Using Specialized Support Vector Machines to Detect Helpful and Unhelpful Product Reviews

Scott Bolter, Teng-Sheng Moh

Department of Computer Science, San Jose State University

ABSTRACT

In this paper we focused on automatically classifying product reviews as either helpful or unhelpful using machine learning techniques, namely, SVM classifiers. Using LIBSVM and a set of Amazon product reviews from 25 product categories, we train models for each product type to determine if a review will be helpful or unhelpful. Previous work has focused on training one classifier for all reviews in the data set, but we hypothesize that a distinct model for each of the 25 product types will improve the accuracy of classification. Furthermore, we developed a framework to inform authors on the fly if their review is predicted to be of great use (helpful) to other readers, with the hypothesis that authors are more likely to rethink their review post and amend it to be of maximum utility to other readers when given guidance.

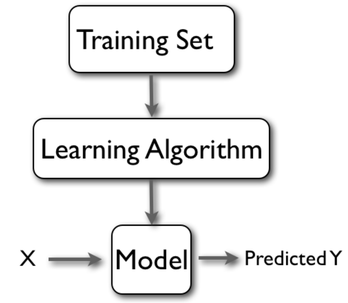
RESEARCH DESIGN AND METHODOLOGY

Building upon past successes with analyzing the utility of product reviews using support vector machines, we employ LIBSVM and a bag of words approach to building many, rather than one, classifiers for each category of products on Amazon.com.
Constructing a cache of classifiers from randomized training sets for each of the product categories, we test these classifiers' accuracy, marking the best performing in each product category.
Using a stricter criteria for marking reviews as eligible for training, our models are specialized to perform strongly against reviews of products residing in their category.
Overall motivation is to maximize the utility of reviews for online shoppers. Specialized classifiers built-in to review submission form might identify unhelpful reviews and influence the author to augment and improve their review of a product.

SYSTEM MODELING AND RESULTS

Supervised Machine Learning

- In machine learning, we attempt to extrapolate trends of a dataset to learn something about a problem and *teach* a model to solve it
  - Models are created based on training data
  - Test data is then fed into models for predictions

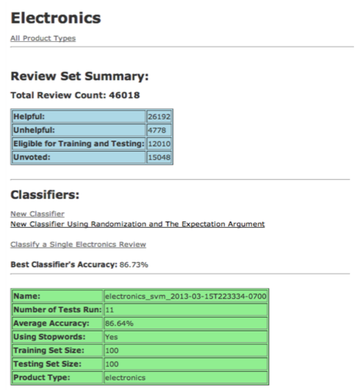


Review Eligibility

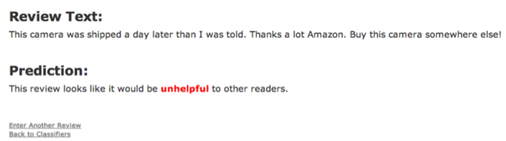
- We only accept reviews for the training and testing sets with:
  - A high degree of agreement (> 0.7)
    - Previous work used helpful votes > 0.6
  - Minimum number of votes (> 3)

Testing Models

- Specify size of testing set, selected from all eligible reviews excluding the reviews used for training
- Compare the predicted class against the known class of each test review to compile accuracy statistics



Predicting Helpfulness of New Reviews



- We use the trained models to implement a framework for predicting the helpfulness of a review as it is submitted
- Hypothesize that authors will attempt to improve their review if they know it is likely going to be unhelpful to other readers

Past Results

- Accuracies of previous work (Kim and Hong used same dataset):

Research	Accuracy (%)
Kim	61.29
Liu	62.85
Hong	69.62

Specialized Model Results

- Of all of the specialized classifiers trained amongst all product types, accuracy is improved over past research
  - Mean: 86.9346%
  - Median: 88.36%

CONCLUSIONS

Using the following principles, previously unexamined by other research in this area, our implementation yielded better accuracy and a new framework for predicting product review helpfulness

