Addressing Complex and Subjective Product-Related Queries with Customer Reviews

Julian McAuley
University of California
San Diego
jmcauley@cse.ucsd.edu

Alex Yang
University of California
San Diego
alexyang@fb.com

ABSTRACT

Online reviews are often our first port of call when considering products and purchases online. When evaluating a potential purchase, we may have a specific query in mind, e.g. 'will this baby seat fit in the overhead compartment of a 747?' or 'will I like this album if I liked Taylor Swift's *1989*?'. To answer such questions we must either wade through huge volumes of consumer reviews hoping to find one that is relevant, or otherwise pose our question directly to the community via a Q/A system.

In this paper we hope to fuse these two paradigms: given a large volume of previously answered queries about products, we hope to automatically learn whether a review of a product is relevant to a given query. We formulate this as a machine learning problem using a mixture-of-experts-type framework—here each review is an 'expert' that gets to vote on the response to a particular query; simultaneously we learn a relevance function such that 'relevant' reviews are those that vote correctly. At test time this learned relevance function allows us to surface reviews that are relevant to new queries on-demand. We evaluate our system, *Moqa*, on a novel corpus of 1.4 million questions (and answers) and 13 million reviews. We show quantitatively that it is effective at addressing both binary and open-ended queries, and qualitatively that it surfaces reviews that human evaluators consider to be relevant.

Keywords

Relevance ranking; question answering; text modeling; reviews; bilinear models

1. INTRODUCTION

Consumer reviews are invaluable as a source of data to help people form opinions on a wide range of products. Beyond telling us whether a product is 'good' or 'bad', reviews tell us about a wide range of *personal experiences*; these include objective descriptions of the products' properties, subjective qualitative assessments, as well as unique use- (or failure-) cases.

The value and diversity of these opinions raises two questions of interest to us: (1) How can we help users navigate massive volumes of consumer opinions in order to find those that are *relevant* to their decision? And (2) how can we address specific *queries* that a user wishes to answer in order to evaluate a product?

To help users answer specific queries, review websites like *Amazon* offer community-Q/A systems that allow users to pose product-specific questions to other consumers.[1] Our goal here is to respond to such queries automatically and on-demand. To achieve this we make the basic insight that our two goals above naturally complement each other: given a large volume of community-Q/A data (i.e., questions and answers), and a large volume of reviews, we can automatically *learn* what makes a review relevant to a query.

We see several reasons why reviews might be a useful source of information to address product-related queries, especially compared to existing work that aims to solve Q/A-like tasks by building knowledge bases of facts about the entities in question:

- General question-answering is a challenging open problem. It is certainly hard to imagine that a query such as "Will this baby seat fit in the overhead compartment of a 747?" could be answered by building a knowledge-base using current techniques. However it is more plausible that some review of that product will contain information that is relevant to this query. By casting the problem as one of surfacing relevant opinions (rather than necessarily generating a conclusive answer), we can circumvent this difficulty, allowing us to handle complex and arbitrary queries.
- Fundamentally, many of the questions users ask on review websites will be those that *can't* be answered using knowledge bases derived from product specifications, but rather their questions will be concerned with subjective personal experiences. Reviews are a natural and rich source of data to address such queries.
- Finally, the massive volume and range of opinions makes review systems difficult to navigate, especially if a user is interested in some niche aspect of a product. Thus a system that identifies opinions relevant to a specific query is of fundamental value in helping users to navigate such large corpora of reviews.

To make our objectives more concrete, we aim to formalize the problem in terms of the following goal:

> *Goal:* Given a query about a particular product, we want to determine how relevant each review of that product is to the query, where 'relevance' is measured in terms of how helpful the review will be in terms of identifying the correct response.

The type of system we produce to address this goal is demonstrated in Figure 1. Here we surface opinions that are identified as being 'relevant' to the query, which can collectively vote (along with all other opinions, in proportion to their relevance) to determine the response to the query.

[1]E.g. `amazon.com/ask/questions/asin/B00B71FJU2`

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883044.

Product: BRAVEN BRV-1 Wireless Bluetooth Speaker

Query: "I want to use this with my iPad air while taking a jacuzzi bath. Will the volume be loud enough over the bath jets?"

customer opinions, ranked by relevance:	vote:
"The sound quality is great, especially for the size, and if you place the speaker on a hard surface it acts as a sound board, and the bass really kicks up."	yes
"If you are looking for a water resistant blue tooth speaker you will be very pleased with this product."	yes
"However if you are looking for something to throw a small party this just doesnt have the sound output."	no
etc.	etc.

Response: Yes

Figure 1: An example of how our system, *Moqa*, is used. This is a real output produced by *Moqa*, given the customer query about the product above. We simultaneously learn which customer opinions are 'relevant' to the query, as well as a prediction function that allows each opinion to 'vote' on the response, in proportion to its relevance. These relevance and prediction functions are learned automatically from large corpora of training queries and reviews.

This simple example demonstrates exactly the features that make our problem interesting and difficult: First, the query ('is this loud enough?') is inherently subjective, and depends on personal experience; it is hard to imagine that any fact-based knowledge repository could provide a satisfactory answer. Secondly, it is certainly a 'long-tail' query—it would be hard to find relevant opinions among the (300+) reviews for this product, so a system to automatically retrieve them is valuable. Third, it is linguistically complex—few of the important words in the query appear among the most relevant reviews (e.g. 'jacuzzi bath'/'loud enough')—this means that existing solutions based on word-level similarity are unlikely to be effective. This reveals the need to learn a complex definition of 'relevance' that is capable of accounting for subtle linguistic differences such as synonyms.

Finally, in the case of Figure 1, our model is able to respond to the query (in this instance correctly) with a binary answer. More importantly though, the opinions surfaced allow the user to determine the answer themselves—in this way we can extend our model to handle general open-ended queries, where the goal is not to answer the question *per se*, but rather to surface relevant opinions that will help the questioner form their own conclusion.

It seems then that to address our goal we'll need a system with two components: (1) A *relevance* function, to determine which reviews contain information relevant to a query, and (2) a prediction function, allowing relevant reviews to 'vote' on the correct answer.

However as we stated, our main goal is *not* to answer questions directly but rather to surface relevant opinions that will help the user answer the question themselves; thus it may seem as though this 'voting' function is not required. Indeed, at *test* time, only the relevance function is required—this is exactly the feature that shall allow our model to handle arbitrary, open-ended, and subjective queries. However the voting function is critical at *training* time, so that with a large corpus of already-answered questions, we can simultaneously learn relevance and voting functions such that 'relevant' reviews are those that vote for the correct answer.

The properties that we want above are captured by a classical machine learning framework known as *mixtures of experts* [18]. Mixtures of experts are traditionally used when one wishes to combine a series of 'weak learners'—there the goal is to simultaneously estimate (a) how 'expert' each predictor is with respect to a particular input and (b) the parameters of the predictors themselves. This is an elegant framework as it allows learners to 'focus' on inputs that they are good at classifying—it doesn't matter if they sometimes make incorrect predictions, so long as they correctly classify those instances where they are predicted to be experts.

In our setting, individual reviews or opinions are treated as experts that get to vote on the answer to each query; naturally some opinions will be unrelated to some queries, so we must also learn how relevant (i.e., expert) each opinion is with respect to each query. Our prediction (i.e., voting) function and relevance function are then learned simultaneously such that 'relevant' opinions are precisely those that are likely to vote correctly. At test time, the relevance function can be used directly to surface relevant opinions.

