RIKER: Mining Rich Keyword Representations for Interpretable Product Question Answering

Jie Zhao
The Ohio State University
zhao.1359@osu.edu

Ziyu Guan
Xidian University
zyguan@xidian.edu.cn

Huan Sun
The Ohio State University
sun.397@osu.com

ABSTRACT

This work studies product question answering (PQA) which aims to answer product-related questions based on customer reviews. Most recent PQA approaches adopt end2end semantic matching methodologies, which map questions and answers to a latent vector space to measure their relevance. Such methods often achieve superior performance but it tends to be difficult to interpret why. On the other hand, simple keyword-based search methods exhibit natural interpretability through matched keywords, but often suffer from the lexical gap problem. In this work, we develop a new PQA framework (named RIKER) that enjoys the benefits of both interpretability and effectiveness. RIKER mines rich keyword representations of a question with two major components, *internal word re-weighting* and *external word association*, which predict the importance of each question word and associate the question with outside relevant keywords respectively, and can be jointly trained under weak supervision with large-scale QA pairs. The keyword representations from RIKER can be directly used as input to a keyword-based search module, enabling the whole process to be effective while preserving good interpretability. We conduct extensive experiments using Amazon QA and review datasets from 5 different departments, and our results show that RIKER substantially outperforms previous state-of-the-art methods in both synthetic settings and real user evaluations. In addition, we compare keyword representations from RIKER and those from attention mechanisms popularly used for deep neural networks through case studies, showing that the former are more effective and interpretable.

KEYWORDS

Product QA, Interpretable Search, Question Representation

ACM Reference Format:
Jie Zhao, Ziyu Guan, and Huan Sun. 2019. RIKER: Mining Rich Keyword Representations for Interpretable Product Question Answering. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330985

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330985

1 INTRODUCTION

Question answering (QA) on e-commerce websites allows customers to acquire useful information for making purchase decisions. Currently, answering product-related questions still largely relies on costly human efforts. It can take several days to get an answer from sales representatives or other experienced customers. In this work, we study the problem of automating product question answering (PQA) [7]: Given a question regarding a product, we aim to return relevant sentences extracted from corresponding customer reviews to provide answer information. One important characteristic of PQA in comparison with open domain answer selection (e.g., [45, 47, 51]) is its high demand in system interpretability, partly because customers may have increasing concerns about hidden advertising agendas and leaked personal information and can distrust a PQA system [18] if puzzled by how they get the results, and also partly because latest policies and regulations are also urging companies to provide interpretations for their algorithms that directly affect users [13, 17].

To the best of our knowledge, all recent PQA works [30, 43, 48] advocate end2end semantic matching methodologies, which tend to be black-box and directly output a matching score for each <question, review sentence> pair. Typically, questions/answers/reviews are first encoded into low-dimensional vector representations, which are then used to generate the matching scores based on some relevance functions (e.g., dot product). For example, question-review and answer-review encoders and relevance functions are simultaneously learned by optimizing a mixture-of-experts objective [30]. In fact, in open domain as well, almost all latest answer selection works [41, 42, 50] fall into this category, e.g., relying on hidden units within deep neural networks (DNN) to represent the relevance of QA sentences pairs [28]. Despite its popularity, the end2end paradigm has received challenges on its lack of interpretability [14]. Even with many recent efforts [3, 5, 10, 37, 39], it is still very difficult to interpret dense vector representations or how an answer is matched with a question. Meanwhile, different attention mechanisms [31, 41, 50] have been incorporated into DNN models to associate the relevant parts across two sentences. Although to some extent the intermediate attentions can help reveal the soft word alignments between two sentences, the models are still less transparent because of the nonlinear relationship between attention scores and outputs [35].

In this work, we do not follow the end2end paradigm of existing approaches, and instead advocate a hybrid framework marrying the advantages of both DNN structures and classic keyword-based Information Retrieval (IR) techniques [4]. Keyword-based IR techniques such as *tf-idf* ranking functions *naturally exhibit much better interpretability* owing to their transparency and intuitiveness:

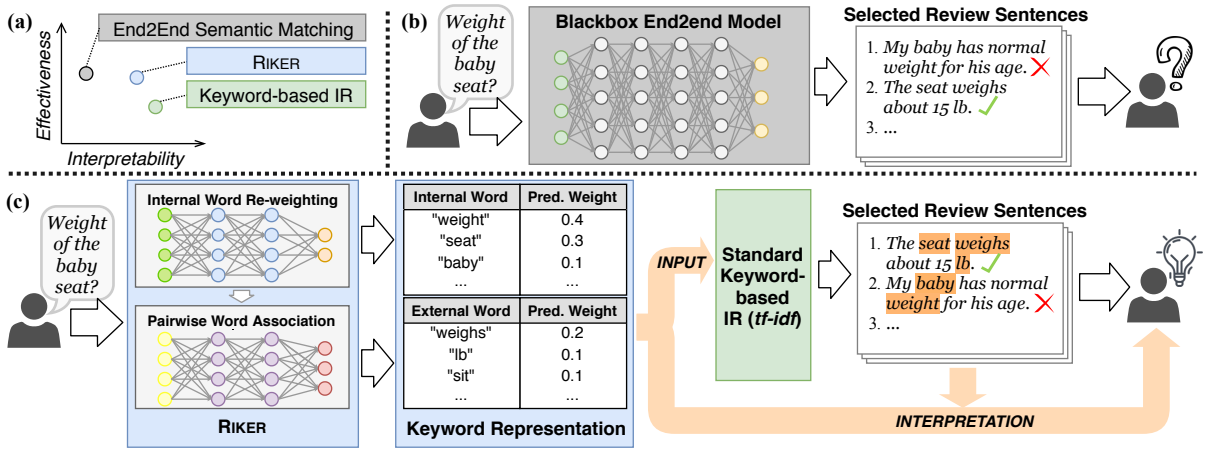

Figure 1: Interpretability and effectiveness of different methodologies. Here we use DNN structures as an example of end2end methods. Our proposed method RIKER mines rich keyword representations for a question, which are used as input to a standard *tf-idf* based IR module. With overlapped words highlighted in orange, such representations also serve as interpretation to the user for the retrieved results and help keep the ranking process transparent.

