ProductQnA: Answering User Questions on E-Commerce Product Pages

Ashish Kulkarni*
India Machine Learning, Amazon
kulkashi@amazon.com

Kartik Mehta*
India Machine Learning, Amazon
kartim@amazon.com

Shweta Garg*
India Machine Learning, Amazon
shwegarg@amazon.com

Vidit Bansal*
India Machine Learning, Amazon
bansalv@amazon.com

Nikhil Rasiwasia
India Machine Learning, Amazon
rasiwasi@amazon.com

Srinivasan H Sengamedu
India Machine Learning, Amazon
sengamed@amazon.com

ABSTRACT

Product pages on e-commerce websites often overwhelm their customers with a wealth of data, making discovery of relevant information a challenge. Motivated by this, here, we present a novel framework to answer both *factoid* and *non-factoid* user questions on product pages. We propose several question-answer matching models leveraging both deep learned distributional semantics and semantics imposed by a structured resource like a domain specific ontology. The proposed framework supports the use of a combination of these models and we show, through empirical evaluation, that a cascade of these models does much better in meeting the high precision requirements of such a question-answering system. Evaluation on user asked questions shows that the proposed system achieves 66% higher precision[1] as compared to IDF-weighted average of word vectors baseline [1].

CCS CONCEPTS

• **Information systems** → **Question answering**; • **Applied computing** → *Online shopping*.

KEYWORDS

question answering; deep learning; chatbot; e-commerce

ACM Reference Format:
Ashish Kulkarni, Kartik Mehta, Shweta Garg, Vidit Bansal, Nikhil Rasiwasia, and Srinivasan H Sengamedu. 2019. ProductQnA: Answering User Questions on E-Commerce Product Pages. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308560.3316597

1 INTRODUCTION

Online e-commerce systems play a vital role in connecting product sellers and end consumers at scale. However, consumers often struggle to navigate through the millions of products on offer and therefore, the success of these systems relies on their ability to seamlessly support customers in their product discovery and research. This has motivated a lot of work in the areas of product search, recommendation, information extraction, summarization, and recently, automatic question answering [17, 20] and chatbots [22]. In this work, we are concerned with the specific problem of answering customer questions on e-commerce product pages. Product detail pages often contain a wealth of information contributed by both sellers (product title, description, features, *etc.*) and customers (reviews, community question-answers, *etc.*). However, in their effort to offer the most comprehensive product information, the amount of data on these pages has grown so much, that for a top selling product, the detail page typically spans over six to eight thousand words, filling up around 15 A4 sheets. Customers also face an increased complexity in product evaluation due to variations ("size" *vs.* "dimension") and implicit references to product features (e.g. for title "20.1 *MP* Point and Shoot Camera *Black*", 20.1 *MP* refers to resolution and *Black* refers to color attribute). On small form factor devices like mobile, customers might benefit from a system that answers their product-related questions without having to browse through the page.

Building such a question-answering system poses some interesting challenges.

Question intent: In addition to product feature-related questions (like, "size" or "resolution"), customers could ask other *factoid* questions like "what's in the box?", "does this work with canon?" or *non-factoid* questions like "is this worth the money?" Understanding question intent is key to generating an appropriate response.

Product attribute-value: The system should account for explicit and implicit references to product attributes and their values in both questions and candidate answer lines.

Semantic matching: Customers often use text variations (*eg.*"anti-shake" to refer to "image stabilization"), thus necessitating semantic matching of question and answer lines.

High precision: Providing incorrect answers would lead to a marred customer experience and add to their frustration.

Lack of training data: Unlike question answering systems for open domain, domain specific systems suffer from scarcity of training data and other resources like structured knowledge bases.

Addressing these challenges for domain specific question answering systems is the primary focus of this work. We believe that building such a system would involve an interplay of different components for identifying question intent, attribute name-value

*These authors made equal contribution
[1]Evaluated at fixed coverage, where coverage is the number of questions that receive an answer. We cannot reveal the exact coverage number due to confidentiality