- More stringent requirements for review training and testing eligibility
- Specialized classification models for each of 25 different product categories
- On-the-fly helpfulness predictions for new review authoring

FUTURE WORK

- Specialized stopword lists
- Further specialization, down to the individual products
- Consider other review attributes as features for training
- Bigrams, trigrams, quadgrams, etc.
- N-fold cross verification

REFERENCES

1. John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL), 2007
2. Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
3. Yu Hong, Jun Lu, Jianmin Yao, Qiaoming Zhu, and Guodong Zhou. 2012. What reviews are satisfactory: novel features for automatic helpfulness voting. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR'12). ACM, New York, NY, USA, 495-504. DOI=10.1145/2348283.2348351 http://doi.acm.org/10.1145/2348283.2348351
4. Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06). Association for Computational Linguistics, Stroudsburg, PA, USA, 423-430.
5. Ying Liu, Jian Jin, Ping Ji, Jenny A. Harding,and Richard Y. K. Fung. 2013. Identifying helpful online reviews: A product designer's perspective. Comput. Aided Des. 45, 2 (February 2013), 180-194. DOI=10.1016/j.cad.2012.07.008 http://dx.doi.org/10.1016/j.cad.2012.07.008
6. www.statista.com/topics/871/online-shopping/

Using Specialized Support Vector Machines to Detect Helpful and Unhelpful Product Reviews

Scott Bolter and Teng-Sheng Moh
Department of Computer Science
San Jose State University, San Jose, US
scott.bolter@sjsu.edu and teng.moh@sjsu.edu

Abstract

In this paper we focused on automatically classifying product reviews as either helpful or unhelpful using machine learning techniques, namely, SVM classifiers. Using LIBSVM and a set of Amazon product reviews from 25 product categories, we train models for each product type to determine if a review will be helpful or unhelpful. Previous work has focused on training one classifier for all reviews in the data set, but we hypothesize that a distinct model for each of the 25 product types will improve the accuracy of classification. Furthermore, we developed a framework to inform authors on the fly if their review is predicted to be of great use (helpful) to other readers, with the hypothesis that authors are more likely to rethink their review post and amend it to be of maximum utility to other readers when given guidance.

Keywords: review helpfulness; machine learning; product review classification; support vector machine; amazon reviews; svm specialization; bag of words.

1. Introduction

As e-commerce continues to account for increasingly large portions of the consumer economy, online product reviews are of greater importance to shoppers. Indeed, to the irritation and detriment of traditional "brick and mortar" stores, web sites such as Amazon.com have turned these businesses into what many have termed "product showrooms". Customers can visit these traditional stores simply to handle and view products in person that they later intend to purchase from an online vendor. Tech-savvy companies such as Amazon.com have even released mobile applications that consumers can install on their cell phones, turning them into bar code scanners with quick access to the product page on Amazon.com, complete with price, shipping details, and product reviews.

Whether the traditional location-based businesses like it or not, online shopping seems to be growing in popularity. Consumers rely heavily on product reviews, ratings, and testimonials to influence their shopping decisions. For this reason, it is desirable to have quick and easy access to the most informative, accurate, and overall helpful reviews.

Web sites such as Amazon.com have taken to providing features where users can supply feedback about product reviews. While this can come in the form of comments and questions posted to the review author, the most simple and straightforward feature is a simple click of an answer ("yes" or "no") to the question, "Was this review helpful to you?" This information is stored and presented to the customer with the metrics of the review (how many people found the review helpful), as well as the option to sort the reviews by helpfulness or recency.

While this is a good start to providing customers with the means to find the most helpful reviews, the obvious limitation is the reliance on users to manually flag whether or not any given review is or is not helpful in their evaluation of a product. This is why it is necessary to develop a mechanism for automatically marking reviews as helpful or unhelpful. This paper aims to do just that through the use of machine learning. By taking sets of reviews we know to be either helpful or unhelpful, we can train an SVM (Support Vector Machine) model to classify reviews that have no previous votes from other customers as helpful or unhelpful .

