Consumer product recommendation with sentimental analyze of online reviews
Dell Amerlina Rios, Matias and Zhang, Xueyuan

For Summer 14 CS922 Machine Learning class instructed by Professor Ming-Hwa Wang

Preface
97% who made a purchase based on an online review found the review to be accurate.

Most popular products have hundreds of reviews. So, finding the most helpful review can save our time, making more satisfied purchase and getting our dreaming product. In this paper, we review previous papers and exploring latest dataset to provide a more adaptive algorithm to generate a recommendation list of existing reviews.

Acknowledgements
We would like to express our very great appreciation to the weather of bay area, which is super good for us to concentrate on our topic.
Contents

Preface

Acknowledgements

Contents

List of tables/figures

Abstract

1 Introduction

1.1 Objective

1.2 What is the problem

1.3 Why this is a project related to this class

1.4 Why other approach is no good

1.5 Why you think your approach is better

1.6 Statement of the problem

1.7 Area or scope of investigation

2 Theoretical bases and literature review

2.1 Definition of the problem

2.2 Theoretical background of the problem

2.3 Related research to solve the problem

2.4 Advantage/disadvantage of those research

2.5 Your solution to solve this problem

2.6 Where your solution different from others

2.7 Why your solution is better

3 Hypothesis or goals

4 Methodology

4.1 Dataset

4.1.1 Amazon reviews

4.1.2 Sentimental words

4.2 Algorithm

4.2.1 Algorithm design

4.2.2 Language used

4.2.3 Tools used

4.2.4 Prototype (optional if time permit)

4.3 Output

4.4 Evaluation

5 Implementation

5.1 Code (refer programming requirements)

5.2 Design document and flowchart

5.2.1 Features

5.2.3 Label useful review

5.2.4 Training and testing

5.2.5 Logistic regression
5.2.6 Flow chart

6 Data analysis and discussion
   6.1 Output generation
   6.2 Output analysis
   6.3 Compare output against hypothesis
   6.4 Abnormal case explanation (the most important task)
      6.4.1 Empty input from raw dataset
      6.4.2 Competition of reviews
      6.4.3 Low quality of review

7 Conclusion and recommendations
   7.1 Summary and conclusions
   7.2 Recommendations for future studies

8 Bibliography

9 Appendices
   9.1 Program flowchart
   9.2 Program source code with documentation
   9.3 Input/output listing
      9.3.1 Input
      9.3.2 Output
   9.4 Other related material
List of tables/figures
Figure 1. Online review sample with sentimental words.
Figure 2. Algorithm design
Figure 3. Generate features
Figure 4. Training/Testing process

Table 1. List of features extracted from the reviews.
Table 2. Feature list
Abstract
Reviews are meant to help customer in buying decision. Given the large number of reviews available and how fast are growing, it became important to automatically identify those that are the most informative. In this paper we use different technique of machine learning combine with sentiment analysis to identify the most helpful review to assist a customer in their buying decision.

[key words] Opinion Mining; Sentiment-based Product Recommendation; Online Reviews; Machine Learning.

1 Introduction

1.1 Objective
The purpose of this paper is to get the most helpful product reviews based on sentiment words in the review dataset.

1.2 What is the problem
Nowadays, there are millions of reviews on internet from products to services. Even, sites dedicated to just show and let users create this information, as Yelp. The generation of reviews by users is an important part of how opinions are formed when somebody want to make a decision to travel, shop or just want to have dinner.

When a user read a review it helps to choose and increase the probability that this user will make a buying decision. For this reason is important to have good quality reviews, this means: detail, objectivity, readability, etc.

Given that there are millions of reviews, it is important to classified them in helpfulness and unhelpfulness. This would help users to have a clear idea for the purchase decision.

The classic approach is sort reviews by helpfulness as Amazon.com site does. The big drawback of this approach is that it takes time to accumulate feedback. Also, the more recent reviews has a disadvantage until they have enough up vote to increase their helpfulness. Another drawback is that if reviews are sorted by helpfulness then it is unlikely that users will get to see those unrated reviews. For that reason this method tend to starve those recent reviews.

