different methods are set to the same values. Specifically, the length of character-level embedding vectors is set to 20; the hidden vector length of GRUs is set as $u = 32$. The mini-batch size for SGD is 128. Hyperparameters that are unique to each method are tuned on the QR validation set. Additionally, we employ a random under-sampling trick on the majority class (negative) in both QA and QR datasets. Different from the classic one-time under-sampling trick (Yan et al. 2015), we conduct this operation on each training epoch. Specifically, for each QA/QR training epoch, we randomly sample negative QA/QR pairs of the same number as the positive pairs and then combine them to form the training set for this epoch. This could alleviate the information loss issue of the classic under-sampling approach (He and Garcia 2009). For fair comparison, all the methods use this under-sampling trick to relieve the class imbalance problem in the datasets. All the MTL methods use the pre-training then fine-tuning training strategy. The number of pre-training epochs for each MTL method is determined on the QR validation set. The number of fine-tuning epochs is determined by early stopping on the validation set.

Compared Methods and Evaluation Metrics

QREM: question review embedding matching. We first convert questions and review sentences into matrices of word vectors by GloVe (Pennington, Socher, and Manning 2014). Then we perform row-wise max, min and average pooling, and concatenate the pooling results to generate fixed-length vectors of questions and review sentences. Finally, For given Q and R we use cosine similarity and threshold trained by logistic regression on the QR task to judge whether R can answer Q.

QAR-net: it is the solution of this paper.

QAR-net-QR: this one employs the QAR-net model but only trains QAR-net with the QR task.

QA-net-QA : this baseline takes the left part of QAR-net (called QA-net), i.e. components in the dotted-line box of the auxiliary task in Fig. 1. It is similar to a state-of-art network for answer selection (dos Santos et al. 2016). We only use QA pairs for training.

QA-net-QR : this baseline uses the same network as above but is trained on QR pairs only.

QA-net-MTL: it employs the QA-net as base network and is trained with both QA and QR tasks. It uses the traditional hard parameter sharing scheme for MTL with task-specific output layers (Ruder 2017).

CQA-hard: it is a variant of the MTL model for CQA (Bonadiman, Uva, and Moschitti 2017). We use 2 pairs of CNNs to process QA and QR pairs respectively, and feed the extracted feature vectors of the QA/QR pair to a fully connected (FC) layer shared by the two tasks.

NLP-soft: this is a soft-sharing MTL model originally designed for dependency parsing in different languages (Duong et al. 2015). We modify its input layer to take pairs as input.

Although we have abundant positive QA pairs, the labeled positive QR pairs are very rare. Since our focus is the positive class, we use precision, recall, F-measure and Precision-Recall curves to evaluate these methods. (He and Garcia 2009).

Table 2: Performance comparison

Method	Precision	Recall	F-measure
QREM	0.1838	0.1667	0.1748
QA-net-QA	0.3525	0.5733	0.4365
QA-net-QR	0.4158	0.5267	0.4647
QA-net-MTL	0.4000	0.5067	0.4471
CQA-hard	0.4103	0.2133	0.2807
NLP-soft	0.4343	0.2867	0.3454
QAR-net-QR	0.4247	0.5267	0.4702
QAR-net	**0.5385**	**0.6067**	**0.5705**

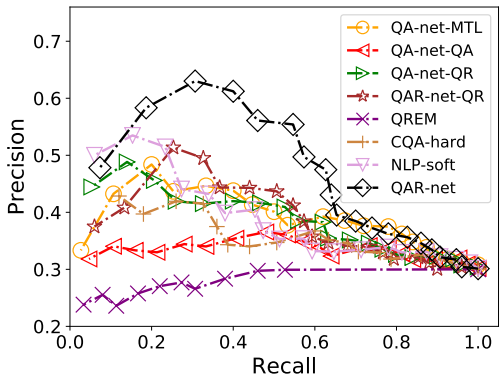

Figure 3: Precision-recall curves.