We further present a framework for applying these models on-the-fly to newly authored reviews. We hypothesize that when given some guidance about what their review might be lacking and informing users before they submit their product reviews of their predicted helpfulness, overall product review quality will improve.

The paper is organized as follows: Section 2 covers related work in this area. Section 3 outlines the data we used for the research. Section 4 presents the algorithms for training and testing our models. Section 5 presents the results of our findings. Section 6 describes the framework for classifying newly authored reviews, and Section 7 presents conclusions and areas for future work.

2. Related Work

As the prevalence of online shopping and product reviews have increased, so too has the interest in generating the best set of reviews to interested parties: customers, vendors, and producers of the goods being sold. Customers rely on this data to make purchasing decisions. Vendors recognize that the more robust their collection of product reviews, the more traffic will filter through their site, thus yielding a beneficial cycle of more customers leaving more product reviews. Finally, producers of the goods being sold can use these product reviews to make design decisions and decide future product direction to ultimately appeal to more customers, essentially using the reviews to continue doing what customers favored, and altering areas where the products were reviewed poorly.

Blitzer et al [3] procured a dataset of Amazon.com product reviews from 25 different product categories. It has been used by several other studies and is used in this research. Their original research focused on detecting the sentiment, or general emotions, of the review to classify positive versus negative experiences with the product. This mirrors the aim of our

research as they used features of the review other than the actual product rating given by the reviewer to detect a positive (high) or negative (low) rating. Here, we use features other than the helpful or unhelpful votes that a review receives to ultimately determine if it would be found as helpful or not to other readers.

Kim et al [8] presented research in which SVM regression was employed to determine which features of the reviews from an Amazon.com dataset yielded themselves to helpfulness predictions. These features included the review length, unigrams (each word of the review text taken as a distinct feature, as we have done in this research), and the product ratings.

While the most commonly known form of spam is unwanted electronic mail, product review spam is also prevalent amongst online shopping review forums. This can take the form of unrelated links, advertising mixed into product review text, or false reviews, perhaps used to artificially increase the rating of a product. Lau et al [9] employed text mining and probabilistic language modeling to detect spam amongst review sets.

Liu et al [10] conducted research on detecting low-quality reviews using different types of biases. They suggested methods to simply strip these reviews from the available set, yielding a ground-truth opinion of the specific product being reviewed.

Hong et al [6] developed an Automatic Helpfulness Voting system, coined "AHV", by building a ranking SVM classifier to assign a score to each of the reviews being tested and then ranking them in order of helpfulness. They built upon earlier successes with SVM classification by attempting to learn user preferences within their models. These included information needs fulfilled by product reviews, the credibility of reviews, and each reviews' consistency with the mainstream opinion of the product. They found that when including these aspects, AHV performance was improved.

Here, we attempt to further the successes of SVM classification by using trained models for detecting helpful versus unhelpful reviews. By training multiple classifiers for the multiple product categories, we aim to improve accuracy predictions and believe that, intuitively, many models that have increased specificity should yield more accurate results. By developing an application to efficiently train, test, and store these classifiers, we gain the benefit of model multiplicity, effectively drilling down from a high-level model for a set "product reviews" to a more refined model for reviews of a specific product type such as "camera and photo".

3. The Data

Our dataset was obtained from the research done by Blitzer et al [3]. These authors made it available for download at http://www.cs.jhu.edu/~mdredze/datasets/sentiment/. The data is entitled "Multi-Domain Sentiment Dataset" and this research makes use of version 2.0. It is a collection of Amazon.com product reviews pulled from multiple product categories (domains). Some product types, such as books and DVDs, have hundreds of thousands of product reviews while others, such as musical instruments may have only a few hundred.

