AUTOMATICALLY PREDICTING THE HELPFULNESS OF ONLINE REVIEWS

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Yadong Zhang

SPRING
2014

AUTOMATICALLY PREDICTING THE HELPFULNESS OF ONLINE REVIEWS

A Project

by

Yadong Zhang

Approved by:

_____, Committee Chair
Dr. Du Zhang

_____, Second Reader
Dr. V. Scott Gordon

_____
Date

Student: Yadong Zhang

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

_____, Graduate Coordinator _____
Dr. Nikrouz Faroughi Date

Department of Computer Science

Abstract

of

AUTOMATICALLY PREDICTING THE HELPFULNESS OF ONLINE REVIEWS

by

Yadong Zhang

Online shopping websites provide platforms for consumers to review products and share opinions. Online reviews provided by the previous consumers are major information source for both consumers and marketers. However, a large number of reviews for a product can make it impossible for readers to read through all the reviews in order to collect information. So it is important to classify and rank the reviews base on their helpfulness to make them easily accessed by readers. It will help consumers finish their information search and decision making more easily. It will also be valuable for product manufactures or retailers to get informative and meaningful consumer feedbacks.

Due to the lack of editorial and quality control, the reviews of product dramatically vary on quality: from very helpful to useless and even spam-like. The helpfulness of reviews is currently assessed manually by the votings from readers. This project experiments with data collected from Amazon through using a supervised machine learning approach to

investigate the task of predicting the helpfulness of online reviews. It discusses the determinants of the helpfulness of online reviews. Eventually it proposes a model which is used to automatically predict the helpfulness of online reviews.

_____, Committee Chair
Dr. Du Zhang

_____
Date

ACKNOWLEDGEMENTS

I take this opportunity to thank Dr. Zhang for his guidance and support, his suggestions and ideas were very essential for the successful completion of this project. I would also thank my parents for their endless love and support throughout my life.

TABLE OF CONTENTS

Page

LIST OF TABLES

Tables Page

LIST OF FIGURES

Figures Page

Chapter 1

INTRODUCTION

Online reviews are a type of product information created by the users based on personal usage experience. Online shopping websites provide platforms for consumers to review products and share opinions. Many consumers rely on online reviews for firsthand information to make purchase decisions. However, a large number of reviews for just one single product have made it impossible for consumers to read through all the reviews and evaluate the true quality of a product. Besides, the quality and the helpfulness of each review also vary. The abundance of the reviews and their uneven quality make it hard for potential consumers to distinguish between useful and useless reviews.

In this project, we focus on modeling the helpfulness of consumer reviews. According to the helpfulness estimation, our regression model assigns each review to one of the five target classes -Extremely Helpful, Very Helpful, Somewhat Helpful, Not very Helpful, Not at all Helpful. The remainder of this paper is organized as following. Chapter 2 introduces the background of this project. Chapter 3 discusses related work. Chapter 4 and Chapter 5 present our proposed approach. The experimentation and evaluation are shown in Chapter 6. Finally Chapter 7 concludes the paper and outlines some future research directions.

Chapter 2

BACKGROUND

Online reviews are a type of product information created by the user based on personal experience. A review can cover: the product features, the shipping speed, the feedback of the customer service and the comparison to other products. Online shopping websites provide a platform for consumers to review products and share opinions. For instance, Amazon.com has one of the most popular forums for user generated reviews. To start to review a product, a reviewer is first asked to use number of stars to indicate the overall product assessment. This star scale is from one to five, with five stars being the best -"I love it" and one being the worst -"I hate it". After the rating, reviewers are asked to provide detailed information about the product and the explanation of the rating. Reviewers are suggested to explain why they like or disliked the product; compare the product to similar products; identify specific attributes and whether the product meets their expectations. Here are two examples of review from Amazon:

- “In this review, I will focus on a feature that has not been covered well by other reviewers. There a few things you need to do to make it work. Firstly you have to enable the HDMI-CEC feature on your TV. HDMI-CEC is marketed under different names by different manufacturers…” (79 of 93 people found this review helpful)
- “I don't have it, and won’t be getting it. It appears to be another canned apps streaming device. I had Google TV. Content was limited on google tv, and

streaming from chrome tabs is in beta, and reported not to work very well…"
(1 of 76 people found this review helpful)

A helpful review likely possessed the following characteristics:

It provides a large quantity of detailed information about the product. For instance, in the first review in the above example, it not only gave information about how to set up the device, but also focused on the opinions that are different from other product description or reviews. Also, the sentence structure is clear and contains less spelling or grammar errors. The providers of these reviews tend to be active and received positive feedback from other consumers. In comparison, the less helpful reviews provide less information and add no additional value to the reader. Specifically in the second review above, the writer of that review mentioned that he didn't have experience with using the product he was reviewing, thus the information he provided was limited and vague.