An answer candidate will be ranked higher if it has more important keywords matched with the question. However, standalone keyword-based IR methods are often not as effective due to the lexical gap problem [38], i.e., they cannot handle mismatched yet relevant words. In addition, they tend to weigh question words only according to their corpus-level statistics like *tf-idf* without considering the specific sentence-level context information. Therefore, to achieve a good balance between interpretability and effectiveness, *we propose a new framework named RIKER, which mines rich keyword representations of a question and uses them to effectively improve the performance of interpretable keyword-based search techniques.*

Figure 1(a) qualitatively shows the trade-off between effectiveness and interpretability for different methodologies, while Figure 1(b) and 1(c) demonstrate how an end2end model and RIKER process the same question *"Weight of the baby seat?"*. An end2end system is often hard for humans to understand or intervene. As exemplified in Figure 1(b), when an irrelevant review sentence is ranked at the top (because the system misinterprets the question intent as "*baby weight*" rather than "*baby seat weight*"), the user may have no choice but keep rephrasing the question to probe the system. In contrast, as Figure 1(c) shows, RIKER can map the question into its keyword representations, which emphasize important internal words (i.e., words that appear in the question) such as *"weight"*, *"seat"* and include relevant external words (i.e., words that are not in the question but exist in the corpus) such as *"weighs"* and *"lb"*. Such easy-to-understand keyword representations can then be used as input to standard *tf-idf* based IR, which keeps the review sentence selection process transparent. Moreover, if unsatisfied with current results, the user can adjust the keywords and their predicted weights (e.g., set the predicted weight of "*baby*" to zero to reduce ambiguity) and anticipate better results.

Specifically, RIKER consists of two neural components: (1) Internal word re-weighting, which predicts the relative importance of words *within* the question, and assigns more weights to those that are more likely to occur in correct answers and less likely in incorrect ones; (2) External word association, which models pairwise word associations between questions and answers, and enables RIKER to infer relevant keywords *outside* the question in the entire vocabulary. For training, we design two novel objective functions respectively for each component. They explicitly encourage assigning higher scores to matched or associated keywords between correct QA pairs and mutually enhance each other, so that RIKER achieves the best overall performance when both objectives are jointly optimized. The mined keyword representations are employed as input to the IR module to alleviate its lexical gap problem and enhance the search performance, and are demonstrated more effective than the state-of-the-art word embedding based query expansion methods [12, 49]. At the same time, they can serve as interpretations to human users and allow RIKER to enjoy the interpretability of the keyword-based search techniques, which we demonstrate both quantitatively and qualitatively through case studies.

In summary, our contributions are as follows:

- To the best of our knowledge, we are the first to advocate interpretability in product QA, which can give customers a better intuition of how their questions are answered and eventually helps earn user trust in the system.
- We design a new PQA framework, RIKER, which jointly identifies important keywords within the question and associates relevant words outside the question based on neural models. Such keyword representations can improve the search performance and meanwhile preserve the interpretability of a keyword-based search module.
- We conduct extensive experiments using 5 PQA datasets from *Amazon.com* as well as a real user study, and show that our framework consistently outperforms existing query expansion and PQA methods. In addition, we also discuss RIKER w.r.t. a widely adopted attention mechanism for DNN models on their interpretability via case studies.

2 OVERVIEW

Traditional keyword-based IR techniques widely used by e-commerce websites have the advantage of being self-explanatory to general

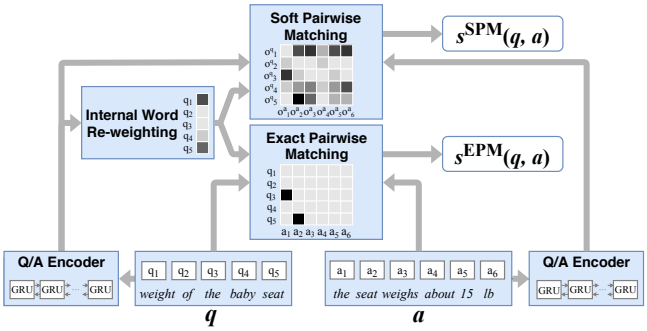

Figure 2: Training structure of Riker.

customers without domain knowledge, and are commonly acknowledged by data scientists as interpretable algorithms among decision trees, rule lists [44], sparse linear models [26], etc. However, naive keyword-based search has drawbacks when applied to PQA: They tend to ignore the relative importance of words given the question context and can not relate different but semantically similar words. To solve this problem, we design Riker that mines better keyword representations for questions. Figure 2 shows the *training* neural architecture of Riker. The *internal word re-weighting* component is learned through exact pairwise matching (Section 3) and predicts higher weights for question words that may exactly appear in the correct answers but rarely in others, whereas the *external word association* component is learned through soft pairwise matching (Section 4) and captures semantic associations between different words. Such keyword representations can preserve the natural interpretability of keyword-based search methods, and also make Riker more effective as experiments will show (Section 5). Additionally, we use case studies to further discuss the interpretability of keyword representations from Riker (Section 6).

3 INTERNAL WORD RE-WEIGHTING

The goal of internal word re-weighting component is to assign different weights to words *within* a given question, according to their relative importance in the context. In this section, we first introduce the details of the re-weighting function, and then describe how it can be trained with an exact pairwise matching objective.

The internal word re-weighting function takes a question q that is a word sequence $[q_1, ..., q_n]$ as input, and outputs a weight for each word, denoted as $f^{\text{int}}(q_i|q)$. As shown in the left part of Figure 2, the question words are mapped to word embeddings and then encoded with a standard bidirectional recurrent neural network (Bi-RNN) with Gated Recurrent Units (GRU) [11] to aggregate sentence-level context information. Specifically, the i-th time step operation of a forward GRU cell parameterized by $\{\vec{W}_r, \vec{b}_r, \vec{W}_u, \vec{b}_u, \vec{W}, \vec{b}\}$is:

$$\begin{aligned} &\vec{r}=\sigma(\vec{W}_r[e(q_i), \vec{h}_{i-1}]+\vec{b}_r) \;\; ; \;\; \vec{u}=\sigma(\vec{W}_u[e(q_i), \vec{h}_{i-1}]+\vec{b}_u) \\ &\vec{\tilde{h}}_i=\phi(\vec{W}[e(q_i), \vec{r}\odot\vec{h}_{i-1}]+\vec{b}) \;\; ; \;\; \vec{h}_i=\vec{u}\odot\vec{h}_{i-1}+(1-\vec{u})\odot\vec{\tilde{h}}_i \end{aligned} \quad (1)$$

Here $[\cdot,\cdot]$ represents vector concatenation, $e(q_i)$ is the looked-up word embedding of word q_i, σ and ϕ represent *sigmoid* and *tanh* activation functions respectively, and $\odot$ is element-wise product. A backward GRU cell is defined symmetrically to encode the question from the last word to the first, with a different set of parameters $\{\overleftarrow{W}_r, \overleftarrow{b}_r, \overleftarrow{W}_u, \overleftarrow{b}_u, \overleftarrow{W}, \overleftarrow{b}\}$. The forward and backward GRU hidden states are concatenated at each position to be the output of the Bi-RNN encoder, i.e., $o_i^q = [\vec{h}_i, \overleftarrow{h}_i]$ for q_i. Next, the Bi-RNN output at each step is passed through a fully connected feedforward network with a sigmoid output layer, followed by a normalization step.

$$s_i = \sigma(W_2\, \sigma(W_1 o_i^q + b_1) + b_2)\,; \quad f^{\text{int}}(q_i|q) = s_i / \sum_{k=1}^{n} s_k \quad (2)$$

Exact pairwise matching. We define the exact pairwise matching (EPM) score for a <q, a> pair as a weighted sum of all the overlapped words (corresponding to the black squares of the exact pairwise matching module in Figure 2):

$$s^{\text{EPM}}(q, a) = \sum_{i=1}^{n} f^{\text{int}}(q_i|q)\, \mathbb{I}(q_i \in a) \quad (3)$$

Here $\mathbb{I}(q_i \in a)$ is an indicator function that returns 1 if q_i appears in a and 0 otherwise.

