Analyzing Reviews for User Expertise

Michael A. Klaczynski
University of Viginia
mak4em@virginia.edu

Akanksha Nichrelay
University of Virginia
an9sx@virginia.edu

Arjun Malhotra
University of Virginia
am2cj@virginia.edu

ABSTRACT

Not all reviews are equally helpful. Certain reviews display a greater degree of expert knowledge about a product than others. For example, a casual camera-owner may comment that their camera sometimes leaves the subject a bit unfocused, whereas a professional photographer might comment on details like IBIS or bitrates. The goal of our project is to separate reviews based on how much knowledge of a product the user displayed when writing the review.

Experimenting on a large set of reviews for a single product (a micro SD card), we used k-means clustering with a number of different types of similarity measures, including Bag-of-Words, TF-IDF, and word2vec - Average word2vec and TF-IDF word2vec. TF-IDF word2vec seemed to be the best measure, as with it the k-means algorithm gave the most well-balanced clusters. Each cluster represented a different level of expertise displayed.

CCS CONCEPTS

• **Information systems** → *Information retrieval*; • **Computer systems organization** → **Embedded systems**; *Redundancy*;

KEYWORDS

Information Retrieval, Natural Language Processing, Machine Learning Classification

ACM Reference Format:
Michael A. Klaczynski, Akanksha Nichrelay, and Arjun Malhotra. 2018. Analyzing Reviews for User Expertise. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Not every online product review is equally helpful to every person. Depending on how much the shopper knows about the product they are looking for, some reviews may be more helpful than others. For example, a professional photographer looking for a new camera would get a lot of use out of a review full of technical details and product comparisons, but someone not familiar with the industry would prefer a simpler review that they understand. Wouldn't it be nice if we could automatically sort these out?

A person's familiarity with a topic is often displayed in the way in which they talk about it. By analyzing their language (and using a bit of machine learning) we can categorize each review based on what level of expertise the review requires to understand. These categories could be used by an online shopper to easily find useful reviews; they could just tell the website how familiar they are with the product at hand, and the website could return a bucket of reviews it thinks the user would understand.

The purpose of this paper was to find ways of identifying these more-intelligent reviews. Using a data set of Amazon reviews, we applied and tested a number of natural language processing and machine learning algorithms with the problem in mind. To the best of our knowledge, we are the first to analyze product reviews in order to determine user expertise level using the unsupervised clustering of reviews.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2 RELATED WORK

Much work has been done in the past with amazon reviews, as there is much data to make use of. Most of it attempts to predict a vague metric of "helpfulness" [3]. Our metric of review intelligence is somewhat less general; since different levels of expertise in reviews are more helpful to different people, we figured it may be useful.

What we are doing is also in the same vein as things like sentiment analysis [6]. Rather than testing for the user's levels of emotions, of course, we are testing for the user's expertise in a field.

Expertise retrieval is an old problem in information retrieval, but research has mostly been focused on helping researchers share knowledge by finding relevant experts [1]. We aim to solve a different but similar problem of helping online shoppers decide on products by finding more informative reviews.

Multiple groups over the years have attempted different ways of taking into account a user's knowledge. Works have focused on ranking documents based on reading level [2] as well as on interpreting a user's topic familiarity based on their queries [4].

3 METHODOLOGY

3.1 Approach

To predict user expertise from reviews, we started with one product. In our data, each user had only one review for the product. From our data set, we focused only on analyzing the product review text column. Other columns were ignored. Since each user had one review each, our problem was now equivalent to finding the knowledge level or the technicality of a review. We converted each review into a vector and then clustered the using Unsupervised Learning Algorithm, K-means. We then use the best model obtained to predict the clusters on unseen data to test our model. Human judgment is used at certain points as will be seen in the experiment and evaluation sections due to lack of proper annotated data and expected outcomes.