Due to the lack of editorial and quality control, the reviews of product vary dramatically on quality: from very helpful to useless and even spam-like. This is a well-known problem that some leading websites have censor systems to filter the spam reviews.
A popular example of such system is the one used by Yelp. They have a system to distinguish between genuine reviews which are posted by the real costumers and fake reviews which are posted by the business owners to boost their reputations[1] . Another example is "helpfulness vote" function provided by online shopping websites like Amazon[2], Sephora[3], etc. Consumers can vote a review as being "helpful" or "not helpful" after they read the review (see Figure 1 for an example). In most of the websites,

reviews can be ranked based on the accumulated votes. However, research [4] showed that this voting function might have three kinds of biases: (1) imbalance vote bias: users tend to value others' opinions positively rather than negatively, (2) winner circle bias: the more votes a review gains, the more default authority it would appear to the readers, which in turn will influence the objectivity of the readers' votes, and (3) early bird bias: the earlier a review is posted, the more votes it will get. The latest published review will always be the least voted one. Since consumers are not obligated to vote such reviews, only a small number of reviews eventually receive sufficient votes.

Figure 1: Example of an Amazon review

As shown in table 2, the metadata of the dataset we used in this project also proved the existence of the bias of voting system. We collected 94,560 reviews from Amazon.com. Only 36% reviews from have votes. Among those 34,878 voted reviews, there're over 75% reviews that have only received less than three helpfulness votes. Figure 2 shows the distribution of the count of the helpfulness vote in our dataset. As a conclusion, newly written reviews and reviews with fewer votes cannot be evaluated fairly in such voting system. Therefore, a model to automatically predict the helpfulness of online reviews is needed. It will be great help for consumers as well as marketers.

Table 1: The ratio of reviews with votes

Category	Review#	Review With Votes#	Ratio
Beauty	15829	6625	41.85%
Electronics	41176	14789	35.92%
Home & Kitchen	12695	4812	37.90%
Health & Personal Care	14113	6129	43.43%
Grocery & Gourmet Food	10747	2523	23.48%
Total	94560	34878	36.88%

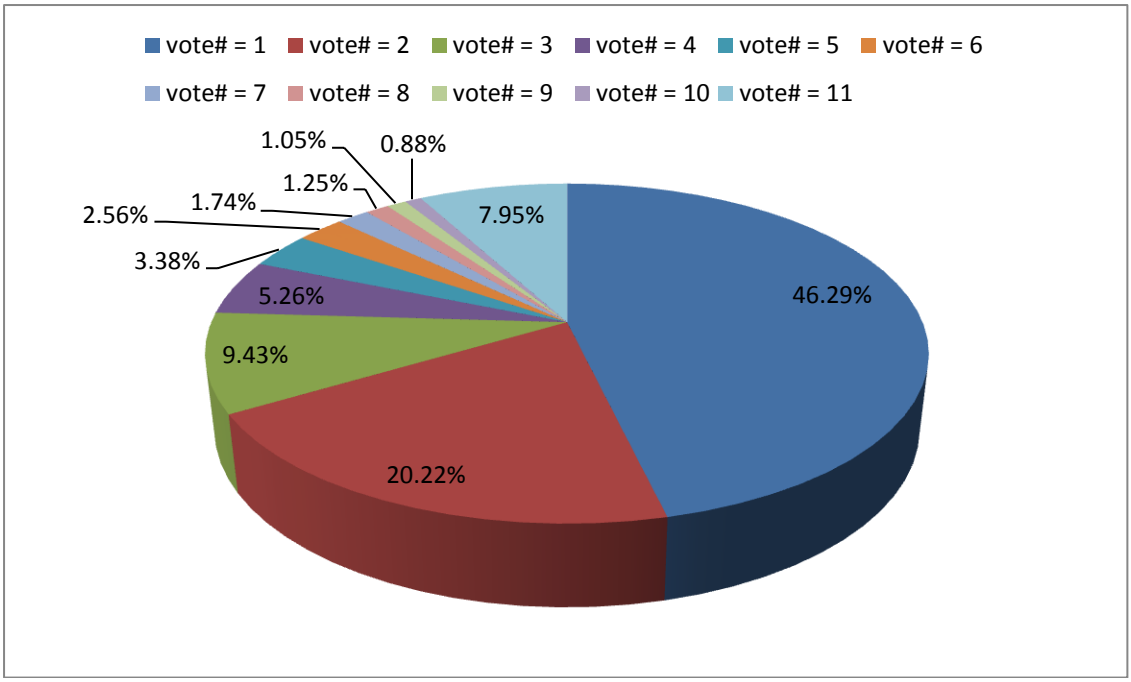

Figure 2: Distribution of review helpfulness vote count

Chapter 3

RELATED WORKS

Automatically evaluating the quality of online reviews has gradually attracted more and more attention in recent years. Most previous works[5] [6] [7] [8] [9] have focused on automatically predicting the quality (helpfulness or usefulness) of reviews by using a set of observed textual or social features. Textual features include features that are based on text statistics such as length of the review, the average length of each sentence, percentage of nouns or adjectives, etc. Social features are information extracted from the reviewer's social context, such as the number of the reviews posted by this author, the past average rating for this author, etc. Most of the current works have formulated the problem of evaluating review quality as a classification problem or a regression problem using observed features. In[10][11][12], authors proposed classification-based approaches supervised by the annotated ground truth, which is manually determined by the feedback from other users regarding their opinion on whether they think the review is helpful or not. Instead of classifying reviews as helpful or unhelpful, some works also considered estimating the helpfulness of reviews by using regression models to generate a quality rating for each review. S. Kim and P. Pantel[6] found that the most useful features to determine the helpfulness of online reviews were the length of the review, unigrams of the review and the rating of the product. They trained a SVM regression system to learn the helpfulness function. To solve the high dimensionality issue of the bag-of-word(BOW) model which is commonly used in mining the textual features, more researchers[13] also applied different dimension reduction techniques to remove

irrelevant, redundant and noisy features in the reviews. Y. Lu[14] demonstrated that prediction accuracy of text-based classifiers can be greatly improved by using regularization on social context. However, there are not enough studies focusing on the social context such as the information about the product and reviewers. In addition, most existing studies are focusing on reviews for specific categories of product. An approach to find a general model, which is based on the review context features and social context features, and is used for predicting the helpfulness of reviews for products from any categories is needed.