Each product category is represented in this dataset in an XML file containing all reviews for that category. Each review in the file is represented between *<review>* and *</review>* tags. All reviews have the attributes listed in Table 1 below.

Table 1. Review Attributes

Attribute Name
product_name
product_type
helpful
rating
title
date
reviewer
review_text

All XML files were parsed and the data was entered into a MySQL database. Parsing was done using Nokogiri (http://nokogiri.org/), a plugin, or "gem", for the Ruby programming language. It has excellent support for parsing both HTML and XML using XPATH and CSS3 Selector. The MySQL database is connected to a Ruby on Rails web application, handling the interface, computation, and data presentation. The application is called Review Helpfulness Predictor, or RHP.

4. Template Usage

While past research has focused on constructing a classifier that could handle all product reviews in the dataset, we aimed to specialize the classification models, training many Support Vector Machines for each of the product categories. Like past efforts, we also employed a unigram (each word of the review text treated as a feature) approach and we also similarly used LIBSVM as an implementation for SVM classifiers. The following areas outline the differences in our implementation:

4.1 Review Eligibility

We added some utility columns to the core data in the reviews table. In the research done by both Kim et al [8] and Hong et al [6] the authors set a "voting rate" of 0.6 to draw a boundary between helpful and unhelpful reviews, labeling the reviews that had been voted on beforehand, and then testing their classifiers on the entire dataset. We took a somewhat different approach and introduced the idea of review eligibility when compiling our training and testing sets. Eligibility was determined by the *use_for_train* column which is flagged as true if the total votes are greater than three and over 70% of voters agree that it is either helpful or not, and false otherwise.

This algorithm was run as a preprocessor step prior to training the classifiers. It was initiated with the click of a button located on the page where reviews are indexed according to product type. Once this button was clicked and this algorithm completed, we were left with a new partitioning of the product type's reviews. Where previously there were

helpful, unhelpful, and reviews that had no votes, the review set was now organized as reviews with no votes, reviews with votes but without the required degree of agreement on their utility (helpful or unhelpful votes were too low or without a clear winner), and lastly, with a subset of reviews that were candidates for training and testing the classifiers. Note that we stipulated that the degree of agreement on a review's utility (as voted by actual Amazon.com users) must be above 70 percent. This meant that a review where 3 of 5 people found it to be helpful would be discarded as ineligible for classifier training because 40 percent of the users that took the time to vote found the review to be unhelpful. We also imposed a minimum on the number of votes the review received. This criterion differed from past research as the above example review would simply have been counted as helpful and used in the model. The reason being that reviews with a stronger degree of agreement, be it on the review's helpfulness or uselessness, would better inform our model during training and increase classification accuracy.

4.2 Selecting Training and Testing Sets

When using supervised learning to train models, it is important to select an appropriate training set. Ideally, this subset of data should be representative of the whole dataset to maximize the accuracy of the model. Initially, our algorithm was very naive, selecting the training data from the database in the order that it was stored. For example, a request for 100 reviews in the training set and 50 reviews in the testing set would return the first 100 reviews that were eligible (as ordered by the id field of the reviews table, the unique, primary key) as the training set and then the next 50 eligible reviews as the testing set. This made the model dependent on how the reviews were loaded into the database, as the reviews with lower primary key identifiers were favored in the training and testing of the model, which introduced problems. For example, if the first 99 eligible reviews happened to be helpful and the next 51 were unhelpful, our model would be plagued with overfitting and the results would be unreliable. To mitigate this, we tried to shape the query based on the product type's overall ratio of helpful to unhelpful reviews. For example, if there were 4,000 unhelpful reviews and 6,000 helpful reviews of grocery items, a request for the training set of size 100 reviews would return the first 40 unhelpful reviews and the first 60 helpful reviews. This approach was still problematic as it continued to favor one sector of the dataset. The final implementation employed randomization to select training and test sets.

