Discovering Relevant Reviews for Answering Product-related Queries

Shiwei Zhang[†§], Jey Han Lau[‡], Xiuzhen Zhang[†*], Jeffrey Chan[†], and Cecile Paris[§]

[†]*RMIT University, Australia. Email: {shiwei.zhang, xiuzhen.zhang, jeffrey.chan}@rmit.edu.au*
[‡]*The University of Melbourne, Australia. Email: jeyhan.lau@gmail.com*
[§]*CSIRO Data61, Australia. Email: cecile.paris@data61.csiro.au*

Abstract—With the increasing popularity of e-commerce, the number of product-related queries generated by customers is growing. Answering these queries manually in real time is infeasible, and so automatic question-answering systems can be immensely helpful. Product queries are, however, very different from open-domain questions: they tend to be product-specific and the answers they demand can be very subjective. Previous research suggests that reviews are a valuable resource for answering product queries, but a key challenge is the language mismatch between user queries and reviews. To address this, we propose two neural models that discover relevant reviews for answering product queries. We demonstrate that our best model produces strong performance, outperforming state-of-the-art systems by consistently finding the most relevant reviews for product queries.

Keywords-Product Question Answering, Mixtures of Experts, Deep Learning

I. INTRODUCTION

E-commerce websites are becoming increasingly sophisticated, and websites such as Amazon, eBay and TaoBao not only sell products, but also serve as a platform for users to ask questions, compare and review products. Online customers, like offline shoppers, often have questions prior to a purchase, and it is infeasible for sellers to provide answers to every customer query in real time due to the high volume of online traffic. As product queries tend to be very product-specific, finding similar queries from other products and retrieving their answers is unlikely to be useful. An alternative solution might be to rely on reviews, as they can provide insights to the query.

To illustrate how reviews might be useful, two product queries, ground truth answers and top-ranked review sentences (by our model) are shown in Table I. We can see that the review sentences are relevant and helpful; in the second case they provide more elaboration than the "Yes" groud truth answer.

There are several studies that leverage reviews to tackle product-related Question Answering (QA). A mixture-of-experts model is trained to distinguish real answers from non-answers based on reviews in [1], [2]. In a similar vein, generative models [3], [4] are proposed to extract useful information from reviews to generate answers. We saw some promising results from these studies, suggesting reviews are

*Corresponding author. Email: xiuzhen.zhang@rmit.edu.au.

Table I
ANSWER AND TOP-RANKED REVIEW FOR TWO QUERIES ON AMAZON.

- Q: If you take this as directed twice a day how long does this size bottle last?
- A: About a month. I skip some doses but my doctor retested and my iron levels increased after 3 month.
- R: Following the directions of taking it twice a day, it may last a whole month.

- Q: Does this have a volume control?
- A: Yes
- R: The remote on the cord is really easy to use, top button is volume up, bottom is volume down, middle is pause/play/answer.

a useful resource for answering product queries. Success in this task has the implication of relieving the community from answering the growing number of product queries manually. It also shortens the time customers waiting for a response and improves their online shopping experience.

With that said, it is difficult to directly train a supervised system to find relevant reviews for product queries, as there is a lack of ground truth, i.e. there is no direct association between product queries and reviews. Creating an annotated data set (by marking the relevance of reviews for each query) would inevitably be a colossal and expensive effort, given the large number of products and reviews.

To tackle this, a better approach is to use existing question and answer pairs as supervision signals. By framing answer prediction as the objective, we can then decompose the problem to two sub-tasks, by learning how to match: (1) a query with a review; and (2) a review with an answer. Post-training, the learnt model for the first sub-task can then be used to extract potentially useful reviews for a query. This idea is first proposed by [1].

A key challenge in our task is the language mismatch between questions and reviews. [1] proposed a number of features and a word-based relevance function to align questions with reviews and reviews with answers. In contrast, we propose neural models that require little feature engineering, and as demonstrated in our experiments, tackle the language mismatch between questions and reviews better. To facilitate replication and future research, we release source code of our

models.[1] A supplementary material for the paper is available online.[2]

To summarise, our main contributions in this paper are:

- We propose two neural models to predict answers for product queries using reviews, with the end goal of using the model to find relevant reviews given a query.
- Our end-to-end neural models use raw texts as input, and as such do not require feature engineering.
- We demonstrate that our best model outperforms current state-of-the-art benchmarks in: (1) predicting answers for a query; and (2) finding the most relevant reviews to a query.

