Sentiment Analysis in Amazon Reviews Using Probabilistic Machine Learning

Callen Rain
Swarthmore College
Department of Computer Science
crain1@swarthmore.edu

Abstract

Users of the online shopping site Amazon are encouraged to post reviews of the products that they purchase. Little attempt is made by Amazon to restrict or limit the content of these reviews. The number of reviews for different products varies, but the reviews provide accessible and plentiful data for relatively easy analysis for a range of applications. This paper seeks to apply and extend the current work in the field of natural language processing and sentiment analysis to data retrieved from Amazon. Naive Bayes and decision list classifiers are used to tag a given review as positive or negative. The number of stars a user gives a product is used as training data to perform supervised machine learning. A corpus contains 50,000 product review from 15 products serves as the dataset of study. Top selling and reviewed books on the site are the primary focus of the experiments, but useful features of them that aid in accurate classification are compared to those most useful in classification of other media products. The features, such as bag-of-words and bigrams, are compared to one another in their effectiveness in correctly tagging reviews. Errors in classification and general difficulties regarding the selection of features are analyzed and discussed.

1 Introduction

As the marketplace for consumer products moves to the Internet, the shopping experience changes in a way that makes much of the information regarding the use products available online and generated by users. This contrasts the way that product information used to be disseminated: through word of mouth and advertising. Since its creation as an online bookstore in 1994, Amazon.com has grown rapidly and been a microcosm for user-supplied reviews. Soon, Amazon opened its reviews to consumers, and eventually allowed any user to post a review for any one of the millions of products on the site. With this increase in anonymous user-generated content, efforts must be made to understand the information in the correct context, and develop methods to determine the intent of the author. Understanding what online users think of its content can help a company market its product as well as mange its online reputation.

The purpose of this paper is to investigate a small part of this large problem: positive and negative attitudes towards products. Sentiment analysis attempts to determine which features of text are indicative of it's context (positive, negative, objective, subjective, etc.) and build systems to take advantage of these features. The problem of classifying text as positive or negative is not the whole problem in and of itself, but it offers a simple enough premise to build upon further.

Much of the work involved in sentiment analysis in content containing personal opinions has been done relatively recently. Pang and Lee (2002) used several machine learning systems to classify

a large carpus of movie reviews. Although Naive Bayes did not perform the best out of their strategies, it did well compared to the baseline provided by human-generated classifying words. Yessenov and Misailovi (2002) did similar work, taking comments off of social networking sites in relation to movie reviews, a somewhat more anonymous context. This idea of informality and its effect on sentiment classification has been researched by Thelwall et al. (2010). Specifically, they tackled the problem of classifying when slang is used and documents lack the uniformity of vocabulary and spelling that a movie review database would contain.

Amazon employs a 1-to-5 scale for all products, regardless of their category, and it becomes challenging to determine the advantages and disadvantes to different parts of a product. Problems with such rating systems are discussed by Hu and Liu (2004). They attempted to categorize opinions about different parts of a product and present these independently to give readers more information that positive/negative sentiment.

2 Methodology

Amazon reviews are plentiful, but a corpus generated from a the average Amazon product is not generally long enough to perform adequate supervised learning on. I chose to analyze a few highly reviewed products because the system would have a higher success rate above the most frequent sense baseline, making the relative effect of each of the extracted features more apparent. Additionally, the logistical questions of selecting truly random product from such a diverse selection seemed like a problem for another project. I opted to download and primarily analyze the reviews for the most reviewed products on the site. Most of these are books (Harry Potter, actually), and the rest are music CDs and movies. The reviews for these (books and other media) were downloaded and viewed as two separate datasets. As well as these, I wanted to test products from a variety of different categories on the site. I downloaded the reviews for the Amazon Kindle, which has a comparably large number compared with the top books and media. Then I downloaded several smaller review sets from products such as Levi Jeans, Kindle Fire, BN Nook, Taylor Swift new 'Red' album, Apple Macbook Pro, and Bill O'Reilly's 'Killing Lincoln'. Relative to other products on the site, these items have a large number of reviews, but they don't really compare to the items in the top 100.

2.1 Data

Downloading and Parsing Although Amazon does not have an API like Twitter to download reviews with, it does have links for every review on every product, so one can technically traverse the site through product IDs. I used two Perl scripts written by Andrea Esuli to obtain the reviews for the Kindle and a few other products. The first script downloads the entire HTML page for the product and the second searches the file for information about the review, such as the product ID, rating, review date, and review text.

