A Simple End-to-End Question Answering Model for Product Information

Tuan Manh Lai
Adobe Research
tlai@adobe.com

Trung Bui
Adobe Research
bui@adobe.com

Sheng Li
Adobe Research
sheli@adobe.com

Nedim Lipka
Adobe Research
lipka@adobe.com

Abstract

When evaluating a potential product purchase, customers may have many questions in mind. They want to get adequate information to determine whether the product of interest is worth their money. In this paper we present a simple deep learning model for answering questions regarding product facts and specifications. Given a question and a product specification, the model outputs a score indicating their relevance. To train and evaluate our proposed model, we collected a dataset of 7,119 questions that are related to 153 different products. Experimental results demonstrate that — despite its simplicity — the performance of our model is shown to be comparable to a more complex state-of-the-art baseline.

1 Introduction

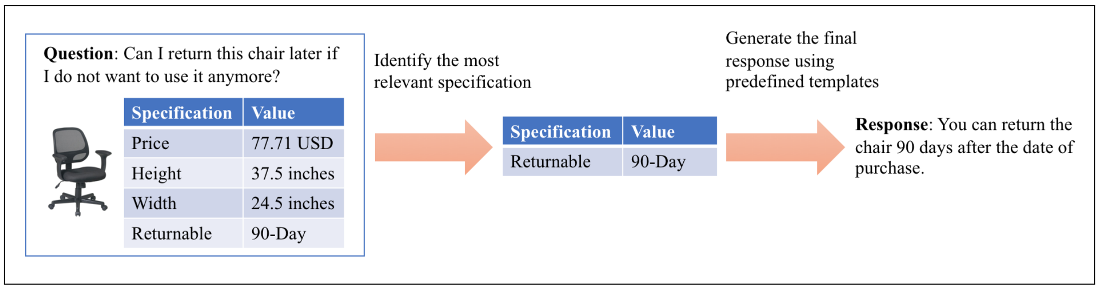
Customers ask many questions before buying products. Developing a general question answering system to assist customers is challenging, due to the diversity of questions. In this paper, we focus on the task of answering questions regarding product facts and specifications. We formalize the task as follows: Given a question Q about a product P and the list of specifications $(s_1, s_2, ..., s_M)$ of P, the goal is to identify the specification that is most relevant to Q. M is the number of specifications of P, and s_i is the i^{th} specification of P. In this formulation, the task is similar to the answer selection problem (Rao et al., 2016; Bian et al., 2017; Shen et al., 2017). 'Answers' shall be individual product specifications in this case. After identifying the most relevant specification, the final response sentence is generated using predefined templates (Cui et al., 2017). Figure 1 illustrates the overall process.

In this paper, we present a simple deep learning model for selecting the product specification that is most relevant to a given question from a set of candidate specifications. Given a question-specification pair, the model will output a score indicating their relevance. To train and evaluate our model, we collected a dataset of 7,119 questions, covering 153 different products. Despite its simplicity, the performance of our model is shown to be comparable to a more complex state-of-the-art baseline.

2 Related Work

2.1 Answer Selection

Answer selection is an active research field and has drawn a lot of attention. Given a question and a set of candidate answers, the task is to identify which of the candidates contains the correct answer to the question. Two types of deep learning frameworks have been proposed for tackling the answer selection problem. One is the Siamese framework (Bromley et al., 1993) and the other is the Compare-Aggregate framework (Wang et al., 2017; Bian et al., 2017; Shen et al., 2017). In the Siamese framework, the same encoder (e.g., a CNN or a RNN) is used to map each input sentence to a vector representation individually. After that, the final output is determined solely based on the encoded vectors. There is no explicit interaction between the sentences during the encoding process. On the other hand, the Compare-Aggregate framework aims to capture more interactive features between sentences in consideration, therefore typically has better performance when evaluated on public datasets such as TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015).



Question: Can I return this chair later if I do not want to use it anymore?

Specification	Value
Price	77.71 USD
Height	37.5 inches
Width	24.5 inches
Returnable	90-Day

Specification	Value
Returnable	90-Day

Response: You can return the chair 90 days after the date of purchase.