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316597

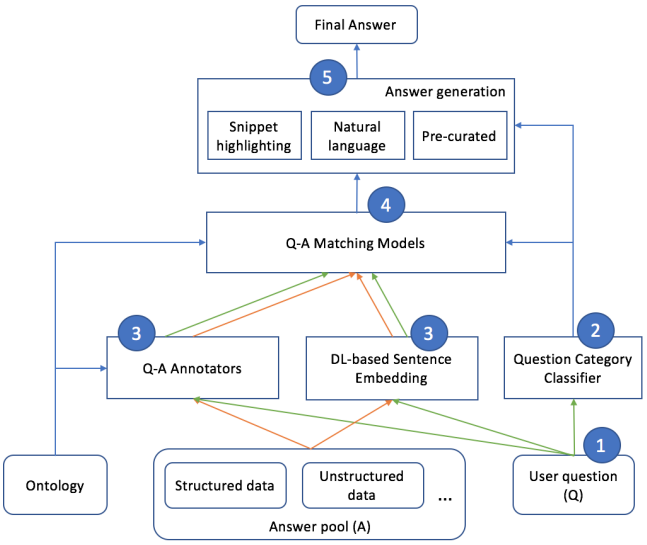


Figure 1: Framework for question-answering leveraging structured and distributed semantics. ① Framework receives a user question; ② Question category classifier classifies the question into one of the predefined categories; ③ Question and answer sentences are processed to generate their ontology-based annotations and deep learning-based embeddings; ④ Matching models rank the answer sentences for their relevance to the question; ⑤ Answer generation component generates the final answer based on the ranked answer sentences.

annotation-based on a structured knowledge base, semantic matching of question and answer lines, and final answer generation. We present a generic framework for in-domain question answering. The framework allows for a graceful combination of deep learning-based distributed semantics and semantics imposed by a structured resource like a domain ontology. Along with a question classifier to identify intent, the proposed system caters to the high precision requirement for a great customer experience. We present a detailed evaluation of different components of the framework and an ablation study underlining their contribution to the system performance.

2 RELATED WORK

The body of work closest to the proposed framework comes from the field of question answering for e-commerce. Yan *et al.* [22] recently presented a task-oriented dialog system that leverages an in-domain knowledge base, search logs and community sites to assist users for online shopping. Distinct from them, SuperAgent [3] takes advantage of in-page product descriptions and user-generated content to answer user questions for a product. While we are also concerned with in-page question answering, we present a more generic solution covering aspects of question understanding, question-answer representation and matching and answer generation. We support the efficacy of the proposed framework via a detailed empirical study.

Contribution of question answering and reading comprehension datasets, notably, TREC [18] and recently, SQUAD [13] and MS MARCO [11] has led to a lot of work in the area of open-domain question answering of factoid questions from a given document collection. Some of the earlier systems [14] made use of text and entity-level surface patterns as clues to right answers. Realizing that these approaches suffered from low recall and did not capture long-distance dependencies, some of the subsequent research extended these with other statistical signals from the corpus [15] or more complex patterns based on deep linguistics [12]. Other approaches based on hand crafted syntactic features [8] have also been explored. Although we are also concerned with answering user questions from a given passage of text, the domain of interest is limited (to e-commerce products, for instance), making it difficult to leverage existing language resources and knowledge bases in the open domain.

With deep learning gaining in popularity, there's a recent body of work in question answering that leverages dense representation of sentences composed from neural word embeddings [10]. Several sentence embedding approaches have emerged based on simple word vector averaging [21] or those leveraging the structure and sequence of words in a sentence using RNN, LSTM or CNN-based [6] architectures. When applied to the question answering task, some of the existing work is based on the semantic similarity of a question and a potential answer in a jointly learned embedding space [9], while others employ a classification or learning-to-rank approach over joint question-answer feature vectors [19]. While the proposed embedding models are inspired from some of the aforementioned approaches, we differ from them in that we complement the distributional semantics learned from these models with the structured semantics imposed by an ontology and combine these in a generic question answering framework. We show that a question answer matching model based on a combination of these features achieves much better results on an in-domain question answering task.

3 PRODUCTQNA FRAMEWORK

Figure 1 gives an overview of proposed ProductQnA (PQnA) framework. We are given a question q and a pool of candidate answer lines $\mathcal{A} = \{a_1, \ldots, a_n\}$. We then pose question answering as a ranking problem, where, the candidate answer lines are ranked based on their relevance to the question q and top-k answers $a'_1, \ldots, a'_k$ ($a'_i \in \mathcal{A}$) are selected for final answer generation if their relevance $s(a'_i)$ exceeds some threshold t. It is possible that none of the answer lines get selected if they all fail to meet the threshold.

We describe the ranking (or question-answer matching) models in more detail in the following sections. The matching models in the proposed question-answering framework (refer to Figure 1) are further aided by several other components which we also describe in detail below.