We evaluate our model using a novel corpus of questions and answers from *Amazon*. We consider both binary questions (such as the example in Figure 1), and open-ended questions, where reviews must vote amongst alternative answers. Quantitatively, we compare our technique to state-of-the-art methods for relevance ranking, and find that our learned definition of relevance is more capable of resolving queries compared to hand-crafted relevance measures.

Qualitatively, we evaluate our system by measuring whether human evaluators agree with the notion of 'relevance' that we learn. This is especially important for open-ended queries, where it is infeasible to answer questions directly, but rather we want to surface opinions that are helpful to the user.

1.1 Contributions

We summarize our contributions as follows: First, we develop a new method, *Moqa*, that is able to uncover opinions that are relevant to product-related queries, and to learn this notion of relevance from training data of previously answered questions. Second, we collect a large corpus of 1.4 million answered questions and 13 million reviews on which to train the model. Ours is among the first works to combine community Q/A and review data in this way, and certainly the first to do it at the scale considered here. Third, we evaluate our system against state-of-the-art approaches for relevance ranking, where we demonstrate (a) the need to learn the notion of 'relevance' from training data; (b) the need to handle heterogeneity between questions, reviews, and answers; and (c) the value of opinion data to answer product-related queries, as opposed to other data like product specifications.

Code and data is available on the first author's webpage.

2. RELATED WORK

The most closely related branches of work to ours are (1) those that aim to mine and summarize opinions and facets from documents (especially from review corpora), and (2) those that study Q/A systems in general. To our knowledge our work is among the first at the interface between these two tasks, i.e., to use consumer reviews as a means of answering general queries about products, though we build upon ideas from several related areas.

Document summarization. Perhaps most related to our goal of selecting relevant opinions among large corpora of reviews is the problem of *multi-document summarization* [25, 30]. Like ours, this task consists of finding relevant or 'salient' parts of documents [7, 30] and intelligently combining them. Most related are approaches that apply document summarization techniques to 'evaluative text' (i.e., reviews), in order to build an overview of opinions or product features [6, 22, 31]. In contrast to our contribution, most of the above work is not 'query-focused,' e.g. the goal is to summarize product features or positive vs. negative opinions, rather than to address specific queries, though we note a few exceptions below.

Relevance ranking. A key component of the above line of work is to learn whether a document (or a phrase within a document) is relevant to a given query. 'Relevance' can mean many things, from the 'quality' of the text [1], to its lexical salience [10], or its diversity compared to already-selected documents [6]. In query-focused settings, one needs a query-specific notion of relevance, i.e., to determine whether a document is relevant in the context of a given query. For this task, simple (yet effective) word-level similarity measures have been developed, such as Okapi BM25, a state-of-the-art TF-IDF-based relevance ranking measure [20, 26]. A natural limitation one must overcome though is that queries and documents may be linguistically heterogeneous, so that word-level measures may fail [3, 46]. This can be addressed by making use of grammatical rules and phrase-level approaches (e.g. ROUGE measures [44]), or through probabilistic language models ranging from classical methods [37] to recent approaches based on deep networks [23, 41]. We discuss ranking measures more in Section 3.1.

Opinion mining. Studying consumer opinions, especially through rating and review datasets is a broad and varied topic. Review text has been used to augment 'traditional' recommender systems by finding the aspects or facets that are relevant to people's opinions [14, 28, 43] and, more related to our goal, to find 'helpful' reviews [4, 9] or experts on particular topics [34]. There has also been work on generating summaries of product features [17], including work using multi-document summarization as mentioned above [6, 22, 31]. This work is related in terms of the data used, and the need to learn some notion of 'relevance,' though the goal is not typically to address general queries as we do here. We are aware of relatively little work that attempts to combine question-answering with opinion mining, though a few exceptions include [33], which answers certain types of queries on *Amazon* data (e.g. "find 100 books with over 200 5-star ratings"); or [45] which learns to distinguish 'facts' from subjective opinions; or [36], which tries to solve cold-start problems by finding opinion sentences of old products that will be relevant to new ones. Though in none of these cases is the goal to address general queries.

Q/A systems. Many of the above ideas from multi-document summarization, relevance ranking, and topical expert-finding have been adapted to build state-of-the-art automated Q/A systems. First is 'query-focused' summarization [7, 24], which is similar to our task in that phrases must be selected among documents that match some query, though typically the relevance function is not learned from training data as it is here. Next (as mentioned above) is the notion that questions, answers, and documents are heterogeneous, meaning that simple bag-of-words type approaches may be insufficient to compare them [3, 46], so that instead one must decompose questions [15] or model their syntax [32]. Also relevant is the problem of identifying experts [5, 21, 35, 40] or high-quality answers [2], or otherwise identifying instances where similar questions have already been answered elsewhere [13, 19], though these differ from our paradigm in that the goal is to select among answers (or answerers), rather than to address the questions themselves.

Naturally also relevant is the large volume of Q/A work from the information retrieval community (e.g. TREC Q/A[2]); however note first that due to the data involved (in particular, subjective opinions) our approach is quite different from systems that build knowledge bases (e.g. systems like Watson [11]), or generally systems whose task is to retrieve a list of objective facts that conclusively answer a query. Rather, our goal is to use Q/A data as a means of learning a 'useful' relevance function, and as such our experiments mainly focus on state-of-the-art relevance ranking techniques.

Table 1: Notation.

Symbol	Description
$q \in \mathcal{Q}, a \in \mathcal{A}$	query and query set, answer and answer set
$y \in \mathcal{Y}$	label set (for binary questions)
$r \in \mathcal{R}$	review and review set
s	relevance/scoring function
v	prediction/voting function
δ	indicator function (1 iff the argument is true)
θ, ϑ, A, B	terms in the bilinear relevance function
ϑ', X, Y	terms in the bilinear prediction function
$p(r\|q)$	relevance of a review r to a query q
$p(y\|r, q)$	probability of selecting a positive answer to a query q given a review r
$p(a > \bar{a}\|r)$	preference of answer a over $\bar{a}$

2.1 Key differences

Though related to the above areas, our work is novel in a variety of ways. Our work is among the first at the interface of Q/A and opinion mining, and is novel in terms of the combination of data used, and in terms of scale. In contrast to the above work on summarization and relevance ranking, given a large volume of answered queries and a corpus of weakly relevant documents (i.e., reviews of the product being queried), our goal is to be as agnostic as possible to the definition of "what makes an opinion relevant to a query?," and to learn this notion automatically from data. This also differentiates our work from traditional Q/A systems as our goal is not to answer queries directly (i.e., to output 'facts' or factoids), but rather to learn a relevance function that will help users effectively navigate multiple subjective viewpoints and personal experiences. Critically, the availability of a large training corpus allows us to learn complex mappings between questions, reviews, and answers, while accounting for the heterogeneity between them.

3. MODEL PRELIMINARIES

Since our fundamental goal is to learn relevance functions so as to surface useful opinions in response to queries, we mainly build upon and compare to existing techniques for relevance ranking.

[2] http://trec.nist.gov/tracks.html

We also briefly describe the mixture-of-experts framework (upon which we build our model) before we describe *Moqa* in Section 4.

3.1 Standard measures for relevance ranking

We first describe a few standard measures for relevance ranking, given a query q and a document d (in our case, a question and a review), whose relevance to the query we want to determine.