Here we employed the Boolean flag *use_for_train* that was set earlier to obtain an entire set of all reviews of this product type that can be used for training and testing. Next, we made use of Ruby's sample method, applicable to any collection of objects. This afforded us the ability to pull an object, a review in our case, from the array using randomly generated indices. By deleting this review from the set of eligible reviews, we shrank this array each time we built up our training and testing sets, ensuring that we avoided duplicate entries in these collections. Note that this randomization yielded the added benefit of a built-in adherence to the ratio of helpful to unhelpful reviews in the original dataset. Since we randomly selected each review for the training and testing sets, the probability that this selection was either helpful or unhelpful mirrored the entire dataset's overall statistics of helpful and unhelpful reviews.

4.3 Building a Classifier

The training of a model in machine learning can be simplified into three entities: input (training data), learning process, and output (a model ready to accept data for classification). The previous section explained how we derive the first entity.

Using a bag of words approach with LIBSVM, we constructed models ready to accept and classify reviews from the test sets. This algorithm relied heavily on an implementation of LIBSVM installed as a Ruby plugin. LIBSVM is an effective and efficient implementation of SVM classification. It was developed by Chih-Chung Chang and Chih-Jen Lin [7]. The web page, http://www.csie.ntu.edu.tw/~cjlin/libsvm/, offers extensive information about this software and links to other language plugins. The Ruby plugin used, rb-lisvm, available at https://github.com/febeling/rb-libsvm, is a Ruby binding to this LIBSVM software. In addition to LIBSVM, we also employed a word stemming plugin from Roman Shterenzon, fast-stemmer. It is available at https://github.com/romanbsd/fast-stemmer. This allowed us to efficiently stem words in our global dictionary and each training document.

After organizing our documents into a set of training vectors mapped against each feature (word) in our global dictionary, we used the LIBSVM class to train the model. The training function accepts the vectors and a parameter object as input. The parameter object stipulates important details about the SVM classifier. These details are explained in depth by Chang et al [7] and this algorithm adopted the parameters that have been proven to work well in similar classification problems. It is worth mentioning that the LIBSVM model contains a method to save its data to a file. This, coupled with our implementation to save the global dictionary used to train the model, allows the application to easily persist and reload any trained classifier for future testing.

4.4 Testing a Classifier

With the models trained and saved as outlined above, we can load and feed in a test set to observe the classifier's accuracy. In machine learning, to predict the accuracy of a model, the test set must be comprised of data with known classes. That is, our reviews in the test set are reviews that have already been voted to be either helpful or unhelpful. Recall that we employed the same flag used for marking training reviews as eligible to create a pool of eligible test reviews.

Applying the models to test sets is fairly straightforward, with the main point being the comparison of the model's prediction of the review against the known class (helpful or unhelpful) of the review. Also of note is the responsibility of a classifier to keep track of its accuracy. This allowed us to create competing classifiers for the same product category to track which is the best performing. Tracking the classifier with the highest accuracy affords the application the ability to choose which classifier is to be used to predict the utility of a review that is entered on-the-fly by a user, explained in the

next section. Lastly we returned the results of testing to the application's classifier controller, to render a view with the results presented on a web page.

5. Results

Both Kim et al [8] and Hong et al [6] also used the Multi-Domain Sentiment Dataset from Blitzer et al [3] for results testing, but as the latter research outperformed the former, we used the results of Hong et al as a baseline for our comparison. Table 2 presents the results of past research.

Table 2. Past Research Classifier Accuracy

Table Column Head	Accuracy (%)
Kim	62.29
Liu	62.85
Hong	69.62

While each evolution of this research has improved classification accuracy, all research thus far has, to our knowledge, focused on lumped-together product type reviews. As shown in Table 3, specializing the models for specific partitions of the review dataset significantly increases the resulting accuracies.