3.1 Ontology

An *ontology* describes the entity types in a domain and their interrelationships. We built an ontology for a large product category, where the entity types comprise products (camera, lens, tripod *etc.*), their attributes (dimension, resolution, *etc.*) and attribute values (20.1 MP, Black *etc.*) and the relationships capture their semantic relatedness, for instance, baby_monitor $\xrightarrow{\text{isA}}$ camera, security_camera $\xrightarrow{\text{hasA}}$

night_vision, resolution $\xrightarrow{\text{hasValue}}$ resolution_value. We bootstrap the ontology from existing in-domain knowledge bases and gazetteers (list of *colors*, *brands etc.*) and further augment it with entities extracted from semi-structured and unstructured corpus of product pages. Product attributes and their values often appear as feature bullets displayed in a tabular fashion on product pages. We exploit such structure on product pages to extract these attributes and their values. We also extract frequently occurring noun phrases, from the unstructured text, which are manually audited and merged into the ontology using Protégé[2]. The ontology that we thus curated, consists of 570 entity types spanning product categories like digital cameras, security cameras, lenses, tripods, bags and cases, batteries, films and others.

3.2 Question-Answer Annotators

An annotator extracts semantics from text by identifying entity mentions (like, *anti-shake* or 20.1 *MP*) in raw text and linking them to their canonical entities (*image_stabilization* and *resolution_value*, respectively) in an ontology. We annotate user questions and candidate answer lines to generate *annotations*, which are triples $\langle e, s_{begin}, s_{end} \rangle$, where, e is an entity in the ontology and s_{begin} and s_{end} define the span of the entity mention in the raw text line. We use three types of annotators:

Regular expression-based: Attribute values (*e.g.* 20.1 MP or 10 GB) often have a well defined signature and could be extracted using a regular expression annotator.

Gazetteer-based: Lists of certain attribute values like color, camera brand *etc.* are often readily available. We leverage these to define gazetteer-based annotators for attributes *color_value*, *camera_brand_value* and others.

Machine learning models: In order to capture semantic variations ("how long does this battery last?" is a reference to *battery_life*), we manually label annotations for a subset of user questions, $Q_{labeled}$ and use a k-NN classifier to annotate an unseen user question q. As distance metric, we use the Jaccard similarity between q and the questions in $Q_{labeled}$.

A union of the outputs from these annotators is then used as the final set of annotations, Q_{annot}, for a question and A_{annot}, for a candidate answer.

3.3 Deep Learning based Sentence Embedding

While annotators provide ontology-based semantic features for a sentence, we also use deep learning-based sentence embeddings leveraging distributional semantics of words and their context. The question and answer embeddings thus obtained serve as another input to the question-answer matching models. The embedding architecture (refer Figure 2) is inspired from the Siamese neural network [4]. Given a sentence, tokenized into words, the network takes as input their word embeddings, typically initialized with embeddings pre-trained on large in-domain corpora. These are then composed together in the following layers, using a bag-of-words or word sequence approach, to obtain the final sentence embedding. For the question-answering task, we project the question and a candidate answer in a shared embedding space and the network parameters are trained to minimize a task-specific loss function.

[2]https://protege.stanford.edu/.

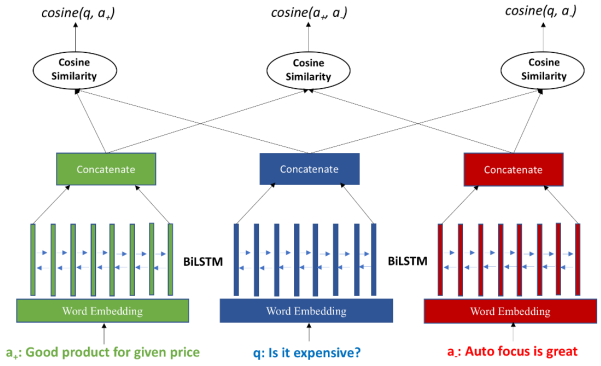


Figure 2: Model architecture for training deep learning-based sentence embedding. q is a question, a_+ is relevant answer to the question and a_- is any irrelevant statement.

We discuss the different sentence embedding approaches and loss functions below.

3.3.1 Sentence embedding using supervised word averaging: For a sentence $s = w_1 \dots w_n$, where, w_i is a word in s and $\boldsymbol{w_i} \in \mathbb{R}^d$ its embedding, the sentence embedding $\boldsymbol{l}$ is computed as: $\boldsymbol{l} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{w_i}$. We initialize word embeddings with random weights and learn them as part of supervised training. This simple approach of averaging word vectors has shown to give comparable performance to complex deep learning models such as LSTM for text classification [5] as well as text similarity problems [1, 21].