Identify the most relevant specification
Generate the final response using predefined templates

Figure 1: Answering questions regarding product facts and specifications

2.2 Customer Service Chatbot

The most closely related branches of work to ours are probably customer service chatbots for e-commerce websites. An example can be the Shopbot [1] of eBay. Shopbot aims at helping consumers narrow down the best deals from eBays over a billion listings. The bot's main focus is to understand the user intent and then make personalized recommendations. Unlike Shopbot, here we do not focus on making product recommendations. Instead we aim to develop a model for answering questions about product specifications. Another example is the SuperAgent (Cui et al., 2017), a powerful chatbot designed to improve online shopping experience. Given a specific product page and a customer question, SuperAgent selects the best answer from multiple data sources such as in-page product information, existing customer questions & answers, and customer reviews of the product. Even though SuperAgent has a component for answering questions about product specifications, the novelties of our work are: 1) a new simple deep learning model for answering questions about product facts and specifications 2) a new method for collecting data to train and evaluate our model.

3 Model Architecture

Given a question and a set of candidate specifications, the goal is to identify the most relevant specification. We aim to train a classifier that takes a question and a specification name as input and predicts whether the specification is relevant to the question. During inference, given a question, the trained classifier is used to assign a score to every candidate specification based on how relevant the specification is. After that, the top-ranked specification is selected.

[1]https://shopbot.ebay.com

A common trait of a number of recent state-of-the-art methods for answer selection is the use of the Compare-Aggregate architecture (Wang et al., 2017; Bian et al., 2017; Shen et al., 2017). Under this architecture, vector representations of smaller units (such as words) of the input sentences are compared. And then the comparison results are aggregated (e.g., by a CNN or a RNN) to determine the relationship of the input sentences. Compared to Siamese models, most Compare-Aggregate models are more complicated and can capture more interactive features between input sentences.

Our task of matching questions and product specifications is similar to the answer selection problem. "Answers" shall be individual product specifications. However, in this case the name of a specification is relatively short. Therefore, our hypothesis is that a well-designed Siamese model would perform as well as a more complicated Compare-Aggregate model. The added complexity of comparing vector representations of smaller units may not be needed as the specification name is already short and descriptive. To this end, we propose a new Siamese model for tackling our problem. We show the overall architecture of our model in Figure 2. Given a question Q and a specification name S, the model calculates a score indicating their relevance through the following layers.

Word Representation Layer. Using word embeddings pre-trained with word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), we transform Q and S into two sequences $Q_e = [\mathbf{e}_1^Q, \mathbf{e}_2^Q, ..., \mathbf{e}_m^Q]$ and $S_e = [\mathbf{e}_1^S, \mathbf{e}_2^S, ..., \mathbf{e}_n^S]$, where $\mathbf{e}_i^Q$ is the embedding of the i^{th} word of the question and $\mathbf{e}_j^S$ is the embedding of the j^{th} word of the specification name. m and n are the lengths of Q and S, respectively.