Cosine similarity is a simple similarity measure that operates on Bag-of-Words representations of a document and a query. Here the similarity is given by

$$cos(q,d) = \frac{q \cdot d}{\|q\|\|d\|}, \tag{1}$$

i.e., the cosine of the angle between (the bag-of-words representations of) the query q and a document d. This can be further refined by weighting the individual dimensions, i.e.,

$$cos_\vartheta(q,d) = \frac{(q \odot d) \cdot \theta}{\|q\|\|d\|}, \tag{2}$$

where $(q \odot d)$ is the Hadamard product.

Okapi BM25 is state-of-the-art among 'TF-IDF-like' ranking functions and is regularly used for document retrieval tasks [20, 27]. TF-IDF-based ranking measures address a fundamental issue with measures like the cosine similarity (above) whereby common—but irrelevant—words can dominate the ranking function. This can be addressed by defining a ranking function that rewards words which appear many times in a selected document (high TF), but which are rare among other documents (high IDF). Okapi BM25 is a parameterized family of functions based on this idea:

$$bm25(q,d) = \sum_{i=1}^{n} \frac{\text{IDF}(q_i) \cdot f(q_i,d) \cdot (k_1+1)}{f(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}. \tag{3}$$

Again q and d are the query and a document, and f and IDF are the term frequency (of a word q_i in the query) and inverse document frequency as described above. 'avgdl' is the average document length, and b and k_1 are tunable parameters, which we set as described in [27]. See [20, 27] for further detail.

Essentially, we treat BM25 as a state-of-the-art 'off-the-shelf' document ranking measure that we can use for evaluation and benchmarking, and also as a feature for ranking in our own model.

Bilinear models. While TF-IDF-like measures help to discover rare but important words, an issue that still remains is that of *synonyms*, i.e., different words being used to refer to the same concept, and therefore being ignored by the similarity measure in question. This is especially an issue in our setting, where questions and reviews are only tangentially related and may draw from very different vocabularies [3, 46]—thus one needs to learn that a word used in (say) a question about whether a baby seat fits in overhead luggage is 'related to' a review that describes its dimensions.

Bilinear models [8, 12, 42] can help to address this issue by learning complex mappings between words in one corpus and words in another (or more generally between arbitrary feature spaces). Here compatibility between a query and a document is given by

$$qMd^T = \sum_{i,j} M_{ij} q_i d_j, \tag{4}$$

where M is a matrix whose entry M_{ij} encodes the relationship between a term q_i in the query and a term d_j in the document (setting $M = I$ on normalized vectors recovers the cosine similarity). This is a highly flexible model, which even allows that the dimensions of the two feature spaces be different; in practice, since M is very high-dimensional (in our application, the size of the vocabulary squared), we assume that it is low-rank, i.e., that it can be approximated by $M \sim AB^T$ where A and B are each rank K.[3] Thus our similarity measure becomes

$$qAB^Td^T = (qA) \cdot (dB). \tag{5}$$

This has an intuitive explanation, which is that A and B project terms from the query and the document into a low-dimensional space such that 'similar' terms (such as synonyms) in the query and the document are projected nearby (and have a high inner product).

3.2 Mixtures of Experts

Mixtures of experts (MoEs) are a classical way to combine the outputs of several classifiers (or 'weak learners') by associating weighted confidence scores with each classifier [18]. In our setting 'experts' shall be individual reviews, each of which lends support for or against a particular response to a query. The value of such a model is that relevance and classification parameters are learned *simultaneously*, which allows individual learners to focus on classifying only those instances where they are considered 'relevant,' without penalizing them for misclassification elsewhere. In the next section we show how this is useful in our setting, where only a tiny subset of reviews may be helpful in addressing a particular query.

Generally speaking, for a binary classification task, each expert outputs a probability associated with a positive label. The final classification output is then given by aggregating the predictions of the experts, in proportion to their confidence (or expertise). This can be expressed probabilistically as

$$p(y|X) = \sum_f \overbrace{p(f|X)}^{\text{confidence in } f\text{'s ability to classify } X} \underbrace{p(y|f,X)}_{f\text{'s prediction}}. \tag{6}$$

Here our confidence in each expert, $p(f|X)$, is treated as a probability, which can be obtained from an arbitrary real-valued score $s(f,X)$ using a softmax function:

$$p(f|X) = \frac{\exp(s(f,X))}{\sum_{f'} \exp(s(f',X))}. \tag{7}$$

Similarly for binary classification tasks the prediction of a particular expert can be obtained using a logistic function:

$$p(y|f,X) = \sigma(v(f,X)) = \frac{1}{1+e^{-v(f,X)}}. \tag{8}$$

Here s and v are our 'relevance' and 'voting' functions respectively. To define an MoE model, we must now define (parameterized) functions $s(f,X)$ and $v(f,X)$, and tune their parameters to maximize the likelihood of the available training labels. We next describe how this formulation can be applied to queries and reviews, and describe our parameter learning strategy in Section 4.2.

4. MOQA

We now present our model, *Mixtures of Opinions for Question Answering*, or *Moqa* for short. In the previous section we outlined the 'Mixture of Experts' framework, which combines weak learners by aggregating their outputs with weighted confidence scores. Here, we show that such a model can be adapted to simultaneously identify relevant reviews, and combine them to answer complex queries, by treating reviews as experts that either support or oppose a particular response.

[3] This is similar to the idea proposed by Factorization Machines [38], allowing complex pairwise interactions to be handled by assuming that they have low-rank structure (i.e., they factorize).

4.1 Mixtures of Experts for review relevance ranking

As described in Section 3.2, our MoE model is defined in terms of two parameterized functions: s, which determines whether a review ('expert') is relevant to the query, and v, which given the query *and* a review makes a prediction (or vote). Our goal is that predictions are correct exactly for those reviews considered to be relevant. We first define our relevance function s before defining our prediction functions for binary queries in Section 4.2 and arbitrary queries in Section 4.3.

Our scoring function $s(r, q)$ defines the relevance of a review r to a query q. In principle we could make use of any of the relevance measures from Section 3.1 'as is,' but we want our scoring function to be *parameterized* so that we can learn from training data what constitutes a 'relevant' review. Thus we define a parameterized scoring function as follows:

$$s_\Theta(r, q) = \underbrace{\phi(r, q) \cdot \theta}_{\text{pairwise similarity}} + \underbrace{\psi(q) M \psi(r)^T}_{\text{bilinear model}}. \quad (9)$$

Here $\phi(r, q)$ is a feature vector that is made up of existing pairwise similarity measures. θ then weights these measures so as to determine how they should be combined in order to achieve the best ranking. Thus $\phi(r, q)$ allows us to straightforwardly make use of existing 'off-the-shelf' similarity measures that are considered to be state-of-the-art. In our case we make use of BM25+ [26] and ROUGE-L [44] (longest common subsequence) features, though we describe our experimental setup in more detail in Section 5.

The second expression in (eq. 9) is a bilinear scoring function between features of the query ($\psi(q)$) and the review ($\psi(r)$). As features we us a simple bag-of-words representation of the two expressions with an $F = 5000$ word vocabulary. As we suggested previously, learning an $F \times F$ dimensional parameter M is not tractable, so we approximate it by

$$M = \underbrace{(\psi(q) \odot \psi(r)) \cdot \vartheta}_{\text{diagonal term}} + \underbrace{\psi(q) A B^T \psi(r)^T}_{\text{low-rank term}}. \quad (10)$$

ϑ (the diagonal component of M) then accounts for simple term-to-term similarity, whereas A and B (the low-rank component of M) are projections that map $\psi(q)$ and $\psi(r)$ (respectively) into K-dimensional space ($K = 5$ in our experiments) in order to account for linguistic differences (such as synonym use) between the two sources of text. Thus rather than fitting $F \times F$ parameters we need to fit only $(2K + 1) \cdot F$ parameters in order to approximate M.