3.3.2 Sentence embedding using LSTM:. As against the bag-of-words approach above, LSTM takes the sequence of words into account. It produces a vector $\overrightarrow{\boldsymbol{l_t}}$ at each word w_t, from its word embedding $\boldsymbol{w_t}$ and that of its previous context $\boldsymbol{w_1} \dots \boldsymbol{w_{t-1}}$. In case of bi-LSTM, $\overleftarrow{\boldsymbol{l_t}}$ is similarly obtained by reversing the order of words in the sentence and taking into account $\boldsymbol{w_t}$ and its context $\boldsymbol{w_n} \dots \boldsymbol{w_{t+1}}$. The concatenation of output vector from each direction, $\overleftrightarrow{\boldsymbol{l}} = \overrightarrow{\boldsymbol{l_n}} || \overleftarrow{\boldsymbol{l_1}}$ is then used as the final sentence representation.

3.3.3 Loss functions: The embedding models discussed above are trained in a supervised manner, where the training data comprises triplets $\langle \boldsymbol{q}, \boldsymbol{a_+}, \boldsymbol{a_-} \rangle$ of embeddings of question, correct answer and an incorrect answer respectively. The training aims to minimize a task-specific loss function which we discuss next.

Weighted Log loss is defined in [7] as: $L_l = -\log p(\boldsymbol{q}, \boldsymbol{a_+}) - \eta \log(1 - p(\boldsymbol{q}, \boldsymbol{a_-}))$ where, $p(\boldsymbol{u}, \boldsymbol{v}) = 1/(1+\exp(-\boldsymbol{u}^T \boldsymbol{v}))$ and $0 < \eta \leq 1$ dampens highly representative negative samples in the training data. We use $\eta = 1$ in the experiments as we have balanced number of negative and positive samples.

Siamese Hinge loss is commonly used for *Siamese architectures* [9] and is defined as: $L_s = \max\{0, M - cosine(\boldsymbol{q}, \boldsymbol{a_+}) + cosine(\boldsymbol{q}, \boldsymbol{a_-})\}$, where M is the margin.

Triplet Hinge loss: We propose a stricter version of the above loss that additionally penalizes the similarity of $\boldsymbol{a_+}$ and $\boldsymbol{a_-}$. Also, inspired from [16], we use different margin for the three components of the loss. In our experiments, this loss function has been found to achieve better results than siamese hinge loss, as we discuss in

more detail in Section 5.1.

$$L_3 = \max\{0, M_1 - \cos(\boldsymbol{q}, \boldsymbol{a}_+)\} + \max\{0, \cos(\boldsymbol{q}, \boldsymbol{a}_-) - M_2\} + \max\{0, \cos(\boldsymbol{a}_+, \boldsymbol{a}_-) - M_3\} \quad (1)$$

3.4 Question-Answer Matching Model

The question-answer matching model receives as input the question and answer feature representations from the annotators and deep learning-based embedding models and generates a final list of answers. We use the following matching models.

Similarity-based ranking model: Given the question embedding $\boldsymbol{q}$ and answer embeddings $\{\boldsymbol{a_1}, \ldots, \boldsymbol{a_n}\}$, the similarity-based ranking model f_{deep} ranks the answers based on their cosine similarity $cos(\boldsymbol{q}, \boldsymbol{a_i})$ to the question in the shared embedding space. A ranked list of answers, with similarity score exceeding a threshold t, is generated as the output.

Annotation-based classification model: Let Q_{annot} and A_{annot} be the set of annotations for a question and a candidate answer respectively. The annotation-based classification model f_{annot} is a binary classifier that returns 1 if any entity $e_q \in Q_{annot}$ subsumes an entity $e_a \in A_{annot}$ and 0 otherwise. An entity e_i is said to subsume an entity e_j if at least one of these assertions holds true in the ontology: $e_i = e_j$, $e_j \xrightarrow{\text{isA}} e_i$, $e_i \xrightarrow{\text{hasA}} e_j$ or $e_i \xrightarrow{\text{hasValue}} e_j$.

Ensemble matching model: One could define an ensemble matching model combining the semantic signals from ontology-based annotations and deep learning-based embedding models. Here, we use a cascade of models, where, the candidate answers are first ranked based on f_{deep} and subsequently filtered by f_{annot} to generate a final list of top-k answers.