3.2 Data

For out data set we picked the 5-core electronics data from Amazon product reviews, which has about 1.6 million reviews [5]. Since we

wanted to work with only one product, we selected the product with maximum available reviews - a collection of about 5000 Amazon reviews on a MicroSD card, "*SanDisk Ultra 64GB MicroSDXC Class 10 UHS Memory Card Speed Up To 30MB/s With Adapter*".

For evaluation purpose, we referred a tech blog which was a microSD card buying guide [7]. We manually collected technical words and phrases that could denote user expertise in the area. We will use these words in building and evaluating our model.

3.3 Clustering

We decided to apply unsupervised learning to cluster the reviews as we did not have any labels to our data. We applied the state of the art K-means clustering. We tried out clusters ranging from 2 to 9. Our main aim was to be able to get similar reviews grouped together so that we can infer the technical expertise of a cluster. We used the elbow curve method to find the best number of clusters produced using the inertia loss against the number of clusters.

3.3.1 Representation of Reviews. To cluster the reviews, we applied different similarity measures to our review text. We tried out four approaches:

(1) Bag of Words
(2) TF-IDF
(3) Average Word2Vec
(4) TF-IDF Word2Vec

In order to do that, we first cleaned our review text using NLTK libraries. We tokenized the reviews and removed the stop words which were taken from the NLTK library. We then removed punctuation marks , and tokens with length less than 2 characters, and also converted the tokens to lower case and applied snowball stemmer. We then applied the same text processing to the set of our technical phrases collected from the tech blog. We converted phrases like "Video Speed Class" and "data transfer rates" to unigrams to simplify our experiment. After processing, there were total 79 technical words obtained. These were stored in a separate set for evaluation.

There were some terms like "GB", "KB", "UHS" that were allowed to bypass the above text processing since these were of value to our process. We simply searched for these words and for cases like "16gb" or misspelled words like "32kbThis", we simple replaced the word with the corresponding term that we were looking for. The above terms were replaced with "gb" and "kb" respectively.

We will call these tokenized technical terms, from the tech blog as the relevant words to our experiment. We counted the number of the relevant words occurring in each review and stored these counts as a feature in our dataframe of sandisk reviews. We will be using these counts later in our evaluation. After obtaining the processed tokens from our user reviews, we joined these tokens to form cleaned reviews and store these sentences in our dataframe to be used later for clustering.

4 EXPERIMENT

The clusters of our reviews were created using various similarity measures. In this section we will describe method and results of each approach. Each of the below similarity measure returned a matrix of vectors to represent each review. To this matrix, we append a feature like 'Total relevant words count' or 'Cumulative TF-IDF of relevant words' per review in order to give some weight to the relevant reviews. We then applied standard scalar to the document matrix along with K-means clustering. We obtain the right number of clusters and then assess the knowledge expertise of a cluster by using two different evaluation methods. We then use these clusters to make predictions on unseen reviews that we manually created to test if the cluster can classify these reviews correctly.

4.1 Bag of Words

To get the Bag of words representation of the reviews, we employed CountVectorizer() class from scikit learn. It uses utf-8, encoding technique. On running fit_transform() on the processed reviews, we got the vector representations of each review. This gave us a sparse matrix, with 4895 columns, the columns represent the total number of unique words in all the reviews. Unfortunately, BoW did not yield clearly distributed clusters. Most of the data points are concentrated in one of the clusters.

```
In [31]:  1 predictions_bow = predictions;
          2 cluster_distribution(predictions_bow)

cluster k=2 :
[4912    3]

cluster k=3 :
[   7    1 4907]

cluster k=4 :
[   1    1    1 4912]

cluster k=5 :
[   1 4911    1    1    1]

cluster k=6 :
[   1    2    2    5    1 4904]

cluster k=7 :
[   1    1 4909    1    1    1    1]

cluster k=8 :
[   3    1    1    1    1    1 4906    1]

cluster k=9 :
[   1    3    1    1    1 4905    1    1    1]
```

Figure 1: Bag-Of-Words clusters with different values of k

4.2 TF-IDF

Next, we used TfidfVectorizer() class from scikit learn to create TF-IDF vector representation of the reviews. Similar to Bag of Words, running fit_transform on the processed reviews, gave us the TF-IDF vector representation of the reviews.