Chapter 4

DATA PREPARATION

Data set

In this project, review data was collected from Amazon.com[2]. The reason we chose Amazon.com is that it has largest number of posted reviews, which ensure us to get a large sample. In addition, a recent study [15] shows that the data on Amazon.com, compared to other popular retail website such as Bestbuy.com, Walmart.com, Dell.com, and Target.com, did not appear to be heavily censored, which helps prevent systemic data bias. Systemic data bias in this case means the tendency that the reviews are censored by the website to appear more positive than they were intended. The reviews from Amazon.com appeared varied level of experience, including extremely negative ones. We crawled 94,560 product reviews from Amazon.com. The products are from best sellers' list of following categories:

- Beauty
- Electronics
- Home & Kitchen
- Health & Personal Care
- Grocery & Gourmet Food

For each record in our dataset, we gathered information on the following variables:

1. Consumer rating: the number of stars a reviewer gives as the overall assessment of the product.

2. Name: the user name of the person who wrote the review.
3. Product review: the full text of the product review.
4. Date: the date on which the review was posted.
5. Reviewers' ranking: the rank of the reviewer, which is posted on Amazon reviewers' profile page. It is determined by the overall helpfulness of all their reviews, factoring in the number of reviews they have written[16].
6. Reviewers' helpful vote percentage: the helpful percentage of the votes received on reviewers' previous reviews.
7. Review number: the number of prior reviews a review's author has written
8. Helpful vote: the number of consumers who have voted the review as helpful.
9. Total vote: the number of consumers who have voted the review.

Data collection

We use an open source web scraping framework Scrapy[17] to crawl data from Amazon.com. The workflow of the data collection is as following:

1. Define the data items. See the definition of data field in Figure 3.

```
from scrapy.item import Item, Field

class AmazonItem(Item):

    name = Field()
    vote = Field()
    voteTotal = Field()
    rating = Field()
    date = Field()
    review = Field()

    reviewerRanking = Field()
    helpfulPercentage = Field()
    reviewNo = Field()
```

Figure 3: Definition of data fields in data crawling

2. Write Spider to extract the data (see appendix for the source code: products.py).

3. Run the Spider to extract the data. See Figure 4 for the command.

```
crapy\amazon\amazon>scrapy crawl products -o data.csv -t csv
```

Figure 4: Command to run Scrapy project

4. Review crawled data. Figure 5 shows a screen shot of the CSV data file.

	A	B	C	D	E	F	G	H	I
1	rating	name	review	date	reviewer Ranking	helpful Percentag	vote	reviewNo	voteTotal
2	5	1	If you need a great e-reader, then this fits the bill. I had an old keyboard Kindle which was just fine but the improvements and the smart lighting system on the paperwhite make this far superior. It's not a tablet - it's just for reading books and whatnot and for what it is designed ffor.	20-Dec-13	813140	74	0	90	1
3	1	321	The lable says it only works for vertext hair loss. Did not know that until I purchased the item. I did not find mentioned anywhere, not on product description, not on their website. Now I cannot even return it, says some hazardous itme policy. What the heck?	15-Sep-10	19795665	48	10	1	21
4	4	829	I was very skeptical even after reading all the positive reviews, but this flat iron didn't disappoint. My hair is silky straight. I love it.	18-Sep-12	11526289	100	1	1	1

Figure 5: Screen shot of crawled data file

Data pre-processing

Handling missing values

When we examined the original data file from web crawling, we found several missing values in two fields: Name and Reviewers' ranking. For the missing attributes in the Name field, we filled them all with string: "NAME". We removed the rows in which Reviewers' ranking are missing.

Data cleaning

We removed duplicated reviews in the data set. We eliminated multiple reviews for a single user-item pair. These duplicates could be caused by the errors in data transmission on the internet, due to which same review might have appeared multiple times. Another possible cause could be that the user's opinion of a product might have changed over time. In the case of multiple reviews over time, we retained the most recent one and discarded the rest of them.

We removed special characters which appeared in review text, such as "\, /, :". Those characters could cause errors in the text mining process.

Data reduction

To ensure the robustness of the regression model and have a more accurate estimate for the helpfulness function, we only use the reviews which received at least ten votes. Reviews where less than ten users have voted the review as helpful or unhelpful were filtered out.

To avoid the potential spam reviewer, we filtered out those reviews written by reviewers who only have published one review and the rating they gave were extreme, such as "1" or "5". Research[18] has found that fake reviews tend to be more extreme than that of legitimate reviews. It also proposed a spam filtering model that contains controls for the number of prior reviews which a review's author has written. According to their study, reviewers with fewer published reviews are more suspicious as a spammer, especially those who have published only one review.