Extracting a Review The reviews for a given product get saved in a text file that is then formatted into a list of tuples consisting of the review text and the score given by the reviewer. Each of the review texts is tokenized, and all punctuation except periods, apostrophes, and hyphens is removed. The first entry in the tuple is this list of tokens. The Amazon website only allows rating from 1 to 5, and using this rating system to classify texts seemed like it would give poor results because of the lack of distinction between reviews receiving similar scores (eg. 4 vs 5). Instead, reviews receiving a 1 or 2 star rating were given a '0' score in the data, whereas reviews receiving 5s received a score of '1'.

Positivity and Amazon It may seem odd that 4s were also not given the positive score, but it remains a fact that the most purchased products on Amazon receive many, many more positive reviews than negative ones. It made the most sense to obtain the most reviews that could be interpreted as negative (1 and 2 stars) and restrict the number of the positive reviews until the two equaled each other and the most frequent sense baseline equaled 50%. This does mean that perhaps some of the defining features in the positive reviews were lost while all of those in the negative ones were preserved, but since only the most frequent features were retrieved anyways, the crucial features in the positive set were most liekly preserved. Perhaps the best classifying

system will only train on negative reviews, attempt to generate specific features just for them, and then tag everything as positive. These observations reveal some weaknesses in the Amazon review model, which seeks to provide an impartial, homogenous analysis of products for shoppers. This "perfect" review doesn't really exist, and it certainly doesn't come from diverse sources. Interviews with some of the top 1000 book reviewers on the site revealed that approximately 88% said that they write mostly positive reviews, 70% are male, and half hold a graduate degree (Pinch and Kesler, 2011). Demographics aside, it remains an interesting question if useful information from reviews can be provided if 95% of them are positive. Comparing different products (books vs. electronics) of different popularities on the same simple 1 to 5 scale might not be the best way to inform shopping choices. There is a decrease in usefulness when one is comparing several products that have all acheived high acclaim. If both the Nook and Kindle e-readers both have thousands of positive reviews, the most accurate way to describe the comparison is "they are both really good." There is no efficient way in the current review system to compare different aspects of them, consider the advantages and disadvantages of each, and make a more informed decision.

2.2 Feature Extraction

Bag of Words A bag of words feature vector consists of all of the words in the article as independent features. In these experiments, all of the words were added to a list and only the top 2000 most frequently occurring words were kept. Each of the words in this list was then compared to the words in the review list, and a dictionary was generated that mapped each of the features to either True or False, denoting whether the feature appeared in the review. This is known as a binary feature vector. The alternate approach would be to collect all of the words in all of the reviews and obtain counts for them, but doing this is computationally intensive and did not, in some small tests, offer more accurate results. It made more sense to select different, more informative features than to fill the vectors solely with bag of words features.

Collocations Since the bag of words feature model assumes the independence of each of the words, it ignores some of the relationships between words that add affect their meanings in the context of the article. For example, the phrase "low price", has a different meaning than "low" and "price" appearing independently. These relationships between words can be captured in the feature vectors for the articles by including common bigrams as well. Using the "bigram" function provided by NLTK, all of the bigrams in each of the articles are saved and sorted by frequency. The top 500 of them are included in the feature vectors for each of the articles.

Handling Negation A more specific case of the issue presented above is the case of negation. Words that occur after a negated word in an article have opposite meanings than what they originally meant and present noise in the data if they are saved as unigrams. For example, the phrase "not like" would indicate a negative review, while "not" and like independently would not necessarily give the same classification. This problem is remedied by adding each of the words after a negative word as "contains(NOT like)". This still does not provide any relation to the "contains(like)" feature, because it is added separately. It instead serves to make the two features more useful in classifying, because they will tend to occur in differently tagged articles.

I found that adding all the words until the next period did not yield better results than just adding the next 3 or 4 words. If a reviewer writes "I don't like the book and I hate the plot!", the "hate" in the sentence should not be stored as a negation. I found that storing the three words after the negative word gave the best accuracy.

Spell Checking Unlike many reviewing sites whose users are professional or well-known reviewers, the bulk of reviews on Amazon are done by anonymous individuals. This lack of accountability in their writing results in much more frequent grammatical errors in the reviews. If a user spells a key word wrong (often "disappointment"), the classifier will ignore the significance of such an important word because if cannot connect it to all of the other "disappointment"s occurring frequently in other negative reviews. The ASpell module was used as a spell checker to attempt to resolve some of

these inconsistencies. The spellchecker can check to see if it contains a given word in its dictionary and it can suggest a list of words if it cannot find the word. This list of returned words can be very large, and it appeared that words that were only slightly misspelled would have a smaller list of suggested words. Spelling was optionally checked as part of the bag of words extraction. As the program checked to see if each word in the feature list appeared in the review document, it checked too see if it had a suggestion that also appeared in the feature list. If it did, the spell-checked word are marked as present.

Part of Speech Tags When a user reviews a product, often their adjectives give the best clues to their opinions. Positive and negative reviews regarding books will contain common nouns appearing often when anyone talks about books, but there are rarely specific adjectives that can be used with both classifications. The corpus was tested with only adjectives in the feature sets, and this assumption was supported.