The objective of exact pairwise matching is to assign higher scores to correct answers than incorrect ones, and therefore guides f^{int} to put more weights on words that occur in the correct answers but not in other answer candidates. Specifically, we use a standard cross-entropy loss function, minimizing which encourages increasing s^{EPM} of the correct answer a^+ compared with a set of randomly sampled non-answers $\mathcal{A}^-$:

$$\mathcal{L}^{\text{EPM}} = -\log \frac{\exp(s^{\text{EPM}}(q, a^+))}{\sum_{a\in\{a^+\}\cup\mathcal{A}^-} \exp(s^{\text{EPM}}(q, a))} \quad (4)$$

Once the internal word re-weighting function f^{int} is learned, we can directly combine the predicted word weights of a question with an off-the-shelf *tf-idf* based IR [22], so that the importance of a word is measured by both its corpus-level statistical importance and the context-aware semantic importance. Specifically, the *tf-idf* weight of each question word in an answer candidate is multiplied with its f^{int} score, before they are summed up as the final score of the answer candidate. Despite being simple, as experiments will show, adopting this re-weighting (RW) strategy helps retrieve better results, which also outperforms some existing baselines.

4 EXTERNAL WORD ASSOCIATION

It is common that a question word semantically aligns with different words in a correct answer, e.g., "*weight*" in q with "*lb*" or "*weighs*" in a in Figure 2. Unfortunately, exact pairwise matching does not exploit such connections, and is therefore incapable of solving the lexical gap problem. This motivates us to design the soft pairwise matching objective to further explore the associations between a question word and other words that are outside the question but likely to appear in correct answers. At the end of this section, we discuss three different strategies that leverage the word associations to expand a question at retrieval time.

Soft pairwise matching. Analogous to the previous section, we define a soft pairwise matching (SPM) score, $s^{\text{SPM}}(q, a)$, based on word associations from both the question side and the answer side. Recall that in Equation 1, the question has already been encoded into $[o_1^q, o_2^q, ..., o_n^q]$. During training, we use the same Bi-RNN (with shared parameters) for questions to encode an answer ($a = [a_1, ..., a_m]$) as $[o_1^a, ..., o_m^a]$. Afterwards, we calculate the external word association score $f^{\text{ext}}(q_i, a_j|q, a)$ for every word pair <q_i, a_j> as the cosine similarity of their corresponding Bi-RNN outputs,

Algorithm 1: General Question Expansion

1 **Result:** Question expansion words and weights
2 **Input:** internal word weights $f^{\text{int}}(q_i|q)$; vocabulary V;
3 expansion similarity function $p(q_i, w)$;
4 hyper-parameters: integer N; float δ
5 **for** q_i *in question* $q = [q_1, \ldots, q_n]$ **do**
6 $p(q_i, q_i) = 0$
7 /* Select N words with highest expansion similarities: */
8 $\{w_{i,1}, \ldots, w_{i,N}\} = \text{top\_k}([p(q_i, w_0), \ldots, p(q_i, w_{|V|})], N)$
9 /* δ controls the contribution of expansion terms */
10 $s_{i,j} = f^{\text{int}}(q_i|q)\, p(q_i, w_j)\, \delta,\ j = 1, \ldots, N$
end
11 Sum $s_{i,j}$ that corresponds to the same word
12 **return** distinct $w_{i,j}$ with corresponding aggregated $s_{i,j}$

and aggregate them into the overall soft pairwise matching score $s^{\text{SPM}}(q, a)$ as follows:

$$f^{\text{ext}}(q_i, a_j|q, a) = \frac{(o_i^q)^T \cdot o_j^a}{|o_i^q| \ \ |o_j^a|} \tag{5}$$

$$s^{\text{SPM}}(q, a) = \sum_{i=1}^{n} f^{\text{int}}(q_i|q) \max_{1 \le j \le m} f^{\text{ext}}(q_i, a_j|q, a) \tag{6}$$

Here $f^{\text{int}}(q_i|q)$ is the internal word weight from Equation 2, representing the relative importance of the i-th word in the question. The intuition behind this SPM function is that an answer should achieve a high score if for each important word of the question, at least one word in the answer matches well with it. We also empirically test for each question word q_i, summing $f^{\text{ext}}(q_i, a_j|q, a)$ of all the answer words a_j instead of using max pooling, which leads to similar results but takes longer time to train. Same as in the previous section, the soft pairwise matching function can also be trained using negatively sampled QA pairs:

$$\mathcal{L}^{\text{SPM}} = -\log \frac{\exp(s^{\text{SPM}}(q, a^+))}{\sum_{a \in \{a^+\} \cup \mathcal{A}^-} \exp(s^{\text{SPM}}(q, a))} \tag{7}$$

Optimizing $\mathcal{L}^{\text{SPM}}$ alone can affect both internal word re-weighting (f^{int}) and external word association (f^{ext}) modules through back-propagation, but the effect on the former can be distant and indirect. Therefore, we propose to jointly optimize both objectives:

$$\min_{\Theta} (\mathcal{L}^{\text{EPM}} + \mathcal{L}^{\text{SPM}}) \tag{8}$$

Here, Θ represents all parameters including word embedding vectors, those in Bi-RNN, and $\{W_1, b_1, W_2, b_2\}$.

Question expansion based on external word associations. Although directly ranking answer candidates through soft pairwise matching seems applicable, it requires encoding all the review sentences with Bi-RNN at testing time, which is computationally expensive for PQA. We propose to use easy-to-compute word similarity functions to transform the neural external word associations from the training phase (Eq. 6) to a small set of expanded keywords for a question, which promotes interpretability as it is easier for humans to check expansion words than pairwise word associations and can also be used to enhance the performance of keyword-based IR.

There has been a broad spectrum of works on question/query expansion (QE) in the past for retrieving web pages [46], emails [23], answers [38] and so on. However, many existing QE techniques are not directly applicable in this PQA scenario (e.g., those trained with click-through data). The most related QE techniques to ours are those using word embeddings [12, 24, 49]. The difference of our QE methods is that we jointly consider internal word weights in training word associations (f^{int} in Eq.6), and find external relevant words not only at the word embedding level, but also at higher levels projected by the trained neural architecture. Algorithm 1 outlines a general query expansion scheme based on which we developed three intuitive variants (**QE1/2/3**) using different *expansion similarity functions* (p /p'/p''). Given an expansion similarity function, it finds N most similar words for each question word (line 8), and their weights are the product of the internal word weights f^{int}, the expansion similarities $p(q_i, w)$ and a hyper-parameter δ (line 9). Duplicated expansion words are aggregated by summing up their individual weights (line 11). The final output is a set of weighted external words, which can be combined with the re-weighted internal words and then input together to the off-the-shelf keyword-based IR method as discussed earlier.