Data transformation

To prepare the Date field for the regression model, we transformed the Date field (data format: "13-Dec-13") to a numeric value which equals to the number of days between the date when the review was posted and the current date.

At Amazon.com, consumers publish their reviews about products online after they have purchased or used the products. Once the reviews have been posted, readers can vote a review as a "Helpful" or "Not Helpful". For a review, its helpfulness can be calculated as the ratio P of the number of consumers who have voted r as "Helpful" to the total number of consumers who have voted r [12]:

$$P = \frac{h}{h + h}$$

where h is the number of people that will find a review helpful and h is the number of people that will find the review unhelpful. We model the prediction of review helpfulness as a classification problem. In order to classify the reviews, thresholds are needed to

assign each review from our dataset to the target class. We define the threshold of the target classes as following:

Table 2: Classification from the percentage of helpful votes

P value	Classes (Label)
greater than 80%	Extremely Helpful
60% to 80%	Very Helpful
40% to 60%	Somewhat Helpful
20% to 40%	Not very Helpful
Less than 20%	Not at all Helpful

After applying the threshold, we removed vote and voteTotal column from our data set. We replaced them with a label column: comment.

Features extraction

One of the goals in this project is to find out the determinants of the helpfulness of online reviews. We extracted features from the following three categories: features from reviews, features from reviewer and features from metadata.

Features from review

Structural features:

Structure features describes structure and the formatting of the reviews. We experimented with the following structural features:

sentence.count: The number of sentence.

token.count : The total number of tokens of a review. It described the length of a review.

token.per.sentence: The average number of tokens in a sentence. It described average sentence length of a review.

We used the components of Sentence Detector and Tokenizer from Apache OpenNLP library[19] to complete the structural feature extraction (See the Java source code in appendix: TextMining.java).

Syntactic features:

The syntactic features capture the linguistic property of the review. We extracted the following features:

noun.percentage: The percentage of tokens which are nouns

verb.percentage: The percentage of tokens which are verbs,

adjective.percentage: The percentage of tokens which are adjectives

adverb.percentage: The percentage of tokens which are adverbs.

We used POSModel Class from Apache OpenNLP library [19] to complete this task. POS(Part of Speech) Tagger marks tokens with their corresponding word type based on the token itself and the context of the token. The OpenNLP POS Tagger uses a probability model to predict the correct pos tag out of the tag set. The tag set used by this model is Penn Treebank tag set[20] (See the Java source code in appendix: TextMining.java).

Semantic Features

The positive or negative sentiment of words in the review is a good indication about the strength of the opinion of the reviewer. We used tm.plugin.sentiment package in R [21] to extract the following five features: *polarity*, *subjectivity*, *pos_refs_per_ref*,

neg_refs_per_ref, *senti_diffs_per_ref*. (See the R source code in appendix: sentiment.r)

Figure 6 shows how each semantic feature was calculated:

$$polarity = \frac{p - n}{p + n}$$

$$subjectivity = \frac{n + p}{N}$$

$$pos\_refs\_per\_ref = \frac{p}{N}$$

$$neg\_refs\_per\_ref = \frac{n}{N}$$

$$senti\_diffs\_per\_ref = \frac{p - n}{N} \quad [22]$$

Figure 6: Sentiment features calculation

p: Number of positive words

n: Number of negative words

N: Number of words

Readability

We assumed that reviews which are highly readable tend to be more helpful. In the contrary, reviews with multiple grammatical error and misspelled words are less helpful for users. We use LanguageTool Java API [23] to implement the language and grammar check for this task. We use the fraction of the number of errors in the review text and the number of sentence as the value of feature *error.per.sentence.* (See Java source code in appendix: TextMining.java)

Features from reviewer

We collected the information of the following features from reviewers' profile page:

reviewerRanking: The rank of the reviewer.

helpfulPercentage: The helpful percentage of the votes received on reviewers' previous reviews.

reviewNo: The number of prior reviews a review's author has written.

The reason we selected these features lies in the fact that people with better reputations in an e-commerce community tend to provide more influential discussions making their reviews more helpful. Reviewers with higher ranking are expected to have better reputation. Since our goal is to predict the helpfulness of a review, we wanted to examine whether the past history of a reviewer can be used to predict the helpfulness of the future review written by the same reviewer. We supposed that reviews from the same author will have similar quality. A reviewer that writes helpful high quality reviews is likely to continue writing good reviews.

Features from meta data

Meta-data features are independent of the review text. We selected the following two features:

rating: Consumer rating.

age: The number of days since the review was posted.

As studies such as [6] have shown consumer rating is one of the key features which can be used to determine the helpfulness of online reviews. We would also like to include it into our selected features. In addition, we suspected that the time stamp of when review has been posted also effect the helpfulness of certain reviews. Older reviews might be more helpful since they might exhaust the aspects worth reviewing, which left less perspective for newer reviewer to add on, instead of repeating the similar reviews.

Chapter 5

DATA MINING

The Support Vector Machine (SVM) is among the best supervised machine learning algorithms. SVM constructs a decision boundary with the largest possible distance to data points, which can be used for classification or regression.