Sentence Length After spending time downloading and reading many reviews, I developed the hypothesis that negative reviews would be shorter than positive ones. After reading some more, it seemed that they were not always shorter in length, but there were often short sentences in them. In an attempt to be blunt and critical, users posting one star reviews like to keep their sentences short and dramatic. I tested this assumption by calculating the average number of words in a sentence in a given document and looking at the distribution to see if there was a pattern. I ended up adding a short sentence feature for documents with average sentence length shorter than 10 and a similar long sentence one for documents with average sentence length longer than 20.

2.3 Classification

Naive Bayes Naive Bayes is a simple but robust classifier applied using Bayes' Rules that yields very useful results. It assumes the independence of each of the features in the vector and for each feature, calculates the probability that it will appear given the class. The probability of a class given the feature set is simply the product of the probability that the class will occur and the probabilities of each of the feature vectors. This process is repeated for each of the possible classes and the text is classified according to the maximum probability. The formal definition is given as:

$$\hat{s} = s \epsilon S argmax P(s) \prod_{j=1}^{n} P(f_j|s)$$

where

$$P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$$

$$P(f_j|s) = \frac{count(f_j, s)}{count(s)}$$

Decision List The Decision list a rule-based tagger that has the advantage of being human readable. One of the major problems with Naive Bayes is the difficulty in identifying which probabilities are causing certain classifications. One can look at the errors made, but there's no way to actually know if there is some feature that is at the root of the problem. The format of a decision list alleviates this problem by making the classification rule-based. The classifier must only determine the existence of the feature at each level and tag appropriately if the feature is in the document. This makes is very easy to intensify which rules the classifier thinks are important, and aids in the removal of features that are causing inaccurate rules.

3 Results

3.1 Parameters

Before comparing the two algorithms directly, it was necessary to find appropriate parameters for them. The systems depend on many subleties of their implementations, but the performances of both the Decision List and Naive Bayes also depend on the number of features that are considered. Additionally, the number of rules that are applied in the Decision List before a tag is made will affect its performance. These parameters do also depend on the size of the test and training sets, but these tests are simply meant to provide an estimate so that the systems can be tested better in other ways.

It was found that the decision list worked best with 100 of the top occuring features considered along a 20 rule limit before the class is picked randomly. Naive Bayes had the highest accuracy when it was used with 800 features.

Decision List Parameters

# Features	Rules	Accuracy
50	2000	0.58914729
90	2000	0.74418605
100	2000	**0.7751938**
200	2000	0.72093023
500	2000	0.65116279
1000	2000	0.59689922
2000	2000	0.59689922
100	10	0.78294574
100	15	0.78294574
100	20	**0.79844961**
100	25	0.79069767
100	30	0.78294574
100	35	0.78294574
100	40	0.78294574

Naive Bayes Parameters

# Features	Accuracy
500	0.7751938
700	0.80620155
800	**0.86821705**
900	0.81395349
1000	0.78294574
1500	0.81395349

3.2 Naive Bayes vs. Decision List

Decision List vs. Nave Bayes

Data Set	Decision List	Nave Bayes
all-books	0.79844961	**0.84496124**
kindle	0.74666667	**0.84**
all-media	0.6828479	**0.79935275**

The Naive Bayes classifier performed better than the decision list classifier with all three of the data sets. Both algorithms performed to the most poorly with the media dataset. I assume this is because the data for that corpus is not as specific as the other two. Apart from them also all being books, the three of the books in the books data set are Harry Potter books, so some such features do determine classifications made. The kindle corpus has a similar advantage of being centered around one single product.

3.3 Feature Analysis

Features and Naive Bayes

Feat.	books	media	kindle
bow	**0.7829**	**0.8220**	**0.8733**
bow/neg.	**0.7906**	**0.8446**	0.86
bow/num.	0.7829	0.8220	0.8666
bow/sent. len.	0.7906	0.8220	**0.8733**
bow/coll.	**0.7829**	0.8155	0.8666
bow/spell	**0.7829**	0.8155	0.8666
bow/neg./sent. len.	0.7829	0.8187	0.8666
adj./adv.	0.6279	0.6990	0.7466

Bag of words ended up being the best feature extraction method that it seemed appropriate to compare all the others against to see if they could raise the accuracy it achieved. Entering words following negative words as negated features showed an improvement as well. Considering the length of sentences in the kindle data set was the only feature that did not reduce the accuracy from the baseline set by the unigrams. Tagging the parts of speech for the unigram features and removing all but the adjectives and adverbs still performed well above the random

Most Informative Bag-of-Words Features

Decision List		Naive Bayes	
Pos.	Neg.	Pos.	Neg.
easy	back	carry	did
love	which	perfect	sent
reader	after	awesome	connect
great	will	loves	return
really	they	easy	bad
read	buy	value	returned
books	screen	eyes	months
much	no	love	try
reading	touch	pocket	canada
price	out	lighter	support

baseline, which is still significant considering how many fewer features were in those vectors.