QE1: Word embedding similarity. This is a basic expansion similarity function that directly leverages trained word embeddings [12]. It is defined as the cosine similarity between the word embeddings of a question word q_i and an arbitrary word w from the vocabulary:

$$p(q_i, w) = \frac{e(w)^T \cdot e(q_i)}{|e(w)| \ |e(q_i)|} \tag{9}$$

Word embeddings in our model are initialized with pre-trained results [33] in the general domain, but further trained to be product domain specific [12] and can be more suitable for query expansion in our settings.

QE2: Higher-level word similarity. During training, the raw word embeddings of a sentence are further encoded by the Bi-RNN, which outputs higher-level representations at each step. We hypothesize that even without context (i.e., surrounding words), such low to high level transformations can help capture word associations more effectively. Therefore, we propose to leverage the learned parameters within the GRU cell. Specifically, we concatenate the word embedding $e(q_i)$ (same for $e(w)$) with zero vectors in both forward and backward directions, and project them by part of the GRU parameters. The projected vectors are concatenated into higher-level representations $e'(q_i)$ and $e'(w)$ and measured by cosine similarity:

$$\begin{aligned} e'(q_i) &= [\phi(\vec{W}[e(q_i), 0] + \vec{b}),\ \phi(\overleftarrow{W}[e(q_i), 0] + \overleftarrow{b})] \\ e'(w) &= [\phi(\vec{W}[e(w), 0] + \vec{b}),\ \phi(\overleftarrow{W}[e(w), 0] + \overleftarrow{b})] \\ p'(q_i, w) &= \frac{e'(w)^T \cdot e'(q_i)}{|e'(w)| \ |e'(q_i)|} \end{aligned} \tag{10}$$

QE3: Context-aware similarity. We further propose a question-context-aware QE strategy, which considers the entire question sentence at testing time and dynamically calculates expansion word similarities based on its Bi-RNN encodings (note for computational efficiency, there is no context considered for the expansion word side). Formally, with o_i^q being the Bi-RNN output corresponding to the i-th question word, and $e'(w)$ the same as in Equation 10, the new expansion similarity is defined as:

$$p''(q_i, w) = \frac{e'(w)^T \cdot o_i^q}{|e'(w)| \ |o_i^q|} \tag{11}$$

Comparing the three variants, QE1 and QE2 are off-line strategies meaning that the expansion similarity between any two words in the vocabulary can be pre-calculated, stored in a table, and ready to look up at testing time. QE2 is slightly more complex than QE1 as it requires to project word embeddings. They both enjoy a higher computational efficiency than QE3, which is an online strategy that needs to dynamically compute the expansion similarities based on the context of a specific question. However, QE3 is closer to how the external word association score f^{ext} is optimized during training, and can potentially achieve the best performance.

5 EXPERIMENTS

Due to the lack of labeled review sentences (being relevant/irrelevant to a question), previous works [30, 43] adopted a *synthetic* evaluation setting using large-scale historical QA pairs. Specifically, answers to all questions are gathered as an entire answer candidate pool for retrieval. For each question, its paired answer given by a human expert (the top rated one if there are many), is treated as correct whereas all other answer candidates as incorrect. The candidate pools for training, dev and testing sets are separated. In addition to this synthetic setting, we also experiment under a *realistic* scenario where we retrieve review sentences to answer questions, and ask human annotators to evaluate their relevance.

5.1 Experimental Setup

Datasets. We use the Amazon QA [30] and review datasets [29] from 5 different departments. We pre-process these raw data slightly differently from [30] and the statistics are summarized in Table 1. First, we do not classify a question as *yes-no* or *open-ended* type. As emphasized in [30], even for yes-no questions, it is critical to find relevant review sentences as supporting evidence. Therefore, we directly treat PQA as a review sentence selection task. Second, we filter out QA pairs whose answers contain less than two words other than stop words or "*yes*"/"*no*", because they contain little useful information for either training or testing. The last column of Table 1 shows that after filtering, we still keep 80%∼97% of all the questions in the raw datasets, much higher than the 44% classified as open-ended in [30], additionally showing that treating PQA as a retrieval problem has a broader coverage. The datasets are randomly divided into train, dev and test set using 7: 1: 2 ratio. The same pre-processed data splits are used for all methods.

Evaluation Measure. For synthetic evaluation, we follow previous work [30, 43] to use average Area Under the Curve (AUC):

$$\text{AUC} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}}\frac{1}{|\mathcal{A}|-1}\sum_{a^-\in\mathcal{A}-\{a^+\}}\mathbb{I}(\text{score}(q,a^+)>\text{score}(q,a^-)),$$

where $\mathcal{Q}$ represents all the questions in dev or test set, and $\mathcal{A}$ is the corresponding set of all the answers. $\mathbb{I}(\text{score}(q,a^+)>\text{score}(q,a^-))$ is a binary indicator function that returns 1 when the paired answer is ranker higher than the non-answer[1]. In the real user study experiment, where humans are asked to annotate the relevance of each retrieved review sentence, we use normalized discounted cumulative gain (nDCG), a popular ranking metric for IR [21].

[1]Same as previous work [30, 43], we approximately assume only the paired answer is correct to a question. AUC essentially measures the percentage of non-paired answers getting lower scores than the paired one, and is less sensitive to the dataset noise compared with other ranking metrics like Mean Reciprocal Rank (MRR) or nDCG, especially when the number of correct answers to a question is significantly smaller than the answer pool size $|\mathcal{A}|$.

Dept.	vocab size	# of QA pairs			subjective
		Train	Dev	Test	Qs (%)
Appliances	31,694	6,156	879	1,760	97.17
Baby	62,267	14,901	2,128	4,259	73.58
Patio	89,449	37,427	5,346	10,695	89.72
Tools	116,635	62,242	8,891	17,785	87.96
Electronics	100,000	179,858	25,694	51,389	81.76

Table 1: Dataset statistics.

Baseline methods. The baseline methods can be divided into those without and with explicit question expansion. Those without explicit QE are: (1) **Moqa** and **BM25+** [30]: They both use the same mixture-of-experts framework to jointly optimize answer-review and question-review relevance functions. Moqa models the relevance functions as approximated bilinear models to match sentence pairs in latent vector spaces, while BM25+ models them as Okapi BM25+ ranking function with a few learnable parameters [27]. We used their published code for both Moqa and BM25+ on our pre-processed datasets. (3) **OQ** and **OQ - stop**: Two naive baselines using the same *tf-idf* IR module as in our framework, one directly using the original question to query and the other with stopwords removed from the question using SpaCy. (4) **RW**: Our internal word re-weighting method as described in Section 3.