We deployed the John Platt's sequential minimal optimization (SMO) algorithm[24] for training a support vector classifier. This model solves multi-class problem by using pairwise classification. The predicted probabilities are coupled using Hastie and Tibshirani's pairwise coupling method[25]. We tested on the development sets various kernels including Normalized Polynomial Kernel , Polynomial Kernel, Pre-computed Kernel Matrix Kernel, Pearson VII function-based universal Kernel and RBF Kernel [26]. The best performing kernel was Polynomial Kernel (See figure 7 for the configuration and kernel information of our model). We applied this model to our training set. The complete classifier output can be found in appendix: classifier output.

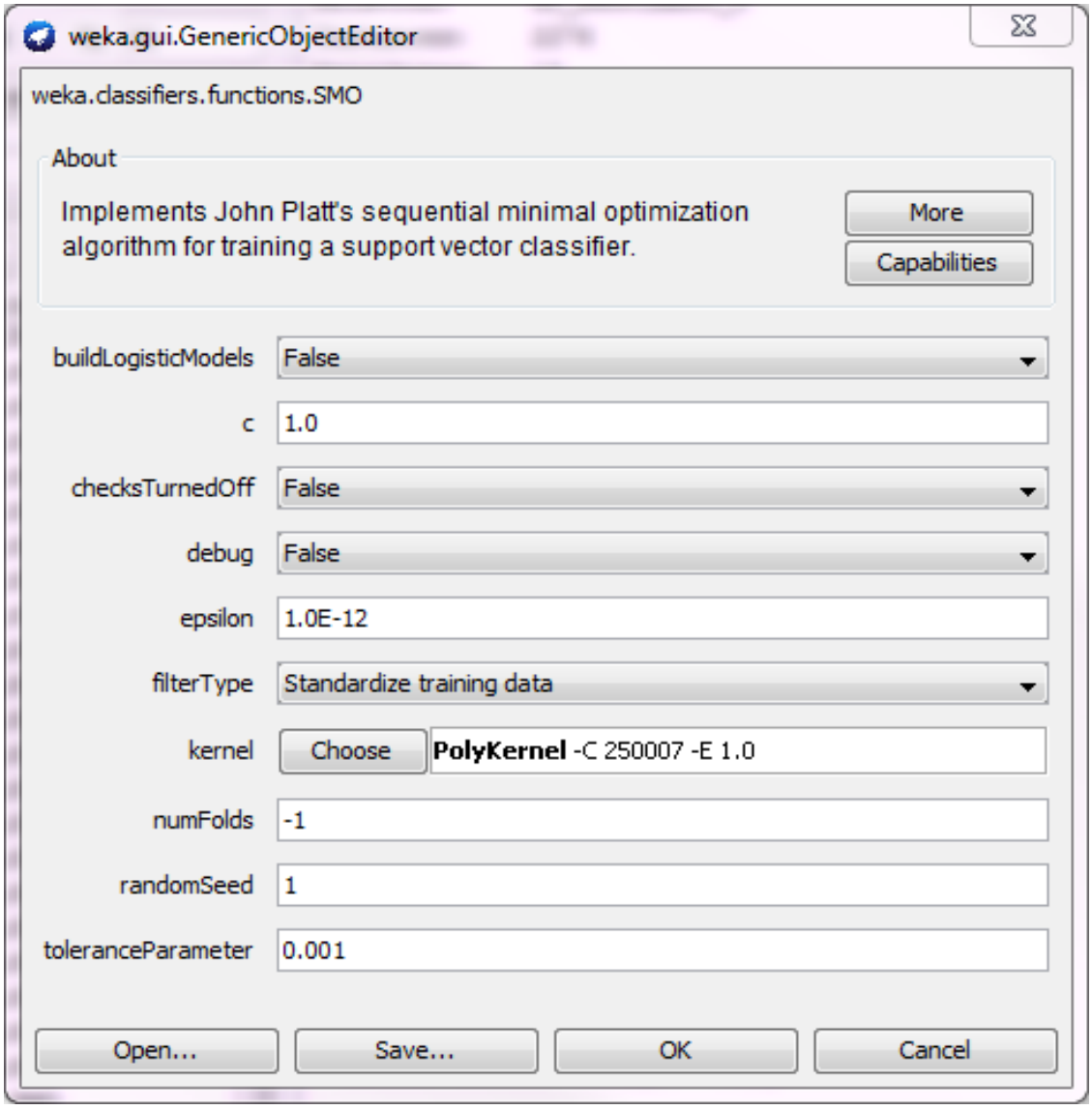

Figure 7: Configuration of SMO Model

Chapter 6

EVALUATION AND DISCUSSION

The original dataset was split into training set and testing set, where 66% of the data was used as training set and remainder was used for testing. The testing set was used to evaluate the effectiveness of the proposed model. First we used all 18 features (shown in table 3) as the input of the SVM model. We got an accuracy of 66.6667 % for correctly classified the test cases. This model has the best performance of precision of 73.3% on predicting the "Extremely Helpful" cases. By looking at the detailed accuracy by class, we found out this model has better performance on predicting extreme cases, such as Extremely Helpful and Not at all Helpful. The accuracy of this model decreased when predicting more moderate cases, such as Somewhat Helpful and Not very Helpful. Please see figure 8 as complete evaluation summary.