Another approach is to automatically assessing the helpfulness of reviews. Our aim is to build a system that can use this new approach using Machine Learning Algorithms, Natural Language Processing and Sentiment Analysis.

1.3 Why this is a project related to this class
The use of Machine Learning Algorithms has grown drastically in different fields. For the purpose of this work, we are going to use a Classifier Algorithm to classify helpful and unhelpful reviews given a big set of reviews.

1.4 Why other approach is no good
In previous works, as the one called Topic Extraction from Online Reviews for Classification and Recommendation [1] They use a sentiment lexicon based on the work of Hu and Liu, Mining and Summarizing Customer Reviews [2]. However we believe the work of Hu and Liu is a little bit old and is not dynamic as others lexicons. For that reason, we decided to use a more advanced and dynamic sentimental lexicon as SentiWordNet. Besides, the use of Classification Algorithms used in the first work was with JRip, Random Forest and Naïve Bayes, instead we want to see what results come up using others algorithms as Logistic Regression, Complementary Naïve Bayes, Hidden Markov Models and Multilayer Perceptron.

1.5 Why you think your approach is better
The use of a dynamic sentiment lexicon as SentiWordNet add significant value to this work. The new words added to this lexicon improve the chances to have a more accurate result in what sentiment feature respects.

1.6 Statement of the problem
The goal of this paper is to develop a proper algorithm to rate the online reviews based on sentimental value and retrieve the most helpful review to help customer make decision.

1.7 Area or scope of investigation
Our scope will be to classified big quantities of reviews using Machine Learning techniques. More specifically, we are going to use classification algorithms in reviews of some products that are selling in Amazon. In order to do that, we give as input different features extracted from the reviews.

To narrow our project we decided to work on reviews related to Electronics Devices like Digital Cameras, Laptops, Tablets among others. We think that this product are enough to see good results. However, we believe this software can be used for other products and even to classified services reviews as the ones from TripAdvisor or Yelp as well.
2 Theoretical bases and literature review

2.1 Definition of the problem
Most online retail company provide review feature long time back. Nowadays, this function is more complex than previous, customer can rate the product, rate the whole purchasing steps from service to products, rate performance, color and other detailed features of the products or write reviews for newcomers to help other people make correct decision before they purchase. And there are new functions that other customer can even comment on any reviews.

We are interested in analyzing the reviews from online retailer companies to make automatic recommendation for the most helpful review. We will focus on calculate sentimental value to help ranking recommended review.

2.2 Theoretical background of the problem
Online reviews are based on real user’s opinions. People add comments on their favourite features, and they will also mention defective features to help interested buyers. Following example is a sample of review. (figure 1 from Amazon, Canon 6D review). Red words are the negative opinion and green words are positive opinion. In the lines, the negative opinion is not critical compare to the positive opinion. If we can calculate a sentimental value for all these 4 opinions, we will get a helpfulness value that can tell us if this review is helpful to other people.

I read up extensively before buying so I was aware of the fact that it doesn't shoot as many frames per sec as the 7D, and that the autofocus might not be quite as good, but that didn't put me off. And since buying I've barely noticed any diff in the two.

But the difference in picture quality is amazing. In the first week after getting it I had five photos on Flickr's Explore page! Outstanding.

Figure 1. Online review sample with sentimental words.

Based on the sentimental words we found in each review, a more complex algorithm can be developed to calculate the helpfulness value.
2.3 Related research to solve the problem
In Topic Extraction from Online Reviews for Classification and Recommendation [1], two techniques to extract topics were used - bi-grams and single nouns - this was done using a combination of shallow NLP and statistical methods. To analyse the Sentiment value, it use sentiment lexicon as the base for the analysis. It will check the sentiment value for each topic, and gives out the polarization on the topic sentence if there is no 4-word-distance of a sentiment word, otherwise the sentiment will be reversed. The classification part will use all 7 features (including sentiment value) generated from previous steps. Result is checked by evaluating the classification performance, in terms of the area under the ROC curve (AUC) using a 10-fold cross validation. After tuning, AUC > 0.7 can be considered as useful. The recommended result will get highest value from the candidate set.