As for the clusters, TF-IDF didn't yield any better clusters than Bag of Words, either. Similar to BoW, K-Means on TF-IDF does not generate well separated clusters. Most of the reviews are concentrated in a single cluster.

4.3 Average Word2Vec

To create average word2vec vector representation of each review text. We first had to get the word2vec representation of every word. To get the word2vec representation of every word we used the gensim library. While creating word2vec representations of the words we used whole 1.7 million reviews of electronics product

```
1 cluster_distribution(predictions_avg_word2vec)
```

```
cluster k=2 :
[2153 2762]

cluster k=3 :
[1397 1772 1746]

cluster k=4 :
[1003 1243 1482 1187]

cluster k=5 :
[1001 1349  973  703  889]

cluster k=6 :
[1213  891  614  890  494  813]

cluster k=7 :
[ 545  465  665 1103  852  681  604]

cluster k=8 :
[639 774 579 660 498 636 423 706]

cluster k=9 :
[411 615 480 654 536 569 415 488 747]
```

Figure 2: Average Word2Vec clusters with different values of k

```
1 cluster_distribution(predictions_tfidf_word2vec)
```

```
cluster k=2 :
[3293 1622]

cluster k=3 :
[1874 1819 1222]

cluster k=4 :
[ 856 1552 1424 1083]

cluster k=5 :
[1318  778  948 1430  441]

cluster k=6 :
[1020  434  844 1287  603  727]

cluster k=7 :
[786 605 565 853 769 425 912]

cluster k=8 :
[817 364 550 521 766 535 720 642]

cluster k=9 :
[544 505 498 742 494 345 618 635 534]
```

Figure 3: TF-IDF word2vec clusters with different values of k

which was about 1.9 GB. This helped our model get acquainted with all the electronics words in the electronics corpus. Only the words with frequency greater than 3 were considered for creating the word2vec. Each word was converted into a 100 dimensional vector. We stored this trained word2vec in a pickle file in order to save hours in retraining our model, with average word2vec and TF-IDF word2vec.

To, get average word2vec representation of individual reviews, we used the equation 1. For each review of the microSD card :-

$$Word2Vec(r) = 1/n[w2v(w1) + w2v(w2) + ...w2v(n)] \quad (1)$$

Where r is the review and w2v(w1),w2v(w2),...,w2v(n) are the word2vec representation of words present in the review. To, get average word2vec representation of individual reviews, we used the equation 1.

$$Word2Vec(r) = 1/n[w2v(w1) + w2v(w2) + ...w2v(n)] \quad (2)$$

4.4 TF-IDF Word2Vec

Using our stored word2vec, we used TfidfVectorizer() function of scikit learn to create our TF-IDF word2vec representations of our reviews. We also stored the cumulative TF-IDF of relevant words per review as a feature in our dataframe and also add this into our TF-IDF word2vec matrix in order to give weights to the reviews containing more relevant words. Where relevancy is defined as the common words between the expert file and the individual reviews.

$$TFIDF{-}w2v(r) = \sum(tfidf_i * w2v(w_i))/\sum tfidf_i \quad (3)$$

Where r is the review and tfidfi are the word2vec representation of words present in the review. To, get TF-IDF word2vec representation of individual reviews, we used the equation 2.

After scaling the data and clustering, we get pretty uniform clusters as can be seen in the figure.

5 EVALUATION

Since we get the most uniform clusters from both Average and TF-IDF word2vecs, we will focus on these two models for the evaluation of the clusters obtained. It is not straightforward to evaluate our clusters since we do not have any labels or annotated data. Thus, we use below two approaches to measure the correctness of clusters.