3.5 Question Category Classifier

Customer questions might span multiple categories (refer to Table 1). Identifying these might help in generating an appropriate response to the question. For instance, one could use question category as an additional feature to the matching models or have separate models based on question categories. Also, in order to maintain the high precision requirement, one might choose not to answer certain categories (e.g. *other*, where, often answer is not available on the page). Certain categories (*"greetings"*, *"shipping_delivery"*, *"warranty"*, *"returns_funds"*,*"used_refurbished"*) have limited surface forms and can be answered with precurated response. We term these categories as *stock categories* and the rest as *non-stock categories*.

Building such a question classifier poses multiple challenges: (1) class ambiguity (e.g. "how expensive is this camera compared to others" question is ambiguous with *price* and *related_product* as candidate classes), (2) spelling mistakes (e.g. "what is *prise*", "what is *brnad*"), (3) complex surface forms (e.g. "does it take picture" is *specs*, but "does it make sound when it takes picture" is *others*) and (4) multiple sub-questions. Also, lack of sufficient training data adds to the complexity of this problem. In order to deal with these challenges, we use deep learning-based architecture. Formally, given a question q, we learn a function $f(\boldsymbol{q})$ that maps it to one of the question categories $\{c_1, \ldots, c_k\}$ as in Table 1. While there are several choices to model $f(\boldsymbol{q})$ (refer to Section 5.2 for an empirical comparison), we use a CNN model similar to the one used by Yoon *et al.* [6]. We propose two extensions to this architecture to make the classifier robust to spelling mistakes and generalize to unseen specs attributes.

Table 1: Question categories and their proportion in data

Question Category	Example	Proportion
specs	What is the weight?	34.3%
compatibility	Will this work with Nikon D300?	10.8%
ratings_and_reviews	What is the customer rating?	5.8%
whats_in_the_box	What comes with camera?	3.6%
returns_refunds	How can I return this package?	2.3%
shipping_delivery	Can I get it delivered to India?	1.6%
related_product	what speaker are people using with the camera	1.6%
warranty	Does it come with a warranty?	1.4%
used_refurbished	Is this a new camera or a refurbished one?	1.0%
greetings	Good evening	0.9%
price	How much does it cost?	0.7%
gibberish	abcd	0.4%
other	How do you access the video footage?	35.6%

Enriching classifier with subword information: We augment our CNN-based question classifier with character n-grams (subwords) [2]. The resulting model (*CNN+Subw*) is found to be robust to spelling mistakes.

Enriching classifier with f_{annot}: Gathering training data for all *specs* attributes and their surface forms is a challenging task. f_{annot} (introduced in Section 3.4) could be used to annotate questions with attribute tags in order to reduce the training data sparsity. For instance, "what is resolution" is annotated as "what is specs_tag". We then train a multi-channel CNN [6], where we use two different inputs (original question for first channel and annotated question for the other channel). We refer to this model as CNN+Subw+f_{annot} and present empirical evaluation in section 5.2.

4 SYSTEM ARCHITECTURE

Based on PQnA framework discussed above, we propose a question answering system. Users can ask questions about the product and the system provides instant answers from three different sources - (1) seller provided *product data* (2) *user reviews* and (3) *community Q&A* (*CQnA*). User questions and all the product detail page data from the three sources are subjected to the proposed PQnA framework to generate the top-3 answers. The question category classifier first classifies the question into one of the question categories. For questions belonging to one of *specs, ratings_and_reviews, compatibility*, and *price*, we then rank the sentences for their relevance to the question using the ranking models. As discussed in Section 3.4, we use a cascade of f_{deep} and f_{annot} as the ensemble matching model

Table 2: Qualitative examples for PQnA system.

User Question	System Answer
How big is this camera?	• dimensions : 3.82 x 4.65 x 5.12 inches. • the camera is big, almost the size of dslr.
Would this be good for beginners?	• Perfect for the first-time GoPro user, or as a second camera. • HERO Session is simple and easy to use.
Does it work under water?	• waterproof to 50ft/ 15m, freezeproof to 14 f/ -10 c, shockproof to 5.8ft, and dustproof. • it takes excellent pics in and out of the water.
Does it have anti-shake?	• image stabilization : yes • also compensates for the shift-type camera shake common in macro shooting
How does it compare with rebel SL1?	**CQnA Question:** What is the difference between this camera and the Rebel SL1? **CQnA Answer:** The SL1 is more compact and the specs are not as good as the T5i
Is it expensive?	For the price this camera is good value for money for an amateur photographer.
Is it compatible with canon lenses?	The EOS Rebel T6 camera is compatible with all Canon lenses in the EF and EF-S lineup.

for *product data* and f_{deep} alone for *user reviews* and *CQnA* data. We use a set of pre-curated answers for questions belonging to *greetings, shipping_delivery, warranty*, and *returns_refunds*. Currently, we do not provide an answer to *whats_in_the_box, related_product* and *other* categories. Table 2 shows examples retrieved from the system.