BiLSTM Layer. We use a bi-directional

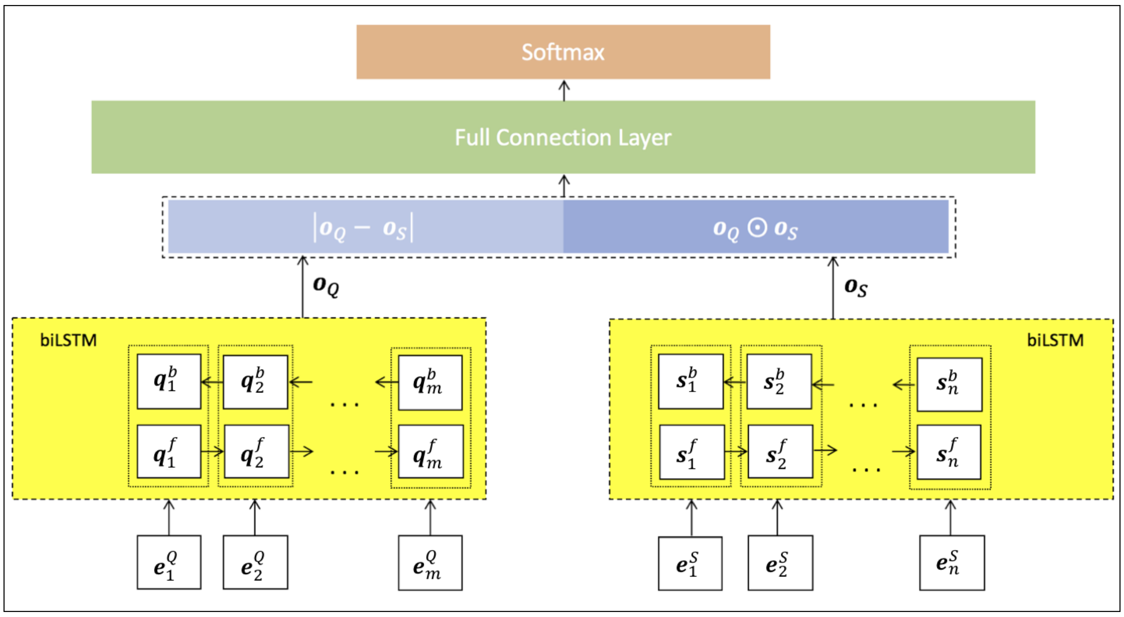


Figure 2: Architecture of our model

LSTM (Hochreiter and Schmidhuber, 1997) to obtain a context-aware vector representation for each position of Q and S. We feed Q_e and S_e individually into a parameter shared bi-directional LSTM model. For the question Q:

$$\mathbf{q}_i^f = \overrightarrow{LSTM}(\mathbf{q}_{i-1}^f, \mathbf{e}_i^Q) \qquad i = 1, ..., m$$
$$\mathbf{q}_i^b = \overleftarrow{LSTM}(\mathbf{q}_{i+1}^b, \mathbf{e}_i^Q) \qquad i = m, ..., 1$$

where $\mathbf{q}_i^f$ is a vector representation of the first i words in the question (i.e., $[\mathbf{e}_1^Q, \mathbf{e}_2^Q, ..., \mathbf{e}_i^Q]$), $\mathbf{q}_i^b$ is a vector representing the context of the last $m - i + 1$ words in the question (i.e., $[\mathbf{e}_m^Q, \mathbf{e}_{m-1}^Q, ..., \mathbf{e}_i^Q]$). Similarly, we use the same bi-directional LSTM to encode S:

$$\mathbf{s}_j^f = \overrightarrow{LSTM}(\mathbf{s}_{j-1}^f, \mathbf{e}_j^S) \qquad j = 1, ..., n$$
$$\mathbf{s}_j^b = \overleftarrow{LSTM}(\mathbf{s}_{j+1}^b, \mathbf{e}_j^S) \qquad j = n, ..., 1$$

The context-aware representation at each position of Q or S is obtained by concatenating the two corresponding output sequences from both directions, i.e., $\mathbf{q}_i = \mathbf{q}_i^f \mid\mid \mathbf{q}_i^b$ and $\mathbf{s}_j = \mathbf{s}_j^f \mid\mid \mathbf{s}_j^b$. The final representations of the question and the specification are generated by applying the max-pooling operation on the context-aware representations. We denote the final representation of the question as $\mathbf{o}_Q$ and denote the final representation of the answer as $\mathbf{o}_S$.

Comparison and Output Layers. Following the approach mentioned in (Tai et al., 2015), two feature vectors are calculated from the final encodings $\mathbf{o}_Q$ and $\mathbf{o}_S$: (1) the absolute difference of the two vectors $|\mathbf{o}_Q - \mathbf{o}_S|$; (2) the element-wise multiplication of the two vectors $|\mathbf{o}_Q \odot \mathbf{o}_S|$. The features are then concatenated and fed into a fully connected layer and a softmax layer to produce the final score indicating the probability that specification S is relevant to question Q.

4 Data Collection

The dataset used for experiments is created using Amazon Mechanical Turk (MTurk) [2], an online labor market. MTurk connects requesters (people who have works to be done) and workers (people who work on tasks for money). Requesters can post small tasks for workers to complete for a fee. These small tasks are referred to as HITs or human intelligence tasks. An example of a HIT is finding objects in an image or transcribing an audio file. Requesters have several options for ensuring their HITs are completed in a high-quality manner. Requesters have the opportunity to determine whether to approve completed HITs before having to pay for them. In addition, requesters can also limit which workers are eligible to complete their tasks based on certain criteria.