II. RELATED WORK

Existing studies on product-related question answering using reviews can be broadly divided into extractive approaches [1], [2], [6], [7] and abstractive approaches [3], [4]. Extractive approaches extract snippets from relevant reviews to answer questions while abstractive methods generate answers by drawing from the vocabulary. With both approaches, a crucial first step is to find relevant reviews to a question.

The closest work to our paper is the system *Mixtures of Opinions for Question Answering* (MOQA), proposed in [1]. MOQA is trained on a large corpus of previously answered questions to learn a relevance function that can surface relevant reviews, where relevance is measured by how well a review helps identify the correct answer. MOQA relies on using traditional word-based relevance function and manual feature engineering to rank reviews.

Other related studies focus on different aspects of the task. [2] study the opinion divergence problem where a question has multiple answers that are opinionated and divergent. [7] propose a question answering system that targets yes-no binary questions. [8] addresses product compatibility related questions.

III. METHODOLOGY

Given a product query, our goal is to surface relevant reviews that can potentially provide answers. Due to the lack of annotated query and review pairs — i.e. we do not have data where relevant and irrelevant reviews for a query are marked — we can't directly train a classification/ranking model to label/rank reviews for a given query.

We do, however, have an abundance of labelled answers for queries (e.g. by mining existing queries that have been answered by users). In order to learn the relevance between review and query, we leverage the supervision signal of community query and answer pairs by building a model whose objective is to compute the probability of an answer

given a query *via reviews*. Formally:

$$P(a|q) = \sum^{r} P(r|q)P(a|r, q)$$

where a, q and r denote answer, query and review respectively.

In other words, we are decomposing the relevance between answer and query ($P(a|q)$) into two functions, by computing the relevance between: (1) review and query ($P(r|q)$); and (2) answer and review ($P(a|r, q)$). This formulation implicitly assumes that product reviews are useful when predicting the most relevant answer given a query; we contend that this is a reasonable assumption, and found that empirically the learnt relevance function for review and query extracts useful reviews based on user studies.

In practice, instead of building ranking models that compute the full probability distribution for a set of answers, we treat it as a pairwise classification problem where the goal of the model is to score an answer for a given query, and the model is optimised by learning to score a true answer higher than a randomly selected non-answer.

This framework of using reviews to score answers is originally proposed in [1], and it is inspired by the mixture-of-experts classifier (MOE) [10], which uses a number of weak classifiers or *experts*, each weighted by its confidence, to make a prediction. By seeing each review r as an *expert*, the term $P(r|q)$ can be interpreted as an expert's confidence, and the term $P(a|r, q)$ as its prediction. [1] calls their model MOQA (Mixtures of Opinions for Question Answering), and it serves as our baseline system for comparison.

MOQA parameterises the relevance functions using simple pairwise similarity metrics (such as BM25+, ROUGE-L) and bilinear models and trains using classical learning models. MOQA originally proposes two models with different objectives for tackling binary yes/no questions and open-ended questions. In this paper, we focus only on the latter, as they are arguably the more interesting questions. For these open-ended questions, MOQA is optimised by training on the logistic loss of the probability of scoring the true answer higher than a randomly selected non-answer. We propose to implement a neural extension of their framework, by exploring by both simple networks and modern Transformer-based architectures [9]. Our neural architecture affords greater flexibility in terms of how we want to model the relationship between query/review/answer; it can also be easily extended to incorporate additional metadata on the questions or reviews as a future direction.

A. Neural Extension

In our neural models, we parameterise the relevance functions $P(r|q)$ and $P(a|r, q)$ with neural networks. We explore both simple networks and Transformer-based networks [9] for these relevance functions, with each encoding different assumptions about how it models the relationship.

Table II
DATA SET STATISTICS

Category	#questions	#products	#reviews	#r/p
Home and Kitchen	184,439	24,501	2,012,777	20
Sports and Outdoors	146,891	19,332	1,013,196	19
Automotive	89,923	12,536	395,872	10
Cell Phones	85,865	10,407	1,534,094	28
Health and Personal Care	80,496	10,860	1,162,587	26
Tools and Home Improv.	101,088	13,397	752,947	18
Patio Lawn and Garden	59,595	7,986	451,473	19

As with MOQA, rather than computing the probability distribution over an answer set ($P(a|q)$), we compute a bounded ($0 - 1$) score for an answer ($S(a|q)$), and optimise based on a margin loss:

$$\min(0, \delta - S(a|q) + S(a'|q))$$

where a is a real answer, a' is a randomly selected non-answer, and δ is the margin hyper-parameter.