Table 3: Feature list

Features
sentence.count
token.count
token.per.sentence
noun.percentage
verb.percentage
adjective.percentage
adverb.percentage
polarity
subjectivity
pos_refs_per_ref
neg_refs_per_ref
senti_diffs_per_ref
error.per.sentence
reviewerRanking
helpfulPercentage
reviewNo
rating
age

```
=== Summary ===

Correctly Classified Instances         516               66.6667 %
Incorrectly Classified Instances       258               33.3333 %
Kappa statistic                          0.5572
Mean absolute error                      0.2599
Root mean squared error                  0.3459
Relative absolute error                 85.1215 %
Root relative squared error             88.4218 %
Coverage of cases (0.95 level)          98.5788 %
Mean rel. region size (0.95 level)      80.491  %
Total Number of Instances              774

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.845    0.091    0.687      0.845    0.758      0.699    0.915     0.655     Not at all Helpful
                 0.273    0.036    0.529      0.273    0.360      0.319    0.813     0.347     Not very Helpful
                 0.843    0.144    0.733      0.843    0.784      0.676    0.893     0.704     Extremely Helpful
                 0.636    0.123    0.652      0.636    0.644      0.517    0.831     0.558     Very Helpful
                 0.329    0.044    0.436      0.329    0.375      0.324    0.699     0.255     Somewhat Helpful
Weighted Avg.    0.667    0.105    0.649      0.667    0.649      0.559    0.852     0.568
```

Figure 8: Evaluation summary

We experimented with various combinations of feature sets. Table 3 shows the selected feature combinations and their corresponding performance.

Table 4: Accuracy from different feature combinations

Accuracy	Rating	Helpful Percentage	Adjective Percentage	Adverb Percentage	Error per Sentence	Review Number	Subjectivity	Token Count	Reviewer Ranking	Age
68.7339%	✓	✓				✓				
68.6047%	✓	✓	✓		✓	✓				
68.4755%	✓	✓	✓		✓	✓	✓			
68.3463%	✓	✓			✓	✓				
68.3463%	✓	✓	✓		✓	✓			✓	✓
68.3463%	✓	✓	✓			✓				
68.2171%	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
68.0879%	✓	✓	✓	✓	✓	✓	✓			
67.9587%	✓	✓	✓			✓			✓	✓
67.8295%	✓	✓	✓	✓	✓	✓				
67.7003%	✓	✓	✓		✓	✓		✓		
67.5711%	✓	✓			✓	✓		✓		
67.4419%	✓	✓	✓	✓	✓	✓	✓	✓		✓
67.3127%	✓	✓	✓		✓					
67.3127%	✓	✓	✓		✓	✓	✓	✓		
67.3127%	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Results show that the highest performing feature combination consisted of the *rating*, the *helpfulPercentage* and the *reviewNo* feature sets, which has a precision of 68.7339%. As we suspected, the reputation of the reviewers is the major determinant of the helpfulness of online reviews. The feature *helpfulPercentage* is an indication of the quality of reviewer's previous reviews and the feature *reviewNo* shows the quantity of their previous reviews. The combination of these features reveal the reputation of a reviewer. Therefore, those two features significantly contribute to higher performance in our model. We found that syntactical features (*noun.percentage, verb.percentage, adverb.percentage*) and semantic features (*polarity, pos_refs_per_ref, neg_refs_per_ref, senti_diffs_per_ref*) did not show any significant improvement in system performance. We found that *rating* is also one of the most influential features on predicting helpfulness of reviews. However, this might be caused by the positive bias, manifested by inflated helpfulness ratings for positive reviews. This positive biasmight misguide consumer to

focus only on the reviews that have been labeled as helpful. As shown in figure 9, consumers tend to rate positive reviews to be more helpful than negative ones.

Compare to the classification model to predict the helpfulness of reviews in study[27], which has average precision of 62.8833% in training set size of 1,000 and average precision of 66.9742% in training set size of 10,000, our model has an average precision of 68.7339% in training set size of 2,276. So our model shows a better performance on correctly predicting the helpfulness of reviews. In the proposed model from study [28], classification precision for GPS product from Amazon reviews is 77.7% for not helpful reviews and 77% for helpful reviews. Among the MP3 Player product reviews, the precision is 69.7% for not helpful reviews and 72% for helpful reviews, which higher than the precision of our model. However, the model proposed in the above study is limited for specific categories of product whereas our model is providing result for general categories.

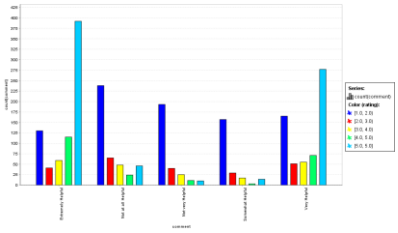

Figure 9: Distribution of label class

Chapter 7

CONCLUSION AND FUTURE WORK

Most websites rely on manual votes from users to measure the helpfulness of reviews. In this paper, we proposed a model to automatically predict the helpfulness of online reviews. We crawled review data from Amazon.com and then trained an SVM model to classify the helpfulness degree of online reviews. Eventually, we applied it to unlabeled reviews. Our best model achieved an accuracy of 68.7339% on predicting and classifying the helpfulness of online reviews. We found that reviewer's reputation is the major determent of the helpfulness of online reviews.