To obtain the final relevance function, we optimize all parameters $\Theta = \{\theta, \vartheta, A, B\}$ using supervised learning, as described in the following section.

4.2 Binary (i.e., yes/no) questions

Dealing with binary (yes/no) questions is a relatively straightforward application of an MoE-type model, where each of the 'experts' (i.e., reviews) must make a binary prediction as to whether the query is supported by the content of the review. This we also achieve using a bilinear scoring function:

$$v_{\Theta'}(q, r) = (\psi(q) \odot \psi(r)) \cdot \vartheta' + \psi(q) X Y^T \psi(r)^T. \quad (11)$$

Note that this is different from the relevance function s in (eq. 9) (though it has a similar form). The role of (eq. 11) above is to vote on a binary outcome; how much weight/relevance is given to this vote is determined by (eq. 9). Positive/negative $v(q, r)$ corresponds to a vote in favor of a positive or negative answer (respectively).

Learning. Given a training set of questions with labeled yes/no answers (to be described in Section 5.2), our goal is to optimize the relevance parameters $\Theta = \{\theta, \vartheta, A, B\}$ and the prediction parameters $\Theta' = \{\vartheta', X, Y\}$ simultaneously so as to maximize the likelihood that the training answers will be given the correct labels. In other words, we want to define these functions such that reviews given high relevance scores are precisely those that help to predict the correct answer. Using the expression in (eq. 6), the likelihood function is given by

$$L_{\Theta,\Theta'}(\mathcal{Y}|\mathcal{Q}, \mathcal{R}) = \prod_{q \in \mathcal{Q}_{\text{yes}}^{(\text{train})}} p_{\Theta,\Theta'}(y|q) \prod_{q \in \mathcal{Q}_{\text{no}}^{(\text{train})}} (1 - p_{\Theta,\Theta'}(y|q)), \quad (12)$$

where $\mathcal{Q}_{\text{yes}}^{(\text{train})}$ and $\mathcal{Q}_{\text{no}}^{(\text{train})}$ are training sets of questions with positive and negative answers, and $\mathcal{Y}$ and $\mathcal{R}$ are the label set and reviews respectively. $p(y|q)$ (the probability of selecting the answer 'yes' given the query q) is given by

$$p_{\Theta,\Theta'}(y|q) = \sum_{r \in \mathcal{R}_{i(q)}} \Bigg\{ \underbrace{\frac{e^{s_\Theta(q,r)}}{\sum_{r' \in \mathcal{R}_{i(q)}} e^{s_\Theta(q,r')}}}_{\text{relevance}} \underbrace{\frac{1}{1 + e^{-v_{\Theta'}(q,r)}}}_{\text{prediction}} \Bigg\}, \quad (13)$$

where $\mathcal{R}_{i(q)}$ is the set of reviews associated with the item referred to in the query q. We optimize the (log) likelihood of the parameters in (eq. 12) using L-BFGS, a quasi-Newton method for non-linear optimization of problems with many variables. We added a simple ℓ_2 regularizer to the model parameters, though did not run into issues of overfitting, as the number of parameters is far smaller than the number of samples available for training.

4.3 Open-ended questions

While binary queries already account for a substantial fraction of our dataset, and are a valuable testbed for quantitatively evaluating our method, we wish to extend our method to arbitrary open-ended questions, both to increase its coverage, and to do away with the need for labeled yes/no answers at training time.

Here our goal is to train a method that given a corpus of candidate answers (one of which is the 'true' answer that a responder provided) will assign a higher score to the true answer than to all non-answers. Naturally in a live system one does not have access to such a corpus containing the correct answer, but recall that this is not required: rather, we use answers only at *training* time to learn our relevance function, so that at test time we can surface relevant reviews *without* needing candidate answers to be available.

Specifically, we want to train the model such that the true answer is given a higher rank than all non-answers, i.e., to train a ranking function to maximize the average Area Under the Curve (AUC):

$$AUC^{(\text{train})} = \frac{1}{|\mathcal{Q}^{(\text{train})}|} \sum_{q \in \mathcal{Q}^{(\text{train})}} \frac{1}{|\mathcal{A}|} \sum_{\bar{a} \in \mathcal{A}} \delta(a(q) > \bar{a}), \quad (14)$$

where $a(q)$ is the 'true' answer for the query q ($\mathcal{A}$ is the answer set) and $\delta(a(q) > \bar{a})$ is an indicator counting whether this answer was preferred over a non-answer $\bar{a}$. In other words, the above simply counts the fraction of cases where the true answer was considered better than non-answers.

In practice, the AUC is (approximately) maximized by optimizing a pairwise ranking measure, where the true answer should be given a higher score than a (randomly chosen) non-answer, i.e., instead of optimizing $p_{\Theta,\Theta'}(y|q)$ from (eq. 13) we optimize

$$p(a > \bar{a}|q) \sum_r \overbrace{p(r|q)}^{\text{relevance}} \underbrace{p(a > \bar{a}|r)}_{a \text{ is a better answer than } \bar{a}}.$$

To do so we make use of the same relevance function s and the same scoring function v used in (eq. 11), with two important differences: First, the scoring function takes a candidate answer (rather than the query) as a parameter (i.e., $v(a, r)$ rather than $v(q, r)$). This is because our goal is no longer to estimate a binary response to the query q, but rather to determine whether the answer a is supported by the review r. Second, since we want to use this function to rank answers, we no longer care that $v(a, r)$ is maximized, but rather that $v(a, r)$ (for the *true* answer) is higher than $v(\bar{a}, r)$ for non-answers $\bar{a}$. This can be approximated by optimizing the logistic loss

$$p(a > \bar{a}|r) = \sigma(v(a, r) - v(\bar{a}, r)) = \frac{1}{1 + e^{v(\bar{a},r)-v(a,r)}}. \quad (15)$$

This will approximate the AUC if enough random non-answers are selected; optimizing pairwise ranking losses as a means of optimizing the AUC is standard practice in recommender systems that make use of implicit feedback [39]. Otherwise, training proceeds as before, with the two differences being that (1) $p(a > \bar{a}|r)$ replaces the prediction function in (eq. 13), and (2) multiple non-answers must be sampled for training. In practice we use 10 epochs (i.e., we generate 10 random non-answers per query during each training iteration). On our largest dataset (*electronics*), training requires around 4-6 hours on a standard desktop machine.

5. EXPERIMENTS

We evaluate *Moqa* in terms of three aspects: First for binary queries, we evaluate its ability to resolve them. Second, for open-ended queries, its ability to select the correct answer among alternatives. Finally we evaluate *Moqa* qualitatively, in terms of its ability to identify reviews that humans consider to be relevant to their query. We evaluate this on a large dataset of reviews and queries from *Amazon*, as described below.

5.1 Data

We collected review and Q/A data from *Amazon.com*. We started with a previous crawl from [29], which contains a snapshot of product reviews up to July 2014 (but which includes only review data). For each product in that dataset, we then collected all questions on its Q/A page, and the top-voted answer chosen by users. We also crawled descriptions of all products, in order to evaluate how description text compares to text from reviews. This results in a dataset of 1.4 million questions (and answers) on 191 thousand products, about which we have over 13 million customer reviews. We train separate models for each top-level category (electronics, automotive, etc.). Statistics for the 8 largest categories (on which we report results) are shown in Table 2.