- We check average count of relevant terms for each review in a cluster.
- We calculate the average TF-IDF of all the relevant words for each review in a cluster.

We will cover the above two in more detail in the following sections. Human judgment was used to interpret the final results of these evaluations.

5.1 Obtaining the clusters

The idea is to find the best number of clusters in the range of 2 to 9 by using elbow curve and then evaluate the correctness of these clusters using the two evaluation methods. The elbow curve measures inertia loss against the number of clusters. We used PCA to visualize the resulting clusters to get an idea of how our clusters look like.

5.1.1 Average word2vec clusters. For average word2vec model, the inertia loss versus the number of cluster plot gives us an elbow at k=4 and k=6, see figure 4. Since the elbow is slightly sharper at k=4, we select this as the best number of clusters obtained.

5.1.2 TF-IDF word2vec clusters. For TF-IDF word2vec model, the inertia loss versus the number of cluster plot gives us an elbow at k=3. The elbow obtained is much sharper than that obtained for average word2vec. We select this as the best number of clusters obtained.

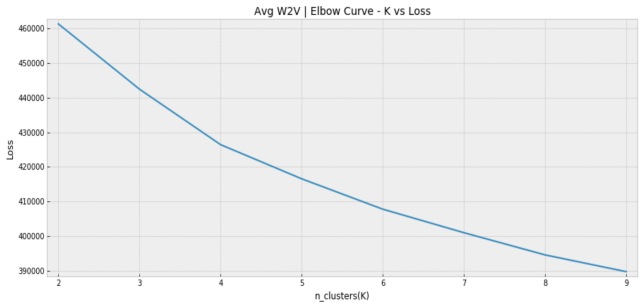

Figure 4: Elbow curve for Average word2vec clusters using inertia loss versus different values of k . Elbow obtained at k=4.

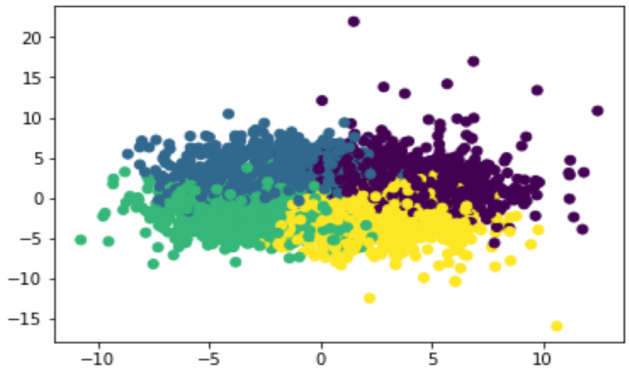

Figure 5: Visualizing resulting clusters with k=4 for average word2vec using PCA.

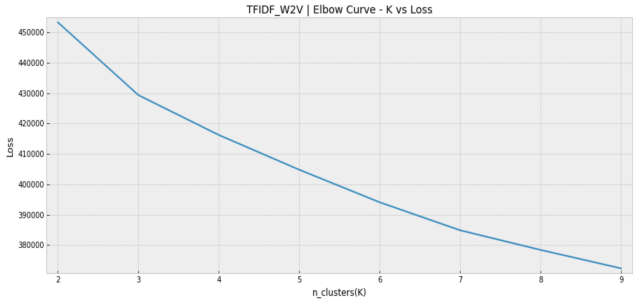

Figure 6: Elbow curve for TF-IDF word2vec clusters using inertia loss versus different values of k . Elbow obtained at k=3.