This project can be further extended in several ways. First, we could examine the validity of our model on larger datasets. We expect our conclusion will hold because this model is general and applicable to reviews from multiple different product categories. Second, we can take more characteristics of reviews into consideration, such as the product categorization and product features mentioned in reviews. Third, we can extend our study into identifying the spam reviewers and further improve the accuracy of our model. We can also incorporate collaborative filtering method into this model, to build a personalized helpfulness prediction model.

Appendix A: Information of the product from the data set

Product Name	Review No	Review With VoteNo
World Pride Hot Lovely Metallic Sweet Lady Hollow Rose Flower Elastic Hair Band Headband - Gold	715	76
HSI PROFESSIONAL 1 CERAMIC TOURMALINE IONIC FLAT IRON HAIR STRAIGHTENER FREE GLOVE + POUCH AND travel size Argan Oil Leave In Hair Treatment. WORLDWIDE DUAL VOLTAGE 110v-220v	6834	2943
THE BEST Vitamin C Serum For Your Face Contains 20% Vitamin C + Amino Complex + Hyaluronic Acid Serum- Potent 20% Vitamin C with 5% Hyaluronic Acid Will Leave Your Skin Radiant & More Youthful Looking By Neutralizing Free Radicals. This Anti Aging Serum Will Finally Give You The Results You've Been Looking For. OZ Naturals is THE MOST EFFECTIVE C SERUM AVAILABLE!	795	675
Hot Sale Beauty Facial Makeup Blender Foundation Puff Sponges Colors by Random	203	56
Nail Art Brushes- Professional Nail Art Brushes- Sable Nail Art Brush Pen, Detailer, Liner **Set of 3	946	68
Rogaine for Men Hair Regrowth Treatment, 5% Minoxidil Topical Aerosol, Easy-to-Use Foam, 2.11 Ounce, 3 Month Supply (Packaging May Vary)	518	445
THE BEST Hyaluronic Acid Serum For Skin - Potent Anti Aging Serum - Best Anti Wrinkle Serum With 5% Vitamin C - Our Customers Call It A Facelift In A Bottle. This Serum Will Plump & Hydrate Dull Skin As It's Designed To Fill Those Fine Lines & Wrinkles. 100% Pure No Cream - Finally An Anti Aging Serum That Actually Does What It Says It Will Do - SELLS OUT FAST!	278	248
Olay Pro-X Advanced Cleansing System 0.68 Fl Oz, 1-Count	2134	835
Aztec Secret Indian Healing Clay Deep Pore Cleansing, 1 Pound	1737	1092
Dotting 5 X 2 Way Marbleizing Dotting Pen Set for Nail Art Manicure Pedicure, 4 Ounce	1669	187
Anchor Hocking Heritage Hill Glass Cookie/Candy Jar, 1-Gallon	453	189
Brita 35503 Pitcher Replacement Filters, 3-Pack	1365	1102
KRUPS 1500813248 Electric Spice and Coffee Grinder with Stainless Steel Blades, Black	2272	1090
Black & Decker CHV1510 Dustbuster 15.6-Volt Cordless Cyclonic Hand Vacuum	2562	595
Ball Jar Heritage Collection Pint Jars with Lids and Bands, Set of 6	514	178
T-fal C798SC64 Ultimate Stainless Steel Copper-Bottom Heavy gauge Multi-Layer Base Dishwasher Safe Oven Safe 12-Piece Cookware Set, Silver	483	301
Pyrex Prepware 2-Cup Measuring Cup, Clear with Red Measurements	968	507
Wilton 570-1121 Easy Flex 3-Piece Silicone Spatula Set, Blue	1098	276

AcuRite 613 Indoor Humidity Monitor	2048	420
OXO Good Grips Bottle Brush	932	154
Google Chromecast HDMI Streaming Media Player	13846	3832
Kindle Paperwhite, 6" High Resolution Display with Next-Gen Built-in Light, Wi-Fi - Includes Special Offers	8878	3137
Roku 3 Streaming Media Player	5982	2918
Apple TV MD199LL/A	3563	1355
Sony BDP-S5100 3D Blu-ray Disc Player with Wi-Fi	1266	604
ARRIS / Motorola SurfBoard SB6141 DOCSIS 3.0 Cable Modem - Retail Packaging - White	1718	601
Samsung UN32EH5300 32-Inch 1080p 60 Hz Smart LED HDTV (Black)	2286	1110
Medialink Wireless-N Broadband Router with Internal Antennas (300 Mbps) - 2.4GHz - 802.11b/g/n - Compatible with Windows 8 / Windows 7 / Windows Vista / Windows XP / Mac OS X / Linux	1935	599
Sabrent 4-Port USB 2.0 Hub with Individual Power Switches and LEDs (HB-UMLS)	975	141
Acer C720 Chromebook (11.6-Inch, 2GB)	727	492
Viva Labs #1 Organic Extra Virgin Coconut Oil - 16 oz	1714	348
Keurig, The Original Donut Shop, K-Cup Packs	2733	463
KIND Nuts & Spices, Dark Chocolate Nuts & Sea Salt, 1.4 Ounce, 12-Count Bars	940	165
Nutiva Organic Chia Seeds, 12-Ounce Bag	515	227
Quest Nutrition Protein Bar Chocolate Chip Cookie Dough Flavor, 2.12 oz,12 Count	307	72
Nature's Way Coconut Oil, 32-Ounce	682	322
Vita Coco 100% Pure Coconut Water, 11.1-Ounce Containers (Pack of 12)	1459	546
Happy Tot Organic Baby Food, Stage 4, Spinach, Mango and Pear, 4.0-Ounce Pouches (Pack of 16)	392	90
Annie's Homegrown Berry Patch Organic Bunny Fruit Snacks,0.8 Ounce, 5-Count Pouches (Pack of 4)	349	101
Grove Square Hot Cocoa, Milk Chocolate, 24 Count Single Serve Cup for Keurig K-Cup Brewers	1656	189
Pampers Sensitive Wipes 7x Box 448 Count	675	76
Quilted Northern Ultra Plush Bath Tissue, 48 Double Rolls (Packaging May Vary)	1407	328
Huggies Natural Care Fragrance Free Baby Wipes Refill, 648 Count (Packaging may vary)	371	75
Playtex Diaper Genie Refill, 270 count (pack of 3)	888	126