A different methods to extract topics in a review is mentioned in Identifying Important Product Aspects from Online Consumer Reviews [3]. Reviews online may have pros and cons which will give out more precise estimation on topics. The author use this kind of reviews to filter out unhelpful topics for the product and rank top 10 topics for further recommendation.

2.4 Advantage/disadvantage of those research
Topic extraction used in both papers can automatically generate a topic list. The advantage of automatically generate topic list is the algorithm will be robust in the future. Topic change will happen for different products. It will be expensive to maintain a static topic list. Also auto detecting topics will auto extends the features we are monitoring on any product. A more meaningful feature list can be generated for different product.

Previous papers prefer using static lexicons as the one proposed by Liu and Hu[2]. That lexicon has more than 10 years and we believe is outdated and it is not a proper representation of sentiment for practical usage.

2.5 Your solution to solve this problem
By checking previous works and latest reviews online, we are generating a more adaptive solution to automatic recommend reviews. We use latest review dataset to train the parameters. To calculate a more precise sentiment value, we are seeking SentiWordNet’s latest dataset. Further more, our calculation are based on words to extract features which will be more adaptive for future usage. Parameters can be trained with new dataset to acquire better performance later for special purpose.

2.6 Where your solution different from others
The main difference between this work and previous works is the use of SentiWordNet as a sentiment lexicon. This lexicon provide 155,287 words rated in a positive, negative and neutral manner.
Besides the use of the lexicon we are using different algorithms for classification. We are going to test the classification of helpful or unhelpful reviews with Complementary Naïve Bayes and Logistic Regression.

Also, the number of reviews we will use is larger and more diverse.

2.7 Why your solution is better
We think our solution improve the sentiment lexicon instead of an old lexicon we use a bigger one. Also, it provides a solution in the long term because more words will be add to the lexicon as more people give feedback for words in SentiWordNet.

Another reason is that we are using Mahout which provide a distributed solution. Mahout will be quite important to classified all the reviews when the corpus is big as the one that we have.

3 Hypothesis or goals
First, we want to assume there is no spam in our dataset. We are getting input data from real world, so it is highly possible that online product reviews will have biased opinion or spam. Spam will looks very similar to a helpful review result with different topics (not relative review or the review is a biased opinion since the reviewer want to demonstrate other products is better than current one). We will consider all these kinds of reviews as spam. Spam filter is a big topic to our paper, we will focus on recommendation first unless more time left.

Second, we would like to assume no biased review in the dataset which not relative to current product. Like this one for Canon 60D:

“The sum of both is a lot of weight, I call it a BRICK, but the actual pics are outstanding”

The reviewer is not happy with the weight of the camera, and it is not helpful since most middle level DSLR camera got similar weight. If the reviewer compare to a normal low end camera, it make sense, otherwise, it is not helpful. So this topic is more like a opinion to all middle level DSLR camera. Biased reviews not just for current product but apply to every similar products. It is not easy to dig this kind of topic, and it will branch out another large topic.

Third, we assume there is no acronym, abbreviation, or initialism. According to acronymfinder.com, three are around 700k these words. Words like “IEEE”, “LOL” or “i c u” can be used in reviews since most reviewer are real person not a journalist. The good thing is most of them are noun. But if it is a sentimental word, we might ignore the words depends on our sentimental dataset.

So we want to assume a potential buyer knows what he is looking for. And we would like to tune our algorithm to retrieve the most helpful reviews for the user.

4 Methodology
4.1 Dataset

4.1.1 Amazon reviews

The project will focus on reviews from Amazon.com. There are multiple collected dataset online and we are going to use data from Stanford Network Analysis Platform, also known as SNAP\(^1\). This site provides review datasets for research purpose only. The data span a period of 18 years, including around 35 million reviews up to March 2013. Reviews include product and user information, ratings and a plaintext review.