5.2 Evaluating with Average count

The most basic evaluation criteria we could think of was to check which review cluster has the most relevant words on average. So, we collected the sum of the relevant words in all reviews of a cluster and then averaged that by the total reviews in the cluster.

$$\sum\{rel \in \mathbb{C}\}/\sum r \in \mathbb{C}$$

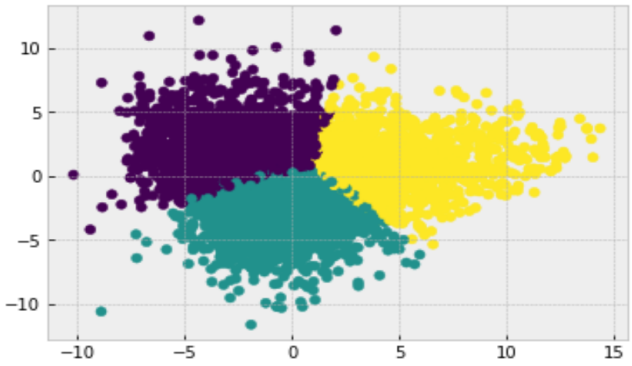

Figure 7: Visualizing resulting clusters with k=3 for TF-IDF word2vec using PCA.

where *rel* are all the relevant words in all reviews belonging to a cluster "C" and *r* r are all the reviews belonging to a cluster "C"

On applying this approach to average word2vec clusters, we get the result shown in figure.

```
Results using avg relevant word count for Avg W2V:
Total relevant word count in reviews in cluster  0 :   2171
Avg relevant word count in reviews in cluster  0 :   2.164506480558325
Total relevant word count in reviews in cluster  1 :   3189
Avg relevant word count in reviews in cluster  1 :   2.5655671761866454
Total relevant word count in reviews in cluster  2 :   10965
Avg relevant word count in reviews in cluster  2 :   7.398785425101215
Total relevant word count in reviews in cluster  3 :   4146
Avg relevant word count in reviews in cluster  3 :   3.4928390901432183
```

Figure 8: Results of checking average relevant word count of reviews in each cluster for average word2vec model where k=4. The results show that the cluster 2 has the most intelligent reviews while cluster 0 has the least intelligent reviews.

The results show that the cluster 2 has the most intelligent reviews with average relevant word count of 7.39 while cluster 0 has the least intelligent reviews with an average relevant word count of 2.16. To get a better idea of what words the clusters consists of we plot the word cloud for these clusters.

As we can see from the word clouds of cluster 2 focuses more on technical terms like 'gb', 'memory', 'speed', 'format' etc. While cluster 0 focuses more on terms like 'great', 'price', 'good' which are not that relevant to user expertise. However, on taking a closer look at the reviews in each cluster, we see that there are many incorrectly classified reviews.

For instance, the most intelligent review cluster i.e. the cluster 2 has some of the least technical reviews as well, as shown. Thus, we concluded that average word2vec is not the best choice for our problem.

Following the same steps for TF-IDF word2vec model, we get the below results for average relevant word count for the clusters.

The results show that the cluster 2 has the most intelligent reviews with average relevant word count of 8.85 while cluster 0 has the least intelligent reviews with an average relevant word count

Figure 9: Visualizing resulting clusters with k=4 for average word2vec using word cloud.

```
#BEST CLUSTER
df_sandisk.loc[df_sandisk['labels_avg_word2vec'] == 2,'CleanedText']
```

```
614    issu

616    work expect sprung higher capac think made bit cheesier earlier paint look clean
```

Figure 10: Misclassified reviews in cluster 2 for average word2vec model.

```
Results using avg relevant word count for TF-IDF W2V:
Total relevant word count in reviews in cluster  0 :   4513
Avg relevant word count in reviews in cluster  0 :   2.4082177161152614
Total relevant word count in reviews in cluster  1 :   5134
Avg relevant word count in reviews in cluster  1 :   2.822429906542056
Total relevant word count in reviews in cluster  2 :   10824
Avg relevant word count in reviews in cluster  2 :   8.857610474631752
```

Figure 11: Results of checking average relevant word count of reviews in each cluster for TF-IDF word2vec model where k=3. The results show that the cluster 2 has the most intelligent reviews while cluster 0 has the least intelligent reviews.

of 2.40. To get a better idea of what words the clusters consists of we plot the word cloud for these clusters.