5 EVALUATION

We use a random sample of 1340 questions (Table 1 shows the distribution) to evaluate the system for coverage (fraction of questions for which we retrieve an answer) and precision (fraction of questions for which top retrieved answer is correct). For comparison, we use IDF-weighted average of word vectors (referred as *IDF-vector-average* hereinafter) which has been found to be a strong baseline for textual similarity tasks [1]. At a fixed coverage[3], our proposed system (as described in Section 4) achieves 66% higher precision as compared to *IDF-vector-average*. In this section, we describe experimental setup and evaluation for various components of the system.

[3]The coverage point is decided based on business requirement. We cannot disclose the absolute value due to confidentiality

Table 3: Comparison of architecture and loss functions. Precision numbers are relative to *IDF-vector-average*

Model Architecture	Loss function	Precision	Latency (in ms) [4]
Supervised Word Averaging	Log loss	-14%	**49**
	Siamese Hinge Loss	21%	
	Triplet Hinge Loss	24%	
LSTM	Log loss	-26%	568
	Siamese Hinge Loss	21%	
	Triplet Hinge Loss	**30%**	

5.1 Evaluation of Matching Models

CQnA provides a natural source of labeled data to train the deep learning-based matching model, f_{deep}. For each question and answer pair in *CQnA*, we generate training triplets by sampling five incorrect answers. Though *CQnA* data consists of rich surface forms, it suffers from some gaps (e.g. Answer lines are often lengthy and are not representative of snippets from product data). In order to alleviate these limitations, we augment *CQnA* data with semi-automatically generated data by leveraging the ontology and question-answer annotators. For each attribute in the ontology, we create a set of question templates (for "price", the question templates include "what is the cost?", *etc.*). We run the annotators on product data to obtain the positive answers for each attribute and use negative sampling to obtain corresponding negative answers. The labeled set thus generated by combining both the resources, comprises over 15M training triplets. For evaluation, we use 10% of *CQnA* questions and sample hundred incorrect answers for each question. For a given question, we score all candidate answers (one correct and hundred incorrect answers) and return the answer with highest score. For each architecture, we compare precision on these highest scored candidates (*P@1*) and report numbers relative to *IDF-vector-average* baseline.

Table 3 shows performance of log loss, siamese hinge loss and triplet hinge loss (Eqn 1) for both LSTM and supervised word averaging models compared to *IDF-vector-average* baseline. Siamese network based loss functions (siamese hinge loss and triplet-hinge loss) perform significantly better than logloss. The proposed triplet-hinge loss (30% improvement for LSTM) has been observed to perform slightly better than standard hinge loss function (21% improvement for LSTM). This is likely due to the stricter nature of triplet-hinge loss function. While both the model architectures show comparative performance (with LSTM model performing slightly better), the supervised word averaging model is 11 times faster than the LSTM model during evaluation time[4]. Due to comparative performance and better latency, we use the supervised word averaging model in rest of the evaluation.

5.2 Evaluation of Question Category Classifier

We train the question category classifier on a randomly sampled set of 7000 questions and use a validation set of 700 questions for tuning

[4]These latency numbers are averaged over scoring 10000 ⟨ query, answer ⟩ pairs and were done on the following configuration: Intel(R) Xeon(R) CPU E5-2665 0 2.40GHz 8 core with 148.84 GB memory

Table 4: Comparison of different architectures for Question Classifier. All numbers reported are relative to FastText.