Intuitively, the model is trained to score a real answer from a non-answer with at least a difference of δ (or a penalty will be incurred).

B. NNQA

NNQA uses FASTTEXT [11] as a sentence encoder to create a vector representation h_q, h_a and h_r for q, a and r respectively, by taking an average of the pre-trained word embeddings in the sentence. Given the sentence encodings, we model the relevance functions with a bilinear function:

$$S(r|q) = \sigma(h_q \mathbf{W_1} h_r^\mathsf{T})$$
$$S(a|r) = \sigma(h_r \mathbf{W_2} h_a^\mathsf{T})$$

where $\mathbf{W_*}$ are model parameters.

We score an answer by combining them:

$$S(a|q) = \sum_{r \in \mathcal{R}} S(a|r) S(r|q)$$

where $\mathcal{R}$ is a set of reviews for q.

Note that in NNQA, we relax the MOE assumption where the accumulated confidence sums to 1 (i.e. $\sum_{r \in \mathcal{R}} S(r|q) = 1.0$). In preliminary experiments we did incorporate this constraint by applying a softmax operation over the confidence scores, but found that this formulation of applying sigmoid to bound confidence scores works better empirically.

C. BERTQA

Our second neural model uses BERT [5], a state-of-the-art Transformer-based model that has shown to perform competitively over a range of NLP tasks, from question-answering to paraphrase detection to natural language inference. BERT is pre-trained over a large corpus, and it can be further fine-tuned to subsequent tasks of interest by adding additional task-specific layers.

Our BERT model (henceforth BERTQA) shares the same framework as NNQA; the core difference is that the relevance functions ($S(r|q)$ and $S(a|r)$) are computed directly with the BERT encoder:[3]

$$S(r|q) = \text{softmax}(W_1^\mathsf{T} \text{BERT}(r, q))$$
$$S(a|r) = \sigma(W_2^\mathsf{T} \text{BERT}(a, r))$$

An important advantage of BERTQA is that it allows for a more fine-grained analysis between two pairs of texts, as: (1) its self-attention mechanism provides a means to assess token-level similarity for all tokens between the pairs; and (2) it uses at sub-word units (i.e. word pieces). For NNQA, as we first generate a compressed representation of the texts (h_r, h_q and h_a) before computing their relevance, we lose the token-level analysis that BERTQA has.

Also, unlike NNQA, for BERTQA we found that empirically it is beneficial to keep the MOE constraint (i.e. $\sum_{r \in \mathcal{R}} S(r|q) = 1.0$), and so we apply a softmax operation for $S(r|q)$ over all reviews in this model.

D. Review Filtering

In all models (our neural models and MOQA), reviews are broken into individual sentences, and a "review" (r) is a review sentence rather than the full review. For this reason, a product typically has a large number of reviews (i.e. review sentences; see statistics at Table II), and it incurs a significant computational overhead when we consider all reviews for each query.

To this end, we explore several methods to pre-rank the reviews for each query, with the idea that most reviews are irrelevant and so can be removed before feeding them to the neural models. We experimented with TF-IDF based unsupervised and machine learning based supervised models, and found that a simple BERT classifier works best.

Our BERT filter (henceforth FLTR) works by first fine-tuning a pre-trained BERT on (query, answer) pairs to predict whether an answer is a real answer for the query.[4] When training our neural models (NNQA or BERTQA), we can optionally apply FLTR to (query, review) pairs to pre-rank the reviews and consider only the top-N reviews. Note that there is a domain mismatch between training and test inference for FLTR, since it's trained on (query, answer) pairs but used for (query, review); but as a preliminary filtering system we found that it works relatively well.

E. Cross-domain Pre-training

The data set we use has a number distinct categories. When training the neural models, we treat each product

---

[3]Note that BERT produces a hidden state for each word in the sentences; we use only the hidden state of the CLS token, which is special token prepended at the start of the sentence pair. Also, the BERT encoder is shared for (r, q) and (a, r), and updated during fine-tuning.

[4]Negative training examples are generated by sampling random answers from other queries.