In this work we are going to use the reviews from Amazon.com related to Electronics devices. We count with 1,241,778 reviews. Each review has the following format:

- product/productId: B00006HAXW
- product/title: Rock Rhythm & Doo Wop: Greatest Early Rock
- product/price: unknown
- review/userId: A1RSDE90N6RSZF
- review/profileName: Joseph M. Kotow
- review/helpfulness: 9/9
- review/score: 5.0
- review/time: 1042502400
- review/summary: Pittsburgh - Home of the OLDIES
- review/text: I have all of the doo wop DVD's and this one is as good or better than the 1st ones. Remember once these performers are gone, we'll never get to see them again. Rhino did an excellent job and if you like or love doo wop and Rock n Roll you'll LOVE this DVD !!

Also another way to collect data is crawling on Amazon. Several scripts provided online, “Amazon reviews downloader and parser” can be used for us to get the latest data for testing.

4.1.2 Sentimental words

Also we use a dataset to search for sentimental words, dataset can be found at SentiWordNet. The dataset is based on WordNet, it is a lexical resource explicitly devised for supporting sentiment classification and opinion mining applications.

4.2 Algorithm

To determine which is the most helpful review we need first to have a dataset of reviews. In this case, we use the SNAP corpus that count with 1,241,778 reviews of Electronic products from Amazon.

\(^1\) [https://snap.stanford.edu/data/web-Amazon.html](https://snap.stanford.edu/data/web-Amazon.html)
The first process over the corpus is to analyse the sentiment of the reviews. Given a review $R_k$ we are going to keep those words that belongs to SentiWordNet (SWN). Let $w_1, w_2, w_3, \ldots, w_n$ the words that belong to $R_k$ and also have a value in SWN we are going to extract as a feature the sum of $w_i$ for all $i$ with $1 \leq i \leq n$ as the sentiment value for that review. The sum will be performed in the 3 dimensions: positive, negative and neutral. In those cases where there are not words from the review that belongs to SWN we assign zero for that feature.

The next process will extract multiple information regarding the review. For example, age that means the number of days since the review was created by the user. Below in table 1 we provide a list of features extracted from the reviews.

<table>
<thead>
<tr>
<th>Type</th>
<th>Feature</th>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>Age</td>
<td>1</td>
<td>The number of days since the review was posted.</td>
</tr>
<tr>
<td>RAT</td>
<td>NormUserRating</td>
<td>1</td>
<td>A normalized rating score obtained by scaling the user’s ranting into the interval $[0,1]$</td>
</tr>
<tr>
<td>SIZE</td>
<td>NumSentence</td>
<td>1</td>
<td>The number of sentences in the review text.</td>
</tr>
<tr>
<td></td>
<td>NumWords</td>
<td>1</td>
<td>The total number of words in the review text.</td>
</tr>
<tr>
<td>SENT</td>
<td>NumPos</td>
<td>3</td>
<td>The sentiment value for the review in positive, negative and neutral.</td>
</tr>
<tr>
<td></td>
<td>SentimentRating</td>
<td>3</td>
<td>The coefficient of NumPos over NormUserRating.</td>
</tr>
<tr>
<td></td>
<td>Density</td>
<td>3</td>
<td>The coefficient of NumPos over NumWords.</td>
</tr>
<tr>
<td>READ</td>
<td>WordsPerSentence</td>
<td>1</td>
<td>The average number of words per sentence.</td>
</tr>
</tbody>
</table>

Table 1 List of features extracted from the reviews.

Finally, for each review we will have a tuple with the feature mentioned in table 1. Those features will be the input for the Classification Algorithm. This will be a supervised algorithm where the algorithm can be trained using the information of the helpfulness value that each review has. After the training we execute a 10-fold cross validation to test how well the model is working.

4.2.1 Algorithm design
4.2.2 Language used

The programming language used will be Java 1.6 or higher. We chose Java as a language because it provide a solid community that create frameworks to improve abstractions and solve very common and recurrent problems. Also, we chose Mahout given his advantage to exploit concurrent computations with MapReduce. However, the last one is not so good when it comes to documentation and stability.