We select the following baselines with question expansion: (5)**PRF-TFIDF**: A classic QE method [6] that expands a query using the top-K *tf-idf* words from the top-N pseudo-relevant sentences returned by IR. (6) **PRF-RM**: A popular QE strategy based on relevance model [25]. Specifically, given original query q_0, the probability of selecting a term t in the reformulated query is: $P(t|q_0) = (1-\lambda)P'(t|q_0)+\lambda\sum_{d\in D_0}1/|D_0|P(t|d)P(q_0|d)$, where D_0 is the set of top-N pseudo-relevant documents returned by q_0 and λ is set at 0.5 following previous work. $P(t|q_0)$ is defined as the normalized *tf-idf* weight of t in q_0 and $P(t|d)$ the add-one smoothed language model: $P(t|d) \propto \text{tf}(t,d)+1$. We select top-K terms with highest probability to expand query q_0. (7) **GloVe** and **GloVe$^{\text{tgt}}$**: Following state-of-the-art word embedding based QE works [12, 49], question words are expanded through similarities (equivalent to $p(q,w_i)$ in Equation 9) measured by two types of word embeddings: (i) those pre-trained by GloVe [33] and (ii) those fine-tuned on our target (tgt) domain corpus with the relevance-based objective designed by [49] for search tasks. Specifically, the QE process is equivalent to Algorithm 1 but without considering learned internal word weights (i.e., fix f^{int} to 1 in line 10). (8) **GloVe+RW** and **GloVe$^{\text{tgt}}$+RW**: Same as above but further take into account our learnt internal word weights (i.e., f^{int} given by Riker in line 10). (9) **Riker+QE1/QE2/QE3**: After Riker is trained, we test the three question expansion variants described in Section 4 respectively.

5.2 Results

Comparing with baselines without QE. The upper half of Table 2 compares all the methods without explicit question expansion. We observe that our internal word re-weighting (RW) strategy outperforms the search baselines OQ and OQ - stop, showing the benefit of dynamically predicting word weights over naive strategies using equal weights or removing stop words (which may fail

	Appliances		Baby		Patio		Tools		Electronics	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BM25+	56.577	56.746	65.799	65.385	65.349	64.497	64.280	65.219	63.913	63.818
OQ	78.120	79.039	87.078	87.600	85.020	84.510	86.131	86.049	86.356	86.408
OQ - Stop	77.448	78.319	86.002	85.963	83.559	83.311	84.873	84.888	84.537	84.683
Moqa	76.367	76.314	83.836	83.349	85.292	85.011	86.673	86.421	87.836	87.824
RW	78.332	79.653	87.743	89.043	85.204	84.792	86.524	86.436	87.063	87.082
PRF-TFIDF	74.382	75.061	84.802	85.102	82.205	82.884	83.534	83.723	82.847	83.492
PRF-RM	78.446	79.801	87.931	88.235	85.469	85.113	86.766	86.688	86.058	86.083
GloVe	65.410	66.918	74.775	74.456	73.760	73.440	77.785	77.676	78.072	78.003
GloVe^{tgt}	76.146	77.293	85.777	85.517	82.748	82.590	84.613	84.442	83.126	83.054
GloVe+RW	77.071	78.628	88.600	88.307	86.147	85.599	87.261	87.156	86.692	86.749
GloVe^{tgt}+RW	79.959	80.981	90.298	90.053	87.884	87.856	89.060	88.918	88.151	88.126
Riker+QE1	78.733	79.920	89.486	89.170	86.744	86.384	86.436	87.836	87.380	87.436
Riker+QE2	80.405	81.427	91.527	91.367	89.427	89.681	90.394	90.175	89.493	89.537
Riker+QE3	**84.415**	**84.629**	**92.262**	**92.086**	**91.365**	**91.368**	**91.969**	**91.643**	**91.095**	**91.010**

Table 2: PQA results (AUC) under the synthetic setting. The upper(lower) half shows methods without(with) explicit QE. Riker+QE2/QE3 achieve the best overall performance, showing that it mines effective keyword representations.

because words such as "*without*", "*last*", "*bottom*" that are often treated as stop words are actually important for product-related questions). RW also achieves comparable or better performances to the end2end Moqa baseline especially when the dataset size gets smaller, indicating that the task to predict word importance is less sensitive to dataset sizes, and is especially suitable for new domains without much available data. Another observation is that with our pre-processing steps, using the original question (OQ)[2] to search has already achieved performances close to Moqa, showing that keyword-based search can be decently effective in product domain. Next, we will discuss that RW also plays a key role for QE to succeed.

Comparing Riker with baselines with QE. The lower half of Table 2 compares all methods with explicit question expansion. Our proposed Riker+QE2/QE3 variants outperform all baselines. GloVe$^{(tgt)}$+RW using word embeddings fine-tuned for retrieval purpose in the target domain corpus is the strongest baseline, but is less effective than some Riker variants because we jointly trained internal word re-weighting and external word association in a unified framework. Notice that word embedding based QE baselines perform poorly if not combined with RW (GloVe$^{(tgt)}$), again showing the importance of internal word re-weighting for PQA because treating question words equally when incorporating external terms will introduce much noise. The classic QE methods PRF-TFIDF and PRF-RM generally works not as well as embedding based ones because the representation power of word embedding can be more expressive than corpus level statistics.

Comparing Riker variants. Riker+QE2 with the novel use of GRU cell parameters outperforms Riker+QE1, demonstrating that the trained neural layer can project word embeddings into more effective higher-level semantic representations. As expected, Riker+QE3, which takes the context information of an entire question into account, performs the best, but it comes at a cost of computing vector similarities at testing time as discussed earlier. In practice, one may consider the trade-off between efficiency and effectiveness when choosing from QE2 and QE3.

[2]This baseline was not tested in [30].

Objective	dev AUC		test AUC	
	RW	QE3	RW	QE3
EPM	87.434	86.563	**88.117**	86.632
SPM	87.104	91.646	87.589	91.426
EPM+SPM	**87.743**	**92.626**	88.041	**92.086**

Table 3: Training objective ablation.

Ablation study of training objectives. Table 3 shows the effect of our proposed training objectives. Due to space limit, we only show the Baby dataset but the results for others are similar. Training using only the EPM objective gets comparable performance as using both objectives for our RW method, but makes QE3 ineffective because without SPM, the Bi-RNN does not learn how to associate words at the high level. The QE3 strategy works the best when the EPM (Eq. 4) and SPM (Eq. 7) objective are optimized simultaneously, showing that with explicit supervision for internal word re-weighting using EPM, Riker can better find the important expansion words to be associated with a question.

Parameter sensitivity. Figure 3 shows that Riker+QE3, our best variant, is insensitive to QE hyper-parameters N and δ on Baby dataset. Similar results are observed on the other datasets. Riker+QE3 significantly outperforms the baseline without QE (i.e., RW in Table 2), whose AUC for the dev/test set is 87.743/89.043, far below the current coordinate ranges in Figure 3. Results on dev and test datasets are similar, with AUCs reaching the best when the expansion scale $\delta = 0.15$. The performance tends to get better as N increases, but involving more and more external words will make keyword-based IR less understandable, and hence we balance effectiveness and interpretability by upper-bounding N at 70.