Table III
ANSWER PREDICTION AUC PERFORMANCE OF ALL MODELS

	MOQA	NNQA	NNQA+PT	NNQA+FLTR+PT	BERTQA+FLTR	BERTQA+FLTR+PT
Home and Kitchen	0.8946	0.8015	0.8878	0.8518	0.9086	**0.9253**
Sports and Outdoors	0.8766	0.8012	0.8871	0.8358	0.9097	**0.9272**
Automotive	0.8445	0.7942	0.8715	0.8015	0.8710	**0.8923**
Cell Phones	0.8746	0.7534	0.8737	0.8254	0.8591	**0.8774**
Health and Personal Care	0.8770	0.8321	0.8952	0.8338	0.9076	**0.9263**
Tools and Home Improv.	0.8719	0.8080	0.8874	0.8375	0.8961	**0.9153**
Patio Lawn and Garden	0.8291	0.7743	0.8752	0.8274	0.8943	**0.9141**
Average	0.8671	0.7950	0.8826	0.8305	0.8923	**0.9111**

category as an independent domain and train a number of separate models. Observing that questions across categories occasionally share similar questions, e.g. queries about compatibility or size, we explore pre-training a general model that combines data from all categories, and then fine-tuning it further for each of the domain. We hypothesise that this training procedure should help tackle both domain-invariant queries (via pre-training) and domain-specific queries (via fine-tuning). We denote models that use cross-domain pre-training with the label PT, e.g. NNQA+PT means the NNQA model is first pre-trained with all data and then fine-tuned for each domain.

IV. EXPERIMENTS

A. Data

We use the Amazon QA and review data set developed by [1] for our experiments. We experiment with seven product categories; statistics for these categories are presented in Table II.

Following MOQA, we split the data in the same train/development/test ratio,[5] and break reviews into sentences. That is, a *review* is in practice a *review sentence*, rather than the full review. The rationale for using review sentences is that from an application perspective, displaying a relevant sentence for a query is more user-friendly than presenting a lengthy full review.

B. Training Details

We tune our neural models based the development partition. For MOQA, we use the open source implementation and its default optimal configuration.[6] When applying FLTR, we keep only the top-10 reviews. For the neural models (NNQA and BERTQA), we set an upper limit for the total number of reviews to 100 (and discard the rest).

---

[5]Note that we are unable to produce the exact same splits used in the original MOQA publication, as the splitting algorithm provided by the authors uses a random seed. For this reason we re-run MOQA on our data for proper comparison.

[6]https://cseweb.ucsd.edu/~jmcauley/

C. Quantitative Evaluation: Answer Prediction

We first present a quantitative assessment of our models. Due to the lack of annotated query and review data, we evaluate the models based on their accuracy of predicting the right answer for a given query.

We follow MOQA and use AUC as the evaluation metric:

$$\frac{1}{\mathcal{Q}} \sum_{q \in \mathcal{Q}} \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} S(a > a')$$

where a is a real answer, a' is randomly-sampled non-answer, $\mathcal{A}$ the set of all non-answers,[7] and $\mathcal{Q}$ the set of all queries.

Intuitively, the AUC calculates for each query the proportion of cases where the model assigns a higher score to the real answer. A score of 1.0 indicates perfect accuracy. We present the full results in Table III.

Looking at MOQA vs. NNQA, we see that MOQA is superior to NNQA in all categories, and about 7% better on average. Applying cross-domain pre-training (NNQA+PT) improves the model substantially, with the largest gain in Automotive; on average NNQA+PT now outperforms MOQA by 2%. But when we apply FLTR to filter away irrelevant reviews and keep only the top-10 reviews (NNQA+FLTR+PT), we see a consistent drop in all categories, suggesting that the FLTR procedure has inevitably discarded potentially useful candidates. We try increasing the number of reviews to keep (up to top-50), but found that its performance is still consistently worse than NNQA+PT.

Next we look at BERTQA. Due to memory constraints, we are unable to run BERTQA using all reviews, and so present only the results of BERTQA+FLTR variants. Encouragingly, even with the filter, BERTQA+FLTR has a superior performance over baseline (MOQA) and all NNQA models, with an average performance improvement of over 3% over MOQA and 10% over NNQA. When we pre-train it over multiple domains (BERTQA+FLTR+PT), we see another substantial improvement, widening the gap between BERTQA and other models further. We hypothesise that the strong performance of BERTQA is likely to be

---

[7]For every query, we randomly sampled 100 non-answers.

attributed to its self-attention architecture, which facilitates a more fine-grained analysis of words between sentences.