4.2.3 Tools used

We will used Mahout as a framework for Machine Learning. To make the Natural Language Processing of the reviews we are going to use openNLP\(^2\). To setup, build the software and resolve the dependencies we will use Maven\(^3\)

\(^2\) [http://opennlp.apache.org/index.html](http://opennlp.apache.org/index.html)

\(^3\) [http://maven.apache.org/](http://maven.apache.org/)
To the development process we will use Eclipse⁴

4.2.4 Prototype (optional if time permit)
None, time limit.

4.3 Output
Output will be a list of reviews with the order calculated by our recommendation algorithm. Review number or user ID will be used to retrieve the review from dataset.

4.4 Evaluation
Amazon provided a boolean value for other reviewers to up vote the helpfulness of the review. And Amazon use this value as recommendation to order the reviews. The most helpful one will have more up votes. Since it is inputed by real human, it is a ground-truth value to evaluate our result. Meanwhile, it is also a challenging to test our results with this value. Product exist more than one year or more popular will gain enough attraction for us to use for testing. We would like to see if our results can compete with real human’s recommendation. Or we learn from the compared results to enhance our algorithm.

5 Implementation
5.1 Code (refer programming requirements)
The Major part of code is implemented in JAVA. We also did use of Bash Scripting to manipulate files, more specifically the command awk to delete a column from a CSV file or to delete some rows that accomplish a criteria.

Another commands used from Bash were head and tail. You can check the manual of the commands but briefly what they do is to return an specific number of line from the top or the final of a file. We use this commands to create the training and test file for the machine learning algorithm. More information is in the README file of the project.

5.2 Design document and flowchart
General idea to implement the algorithm is try to generate results level by level.

⁴ http://www.eclipse.org/
At the beginning we prepare data for training and testing. Raw data is downloaded from SNAP dataset website. We use JAVA to parse the huge dataset into JAVA object array, calculate features and output into testing file for training and testing.

Then we use Logistic Regression model as our classifier. Mahout implementation uses Stochastic Gradient Descent (SGD) to train the model. And the results of the parameters from training will be used for testing.

5.2.1 Features
The starting point of the whole classification process is to extract features from a raw dataset. The feature extraction is a main part of the process because the use of the predictor variables is a big part in a machine learning algorithm.

To extract the variables we have different levels of abstractions. A record from a file is represented with a FeatureItem Object. The review per se is also represented with an object with the same name.

Here, is important to mention the extensive use of NLP that we did using two frameworks OpenNLP and Stanford Core NLP. We use OpenNLP for all related to tokenization, sentences detection and POS Tagging. On the other hand, we use Stanford Core NLP to lemmatize words from the review to have a more accurate sentimental value. The framework for Stanford gave us a feature that is not yet available in OpenNLP and was needed for the purpose of the project.

The lemmatizer works as follow: given a word like got it returns the infinitive conjugation of the verb, in this case will be get. The reason to use lemmatizer is because in the sentimental dictionary built with SentiWordNet the words are annotated in its infinitive conjugation. For example, the word got has a value in its three dimensions (positive, negative and objective) but the word conjugated in past not.

We distinguish 3 types of features. The first one are those numeric values that are strongly related with the rate of the user, for example: normUserRate. The others are those numeric values related with the sentiment of the review, such as: numSentimental(+,-,o). Furthermore, those values related to a mixture between rate from the user and sentimental value, such as sentimentRating (+,-,o) and sentimentRating (+,-,o). Also, we are using the age, in number of days, since the review was posted and the number of sentences and words.

Below there is a detailed list of the properties of each review: product ID, title, price, user ID, profile name, helpfulness, score, time, summary and text.

The features extracted were:

<table>
<thead>
<tr>
<th>Output for training and testing</th>
<th>Input from raw data</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>time</td>
</tr>
</tbody>
</table>
Table 2. Feature list