3.4 Cross-Product Results

Train on Kindle

Test	# Reviews	Dec. List	Naive Bayes
Kindle	6316	0.7467	0.84
Fire	3198	0.6734	0.8394
Nook	182	0.7415	1
MacBook	192	0.6379	0.8824
T.Swift	210	0.8342	0.6523
Levi	573	0.5524	0.7027
Lincoln	3895	0.6639	0.7388

Here, the classifiers were trained on the Kindle dataset and tested on each of the smaller datasets. It is interesting to note that moving futher from the Kindle in related categories makes the accuracy go down. This is because the feature vectors for the Kindle contain words like "screen", "read", and "lightweight" which would not apply to the Taylor Swift album but would still marginally apply to the MacBook and would definitely apply to the Nook. It should also be noted that the datasets with fewer numbers of reviews are easier to tag. The Naive Bayes classifier tagged the Nook at 100%, but failed to tag the Taylor Swift album at over 66% even though it was small. To be tagged at the extreme levels of accuracy, a small dataset must be categorically related to the product that the system was trained on.

4 Analysis

Finding optimal parameter values for the two classifiers made it apparent how drastically they can change the accuracy of the algorithm. By having evidence that the number of features is so important, it also gives a sense of how important the composition of these vectors is. One thing that I think was missing from my implementation was a greater consideration of how many features from each category I was using. The bag of words features outperformed the rest by a large margin. Part of this could have been the fact that I had close to 800 bag of words features in the vectors for Naive Bayes, but only as many of the other features as I had documents. This would naturally favor the unigram model to be more successful. I could have implemented some of the auxiliary features I was testing with more occurences per document. For example, I only assigned a sentence length feature occassionally, and when I did, it was the only one of its kind in the vector. Perhaps it just did not get assigned to vectors enough in the data to yield measureable results comparable to bag of words.

Regardless of this, bag of words is clearly a robust method of feature extraction. Better implementation of the others would have only added to its accuracy. The most frequently used bag-of-words features for each of the classifiers definitely resemble logical positively and negatively connotated words. The negative words are definitely less accurate, with words like "after" and "try" being the worst of them. They can be seen wokring in some sentences, like "I try every day to read, but I hate it!", but the words don't have meanings a strong as some of the others. The fact that the unigram model did do well means that much of how a user feels can be approximated by considering each of the words that they write independently of one another. This fact may not be true of simply positive or negative classification, where more sophisticated methods of feature extraction would have to be pursued, but it does mean that very accurate systems could be built on a large scale to detect simple sentiment like this.

The results of the cross-product training and testing can tell us that narrowing training to a specific

category of products will greatly increase its performance. It became clear that training on electronics components and testing on books and clothing would be very innaccurate on a larger scale. The basic words that users write to explain their simple satisfaction or disatisfaction with a product would yield a certain amount of success, but in order to gain any deeper information, one would have to consider more about the physical properties of the objects and what a typical user might want to comment on.

5 Conclusion

Generally, the results of this experiement were very successful. The classifiers managed to accurately tag a great amount of user-generated data much futher past the random baseline of 50%. Most of the new features that I tested were relatively unsuccessful, but that is most likely due to their implementation relative to the bag-of-words. I think the theory behind their use is sound and they could be implemented together in more successful way in later experiments.

As mentioned above, this work could be extended to make the system of numbered star rating more useful to users. The success of the bag-of-words feature extraction could be used to make systems that analyze more diverse sets of data, but it may have more use in smaller datasets. The systems performed reasonably well on small data sets even when they trained and tested on products that were completely different. This could be applied not to the testing of different products, but instead to the testing of different features of a product. Something missing from a quick glance at a product page is the knowledge of what the best features of that product are. The classifying systems here could be used to determine if the screen of the Kindle is better than that of the Nook, or which has a nicer keyboard. These questions are more useful to readers than simple stars, and the necessary features are in the text. Users do reflect on specific components of products when they review, but that information is lost in a way when so many are gathered together.

References

M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

S. Mukherjee and P. Bhattacharyya. Feature specific sentiment analysis for product reviews. *Computational Linguistics and Intelligent Text Processing*, pages 475–487, 2012.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

T. Pinch and F. Kesler. How aunt ammy gets her free lunch: A study of the top-thousand customer reviewers at amazon. com, 2011.

M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010.

K. Yessenov and S. Misailovic. Sentiment analysis of movie review comments. *Methodology*, pages 1–17, 2009.