D. Qualitative Survey: Relative Review Quality

Recall that our end goal is to extract helpful reviews that can provide answers for a product query. Although our quantitative evaluation (Section IV-C) demonstrates that BERTQA is a good model, its ability to discover useful reviews has yet to be directly assessed.

To this end, we use the trained relevance function ($S(r|q)$) of the models to select the most relevant review sentence for some queries, and ask users to judge the relevance of these reviews. We use the Amazon Mechanical Turk as the crowdsourcing platform.[8]

For the survey, we test three models: MOQA, NNQA+PT and BERTQA+FLTR+PT. MOQA serves as our baseline, and it has been previously shown to outperform unsupervised retrieval baseline (Okapi-BM25+/ROUGE) [1]. NNQA+PT and BERTQA+FLTR+PT are chosen because they are the best neural models based on their answer prediction performance.

We experiment with three product categories: "Cell Phones" (most popular category, with an average of 28 reviews per product); "Patio Lawn and Garden" (least popular category); and "Sports and Outdoors" (average in terms of popularity). For each category, we randomly selected 100 questions, and for each question, we select the most relevant review ranked by each of the 3 models. A crowdworker is presented the product query, product image, and the 3 top-ranked reviews, and they are asked to select the most relevant review based on the query.

We present 3 queries in a HIT, and one of the queries is a control question where the 3 "reviews" consist of a real answer and 2 randomly selected non-answers. Each HIT is annotated by 7 workers. The control question serves a check to confirm that our workers are performing the task properly.[9]

We present the survey results in Table IV. For each category, we calculate the number of times each model is selected by the user. Across 3 categories, we see that BERTQA+FLTR+PT reviews are consistently selected by users as being most relevant. Surprisingly, MOQA performs better than NNQA+PT in 2 out of 3 categories, showing that better performance in answer prediction does not necessarily translate to review extraction.

Considering that the candidate set for MOQA is the full set of reviews, while for NNQA+FLTR and BERTQA+FLTR+PT it is up to 100 or 10 reviews respectively, we expect that BERTQA+FLTR+PT might be handicapped. The user study, however, reveals that this is

---

[8]https://www.mturk.com/.

[9]We turn on the Masters qualification requirement for survey, and found that all workers answer the control questions correctly.

Table IV
NUMBER OF SELECTED REVIEWS FOR MOQA, NNQA+PT AND BERTQA+FLTR+PT

	MOQA	NNQA+PT	BERTQA+FLTR+PT
Sports.	24	32	**44**
Cell Phones	35	17	**48**
Patio Lawn.	31	30	**39**

Table V
USER STUDY ON COMPARISON OF THE TOP-1 RANKED REVIEW FROM FLTR AND BERTQA+FLTR+PT

	FLTR	BERTQA+FLTR+PT
Sports and Outdoor	42	**58**
Cell Phones	36	**64**
Patio Lawn and Garden	48	**52**

clearly not the case. In fact, our results suggest that relevant reviews can often be found among the 10 candidates, suggesting that pre-ranked reviews (by FLTR) are good.

This then raises a natural question: if the top ten FLTR reviews are already good, do we need re-rank them with BERTQA? To better understand this, we conduct a second survey to compare BERTQA+FLTR+PT vs. FLTR, We follow the same procedure as before, by randomly selecting one hundred questions from the 3 categories. For each question, we select the top review ranked by BERTQA+FLTR+PT and FLTR, and ask workers to select the more relevant review.

Table V shows the number of reviews selected by users for the two models. Encouragingly, BERTQA+FLTR+PT reviews are consistently selected as being more relevant, showing that the re-ranking step by BERTQA is important.
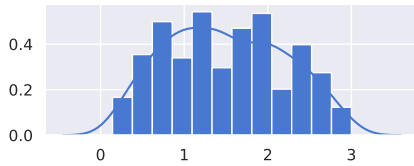
E. Qualitative Survey: Absolute Review Quality

Table VI
AVERAGE HELPFULNESS SCORES FOR 2 REVIEWS.