Table 2: Dataset Statistics.

Dataset	questions (w/ answers)	products	reviews
electronics	314,263	39,371	4,314,858
home and kitchen	184,439	24,501	2,012,777
sports and outdoors	146,891	19,332	1,013,196
tools and home impr.	101,088	13,397	752,947
automotive	89,923	12,536	395,872
cell phones	85,865	10,407	1,534,094
health and personal care	80,496	10,860	1,162,587
patio lawn and garden	59,595	7,986	451,473
total	1,447,173	191,185	13,498,681

5.2 Labeling yes/no answers

Although the above data is already sufficient for addressing open-ended questions, for binary questions we must first obtain additional labels for training. Here we need to identify whether each question in our dataset is a yes/no question, and if so, whether it has a yes/no answer. In spite of this need for additional labels, addressing yes/no questions is valuable as it gives us a simple and objective way to evaluate our system.

We began by manually labeling one thousand questions to identify those which were binary, and those which had binary answers (note that these are not equivalent concepts, as some yes/no questions may be answered ambiguously). We found that 56.1% of questions are binary, and that 76.5% of these had conclusive binary answers. Of those questions with yes/no answers, slightly over half (62.4%) had positive (i.e., 'yes') answers.

Note that the purpose of this small, manually labeled sample is not to train *Moqa* but rather to evaluate simple techniques for automatically labeling yes/no questions and answers. This is much easier than our overall task, since we are *given* the answer and simply want to determine whether it was positive or negative, for which simple NLP techniques suffice.

To identify whether a question is binary, a recent approach developed by *Google* proved to be effective [16]. This approach consists of a series of complex grammatical rules which are used to form regular expressions, which essentially identify occurrences of 'be', modal, and auxiliary verbs. Among our labeled data these rules identified yes/no questions with 97% precision at 82% recall. Note that in this setting we are perfectly happy to sacrifice some recall for the sake of precision—what we want is a sufficiently large sample of labeled yes/no questions to train *Moqa*, but we are willing to discard ambiguous cases in order to do so.

Next we want to label *answers* as being yes/no. Ultimately we trained a simple bag-of-unigrams SVM, plus an additional feature based on the first word only (which is often simply 'yes' or 'no'). Again, since we are willing to sacrifice recall for precision, we discarded test instances that were close to the decision hyperplane. By keeping only the 50% of instances about which the classifier was the most confident, we obtained 98% classification accuracy on held-out data.

Finally we consider a question only if *both* of the above tests pass, i.e., the question is identified as being binary *and* the answer is classified as yes/no with high confidence. Ultimately through the above process we obtained 309,419 questions that we were able to label with high confidence, which can be used to train the binary version of *Moqa* in Section 5.4.1.

5.3 Baselines

We compare *Moqa* against the following baselines:

rand ranks and classifies all instances randomly. By definition this has 50% accuracy (on average) for both of the tasks we consider. Recall also that for yes/no questions around 62% are answered affirmatively, roughly reflecting the performance of 'always yes' classification.

Cosine similarity (c). The relevance of a review to a query is determined by their cosine similarity, as in (eq. 1).

Okapi-BM25+ (o). BM25 is a state-of-the-art TF-IDF-based relevance measure that is commonly used in retrieval applications [20, 27]. Here we use a recent extension of BM25 known as BM25+ [26], which includes an additional term ($\delta \sum_{i=1}^{n} \text{IDF}(q_i)$) in the above expression in order to lower-bound the normalization by document length.

ROUGE-L (r). Review relevance is determined by ROUGE metrics, which are commonly used to measure similarity in document summarization tasks [44]. Here we use ROUGE-L (longest common subsequence) scores.

Learning vs. non learning (-L). The above measures (c), (o), and (r) can be applied 'off the shelf,' i.e., without using a training set. We analyze the effect of applying maximum-likelihood training (as in eq. 12) to tune their parameters (c-L, o-L, etc.).

Mdqa is the same as *Moqa*, except that reviews are replaced by product descriptions.

The above baselines are designed to assess (1) the efficacy of existing state-of-the-art 'off-the-shelf' relevance measures for the ranking tasks we consider (c, o, and r); (2) the benefit of using a training set to optimize the relevance and scoring functions (c-L, o-L, etc.); (3) the effectiveness of reviews as a source of data versus other potential knowledge bases (*Mdqa*); and finally (4) the influence of the bilinear term and the performance of *Moqa* itself.

For the baselines above we use a linear scoring function in the predictor ($v_{\Theta'}(q,r) = (\psi(q) \odot \psi(r)) \cdot \vartheta'$), though for *Mdqa* and *Moqa* we also include the bilinear term as in (eq. 11). Recall that our model already includes the cosine similarity, ROUGE score, and BM25+ measures as features, so that comparison between the baseline 'cro-L' (i.e., all of the above measures weighted by maximum likelihood) and *Moqa* essentially assesses the value of using bilinear models for relevance ranking.

For all methods, we split reviews at the level of *sentences*, which we found to be more convenient when surfacing results via an interface, as we do in our qualitative evaluation. We found that this also led to slightly (but consistently) better performance than using complete reviews—while reviews contain more information, sentences are much better targeted to specific product details.

5.4 Quantitative evaluation

5.4.1 *Yes/no questions*

We first evaluate our method in terms of its ability to correctly classify held-out yes/no questions, using the binary groundtruth described above. Here we want to measure the classification accuracy (w.r.t. a query set $\mathcal{Q}$):

$$\text{accuracy}(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \underbrace{\delta(q \in \mathcal{Q}_{\text{yes}})\delta(p(y|q) > \frac{1}{2})}_{\text{true positives}} + \underbrace{\delta(q \in \mathcal{Q}_{\text{no}})\delta(p(y_q) < \frac{1}{2})}_{\text{true negatives}},$$

i.e., the fraction of queries that were given the correct binary label.

We found this to be an incredibly difficult measure to perform well on (for any method), largely due to the fact that some fraction of queries are simply not addressed among the reviews available. Fortunately, since we are training probabilistic classifiers, we can also associate a *confidence* with each classification (i.e., its distance from the decision boundary, $|\frac{1}{2} - p(y|q)|$). Our hope is that a good model will assign high confidence scores to exactly those queries that can be (correctly) addressed. To evaluate algorithms as a function of confidence, we consider the accuracy@k:

$$\text{A@k} = \text{accuracy}\Big(\underbrace{\operatorname*{argmax}_{\mathcal{Q}'\in\mathcal{P}_k(\mathcal{Q})} \sum_{q\in\mathcal{Q}'} |\frac{1}{2} - p(y|q)|}_{k \text{ most confident predictions}}\Big), \qquad (16)$$

Where $\mathcal{P}_k(\mathcal{Q})$ is the set of k-sized subsets of $\mathcal{Q}$.