Real User Evaluation So far, we have been experimenting under the *synthetic* setting, where Riker performs the best at selecting answer sentences. However, one might suspect such performance gain could come from that Riker is trained to optimize answer ranking, whereas the existing state-of-the-art PQA method Moqa [30] jointly models the relevance between questions, answers and reviews in a comprehensive probabilistic model.[3]

[3]It is fair to compare Riker with other baselines such as GloVe^{tgt}(+RW) under the synthetic evaluation, since they all directly match QA pairs, not through reviews.

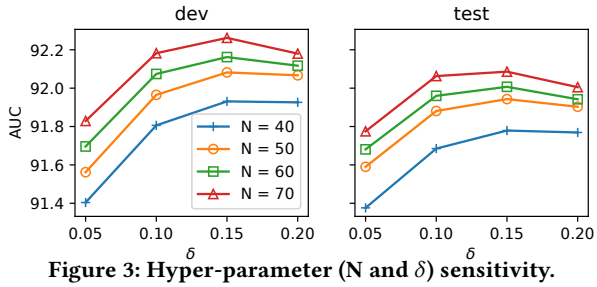

Figure 3: Hyper-parameter (N and δ) sensitivity.

Dept.	nDCG@10 (%)								Fleiss' Kappa
	User1		User2		User3		**Avg**		
	Moqa	QE3	Moqa	QE3	Moqa	QE3	Moqa	QE3	
Appl.	18.60	55.22	42.36	56.60	20.74	43.67	27.24	**51.83**	0.335
Baby	26.38	73.97	25.50	66.29	14.47	54.13	23.78	**64.80**	0.459
Patio	23.89	61.80	26.53	49.30	24.68	56.64	25.03	**55.91**	0.293
Tools	25.03	49.67	26.62	42.15	33.49	43.55	28.38	**45.12**	0.392
Elec.	27.51	59.57	35.49	53.42	21.81	58.47	28.27	**57.15**	0.370

Table 4: Real user evaluation of Moqa and RIKER+QE3.

In order to show RIKER's generalization ability to selecting review sentences, we conduct a real user study where different methods retrieve *review sentences* corresponding to the product to answer questions. We randomly sampled 200 questions, 40 from each department, and there are on average 474 review sentences as answer candidates for each question.[4] For every retrieved sentence, three different human annotators were asked to score it with 2, 1 or 0, corresponding to being *relevant*, *partly relevant* and *irrelevant*.[5] The results are summarized in Table 4. In this realistic setting, RIKER+QE3 still substantially outperforms Moqa, indicating that using RIKER predicted keyword representations of a question can also search product reviews more effectively than latent semantic representation approaches. This is partly because review sentences often contain more personal experience as well as sentiment-related words/phrases than human provided answers in our training sets, and their sentence-level semantics can be hard to match with a question. However, once the keywords associated with a question are predicted, relevant review sentences can be more easily retrieved using keyword-based IR. Fleiss' Kappa scores [15] are used to measure the consistency among different users' labels, which show fair and moderate agreements (>0.35) for most departments.

6 DISCUSSION

Section 5 shows that keyword representations mined by RIKER can enhance the performance of keyword-based IR, in comparison with various existing query expansion and PQA methods, leading to a more effective PQA framework with the ranking process still interpretable based on matched keywords. In this section, we will further compare RIKER with attention mechanisms[9, 31, 41, 42], which have recently been widely used in DNNs for sentence pair modeling and can provide interpretations of how two sentences are matched by revealing word-level associations. As we need to first clarify how to apply the attention mechanisms to our task, we separate the discussion from other methods in Section 5.

6.1 An Attention-based Deep QA Model

We briefly show how to adapt an end2end DNN QA model to the PQA setting. We choose a standard word pairwise attention mechanism [31] that are proved effective in many recent sentence pair modeling works [9, 41, 42] . For fair comparison, we use the exact same Bi-RNN architecture as in RIKER to encode the question/answer sentences and compute attention from its outputs $[o_1^q, ..., o_n^q]$ and $[o_1^a, ..., o_m^a]$. The final matching score is S^{DNN}:

$$e_{i,j} = F(o_i^q)^T \cdot F(o_j^a) \quad 1 \le i \le n,\ 1 \le j \le m \tag{12}$$

$$\bar{o}_i^q = \sum_{j=1}^{m} \frac{\exp(e_{i,j})}{\sum_{k=1}^{m} \exp(e_{i,k})} o_j^a, \quad \bar{o}_j^a = \sum_{i=1}^{n} \frac{\exp(e_{i,j})}{\sum_{k=1}^{n} \exp(e_{k,j})} o_i^q \tag{13}$$

$$v^q = \sum_{i=1}^{n} G([o_i^q, \bar{o}_i^q]), \quad v^a = \sum_{j=1}^{m} G([o_j^a, \bar{o}_j^a]) \tag{14}$$

$$s^{\text{DNN}} = H([v^q, v^a]) \tag{15}$$

Here, F, G and H are different feedforward networks and are all instantiated with one hidden layer in our experiments. The same objective function (Eq. 4 or 7 but using s^{DNN}), regularization and training procedure as RIKER is used to train this DNN model.

Using attention for question expansion We compare the learned attention from this DNN model head-to-head with RIKER's word associations for question expansion. Specifically, we adapt the same QE2 and QE3 strategy in Section 4 for the attention mechanism, with the word similarity functions changed to:

$$p'^{\text{ATTN}}(q_i, w) = F(e'(w))^T F(e'(q_i))/(|F(e'(w))|\ |F(e'(q_i))|) \tag{16}$$

$$p''^{\text{ATTN}}(q_i, w) = F(e'(w))^T F(o_i^q)/(|F(e'(w))|\ |F(o_i^q)|) \tag{17}$$

Here F is the trained feedforward net in Equation 12. We denote these two baselines as **ATTN+QE2** and **ATTN+QE3**.

6.2 Case Studies

We first show some qualitative results to help illustrate the interpretability of RIKER's word associations in comparison with that of DNN. *How to evaluate interpretability is still an open issue, but intuitively in the PQA scenario, a method that comes up with more relevant keywords and uses them to retrieve better answer sentences is more interpretable than one with less relevant words and finds less useful answers.* Table 5 demonstrated using QE3 strategy to expand several common questions in the largest Electronics dataset. In general, RIKER's expansion words appear to be better. For example, for the last question about turning off the alarm, RIKER+QE3 suggests words such as"*alert*", "*disarm*", "*deactivate*", while ATTN+QE3's result seems less relevant. Due to space limit, we omit examples from other datasets and the qualitative comparison between RIKER+QE2 and ATTN+QE2, but we observed the same phenomenon that RIKER can associate more relevant words.