We crawled the information of products listed in the Home Depot website [3]. For each product, we create HITs where workers are asked to write

[2]https://www.mturk.com
[3]https://www.homedepot.com

questions regarding the specifications of the product. Figure 3 shows a sample HIT, including the instructions, which are shown to every participated worker. In this sample HIT, a question for the specification “Product Height (in.)” can be “How tall is this shredder?” or “What is the height of this shredder?”. To work on the HITs, workers are required to have a 98% HIT approval rate, a minimum of 800 HITs approved, and be located in the United States or Canada. The constraints ensure that the participated workers can provide good questions in English. The final dataset consists of 7,119 question-specification pairs in total, covering 369 kinds of specifications extracted from 153 products. Even though in this work we focus on products listed in the Home Depot website, the data collection process is applicable to other popular e-commerce websites such as Amazon whose product pages typically have a section for product facts and specifications.

5 Experiments and Results

5.1 Training and Evaluation

We set up two different experimental settings. The only difference between the two settings is the way in which we split up the collected HomeDepot dataset into training set, development set, and test set:

1. We divide the dataset so that the test set has no products in common with the training set or the development set.

2. We divide the dataset so that the test set has no specifications in common with the training set or the development set. This is different from the first setting, because two different products may have some specifications in common. For example, a chair and a table usually have a same specification called ‘Product Weight’.

In both settings, the proportions of the training set, development set, and test set are roughly 80%, 10%, and 10% of the total questions, respectively.

During training, the objective is to minimize the cross entropy of all question-specification pairs in the training set:

$$loss(\theta) = -log \prod_i p_\theta(y^{(i)}|Q^{(i)}, S^{(i)})$$

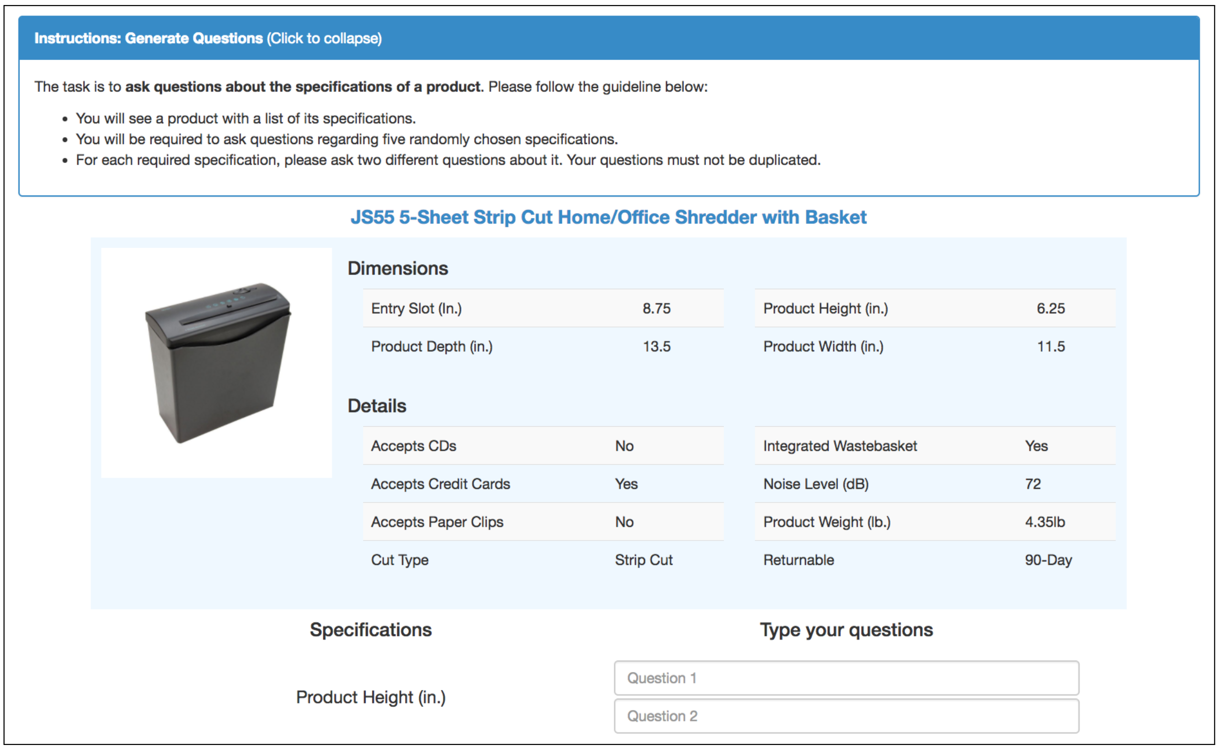
where $Q^{(i)}$ and $S^{(i)}$ represent a question-specification pair in the training set, $y^{(i)}$ indicates whether specification $S^{(i)}$ is relevant to question $Q^{(i)}$, and p_θ is the predicted probability with model weights θ. We use all possible question-specification pairs for training. In other words, if there are k questions about a product and the product has h specifications, there are $h \times k$ question-specification examples related to the product, and exactly k of them are positive examples. During testing, for every question about a product, we sort the specifications of the product in descending order based on the predicted probability of being relevant. After that, we calculate the precision at 1 (P@1), precision at 1 (P@2), and precision at 3 (P@3) of our model.