Table 3: Performance of *Moqa* against baselines in terms of the accuracy@50%; only learning (i.e., -L) baselines are shown as non-learning baselines are not applicable to this task.

	rand	ro-L	cro-L	*Moqa*	red. in error vs. cro-L
electronics	50%	78.9%	79.7%	**82.6%**	3.7%
home and kitchen	50%	70.3%	64.6%	**73.6%**	13.9%
sports and outdoors	50%	71.9%	72.8%	**74.1%**	1.8%
tools and home impr.	50%	70.7%	69.0%	**73.2%**	6.1%
automotive	50%	74.8%	76.6%	**78.4%**	2.3%
cell phones	50%	74.6%	76.3%	**79.4%**	4.1%
health and personal care	50%	61.7%	75.5%	**76.2%**	0.9%
patio lawn and garden	50%	74.6%	75.4%	**76.8%**	1.8%
average	50%	72.2%	73.7%	**76.8%**	4.3%

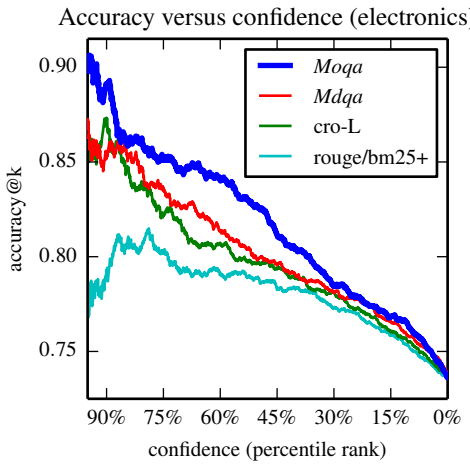

Figure 2: Accuracy as a function of confidence. *Moqa* correctly assigns high confidence to those queries it is able to accurately resolve.

Table 3 shows the performance of *Moqa* and baselines, in terms of the accuracy@50% (i.e., for the 50% of predictions about which each algorithms is most confident). Note that only methods with learning (-L) are shown as non-learning approaches are not applicable here (since there is no good way to determine parameters for a binary decision function in eq. 13 *without* learning). Here *Moqa* is substantially more accurate than alternatives, especially on larger datasets (where more data is available to learn a meaningful bilinear map). Among the baselines ro-L (ROUGE+Okapi BM25+ with learned weights) was the second strongest, with additional similarity-based features (cro-L) helping only slightly.

Figure 2 shows the full spectrum of accuracy as a function of confidence on 'electronics' queries, i.e., it shows how performance degrades as confidence decreases (other categories yielded similar results). Indeed we find that for all methods performance degrades for low-confidence queries. Nevertheless *Moqa* remains more accurate than alternatives across the full confidence spectrum, and for queries about which it is most confident obtains an accuracy of around 90%, far exceeding the performance of any baseline. Figure 2 also shows the performance of *Mdqa*, as we discuss below.

5.4.2 *Open-ended questions*

In Table 4 we show the performance of *Moqa* against baselines for open-ended queries on our largest datasets. Cosine similarity (c) was the strongest non-learning baseline, slightly outperforming the ROUGE score (r) and BM25+ (o, not shown for brevity). Learning improved all baselines, with the strongest being ROUGE and BM25+ combined (ro-L), over which adding weighted cosine similarity did not further improve performance (cro-L), much as we found with binary queries above. *Moqa* was strictly dominant on all datasets, reducing the error over the strongest baseline by 50.6% on average.

5.4.3 *Reviews versus product descriptions*

We also want to evaluate whether review text is a better source of data than other sources, such as product descriptions or specifications. To test this we collected description/specification text for each of the products in our catalogue. From here, we simply interchange reviews with descriptions (recall that both models operate at the level of sentences). We find that while *Moqa* with descriptions (i.e., *Mdqa*) performs well (on par with the strongest baselines), it is still substantially outperformed when we use review text. Here *Moqa* yields a 37.5% reduction in error over *Mdqa* in Table 4; similarly in Figure 2, for binary queries *Mdqa* is on par with the strongest baseline but substantially outperformed by *Moqa* (again other datasets are similar and not shown for brevity).

Partly, reviews perform better because we want to answer subjective queries that depend on personal experiences, for which reviews are simply a more appropriate source of data. But other than that, reviews are simply more abundant—we have on the order of 100 times as many reviews as descriptions (products with active Q/A pages tend to be reasonably popular ones); thus it is partly the sheer volume and diversity of reviews available that makes them effective as a means of answering questions.

We discuss these findings in more detail in Section 6.

5.5 Qualitative evaluation

Finally, we evaluate *Moqa* qualitatively through a user study. Although we have shown *Moqa* to be effective at correctly resolving binary queries, and at maximizing the AUC to select a correct answer among alternatives, what remains to be seen is whether the relevance functions that we learned to do so are aligned with what humans consider to be 'relevant.' Evaluating this aspect is especially important because in a live system our approach would presumably not be used to answer queries directly (which we have shown to be very difficult, and in general still an open problem), but rather to surface relevant reviews that will help the user to evaluate the product themselves.

Here we use the relevance functions $s_\Theta(q, r)$ that we learned in the previous section (i.e., from Table 4) to compare which definition of 'relevance' is best aligned with real users' evaluations—note that the voting function v is not required at this stage.

We performed our evaluation using *Amazon's Mechanical Turk*, using 'master workers' to evaluate 100 queries from each of our five largest datasets, as well as one smaller dataset (*baby*) to assess whether our method still performs well when less data is available for training. Workers were presented with a product's title, image, and a randomly selected query (binary or otherwise). We then presented them the top-ranked result from our method, as well as the top-ranked result using Okapi-BM25+/ROUGE measures (with tuned parameters, i.e., ro-L from Table 4); this represents a state-of-the-art 'off-the-shelf' relevance ranking benchmark, with parameters tuned following best practices; it is also the most competitive baseline from Table 4. Results were shown to evaluators in a random order without labels, from which they had to select whichever they considered to be the most relevant.[4] We also asked workers whether they considered a question to be 'subjective' or not, in order to evaluate whether the subjectivity of the question impacts performance. A screenshot of our interface is shown in Figure 3.

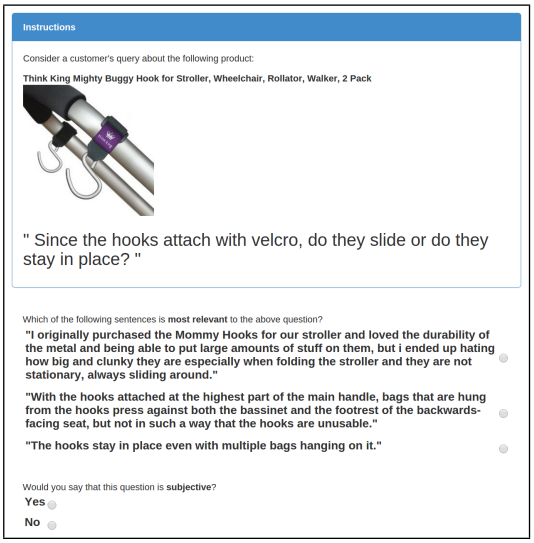

Figure 3: A screenshot of our interface for user evaluation.

Results of this evaluation are shown in Figure 4. On average, *Moqa* was preferred in 73.1% of instances across the six datasets we considered. This is a significant improvement; improvements were similar across datasets (between 66.2% on Sports and Outdoors and 77.6% on Baby), and for both subjective and objective queries (62.9% vs. 74.1%). Ultimately *Moqa* consistently outperforms our strongest baseline in terms of subjective performance, though relative performance seems to be about the same for objective and subjective queries, and across datasets.