We next quantitatively compare the keyword representations mined by RIKER and those by attention mechanisms in terms of how much they help boost the *tf-idf* based search framework. This evaluation method is also the same as the so-called *functionally-grounded*

[4]The average number of reviews sentences for each product in the datasets: Appliance–342.88, Baby–720.87, Patio–319.96, Tools–297.92, Electronics–686.23.
[5]We adopt these three scores since annotators found it hard to label many review sentences as definitely relevant or not, due to the subjectivity of PQA.

Question	ATTN+QE3	Riker+QE3
"can i use this for outdoors? thank..."	sensitive outside weather aviation camping d100 cop extensively sonar 60csx eave coordinate 1980 beltronics fahrenheit resonate ra dingy enemy 9500ix	outside outdoor backyard camping weather windy unprotected rain indoor rv camper atmosphere weatherproof hiking rooftop wherever desert
"how large be the base of the light stand? i need to know if these stand can fit in a smallish space."	ceiling riser plywood sanus symmetrical perpendicular shelf finesse firing endure attached peerless 50-inch cushioning sleeker stretchy 15inch sail	folding standing spaced headboard collapse stool heel pray centimeter small raise hang 161 hanging chair bookcase strong ledge pedestal liking
"how do you turn the alarm off?"	switched hardwired goodbye elevator reseat instantaneously tutorial uninterrupted shutdown shaft hitch shutoff swiping uninstalled direction quadrant	alert disarm deactivate beeping buzzer countdown unplug regulate annoying beep inactive disable checkbox snooze timer shutoff inactivity

Table 5: Qualitative comparison of word expansion with QE3. In general, the top few words expanded by Riker+QE3 (e.g., "*outside*", "*folding*", "*alert*") is more relevant than those using DNN attentions (e.g., "*sensitive*", "*ceiling*", "*switched*").

evaluation methodology suggested by [13] for interpretable machine learning, which advocates using the performance improvement of some interpretable model (i.e., the *tf-idf* based search in our setting) as proxy to evaluate the explanation/interpretation quality. We prefer this evaluation method to user studies (e.g., hiring humans to score the expanded keywords such as those in Table 5) in that it is more objective, much cheaper and easier to conduct. For example, it is not easy for humans to compare the two columns in Table 5 at large scale, especially when both contain some relevant keywords.

Table 6 summarizes the comparison between Riker and ATTN with both QE2 and QE3. We also list the performance of end2end DNN at the bottom despite its known lack of interpretability. As we can see, all methods outperform the RW baseline, indicating that the learned pairwise attentions from DNN indeed associate interpretable expansion words. Consistent with Riker's results, ATTN+QE2 is less effective than ATTN+QE3. Importantly, both ATTN+QE2 and ATTN+QE3 are inferior to their counterparts using Riker, showing that our methods can generate more interpretable expansion words. The reason is that the attention within DNN is trained only as intermediate weights used for aggregating latent word representations (Eq. 13), while our training objective directly optimize the score function s^{SPM}, which is the sum of pairwise word associations (Eq. 6). The end2end DNN model achieves the best performance when the dataset size is large, but its interpretability is worse than ATTN+QE2/QE3 as the last few steps (Eq. 14 and 15) are non-linear and the low-dimensional dense feature vectors still need further interpretation [14, 20]. *Our framework can be viewed as a model that constrains the question and review sentences to be matched in the lexical space with sparse bag-of-word features, and is optimized to strike a good balance between effectiveness and interpretability.*

7 RELATED WORK

(Product) answer sentence selection. Product QA on large-scale Amazon datasets is first studied in [30], which proposes a mixture-of-experts framework to jointly model review-answer and review-question relevance with latent vector representations. Wan et al.

	Appl.	Baby	Patio	Tools	Elec.
RW	79.653	89.043	84.792	86.436	87.082
Riker+QE2	81.427	91.367	89.681	90.175	89.537
ATTN+QE2	78.405	89.434	86.965	88.749	88.112
Riker+QE3	**84.629**	**92.086**	91.368	91.643	91.010
ATTN+QE3	80.682	90.070	87.729	89.428	88.587
DNN (end2end)	82.115	90.318	**91.756**	**94.158**	**95.927**

Table 6: Quantitative comparison of Riker, end2end DNN and using its ATTeNtion for QE on test set. For methods using keyword-based search (except end2end DNN), higher retrieval performance (AUC) corresponds to better interpretability reflected by the *tf-idf* proxy [13].

[43] build upon the above framework to handle the situation where questions can have multiple answers and reviews can be subjective by including more features (e.g., reviewer expertise and biases). Similar to [30], our framework currently models text information only and optimizes one correct answer, but can be extended to the scenario in [43] by averaging the scores of all correct answers and incorporating non-text features in our objective functions. Yu et al. [48] focus on "yes-no" questions only, and shows that learning latent product aspects and aspect-specific embeddings can help make binary predictions. Outside the product domain, answer sentence selection (e.g., [41, 42]) has been a popular topic in general. Our internal word re-weighting function and soft pairwise matching module are related to attention mechanism (e.g., [50]) and pairwise semantic interactions (e.g., [31, 41]), which are two popular techniques applied in general answer sentence selection models, but differ from them in the sense that we use the explicit weights (f^{int} and f^{ext}) to directly rank answers, while their techniques are integrated in latent vector representations in a less interpretable end2end fashion.

Query expansion. Query expansion aims to automatically expand a query with additional terms to get better search results, and has been a longstanding topic [6, 23, 25, 38, 46]. Question expansion strategies in our work (especially QE1) are most related to recent word embedding based techniques such as [24]. Diaz et al. [12] further show that fine-tuning word embeddings on domain specific corpus can help get better performance, but it requires re-training word embeddings for every query. Zamani et al. [49] propose learning offline word embeddings based on "relevance" instead of "proximity". In this work, we do not focus on developing novel QE strategies, and instead apply existing or most intuitive ones to employ the learned word weights and associations by Riker and test its effectiveness. It is interesting for future work to see whether QE techniques learned from other data sources may be combined with Riker to achieve overall better performances.

Interpretable machine learning. Interpretable machine learning receives a lot of attention recently in a broad range of fields [16–18]. Some works focus on understanding general machine learning algorithms including deep neural networks, e.g., through proxy models [37], salient mapping [3], interpreting latent semantic representations [5], or adversarial networks [10]. On the application level, people have explored making systems more interpretable to users including recommendation systems [2], visual QA [32], multiple choice answers [40] and so on. To the best of our knowledge,

we make the first effort towards interpretable PQA and advocate mining keyword, rather than vector, representations of a question to boost the effectiveness of keyword-bases search.

8 CONCLUSION

This work proposes a new hybrid framework combining the advantages of deep models and keyword-based search towards effective yet interpretable PQA. We employ an easily interpretable *tf-idf* based IR module to rank answers, but in order to address the lexical gap problem, we propose Riker to mine rich keyword representations for customer questions, consisting of re-weighted internal words and associated external words. Experimental results show that the mined keyword representations can help improve PQA performance substantially over existing PQA methods, while at the same time preserve good interpretability of the keyword-based search paradigm.