We compare the performance of our model with the unigram model mentioned in (Yu et al., 2014) and the IWAN model proposed in (Shen et al., 2017). The unigram model is a simple bag-of-words model. It first generates a vector representation for each input sentence by summing over the embeddings of all words in the sentence. The final output is then determined based on the generated vector representations. The unigram model is less complicated than our model. On the other hand, the IWAN model belongs to the Compare-Aggregate framework, and it is more sophisticated than our model. In addition to comparing between the fine-grained word representations of the input sentences, the IWAN model also has an inter-weighted layer for evaluating the importance of each word in each sentence. The IWAN model currently achieves state-of-the-art performance on public datasets such as TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015).

We make use of the GloVe word embeddings (Pennington et al., 2014) when training the models. We did try the word2vec word embeddings (Mikolov et al., 2013), however they gave worse performances than GloVe. We tune the hyper-parameters of each model using the development set.

5.2 Results

Table 1 shows the performances of all the models in the first setting. Table 2 shows the performances of all the models in the second setting. The IWAN model and our model clearly outperform the unigram model. In addition, in both settings, our model’s performance is comparable to


Instructions: Generate Questions (Click to collapse)

The task is to **ask questions about the specifications of a product**. Please follow the guideline below:

- You will see a product with a list of its specifications.
- You will be required to ask questions regarding five randomly chosen specifications.
- For each required specification, please ask two different questions about it. Your questions must not be duplicated.

JS55 5-Sheet Strip Cut Home/Office Shredder with Basket

Dimensions

Entry Slot (In.)	8.75	Product Height (in.)	6.25
Product Depth (in.)	13.5	Product Width (in.)	11.5

Details

Accepts CDs	No	Integrated Wastebasket	Yes
Accepts Credit Cards	Yes	Noise Level (dB)	72
Accepts Paper Clips	No	Product Weight (lb.)	4.35lb
Cut Type	Strip Cut	Returnable	90-Day

Specifications	Type your questions
Product Height (in.)	Question 1
	Question 2

Figure 3: An example of a HIT

Model	P@1	P@2	P@3
Unigram	0.802	0.904	0.927
IWAN	**0.852**	0.927	**0.964**
Our model	0.850	**0.930**	**0.964**

Table 1: Test results in the setting where the test set has no product in common with the training set or the development set

Model	P@1	P@2	P@3
Unigram	0.399	0.529	0.627
IWAN	0.525	**0.661**	**0.789**
Our model	**0.563**	0.640	0.759

Table 2: Test results in the setting where the test set has no specification in common with the training set or the development set

the performance of the IWAN model despite being much simpler. We measured the speeds of our model and the IWAN model. Our proposed model is about 8% faster than the IWAN model. In addition, we see that all models perform worse in the second setting than the first setting. This may due to the fact that in the first setting two different products in the train set and the test set may still have many specifications in common (e.g., a LG TV and a Samsung TV).

6 Conclusion

In this work we explore the task of answering questions related to product facts and specifications. We propose a new, simple deep learning model for tackling the task. To train and evaluate the model, we collected a dataset of question-specification pairs using MTurk. Experimental results show that our model's performance is comparable to a state-of-the-art baseline despite having less complexity. Our proposed model takes less time for training and inference than the state-of-the-art baseline.

Recently, researchers collected a large volume of community question answering data and a large volume of product reviews from the Amazon website (McAuley and Yang, 2016). In the future, we plan to investigate transfer learning techniques to utilize this large dataset for improving the performance of our proposed model.

References

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *CIKM*.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7:669–688.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. Superagent: A customer service chatbot for e-commerce websites. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9 8:1735–80.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *WWW*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Yi Yang, Scott Wen-tau Yih, and Chris Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. ACL Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen G. Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.