5.5.1 *Examples*

Finally, a few examples of the output produced by *Moqa* are shown in Figure 5. Note that none of these examples were available at training time, and only the question (along with the product being queried) are provided as input. These examples demonstrate a few features of *Moqa* and the data in question: First is the wide variety of products, questions, and opinions that are reflected in the data; this linguistic variability demonstrates the need for a model that *learns* the notion of relevance from data. Second, the questions themselves (like the example from Figure 1) are quite different from those that could be answered through knowledge bases; even those that seem objective (e.g. "how long does this stay hot?") are met with a variety of responses representing different (and sometimes contradictory) experiences; thus reviews are the perfect source of data to capture this variety of views. Third is the heterogeneity between queries and opinions; words like "girl" and "tall" are identified as being relevant to "daughter" and "medium," demonstrating the need for a flexible model that is capable of learning complicated semantics in general, and synonyms in particular.

Also note that while our bilinear model has many thousands of parameters, at test time relevance can be computed extremely efficiently, since in (eq. 10) we can project all reviews via B in ad-

[4]We also showed a randomly selected result, and gave users the option to select *no* result. We discarded cases with overlaps.

Table 4: Performance of *Moqa* against baselines (a key is shown at right for baselines from Section 5.3). Reported numbers are average AUC (i.e., the models' ability to assign the highest possible rank to the correct answer); higher is better.

Dataset	rand	c	r	ro-L	cro-L	*Mdqa*	*Moqa*	red. in error vs. cro-L	red. in error vs. *Mdqa*
electronics	0.5	0.633	0.626	0.886	0.855	0.865	**0.912**	65.6%	54.5%
home and kitchen	0.5	0.643	0.635	0.850	0.840	0.863	**0.907**	73.5%	48.1%
sports and outdoors	0.5	0.653	0.645	0.848	0.845	0.860	**0.885**	35.1%	22.5%
tools and home impr.	0.5	0.638	0.632	0.860	0.817	0.834	**0.884**	58.8%	43.7%
automotive	0.5	0.648	0.640	0.810	0.821	0.825	**0.863**	30.4%	27.7%
cell phones	0.5	0.624	0.617	0.768	0.797	0.844	**0.886**	78.7%	37.5%
health and personal care	0.5	0.632	0.625	0.818	0.817	0.842	**0.880**	52.7%	31.9%
patio lawn and garden	0.5	0.634	0.628	0.835	0.833	0.796	**0.848**	10.2%	34.4%
average	0.5	0.638	0.631	0.834	0.828	0.841	**0.883**	50.6%	37.5%

rand	random
c	cosine similarity
r	ROUGE measures
o	Okapi BM25+
-L	ML parameters
Moqa	our method
Mdqa	w/ descriptions

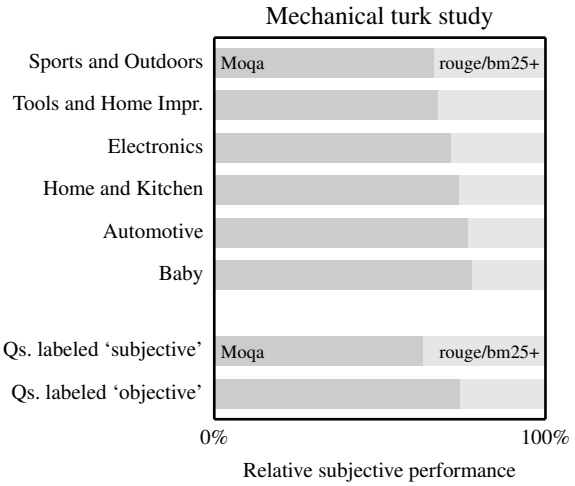

Figure 4: User study. Bars indicate the fraction of times that opinions surfaced by *Moqa* are preferred over those of the strongest baseline (a tuned combination of BM25+ and the ROUGE score, ro-L from Section 5.3).

vance. Thus computing relevance takes only $O(K+|q|+|r|)$ (i.e., the number of projected dimensions plus the number of words in the query and review); in practice this allows us to answer queries in a few milliseconds, even for products with thousands of reviews.

6. DISCUSSION AND FUTURE WORK

Surprisingly, performance for open-ended queries (Table 4) appears to be better than performance for binary queries (Table 3), both compared to random classification and to our strongest baseline, against our intuition that the latter task might be more difficult. There are a few reasons for this: One is simply that the task of differentiating the true answer from a (randomly selected) non-answer is 'easier' than resolving a binary query; this explains why outperforming a random baseline is easier, but does not explain the higher relative improvement against baselines. For the latter, note that the main difference between our method and the strongest baseline is the use of a bilinear model; while a highly flexible model, it has far more parameters than baselines, meaning that a large dataset is required for training. Thus what we are seeing may simply be the benefit of having substantially more data available for training when considering open-ended questions.

Also surprising is that in our user study we obtained roughly equal performance on subjective vs. objective queries. Partly this may be because subjective queries are simply 'more difficult' to address, so that there is less separation between methods, though this would require a larger labeled dataset of subjective vs. objective queries to evaluate quantitatively. In fact, contrary to expectation only around 20% of queries were labeled as being 'subjective' by workers. However the full story seems more complicated—queries such as "how long does this stay hot?" (Figure 5) are certainly labeled as being 'objective' by human evaluators, though the variety of responses shows a more nuanced situation. Really, a large fraction of seemingly objective queries are met with contradictory answers representing different user experiences, which is exactly the class of questions that our method is designed to address.

6.1 Future work

We see several potential ways to extend *Moqa*.

First, while we have made extensive use of reviews, there is a wealth of additional information available on review websites that could potentially be used to address queries. One is rating information, which could improve performance on certain evaluative queries (though to an extent we already capture this information as our model is expressive enough to learn the polarity of sentiment words). Another is user information—the identity of the questioner and the reviewer could be used to learn better relevance models, both in terms of whether their opinions are aligned, or even to identify topical experts, as has been done with previous Q/A systems [2, 5, 21, 35, 40].

In categories like electronics, a large fraction of queries are related to compatibility (e.g. "will this product work with X?"). Addressing compatibility-related queries with user reviews is another promising avenue of future work—again, the massive number of potential product combinations means that large volumes of user reviews are potentially an ideal source of data to address such questions. Although our system can already address such queries to some extent, ideally a model of compatibility-related queries would make use of additional information, for instance reviews of *both* products being queried, or the fact that compatibility relationships tend to be symmetric, or even co-purchasing statistics as in [29].

Finally, since we are dealing with queries that are often subjective, we would like to handle the possibility that they may have multiple and potentially inconsistent answers. Currently we have selected the top-voted answer to each question as an 'authoritative'

Binary model:

Product: Schwinn Searcher Bike (26-Inch, Silver) (amazon.com/dp/B007CKH61C)
Question: "Is this bike a medium? My daughter is 5'8"."

Ranked opinions and votes: "The seat was just a tad tall for my girl so we actually sawed a bit off of the seat pole so that it would sit a little lower." (yes, .698); "The seat height and handlebars are easily adjustable." (yes, .771); "This is a great bike for a tall person." (yes, .711)

Response: Yes (.722)
Actual answer (labeled as 'yes'): My wife is 5'5" and the seat is set pretty low, I think a female 5'8" would fit well with the seat raised.

Product: Davis & Sanford EXPLORERV Vista Explorer 60" Tripod (amazon.com/dp/B000V7AF8E)
Question: "Is this tripod better then the AmazonBasics 60-Inch Lightweight Tripod with Bag one?"