Table 3. Specialized Review Helpfulness Predictor Results

Product Category	Best Accuracy (%)
Apparel	90.82
Automotive	81.82
Baby	92.0
Beauty	93.55
Books	81.91
Camera & Photo	89.09
Computer & Video Games	90.55
DVD	79.18
Electronics	86.73
Gourmet Food	90.73
Grocery	88.36
Health & Personal Care	90.55
Jewelry & Watches	89.0
Kitchen & Housewares	93.45
Magazines	85.82
Music	77.0
Musical Instruments	93.82
Office Products	95.0
Outdoor Living	90.82
Software	85.91
Sports & Outdoors	89.36
Tools & Hardware	N/A
Toys & Games	90.18
Video	80.64

For each product type, we trained three classifiers with training set sizes of 100, 300, and 600 product reviews wherever possible. As the table shows, our accuracies for the Amazon.com product reviews are considerably higher than previous work, with some caveats. In the "tools & hardware" category, for example, only 40 reviews met our criteria for training and testing eligibility. Among these, only a small number were voted unhelpful. Whereas most product types had classifiers with a minimum training size of 100 reviews, here we were only able to train the model with 20 reviews, leaving 20 for testing. This resulted in overfitting our model and without a wide array of test reviews available, the accuracy was artificially high, and therefore omitted in the table. While this anomaly is present in a few other product types that are less popular with consumers on Amazon.com, this artificially high accuracy seems to be the exception. Even the most popular categories such as books, DVDs, and music, all have classifiers that performed quite well in comparison to past research. It is also clear that the most specific product categories tended to have the higher accuracies. For example, while the camera and photo product type could actually be considered "electronics" and lumped into this product type, the fact that it is separate and houses distinct, more specific reviews, lent itself to RHP's specialized models. This is what one would expect as the dictionary for electronics is far broader than any sub-category of electronic products might be, thus increasing the challenge of accurate classification.

6. Classifying a Single New Review

While the testing algorithm encompasses predicting a single review as well (executed with a user-entered review as the sole member of the test set), we explain this framework explicitly as, to our knowledge, it has yet to be suggested in previous work. The screen shot in Figure 1 shows what an indexing of classifiers page looks like for the "camera and photo" product type.

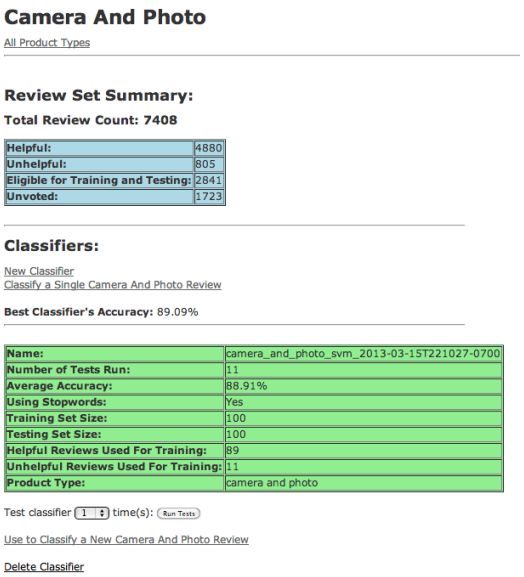


Figure 1. Classifier View

The bottom table holds information about only one of three classifiers that was trained for this product category. Indeed, there are two others below that are not shown, one of which is the most accurate, with an accuracy of 89.09%. Although we built in links to use each of the classifiers to predict a single, user-entered review, the link at the top automatically used the classifier with the highest accuracy. The process is shown in Figure 2 below.