Main Results

The comparison results are shown in Table 2. QREM performs poorly, which indicates that a naive text embedding constructed by pooling and concatenation operations on pre-trained word embeddings cannot handle the problem. QA-net-QA achieves the lowest precision among all the deep learning baselines. This is because it is not trained on the QR task and therefore cannot enjoy performance boosting by exploiting the QR data. Its high recall indicates the two tasks indeed share common answer patterns. QA-net-QR and QAR-net-QR exhibit inferior performance compared to our QAR-net. This indicates only relying on QR pairs with rare positive instances cannot well capture the mapping patterns of QR. QA-net-MTL performs better than QA-net-QA but worse than QA-net-QR, in terms of F1. This means the

naive hard-sharing scheme cannot well leverage the QA data to boost the performance of the QR task. The other two MTL baselines, CQA-hard and NLP-soft, are the worst among the deep learning baselines, in terms of F1. The reasons could be (1) they do not use attention mechanisms; (2) their model sharing schemes may not be suitable for our problem. With a carefully designed model sharing scheme and proper attention mechanisms with regularization, QAR-net outperforms all the baselines. To assess their ability of ranking answers before non-answers, Figure 3 shows precision-recall curves of all the methods. It also clearly shows our method's superiority.

Pre-training Degree and Regularization Terms

Here we investigate how the pre-training degree affects the performance. We vary the number of pre-training epochs for QAR-net from 10 to 100. The obtained model is fine-tuned on the QR training set and tested on the QR validation set. For each test value of epoch number, the process is repeated 20 times in consideration of the randomness in under-sampling. The results are shown in Figure 4. We can observe that the performance first increases, then decreases with some fluctuations. The best performance is achieved around 20 epochs. These observations indicate long time pre-training with the QA task cannot further benefit the QR task. The reason could be due to the distribution and data size differences between the two tasks.

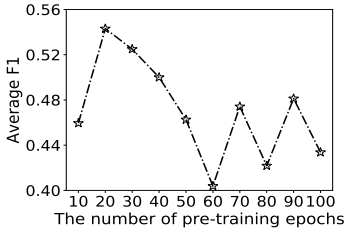

Figure 4: Validation performance of QAR-net when varying the number of pre-training epochs.

To prove the effectiveness of the regularization terms Eqs. (16) and (17), we conduct an ablation study: 1) we remove one of the two terms; 2) we remove both terms. The results are shown in Table 3. It clearly shows that the two regularization terms are crucial to the performance. When removing both of them, the performance takes a nose dive. The results imply that the two terms can effectively force the attention modules to capture important patterns for QR matching.

Table 3: Ablation study

Method	Precision	Recall	F-measure
No regularizations	0.4118	0.4200	0.4158
$\mathcal{R}_2$ only	0.4690	0.4533	0.4610
$\mathcal{R}_1$ only	0.4938	0.5267	0.5097
$\mathcal{R}_1 + \mathcal{R}_2$	0.5385	0.6067	0.5705

Parameter Study

k, μ and η are three hyperparameters exclusive to QAR-net. k is the hidden layer size of the self-attention module. μ and η are regularization hyperparameters. We investigate their impact on model performance with QR validation set. For each hyperparameter, we fix the other two and decide the test range by probing some values. The results are shown in Figure 5. Regarding the regularization hyperparameters, we can see the performance is the best when they are given a moderate value, indicating the usefulness of the regularization. The curve of k has a similar pattern. Besides, the performance is sensitive to k. This can be explained: (1) when k is small, the model cannot well capture the problem complexity; (2) when k is large, we could easily overfit the QR training data since the positive QR pairs are very limited (the self-attention module can only be trained by QR signals). We finally set k = 32, μ = 1e-1 and η = 1e-1.

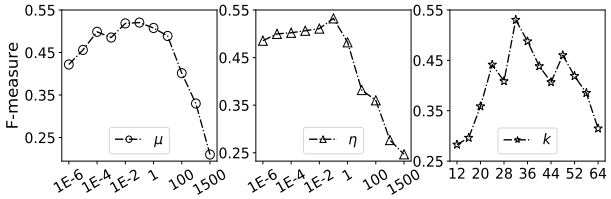

Figure 5: Parameter tuning.