Figure 12: Visualizing resulting clusters with k=3 for TF-IDF word2vec using word cloud.

The word cloud in this case do not offer much analysis as many of the common terms occur in all 3 clusters. This suggests that we can actually omit these words and add to our list of stop words. However, on taking a closer look at the reviews in each cluster, we see that the reviews are correctly classified. We move on to our next evaluation metric to get better insights and the accuracy of our interpretation.

5.3 Evaluating with cumulative TF-IDF

To get better results, instead of using the simple average count of relevant words in a cluster, we collect the cumulative sum of TF-IDF of the relevant words in all reviews of a cluster and then averaged that by the total reviews in the cluster. The term frequency is obtained from the microSD card product reviews while the Inverse Document Frequency is obtained from the whole corpus of Electronics product reviews. This ensures that the IDF gets assessed correctly and eliminates chances of error in giving weights to our reviews. So for each relevant technical word in a review, we calculate the TF-IDF weight and then add all these TF-IDF weights for a review. This gives us the cumulative TF-IDF weight of relevant words in a review. This is then averaged across all reviews in a cluster as shown below.

$$\sum\{(tfidf-rel) \in \mathbb{C}\}/\sum r \in \mathbb{C}$$

Where tfidf-rel is the TF-IDF of the relevant words in all the reviews of a cluster and r are the reviews in the cluster.

On applying this approach to TF-IDF word2vec clusters, we get the result shown in figure.

```
Cummulative TF-IDF of reviews in cluster  0 :   1304.4987290736003
Avg Cummulative TF-IDF of reviews in cluster  0 :   0.6961039109250802
Cummulative TF-IDF of reviews in cluster  1 :   1548.7673966783148
Avg Cummulative TF-IDF of reviews in cluster  1 :   0.8514389206587767
Cummulative TF-IDF of reviews in cluster  2 :   3368.9542841171065
Avg Cummulative TF-IDF of reviews in cluster  2 :   2.7569183994411675
```

Figure 13: Results of checking average cumulative TF-IDF relevant word count of reviews in each cluster for TF-IDF word2vec model where k=3. The results show that the cluster 2 has the most intelligent reviews while cluster 0 has the least intelligent reviews.

The results show that the cluster 2 has the most intelligent reviews with average cumulative TF-IDF relevant word count of 2.75 while cluster 0 has the least intelligent reviews with an average cumulative TF-IDF relevant word count of 0.69. It is also important to note that the average cumulative TF-IDF relevant word count of cluster 1 is much closer to cluster 0 and thus we can conclude that cluster 0 and cluster 1 are not so knowledgeable while cluster 2 contains the most informative reviews. This result is consistent with our findings using the average relevant word count evaluation metric. Thus, we select this TF-IDF word2vec cluster model with k=3 as our final model to predict the intelligence level of a review and thus of the reviewer.

6 RESULTS

To test our selected TF-IDF word2vec cluster model with k=3, we predict the cluster on new set of reviews. These new reviews are generated manually and are not present in the training data of

microSD card reviews or even in the tech blog guide. Our aim to assess whether our model works correctly in classifying certain reviews to the appropriate knowledge levels. Since we do not have any labels, neither we can measure accuracy of the model on the whole nor we have existing baseline to compare with. So we rely on human judgment on test out the system. The idea is given a review, the review should be assigned the right cluster depending upon how knowledgeable or not knowledgeable the review is. If the review appears to a human judge as knowledgeable, our model should add the review to cluster 2, as per our observations above. While if the review does not appear to be much informative, the model should classify it into either cluster 0 or cluster 1.