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>normUserRate (normalized user rating)</td>
<td>normalized score given by the user</td>
</tr>
<tr>
<td>numSentences (number of sentences)</td>
<td>sentences in review text</td>
</tr>
<tr>
<td>numWords (number of words)</td>
<td>words in review text</td>
</tr>
<tr>
<td>numSentimental+</td>
<td>number of words with positive sentimental</td>
</tr>
<tr>
<td>numSentimental-</td>
<td>number of words with negative sentimental</td>
</tr>
<tr>
<td>numSentimentalo</td>
<td>number of words with no sentimental</td>
</tr>
<tr>
<td>sentimentRating+</td>
<td>positive sentimental over rating</td>
</tr>
<tr>
<td>sentimentRating-</td>
<td>negative sentimental over rating</td>
</tr>
<tr>
<td>sentimentRatingo</td>
<td>no sentimental over rating</td>
</tr>
<tr>
<td>density+</td>
<td>positive sentimental over number of words</td>
</tr>
<tr>
<td>density-</td>
<td>negative sentimental over number of words</td>
</tr>
<tr>
<td>densityo</td>
<td>no sentimental over number of words</td>
</tr>
<tr>
<td>wordsPerSentence</td>
<td>normalized words per sentence</td>
</tr>
</tbody>
</table>

5.2.3 Label useful review

For this project we considered that a review is *useful* if for a certain product has the highest value of upvotes. For example, given a product A and a review with the highest helpfulness value for A is 9/10, we considered that the number of upvotes is 9 and those reviews that has this value are “useful” as well. Otherwise, the helpfulness is set as “unuseful”. This usefulness indicator will be our target variable. The SGD algorithm when is trained will learn from this variable.

5.2.4 Training and testing

To test and train the algorithm we use two approaches. The first one was to use a 10-fold cross validations and the other one was to split the feature file in 90% for training and 10% for testing. We count with more than 1 million reviews related to the electronic area so was quite flexible to choose training and testing data.

5.2.5 Logistic regression

Logistic regression is a model used for prediction of the probability of occurrence of an event. It makes use of several predictor variables that may be either numerical or categories. Mahout use Stochastic Gradient Descent (SGD) as Logistic regression model. We are interested to get
the useful reviews from the list and see how good can predict this algorithm the usefulness of a review.

5.2.6 Flow chart
Flowchart of how we extract features from raw data set. (Figure 3)

Flowchart of how we train and test our features. Main code is implement with SGD in Mahout. Further exploration using Weka with same data set.
6 Data analysis and discussion

6.1 Output generation
From the source code you could see that we mainly evaluated our experiments with the AUC value. AUC is the value under the curve and means how good the result beyond pure random decision like throwing a coin. For have a result that means something is necessary to have a value above 0.5, which means that the predictions are equal to decided useful or unuseful.

Another output from the experiments that we executed were the confusion matrix. The confusion matrix indicates the expected vs actual classification numbers. For a good classifier, we should expect the numbers along the diagonal to predominate. The matrix in this case consist in a 2x2 matrix. Where the rows and the columns are useful and unuseful. For example a matrix like:

<table>
<thead>
<tr>
<th></th>
<th>useful</th>
<th>unuseful</th>
</tr>
</thead>
<tbody>
<tr>
<td>useful</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>unuseful</td>
<td>0</td>
<td>5000</td>
</tr>
</tbody>
</table>

A perfect Confusion Matrix. From 5500 reviews all the predictions were 100% accurate.

We implemented with Mahout the SGD algorithm to run the AUC. From the total features extracted 992332 we used the 90% for training and the 10% for testing purpose. The result for the AUC was 0.5919.
As Mahout have implemented just a few classifiers and the documentation is still poor we had a very hard time to implement the SGD algorithm. Some of the algorithms that Mahout has are not implemented to run in parallel. Given the limited time to finish the project and the hard time that we had with Mahout we wanted to try the features extracted with others classification algorithms in order to have more result to compare. For this purpose we use Weka\textsuperscript{5} which is a tool very straightforward to run machine learning algorithms.

We tested the features extracted with Weka (Waikato Environment for Knowledge Analysis) This is a non-parallel implementation to run in a single computer environment. With Weka we ran this algorithms: Random Forest, Decision Tree (J48 implementation), Complementary Naïve Bayes, and Multilayer Perceptron.