Attention Visualization

In QAR-net, we use two attention mechanisms in order to capture question focus and answer patterns in the paired input. In Figure 6, we present a visualization of the attention vectors obtained on two positive QR pairs. The highlighted words with stronger red color indicates larger attention values. We can see the two attention mechanisms work as expected. $\mathbf{a}^{\overleftarrow{QR}}$ and $\mathbf{a}^{R}$ roughly capture the question focus, while $\mathbf{a}^{\overrightarrow{QR}}$ seems to reflect answer related words. We also investigated some negative pairs. The patterns captured by the bi-attention module for negative pairs seem to be random, since there is actually no related answer pattern in negative sentences. $\mathbf{a}^{\overleftarrow{QR}}$ and $\mathbf{a}^{R}$ tend to highlight noun words and sometimes can capture aspect-related words.

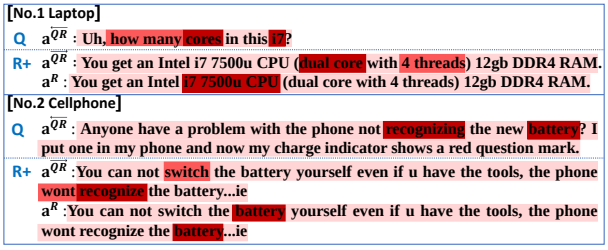

Figure 6: Attention visualization.

Conclusion

In this work, we proposed a novel multi-task attentive model named QAR-net to identify plausible answers from product reviews for user questions. QAR-net can well leverage large scale user generated QA data to help QR matching. Experiments on data collected from Amazon showed that QAR-net is effective and outperform baseline methods.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant Nos. 61522206, 61672409, 61876144, 61876145), the Major Basic Research Project of Shaanxi Province (Grant No. 2017ZDJC-31), and Shaanxi Province Science Fund for Distinguished Young Scholars (Grant No. 2018JC-016).

References

Bonadiman, D.; Uva, A.; and Moschitti, A. 2017. Effective shared representations with multitask learning for community question answering. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, 726–732.

Bordes, A.; Chopra, S.; and Weston, J. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

dos Santos, C. N.; Tan, M.; Xiang, B.; and Zhou, B. 2016. Attentive pooling networks. *CoRR, abs/1602.03609*.

Duong, L.; Cohn, T.; Bird, S.; and Cook, P. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, 845–850.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

He, H., and Garcia, E. A. 2009. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* 21(9):1263–1284.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, F.; Tang, Y.; Huang, M.; and Zhu, X. 2009. Answering opinion questions with random walks on graphs. In *ACL&AFNLP*, 737–745. Association for Computational Linguistics.

Li, J.; Liu, C.; Yu, J. X.; Chen, Y.; Sellis, T. K.; and Culpepper, J. S. 2016. Personalized influential topic search via social network summarization. *IEEE Trans. Knowl. Data Eng.* 28(7):1820–1834.

Lin, Z.; Feng, M.; dos Santos, C. N.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *CoRR* abs/1703.03130.

Liu, B. 2012. *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers.

McAuley, J., and Yang, A. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, 625–635. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Moghaddam, S., and Ester, M. 2011. Aqa: aspect-based opinion question answering. In *ICDMW*, 89–96. IEEE.

Nakov, P.; Hoogeveen, D.; Màrquez, L.; Moschitti, A.; Mubarak, H.; Baldwin, T.; and Verspoor, K. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 27–48.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kociský, T.; and Blunsom, P. 2015. Reasoning about entailment with neural attention. *CoRR* abs/1509.06664.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Seo, M. J.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 6000–6010.

Wan, M., and McAuley, J. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *ICDM*, 489–498. IEEE.

Wang, B.; Wang, C.; Bu, J.; Chen, C.; Zhang, W. V.; Cai, D.; and He, X. 2013. Whom to mention: expand the diffusion of tweets by@ recommendation on micro-blogging systems. In *WWW*, 1331–1340. ACM.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Meeting of the Association for Computational Linguistics*, 189–198.

Xiong, C.; Zhong, V.; and Socher, R. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Yan, Y.; Chen, M.; Shyu, M.-L.; and Chen, S.-C. 2015. Deep learning for imbalanced multimedia data classification. In *2015 IEEE International Symposium on Multimedia (ISM)*, 483–488. IEEE.

Yang, Y., and Hospedales, T. M. 2016. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*.

Yu, J.; Zha, Z.-J.; and Chua, T.-S. 2012. Answering opinion questions on products by exploiting hierarchical organization of consumer reviews. In *EMNLP&CoNLL*, 391–401. Association for Computational Linguistics.