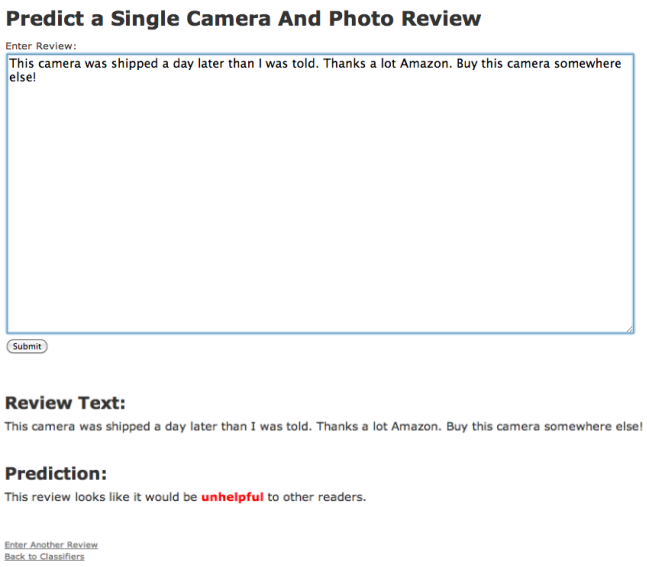


Figure 2. Predicting a Single New Review

As this review is concerned only with the vendor and the shipping satisfaction of the author, we suspected it would have little utility to other readers looking for information about the product itself. Our classifier's prediction agreed, marking this review as unhelpful. While simple in nature, we believe that this framework could be implemented in online shopping websites and would help users to author higher quality reviews.

7. Conclusions and Future Work

While limited in novelty, this project confirms an admittedly intuitive suspicion, that specialized machine learners perform better than general models. We incorporated a dataset from XML into a more manageable and efficient format in a MySQL database. Using queues from past research, we pursued Support Vector Machines as an avenue for machine learning to automatically predict review helpfulness. Using Ruby on Rails, employing a Model-View-Controller architecture, and an implementation of LIBSVM, we built the Review Helpfulness Predictor (RHP) web application with functionality to train and test models, and predict single product reviews as either helpful or unhelpful. As proven by our results, our hypothesis of creating specialized machine learners to increase performance on a partitioned dataset is an effective way to classify product reviews. Aside from these improved results, we also offered a framework for selecting and employing a classifier to perform on-the-fly classification of product reviews entered by customers, with the end goal of improving review quality to the benefit of customers, online vendors, and manufacturers.

As our hypothesis appears valid, we see no reason why machine learners should not be even further specialized by partitioning the product types further. Given adequate numbers of reviews for single products, it is not inconceivable to have a classifier trained specifically for one popular product sold. We also suspect that incorporating more words into a single feature (using bigrams, trigrams, quadgrams, etc.) during training, rather than treating each word as a distinct feature might have a positive effect on performance. For example, we treated the presence of "focal length" as one feature in a 'camera and photo' review rather than "focal" and "length" as distinct features.

As part of a further analysis of the relationship between kinds of product reviews used for training versus the kind of product review we are predicting as helpful or unhelpful, we have created an additional model type, the MAKifier (Many Applicable Kinds) classifier. As opposed to the classifier model, a MAKifier can have many kinds of product reviews used for the training set, and is used to predict the helpfulness of reviews from one kind of product. This affords us a sliding scale from a highly specialized machine learner to the type of classifier used by previous research (one classifier for all kinds of product reviews). Our preliminary testing included using all 25 product types to train a model and then test it against only 'camera and photo' reviews. Not surprisingly, this yielded an accuracy close to, but slightly less, than previous research, 60.7%. We also tested 'camera and photo' reviews against classifiers that were trained with fairly unrelated kinds of products, 'baby', 'dvd', 'office products', 'beauty', 'magazines', and 'health and personal care'. These tests yielded accuracies of 52% and 55%, only slightly higher than simple chance. One interesting observation from early MAKifier testing was the improved accuracy when training reviews are used from a kind of product that is similar to the one used for testing. For example, when a learner was trained with 'dvd' and 'video' reviews and was tested against 'book' reviews, the accuracy was significantly higher at 68% as opposed to testing 'book' reviews against 'apparel', 'automotive', and 'kitchen and housewares' reviews, where accuracy was only 51%. We suspect this is due to the common elements of reviews from 'dvd', 'video', and 'books' such as story, emotion evocation, and prose commentary. This warrants further research to extrapolate indicators of similarities between different kinds of products, as well as the relationship between the number of product types used for training a model versus the number of product types used for testing. To further explore and verify the improved accuracy, further research warrants a more extensive 10-fold or even n-fold cross verification.