The results are below:

**Random Forest**

Instances: 992332  
Test mode: 4-fold cross-validation

--- Results ---

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>927520</th>
<th>93.4687 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>64812</td>
<td>6.5313 %</td>
</tr>
<tr>
<td>Kappa statistic</td>
<td>0.0808</td>
<td></td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.118</td>
<td></td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>992332</td>
<td></td>
</tr>
</tbody>
</table>

--- Detailed Accuracy By Class ---

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.053</td>
<td>0.005</td>
<td>0.429</td>
<td>0.053</td>
<td>0.094</td>
<td>0.578</td>
<td>useful</td>
</tr>
<tr>
<td>0.995</td>
<td>0.947</td>
<td>0.939</td>
<td>0.995</td>
<td>0.966</td>
<td>0.578</td>
<td>useless</td>
</tr>
</tbody>
</table>

--- Confusion Matrix ---

```
   a   b <--- classified as
3349   60357 |   a = useful
4455   924171 |   b = useless
```

**Decision Tree (J48)**

Instances: 992332  
Test mode: 4-fold cross-validation

\textsuperscript{5} [http://www.cs.waikato.ac.nz/ml/weka/](http://www.cs.waikato.ac.nz/ml/weka/)
Results

Correctly Classified Instances 928626 93.5802 %
Incorrectly Classified Instances 63706 6.4198 %
Kappa statistic 0
Mean absolute error 0.1202
Total Number of Instances 992332

Detailed Accuracy By Class

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>useful</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.936</td>
<td>1</td>
<td>0.967</td>
<td>0.5</td>
<td>unuseful</td>
</tr>
</tbody>
</table>

Confusion Matrix

```
a  b  <-- classified as
0  63706 | a = useful
0 928626 | b = unuseful
```

Complementary Naïve Bayes
Instances: 992332
Test mode: 10-fold cross-validation

Results

Correctly Classified Instances 649792 65.4813 %
Incorrectly Classified Instances 342540 34.5187 %
Kappa statistic 0.0109
Mean absolute error 0.3452
Total Number of Instances 992332

Detailed Accuracy By Class

```
<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.356</td>
<td>0.325</td>
<td>0.07</td>
<td>0.356</td>
<td>0.117</td>
<td>0.516</td>
<td>useful</td>
</tr>
<tr>
<td>0.675</td>
<td>0.644</td>
<td>0.939</td>
<td>0.675</td>
<td>0.785</td>
<td>0.516</td>
<td>unuseful</td>
</tr>
</tbody>
</table>
```

Confusion Matrix

```
a  b  <-- classified as
22697 41009 | a = useful
301531 627095 | b = unuseful
```
**Multilayer Perceptron**

Instances: 992332  
Test mode: 10-fold cross-validation

--- Results ---

Correctly Classified Instances 928626 93.5802 %  
Incorrectly Classified Instances 63706 6.4198 %  
Kappa statistic 0  
Mean absolute error 0.1178  
Total Number of Instances 992332

--- Detailed Accuracy By Class ---

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.616</td>
<td>useful</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.936</td>
<td>1</td>
<td>0.967</td>
<td>0.616</td>
<td>unuseful</td>
</tr>
</tbody>
</table>

--- Confusion Matrix ---

a b <-- classified as  
0 63706 | a = useful  
0 928626 | b = unuseful

### 6.2 Output analysis

From our Mahout implementation of SGD we had a 0.59 AUC which we consider is a good starting point. It means that there is information in the feature file that is useful to determine whether a review is *useful* or *unuseful*. We have to keep in mind that we are not using complicated features as the original paper used. They used for example, the number of complex words in the review text, the average number of syllables per word, the number of years of formal education required to understand the review, a standard readability score on a scale from 1 (30 - very difficult) to 100 (70 - easy), among others. Those features let see how good the review is in term of readability, understandability and complexity.

Another important difference between the original paper and our paper is the extraction of topics from a review and the generation of predictor variables with that information.

Our project showed that there is information in our features and the algorithm prove that can decided with 10% more accurate than throwing a coin if a review is useful or not. Also, we can mention that classify correctly almost 93% of the reviews although the accuracy looks
impressive is necessary clarify that most of the reviews are *unuseful* so, the algorithm is better trained in this direction.

6.3 Compare output against hypothesis

Our hypothesis was that our result wouldn’t be as good as the ones obtained in the original paper, but however we could have information to predict if a review was useful or not.