Test Review	Predicted Cluster	Meet Expectations?
My sd card is 16 gb with fast read write speed class about 20 mm	cluster 2	Yes
great product	cluster 0	Yes
great for my phone	cluster 1	Yes
the sd card is not so great as it is really slow	cluster 2	No
the sd is not so great as it is really slow	cluster 1	Yes
the card is not so great as it is really slow	cluster 2	No

Figure 14: Predictions on new unseen reviews by our selected TF-IDF word2vec cluster model with k=3. The meet expectations column refers to the expectations of a human judge.

We predicted clusters for each of the above test review and checked it against human judgment if the predict cluster is correctly classifying the review. As we can see from the above table, the model is making predictions correctly in most cases. The first review in the table offers the most information and it is classified as knowledgeable as expected. The second and third review are not so knowledgeable and are correctly put in cluster 0 and 1 respectively.

We also see that some of the words like 'card' have been assigned higher weights as was expected. This can be seen in the last three reviews. The three reviews are conveying the same idea which is not very informative. The occurrence of 'card' word is causing the cluster to misclassification of the review by our model. Such words can be added to the list of a product stop-word list that can help eliminate such words that occur a lot for one product. Since our IDF was calculated on the whole corpus of electronics product reviews, 'card' will have an high IDF but for the product the term is not relevant.

Other than these stop words, the model works correctly in classifying the reviews to their correct knowledge level and thus help us predict the expertise of the reviewers for the particular domain.

7 CONCLUSION

In conclusion, we successfully analyzed product reviews to predict the user expertise in the domain of the product. We came up with a methodology for analyzing reviews using K-means clustering and two evaluation approaches to measure the accuracy of our clusters. Our process did require some human judgment which was expected due to the lack of annotated data and expected outcome labels for the intelligence level of our reviews. We see that using TF-IDF word2vec as similarity measure and K-means as our clustering algorithm, we get a model with three clusters that can predict the user expertise based on the technicality or the relevance of the review text.

8 LIMITATIONS AND FUTURE WORK

Armed with a method of analyzing the level of expertise shown in a certain area (SD cards), we can expand this into any topic, and create a more general algorithm that can be applied to all products. However, there are some limitations to this approach. Most critical limitation is the requirement of annotated relevant words for each product or domain that will help us judge the intelligence of a cluster. We handled this problem by manually collecting technical words from a technical blog post. On a larger scale, to get this annotated data, we will require domain experts to annotate the data. This limitation can also be addressed in a cyclic feedback process where we keep updating our annotated data based on the performance of our system. One other limitation is that this is an exhaustive process, meaning that user expertise is domain dependent and we will need to independently assess user expertise in every possible domain in order to judge the user's expertise level for every domain.

There is a lot of scope in this project. We can also investigate the correlations between the users and the reviews they write. It may yield interesting results to keep track of which users. This can be very useful for a variety of tasks, such as recommending documents or products. Someone with little knowledge of photography would not necessarily benefit from knowing the exact resolution, lens aperture and range of a given camera, nor would he likely need either quality to be spectacular; it might be better to recommend cameras in a way that prioritizes lower pricing over quality.

REFERENCES

[1] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, Luo Si, et al. 2012. Expertise retrieval. *Foundations and Trends® in Information Retrieval* 6, 2–3 (2012), 127–256.

[2] K. Collins-Thompson, P. N. Bennett, R. W. White, S. De La Chica, and D. Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM international conference on Information and knowledge*. 403–412.

[3] Gerardo Ocampo Diaz and Vincent Ng. 2018. Modeling and Prediction of Online Product Review Helpfulness: A Survey. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 698–708.

[4] Rosie Jones Kumaran, Giridhar and Omid Madani. 2005. Biasing web search results for topic familiarity. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 271–272.

[5] Julian McAuley. 2016. Amazon product data. *URL http://jmcauley. ucsd. edu/data/amazon* (2016).

[6] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.

[7] TechSpot. 2018. microSD Card Buying Guide. https://www.techspot.com/guides/1591-microsd-buying-guide/