Lastly, we believe that the interface for predicting a single review and reporting the predicted helpfulness to the author could benefit from added details. For example, rather than simply stating whether their review looks to be either helpful or unhelpful to other readers, enhancing the feedback to include specific content that could be added or improved upon would ultimately guide the author toward creating a review of higher quality and helpfulness.

References

[1] Mumtaz M. Al-Mukhtar and Yasmine M. Tabra. 2012. An effective spam filter based on a combined support vector machine approach. Int. J. Internet Technol. Secur. Syst. 4, 1 (January 2012), 42-54. DOI=10.1504/IJITST.2012.045149 http://dx.doi.org/10.1504/IJITST.2012.045149

[2] Izzat Alsmadi, Mohammed Al-Kabi, Abdullah Wahbeh, Qasem Al-Radaideh, and Emad Al-Shawakfa. 2011. The Effect of Stemming on Arabic Text Classification: An Empirical Study. Int. J. Inf. Retr. Res. 1, 3 (July 2011), 54-70. DOI=10.4018/IJIRR.2011070104 http://dx.doi.org/10.4018/IJIRR.2011070104

[3] John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL), 2007

[4] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. 2005. Working Set Selection Using Second Order Information for Training Support Vector Machines. J. Mach. Learn. Res. 6 (December 2005), 1889-1918.

[5] Anindya Ghose and Panagiotis G. Ipeirotis. 2007. Designing novel review ranking systems: predicting the usefulness and impact of reviews. In Proceedings of the ninth international conference on Electronic commerce (ICEC '07). ACM, New York, NY, USA, 303-310. DOI=10.1145/1282100.1282158 http://doi.acm.org/10.1145/1282100.1282158

[6] Yu Hong, Jun Lu, Jianmin Yao, Qiaoming Zhu, and Guodong Zhou. 2012. What reviews are satisfactory: novel features for automatic helpfulness voting. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR'12). ACM, New York, NY, USA, 495-504. DOI=10.1145/2348283.2348351 http://doi.acm.org/10.1145/2348283.2348351

[7] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[8] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06). Association for Computational Linguistics, Stroudsburg, PA, USA, 423-430.

[9] Raymond Y. K. Lau, S. Y. Liao, Ron Chi-Wai Kwok, Kaiquan Xu, Yunqing Xia, and Yuefeng Li. 2012. Text mining and probabilistic language modeling for online review spam detection. ACM Trans. Manage. Inf. Syst. 2, 4, Article 25 (January 2012), 30 pages. DOI=10.1145/2070710.2070716 http://doi.acm.org/10.1145/2070710.2070716

[10] Ying Liu, Jian Jin, Ping Ji, Jenny A. Harding,and Richard Y. K. Fung. 2013. Identifying helpful online reviews: A product designer's perspective. Comput. Aided Des. 45, 2 (February 2013), 180-194. DOI=10.1016/j.cad.2012.07.008

[11] Thomas L. Ngo-Ye and Atish P. Sinha. 2012. Analyzing Online Review Helpfulness Using a Regressional ReliefF-Enhanced Text Mining Method. ACM Trans. Manage. Inf. Syst. 3, 2, Article 10 (July 2012), 20 pages. DOI=10.1145/2229156.2229158 http://doi.acm.org/10.1145/2229156.2229158

[12] Michael P. O'Mahony and Barry Smyth. 2010. Using readability tests to predict helpful product reviews. In Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO '10). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, Paris, France, France, 164-167

[13] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2005. Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.