Model Architecture	Multi class Accuracy	AUC (weighted)
Logistic Regression	-6.5%	-3.1%
FastText	0.0%	0.0%
LSTM	-0.1%	0.4%
CNN	1.2%	0.3%
CNN+Subw	5.5%	2.6%
CNN+Subw+f_{annot}	**8.4%**	**3.3%**

the model hyper-parameters. We evaluate multiple classification models—logistic regression with bag-of-words features, FastText[5], LSTM and CNN. For the FastText architecture, we choose *ngram size* and *number of epochs* based on cross validation. We use similar setting as Yoon *et al.* [6] for CNN model and a single hidden layer of 128 dimensions for LSTM. We implemented the CNN and LSTM models in Keras[6] and chose best epoch based on performance on validation set. We use set of 1340 questions (refer to Section 4) and multi-class accuracy and multi-class weighted AUC for evaluation and report numbers relative to FastText architecture.

The logistic regression classifier trained with bag-of-word features leads to a drop of 6.5% in multi-class accuracy (refer to Table 4) signifying the complexity of the problem. The CNN model achieves 1.2% improvement whereas no significant improvements are observed using LSTM architecture. CNN with subword embeddings achieves 5.5% improvements and leveraging f_{annot} further improves this to 8.4%.

6 SYSTEM ABLATION STUDY

We analyze the effect of different components on system metrics using the same set of 1340 questions used for evaluating the overall system. We compute precision (*P@1*) and coverage at different thresholds and show the precision-coverage plot for each analysis. We select fixed coverage based on system requirement and report relative precision numbers for each setting[7].

6.1 Effect of Question Category Classification

We evaluate three settings:

NoQC: All questions are treated as belonging to *specs* category and evaluated using ensemble matching model (refer to Section 3.4).

EnhancedQC: We use canned response for stock categories, ensemble matching model for three categories (*specs*, *price* and *compatibility* categories), and we provide no response for three remaining categories (*other*, *related_product* and *whats_in_box*).

StockQC: For this setting, we use the *EnhancedQC* classifier with a single difference that all non-stock categories are considered as *specs*. We use canned response for stock category questions and evaluate all non-stock categories using ensemble matching model. The motivation for this setting is that stock categories have limited surface forms and it is easier to detect them as compared to detecting non-stock categories which can have large number of surface forms.

Figure 3a shows precision-coverage curve for this analysis[8, 9]. At fixed coverage, *StockQC* setting achieves 14% improvement over *NoQC* setting. Using *EnhancedQC* setting, this improvement further increases to 19%. It can be concluded that significant improvement in precision is observed by having a simplified classifier that can detect stock categories and further precision improvements are observed by having an enhanced classifier (EnhancedQC setting).

6.2 Comparison of Matching Models

We study the performance of different matching models (as introduced in Section 3.4) against the *IDF-vector-average* baseline (refer to Figure 3b [10]. f_{annot} provides only one operating point whereas other models provide flexibility to choose operating point based on performance requirements. f_{deep}, by virtue of exploiting semantic signals in the training data, shows 85% better precision over the unsupervised *IDF-vector-average* baseline. Using cascade of f_{annot} and f_{deep} (refer to Section 3.4), this further improves to 93%.

6.3 Comparison of Data Sources

We analyze the contribution of three data sources, *product data*, *user reviews* and *CQnA*, on the system performance. We restrict this analysis to only specs questions and f_{deep} model. For *CQnA* evaluation, we obtain two similarity scores based on (1) user question and *CQnA* question similarity and (2) user question and *CQnA* answer similarity. We take maximum of these two similarity scores for selecting answer from *CQnA* data. For the evaluation with combined data setting, we use one answer from each source and a question is considered as positively answered if at least one answer is relevant.

As can be observed (refer to Figure 3c[11]), best results are obtained from the product data whereas performance on reviews and *CQnA* are comparable. By combining answers from these three sources, the performance improves drastically (11% improvement as compared to answer only from product data).

7 CONCLUSION

We presented a novel end-to-end framework for answering questions on e-commerce product pages. Based on this framework, we propose a question answering system, which uses deep learning-based ranking model and ontology-based matching to answer question from three sources—product data, reviews and CQnA. A CNN-based question intent classifier helps in further improving the precision of the system. Our proposed system, using question classifier and cascade of deep learning-based ranking and annotation-based matching, achieves 66% higher precision as compared to *IDF-vector-average* baseline.