Ranked opinions and votes: "However, if you are looking for a steady tripod, this product is not the product that you are looking for" (no, .295); "If you need a tripod for a camera or camcorder and are on a tight budget, this is the one for you." (yes, .901); "This would probably work as a door stop at a gas station, but for any camera or spotting scope work I'd rather just lean over the hood of my pickup." (no, .463);

Response: Yes (.863)
Actual answer (labeled as 'yes'): The 10 year warranty makes it much better and yes they do honor the warranty. I was sent a replacement when my failed.

Open-ended model:

Product: Mommy's Helper Kid Keeper (amazon.com/dp/B00081L2SU)
Question: "I have a big two year old (30 lbs) who is very active and pretty strong. Will this harness fit him? Will there be any room to grow?"

Ranked opinions: "So if you have big babies, this may not fit very long."; "They fit my boys okay for now, but I was really hoping they would fit around their torso for longer."; "I have a very active almost three year old who is huge."

Actual answer: One of my two year olds is 36lbs and 36in tall. It fits him. I would like for there to be more room to grow, but it should fit for a while.

Product: Thermos 16 Oz Stainless Steel (amazon.com/dp/B00FKPGEBO)
Question: "how many hours does it keep hot and cold ?"

Ranked opinions: "Does keep the coffee very hot for several hours."; "Keeps hot Beverages hot for a long time."; "I bought this to replace an aging one which was nearly identical to it on the outside, but which kept hot liquids hot for over 6 hours."; "Simple, sleek design, keeps the coffee hot for hours, and that's all I need."; "I tested it by placing boiling hot water in it and it did not keep it hot for 10 hrs."; "Overall, I found that it kept the water hot for about 3-4 hrs.";

Actual answer: It doesn't, I returned the one I purchased.

Figure 5: Examples of opinions recommended by *Moqa*. The top two examples are generated by the binary model, the bottom two by the open-ended model. Note that none of these examples were available at training time, and only the question is provided as input (the true answer and its label are shown for comparison). Opinions are shown in decreasing order of relevance. Note in the second example that *all* opinions get to vote in proportion to their relevance; in this case the many positive votes among less-relevant opinions outweigh the negative votes above, ultimately yielding a strong 'yes' vote.

response to be used at training time. But handling multiple, inconsistent answers could be valuable in several ways, for instance to automatically identify whether a question is subjective or contentious, or otherwise to generate relevance rankings that support a spectrum of subjective viewpoints.

7. CONCLUSION

We presented *Moqa*, a system that automatically responds to product-related queries by surfacing relevant consumer opinions. We achieved this by observing that a large corpus of previously-answered questions can be used to *learn* the notion of relevance, in the sense that 'relevant' opinions are those for which an accurate predictor can be trained to select the correct answer as a function of the question and the opinion. We cast this as a mixture-of-experts learning problem, where each opinion corresponds to an 'expert' that gets to vote on the correct response, in proportion to its relevance. These relevance and voting functions are learned automatically and evaluated on a large training corpus of questions, answers, and reviews from *Amazon*.

The main findings of our evaluation were as follows: First, reviews proved particularly effective as a source of data for answering product-related queries, outperforming other sources of text like product specifications; this demonstrates the value of *personal experiences* in addressing users' queries. Second, we demonstrated the need to handle heterogeneity between various text sources (i.e., questions, reviews, and answers); our large corpus of training data allowed us to train a flexible bilinear model that it capable of automatically accounting for linguistic differences between text sources, outperforming hand-crafted word- and phrase-level relevance measures. Finally, we showed that *Moqa* is quantitatively able to address both binary and open-ended questions, and qualitatively that human evaluators prefer our learned notion of 'relevance' over hand-crafted relevance measures.

References

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, 2008.

[2] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: a case study of Stack Overflow. In *KDD*, 2012.

[3] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR*, 2000.

[4] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *World Wide Web*, 2009.

[5] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: the case of Yahoo! Answers. In *KDD*, 2008.

[6] G. Carenini, R. Ng, and A. Pauls. Multi-document summarization of evaluative text. In *ACL*, 2006.

[7] Y. Chali and S. Joty. Selecting sentences for answering complex questions. In *EMNLP*, 2008.

[8] W. Chu and S.-T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *World Wide Web*, 2009.

[9] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. How opinions are received by online communities: A case study on amazon.com helpfulness votes. In *World Wide Web*, 2009.

[10] G. Erkan and D. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 2004.

[11] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. Building Watson: An overview of the DeepQA project. In *AI Magazine*, 2010.

[12] W. Freeman and J. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *CVPR*, 1996.

[13] R. Gangadharaiah and B. Narayanaswamy. Natural language query refinement for problem resolution from crowd-sourced semi-structured data. In *International Joint Conference on Natural Language Processing*, 2013.

[14] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

[15] S. Harabagiu, F. Lacatusu, and A. Hicki. Answering complex questions with random walk models. In *SIGIR*, 2006.

[16] J. He and D. Dai. Summarization of yes/no questions using a feature function model. *JMLR*, 2011.

[17] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[18] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991.

[19] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM*, 2005.

[20] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments. In *Information Processing and Management*, 2000.

[21] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *Conference on Information and Knowledge Management*, 2007.

[22] R. Katragadda and V. Varma. Query-focused summaries or query-biased summaries? In *ACL Short Papers*, 2009.

[23] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.

[24] J. Li and L. Sun. A query-focused multi-document summarizer based on lexical chains. In *NIST*, 2007.

[25] C.-Y. Lin and E. Hovy. From single to multi-document summarization: A prototype system and its evaluation. In *ACL*, 2002.

[26] Y. Lv and C. Zhai. Lower-bounding term frequency normalization. In *CIKM*, 2011.

[27] C. Manning, P. Raghavan, and H. Schültze. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[28] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM Conference on Recommender Systems*, 2013.

[29] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Knowledge Discovery and Data Mining*, 2015.

[30] K. McKeown and D. Radev. Generating summaries of multiple news articles. In *SIGIR*, 1995.

[31] X. Meng and H. Wang. Mining user reviews: From specification to summarization. In *ACL Short Papers*, 2009.

[32] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question answer classification. In *ACL*, 2007.

[33] A. Nazi, S. Thirumuruganathan, V. Hristidis, N. Zhang, and G. Das. Answering complex queries in an online community network. In *ICWSM*, 2015.

[34] A. Pal and S. Counts. Identifying topical authorities in microblogs. In *Web Search and Data Mining*, 2011.

[35] A. Pal, R. Farzan, J. Konstan, and R. Kraut. Early detection of potential experts in question answering communities. In *UMAP*, 2011.

[36] D. H. Park, H. D. Kim, C. Zhai, and L. Guo. Retrieval of relevant opinion sentences for new products. In *SIGIR*, 2015.

[37] J. Ponte and B. Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998.

[38] S. Rendle. Factorization machines. In *ICDM*, 2010.

[39] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[40] F. Riahi, Z. Zolaktaf, M. Shafiei, and E. Milios. Finding expert users in community question answering. In *World Wide Web*, 2012.

[41] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015.

[42] J. Tenenbaum and W. Freeman. Separating style and content with bilinear models. *Neural Computation*, 2000.

[43] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Knowledge Discovery and Data Mining*, 2010.

[44] C. yew Lin. ROUGE: a package for automatic evaluation of summaries. In *ACL Workshop on Text Summarization Branches Out*, 2004.

[45] H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *EMNLP*, 2003.

[46] K. Zhang, W. Wu, H. Wu, Z. Li, and M. Zhou. Question retrieval with high quality answers in community question answering. In *CIKM*, 2014.