- Q: How many screen protector come with this package?
- R: Also, I was a little concerned that there was only 1 screen protector in the package.
- Score: 2.85

- Q: What're the dimensions of the largest planter?
- R: The planters are a nice size for plants.
- Score: 1.56

In the previous section, we conduct user studies to compare a number of models on how well they find relevant reviews. Although BERTQA+FLTR+PT is shown to outperform the other models (relative performance), we are yet to measure the actual utility of its reviews (absolute performance). To this end we conduct a second user study and ask annotators to judge how well the reviews answer

Figure 1. **Distribution of average helpfulness score. (Mean = 1.50, std = 0.70)**

Q:	Will these gloves keep hands warm while **driving in winter**?
R:	No, these aren't "warm" gloves, but they do still take the edge off **gripping an ice cold steering wheel**.
Q:	I want to use these pants for **surf fishing**. are they any good?
R:	The pants fit well and keep their shape **in the water**.
Q:	How is the microphone. do you **sound clear to other callers**?
R:	The headset is quite comfortable and is very **clear on both ends of the conversation**.

the questions. Note that we test only the reviews selected by BERTQA+FLTR+PT here.

We randomly selected 200 questions from the same three categories we used previously. Annotators are asked to rate how well/helpful a review answers the question on an ordinal scale from 0 to 3, where 0 indicates that review does not answer the question and is not relevant, and 3 indicates that review directly answers the question. The final helpfulness score for each review is the average rating given by 7 annotators; we present some examples in Table VI.

The distribution of the helpfulness scores is displayed in Figure 1. Interestingly, the results are rather mixed; we can see that not all of reviews are helpful, suggesting that there is still plenty of room for improvement.

F. Language Mismatch

The language used in queries and reviews tend to be very different, and we hypothesise that the strength of BERTQA lies in its ability to align paraphrases without overlapping words (owing to its sub-word tokenisation and deep self-attention layers). To this end, we calculate word-level Jaccard similarity between each question and the most relevant review as ranked by MOQA and BERTQA+FLTR+PT across all categories. The average Jaccard similarity for MOQA and BERTQA+FLTR+PT is 0.097 and 0.069 respectively. The significantly lower similarity value of BERTQA+FLTR+PT confirms our hypothesis: even though BERTQA+FLTR+PT finds more relevant reviews, its reviews tend to have less overlapping words with questions. We present several examples of top-ranked reviews by BERTQA+FLTR+PT in Table VII. These examples demonstrate that the reviews selected by BERTQA+FLTR+PT uses different words/phrases to the questions but their meanings are similar.

V. CONCLUSION

Discovering helpful reviews to answer product-related queries is a challenging problem due to the language mismatch between queries and reviews, and the lack of annotated reviews. In this work, we adopt a mixtures of expert inspired framework, by leveraging reviews to predict answers for queries. We present two neural models, a simple model (NNQA) and a Transformer-based model (BERTQA). We evaluate the models in terms of answer prediction and their ability to find relevant reviews, and show that BERTQA produces the best performance in both tasks, demonstrating the importance of its architecture which allows a more fine-grained analysis between two sources of text. As a future direction, we are interested in using the extracted reviews to generate natural answers for product queries.

REFERENCES

[1] J. McAuley and A. Yang, "Addressing complex and subjective product-related queries with customer reviews," in *WWW*, 2016.

[2] M. Wan and J. McAuley, "Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems," in *ICDM*, 2016.

[3] S. Gao, Z. Ren, *et. al*, "Product-aware answer generation in e-commerce question-answering," in *WSDM*, 2019.

[4] S. Chen, C. Li, *et. al*, "Driven answer generation for product-related questions in e-commerce," in *WSDM*, 2019.

[5] J. Devlin, M.-W. Chang, *et. al*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[6] Q. Yu, W. Lam, and Z. Wang, "Responding e-commerce product questions via exploiting qa collections and reviews," in *COLING*, 2018.

[7] Q. Yu and W. Lam, "Aware answer prediction for product-related questions incorporating aspects," in *WSDM*, 2018.

[8] H. Xu, S. Xie, *et. al*, "Dual attention network for product compatibility and function satisfiability analysis," in *AAAI*, 2018.

[9] A. Vaswani, N. Shazeer, *et al.*, "Attention is all you need," in *NIPS*, 2017.

[10] R. A. Jacobs and M. I. Jordan *et al.*, "Adaptive mixtures of local experts." *Neural computation*, 1991.

[11] A. Joulin, E. Grave, and P. B. T. Mikolov, "Bag of tricks for efficient text classification," in *EACL*, 2017.