[5]https://github.com/facebookresearch/fastText

[6]https://keras.io/backend/

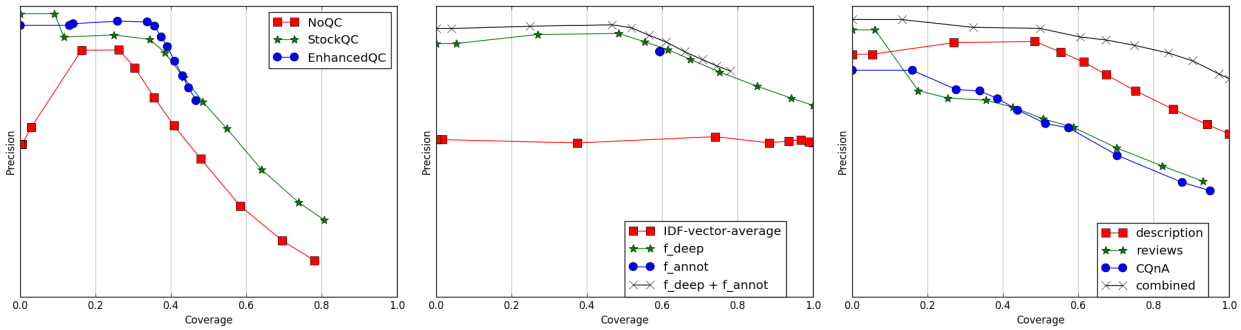
[7]Exact precision and coverage values are not disclosed due to confidentiality

[8]Coverage does not go till 100% due to f_{annot} filtering and filtering of question categories for *EnhancedQC* setting

[9]The *NoQC* setting has very low precision at low coverage, likely due to poor performance of deep learning model for questions belonging to "other" category

[10]f_{deep} has very low precision at low coverage due to errors introduce by incorrect parsing of data (e.g. "From the Manufacturer." answer for "who is the manufacturer?" question gets a score of 1 due to perfect word match after stopwords removal)

[11]The curves for reviews and *CQnA* data source do not go till coverage of 100% as reviews and *CQnA* are not available for newly introduced products



(a) (b) (c)

Figure 3: (a) Precision-coverage curve for different question classifier settings as evaluated on all test questions; (b) Precision-coverage curve for different matching models as evaluated on specs questions in test dataset; (c) Precision-coverage curve for different data sources as evaluated on specs questions in test dataset.

REFERENCES

[1] S. Arora, Y. Liang, and T. Ma. 2016. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. (2016).

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2016. Enriching Word Vectors with Subword Information. *CoRR* (2016).

[3] Lei Cui, Furu Wei, Shaohan Huang, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. SuperAgent: A Customer Service Chatbot for E-commerce Websites, In Proceedings of ACL 2017, System Demonstrations. 97–102.

[4] Y. LeCun E. Sackinger J. Bromley, I. Guyon and R. Shah. 1994. Signature Verification using a Siamese Time Delay Neural Network. In *NIPS*.

[5] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).

[6] Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. *arXiv preprint arXiv:1408.5882* (2014).

[7] S. Lee and Y. Hu. 2015. Joint Embedding of Query and Ad by Leveraging Implicit Feedback. In *EMNLP*.

[8] X. Li. 2003. Syntactic Features in Question Answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*.

[9] B. Xiang M. Tan, C. Santos and B. Zhou. 2015. LSTM-based Deep Learning Models for Non-Factoid Answer Selection. *arXiv preprint arXiv:1511.04108* (2015).

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[11] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. *arXiv preprint arXiv:1611.09268* (2016).

[12] F. Peng, R. Weischedel, A. Licuanan, and J. Xu. 2005. Combining Deep Linguistics Analysis and Surface Pattern Learning: A Hybrid Approach to Chinese Definitional Question Answering. In *HLT/EMNLP*.

[13] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *CoRR* (2016).

[14] D. Ravichandran and E. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *ACL*.

[15] D. Ravichandran, A. Ittycheriah, and S. Roukos. 2003. Automatic Derivation of Surface Text Patterns for a Maximum Entropy based Question Answering System. In *HLT-NAACL–Short Papers*.

[16] C. N. Santos, B. Xiang, and B. Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. *arXiv preprint arXiv:1504.06580* (2015).

[17] R. Soricut and E. Brill. 2006. Automatic Question Answering using The Web: Beyond the Factoid. *Information Retrieval* (2006).

[18] E. M. Voorhees. 2001. The TREC Question Answering Track. *Natural Language Engineering* (2001).

[19] D. Wang and E. Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *ACL-IJCNLP (Volume 2: Short Papers)*.

[20] M. Wang. 2006. A Survey of Answer Extraction Techniques in Factoid Question Answering. *Computational Linguistics* (2006).

[21] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. *arXiv preprint arXiv:1511.08198* (2015).

[22] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li. 2017. Building Task-Oriented Dialogue Systems for Online Shopping. In *AAAI*.