Here, we have to say that our project meet our expectations. There was a lot of work to extract the features from the reviews and also to implement the SGD algorithm. Finally, we could see that the result is 0.6 for the AUC which is closer to the 0.7 value obtained in the original paper.

6.4 Abnormal case explanation (the most important task)

6.4.1 Empty input from raw dataset
Some of the review from raw dataset empty, we are checking parsed value before generating training/test data. Extra script provided if input data have empty value inside.

6.4.2 Competition of reviews
There is high priority multiple reviews for one product are all useful, and we are labeling it with better rating for each product, which means our automatic labeling methods may not label it correctly. From training or testing side, it is not very accurate to get the dreaming result.

6.4.3 Low quality of review
It is possible that in one product, non-review are good. The possibility might be the product is not good, and no one want to comment. In this situation, we cannot find any useful review with this product. And our training/testing data may get affected and finally affect our test results. We would like to take this situation out from our testing/training data.

7 Conclusion and recommendations

7.1 Summary and conclusions

To sum up, we processed almost 1 million reviews related to electronics products from Amazon site. This result in a large feature file integrated by predictor variables calculated with Natural Language Processing techniques and Sentimental evaluations of different words. Finally, we implemented a classifier based in Stochastic Gradient Descent and measure the performance to distinguish between a *useful* and *unuseful* review. Furthermore, we measure the performance of others classifiers implemented in Weka.

In conclusion, the set of features that we decided to use show that there is useful information to classify reviews. Although, the accuracy is not significant with just a few features we are 10% behind of the original work.
7.2 Recommendations for future studies

As things that somebody can improve in a future work we think that there is a couple of decision that can change the results drastically.

First, use a more evenly distributed reviews in terms of ratio useful - unuseful. As we had a little more than 1 million reviews, we were tempted to use all the reviews. As a matter of fact, we did it but looking backward we believe that it would be better had a closer 50% of useful reviews and 50% of unuseful reviews. Too much unuseful reviews can bias the learning process of the algorithm and give wrong answers to those reviews that were useful.

Second, to process with NLP the text of the review use only Stanford Core NLP. We used a mix between Core NLP and OpenNLP because the last one doesn't have a lemmatizer. However, the Core NLP is quite simple to use and less cumbersome. It also brings a sentimental analyser. If at the beginning of the project we would use just Core NLP the code would be more readable and we didn't have to use SentiWordNet and OpenNLP.

Third, instead of use all the products in the electronic spectrum we would may use just some products as they use in the original paper. We believe that in that way we could have a more accurate classifier given that there should be more correlation between the reviews.

Forth and finally, use LDA algorithm to extract topics of the reviews and use those topics to add new features as in the original paper. This, could be done using Mahout.

8 Bibliography


9 Appendices
9.1 Program flowchart
Flowchart of how we extract features from raw data set. (Figure 3)

Figure 3. Generated features

Flowchart of how we train and test our features. Main code is implement with SGD in Mahout. Further exploration using Weka with same data set.

Figure 4. Training/Testing process
9.2 Program source code with documentation
Source code and documents are maintained on https://github.com/matias2681/ml-term-project. Please refer to README file in the repository for how to setup and run it. There might be some difference for different JVM. Known issue is openJDK may cause eclipse complain @Override for several positions in source code, please remove or comment them out before start running eclipse.

9.3 Input/output listing

9.3.1 Input
Under “src/main/resources”
1. “SentiWordNet_3.0.0_20130122.txt” - dataset for calculating sentiment value.
2. “en-pos-perceptron.bin”, “en-sent.bin” and “en-token.bin” - dataset for openNLP to detect POS tag of words.
3. “Electronics.txt” - raw dataset of reviews from SNAP. (Please download separately)
4. “test_Arts.txt” - sample raw dataset

9.3.2 Output
1. “features.csv” - features extracted from dataset.
2. “features-no-nan.csv” - remove empty value from “features.csv”.
3. “train.csv” - splitted data from “features.csv” for training.
4. “test.csv” - splitted data from “features.csv” for testing.

9.4 Other related material
No other related material needed.