ACKNOWLEDGMENTS

This research was sponsored in part by the Army Research Office under cooperative agreements W911NF-17-1-0412, NSF Grant IIS1815674, and Ohio Supercomputer Center [8]. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, Vol. 16. 265–283.
[2] Q. Ai, V. Azizi, X. Chen, and Y. Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *arXiv preprint arXiv:1805.03352* (2018).
[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.
[4] R. Baeza-Yates, B. Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463.
[5] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796* (2017).
[6] C. Buckley, G. Salton, J. Allan, and A. Singhal. 1995. Automatic query expansion using SMART: TREC 3. *NIST special publication sp* (1995), 69–69.
[7] D. Carmel, L. Lewin-Eytan, and Y. Maarek. 2018. Product Question Answering Using Customer Generated Content-Research Challenges. In *SIGIR*. 1349–1350.
[8] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. http://osc.edu/ark:/19495/f5s1ph73
[9] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038* (2016).
[10] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*. 2172–2180.
[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[12] F. Diaz, B. Mitra, and N. Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* (2016).
[13] F. Doshi-Velez and B. Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).
[14] M. Du, N. Liu, and X. Hu. 2018. Techniques for interpretable machine learning. *arXiv preprint arXiv:1808.00033* (2018).
[15] J. L. Fleiss and J. Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement* 33, 3 (1973), 613–619.
[16] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. 2018. Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning. *arXiv preprint arXiv:1806.00069* (2018).
[17] B. Goodman and S. Flaxman. 2016. European Union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813* (2016).
[18] D. Gunning. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* (2017).
[19] Matthew H. and Ines M. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear* (2017).
[20] S. Jain and B. C. Wallace. 2019. Attention is not Explanation. *arXiv preprint arXiv:1902.10186* (2019).
[21] K. Järvelin and J. Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM TOIS* 20, 4 (2002), 422–446.
[22] K S. Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management* 36, 6 (2000), 809–840.
[23] S. Kuzi, D. Carmel, A. Libov, and A. Raviv. 2017. Query Expansion for Email Search. In *SIGIR*. 849–852.
[24] S. Kuzi, A. Shtok, and O. Kurland. 2016. Query expansion using word embeddings. In *CIKM*. 1929–1932.
[25] V. Lavrenko and W. B. Croft. 2001. Relevance Based Language Models. In *SIGIR*.
[26] Z. C. Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).
[27] Y. Lv and C. Zhai. 2011. Lower-bounding term frequency normalization. In *CIKM*. 7–16.
[28] G. Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631* (2018).
[29] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.
[30] J. McAuley and A. Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *WWW*. 625–635.
[31] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* (2016).
[32] D. H. Park, L. A. Hendricks, Z. Akata, A. Rohrbach, B. Schiele, T. Darrell, and M. Rohrbach. 2018. Multimodal Explanations: Justifying Decisions and Pointing to the Evidence. In *CVPR*.
[33] J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.
[34] J. Pérez-Iglesias, J. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein. 2009. Integrating the probabilistic models BM25/BM25F into Lucene. *arXiv preprint arXiv:0911.5046* (2009).
[35] R. Pryzant, S. Basu, and K. Sone. 2018. Interpretable Neural Architectures for Attributing an Ad's Performance to its Writing Style. In *EMNLP Workshop BlackboxNLP*. 125–135.
[36] R. Řehůřek and P. Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *LREC Workshop on New Challenges for NLP Frameworks*. 45–50.
[37] M. T. Ribeiro, S. Singh, and C. Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*. 1135–1144.
[38] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *ACL*. 464–471.
[39] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.. In *ICCV*. 618–626.
[40] R. Sharp, M. Surdeanu, P. Jansen, M. A. Valenzuela-Escárcega, P. Clark, and M. Hammond. 2017. Tell me why: Using question answering as distant supervision for answer justification. In *CoNLL*. 69–79.
[41] G. Shen, Y. Yang, and Z. H. Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In *EMNLP*. 1190–1200.
[42] Y. Tay, L. A. Tuan, and S. C. Hui. 2018. Multi-Cast Attention Networks for Retrieval-based Question Answering and Response Prediction. *arXiv preprint arXiv:1806.00778* (2018).
[43] M. Wan and J. McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *ICDM*. 489–498.
[44] F. Wang and C. Rudin. 2015. Falling rule lists. In *AISTATS*.
[45] M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL*.
[46] J. Xu and W. B. Croft. 1996. Query expansion using local and global document analysis. In *SIGIR*. 4–11.
[47] Y. Yang, W. T. Yih, and C. Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. 2013–2018.
[48] Q. Yu and W. Lam. 2018. Aware Answer Prediction for Product-Related Questions Incorporating Aspects. In *WSDM*. 691–699.
[49] H. Zamani and W. B. Croft. 2017. Relevance-based Word Embedding. In *SIGIR*. 505–514.
[50] X. Zhang, S. Li, L. Sha, and H. Wang. 2017. Attentive Interactive Neural Networks for Answer Selection in Community Question Answering.. In *AAAI*. 3525–3531.
[51] J. Zhao, Y. Su, Z. Guan, and H. Sun. 2017. An End-to-End Deep Framework for Answer Triggering with a Novel Group-Level Objective. In *EMNLP*.

A IMPLEMENTATION DETAILS

Some extra implementation details of this work are clarified here. For data pre-processing, we use SpaCy [19] to lemmatize and lowercase words. The vocabulary (Table 1) is constructed by including all words from the training set that either appear in the GloVe[33] pre-trained vocabulary or appear more than 5 times, except for the Electronics department, where for efficiency reasons we further truncate the vocabulary size to 100K from more than 300K words obtained by the above preprocessing method.

For the question expansion baselines PRF-TFIDF and PRF-RM (Table 2), we grid search the pseudo-relevant sentence number N from [10, 20, ..., 60] and top expansion word number K from [10, 20, ..., 80]. For all other QE baselines using Algorithm 1, we grid search N from [40, 50, 60, 70] and δ from [0.05, 0.10, 0.15, 0.20]. We select the best combination for each dataset based on dev set. For keeping the same level of interpretability, the expanded queries are evaluated through the same *tf-idf* based IR across our experiments, which is different from the language model based IR used in state-of-the-art word embedding based QE works [12, 49].

In the evaluation of the end2end DNN baseline (Table 6), because of the high computational cost to encode all answer candidates through Bi-RNN, we approximate its AUC performance by randomly sampling 1000 negative answers for each question following the method used in [30].

We use Tensorflow [1] to implement Riker. For each training epoch, we randomly sample 5 non-answers from the train/dev answer pool for each QA pair. We tune the model hyper-parameters based on the Baby domain dev set because of its moderate size for efficiency concerns, and use them for all other domains. We use the Adam optimizer with the learning rate set at 5e-4 and batch size set at 64. We add L2 regularization with coefficient 1e-4. For the keyword-based IR module, we employ an off-the-shelf inverted index based IR tool [36] and use the standard BM25 function with default parameters [34] to rank answers or review sentences. Source code and data will be available at: https://github.com/jiez-osu/PQA.