Luvs With Ultra Leakguards Size 4 Diapers 160 Count	948	475
NatureWise Garcinia Cambogia Extract Natural Appetite Suppressant and Weight Loss Supplement, 180 Count, 500mg	3389	2575
Angel Soft, Double Rolls, [4 Rolls*12 Pack] = 48 Total Count	765	118
Crest 3D White Whitestrips, Professional Effects w/Advanced Seal Whitening Treatment, 20 Treatments (Packaging May Vary)	1706	691
Fitbit Flex Wireless Activity + Sleep Wristband, Black	3594	1602
Sparkle Paper Towels, 24 Giant Rolls, Pick-A-Size, White	370	63
	94560	34878

Appendix B: Source Code

preprocessing.r

```
d <-read.csv("C:\\Users\\someone\\Desktop\\project\\data\\data.csv")
d.sub <- subset(d,(d$reviewNo > 1|(d$rating!=5 & d$rating!=1))& d$voteTotal >10)
age <- (Sys.Date() - as.Date(d.sub$date, format='%d-%b-%y'))
label <- d.sub$vote/d.sub$voteTotal
df <- data.frame(d.sub$rating, d.sub$name, d.sub$review, age, d.sub$reviewerRanking,
d.sub$helpfulPercentage, d.sub$reviewNo, label)
names(df) <- c("rating", "name", "review", "age", "reviewerRanking",
"helpfulPercentage", "reviewNo", "label")
write.csv(df, "C:\\Users\\someone\\Desktop\\project\\R\\df10.csv", row.names = FALSE)
```

sentiment.r

```
score <- function (corpus, control, scoreFUNS, replace = TRUE)
{
   if (missing(control)) {
      control = list(tolower = TRUE, removePunctuation = TRUE,
         removeNumbers = TRUE, removeWords = list(stopwords("english")),
         stripWhitespace = TRUE, stemDocument = TRUE, minWordLength = 3,
         weighting = weightTf)
   }
   if (missing(scoreFUNS)) {
      scoreFUNS = list(polarity = list(), subjectivity = list(),
         pos_refs_per_ref = list(), neg_refs_per_ref = list(),
         senti_diffs_per_ref = list())
   }
   tdm <- TermDocumentMatrix(corpus, control = control)
   res <- list()
   for (n in names(scoreFUNS)) {
      args <- unlist(scoreFUNS[[n]])
      if (is.null(args)) {
         res[[n]] <- eval(call(n, tdm))
      }
      else {
         res[[n]] <- eval(call(n, tdm, args))
      }
   }
   dfres <- as.data.frame(res)
   meta <- DMetaData(corpus)
   if (replace) {
```

```
      MetaID <- meta$MetaID
      meta[, colnames(dfres)] <- dfres
      DMetaData(corpus) <- meta
   }
   else {
      DMetaData(corpus) <- cbind(DMetaData(corpus), dfres)
   }
   corpus
}

d <-read.csv("C:\\Users\\someone\\Desktop\\project\\R\\test10.csv")
d.corpus <- Corpus(VectorSource(d$review))
d.score <- score(d.corpus)

df <- data.frame(d,meta(d.score))
dfd <- df[, which(names(df)!="MetaID" & names(df)!="review" & names(df)!="name")]
dfd.sub <- subset(dfd, polarity!="NaN")

write.csv(dfd.sub, "C:\\Users\\someone\\Desktop\\project\\R\\df_sentiment_c.csv",
row.names = FALSE)
```

DataDetails.java

```
public class DataDetails {

        String helpfulPercentage = null;
        String rating = null;
        String name = null;
        String review = null;
        String age = null;
        String reviewerNo = null;
        String label = null;
        String comment = null;
        String reviewerRanking = null;
        float sentenceCount = 0;
        float tokenCount = 0;
        float tokenPerSentence = 0;
        float nounPercentage = 0;
        float verbPercentage = 0;
        float adverbPercentage = 0;
        float adjectivePercentage = 0;
        float errorPerSentence = 0;
```

```
public String getComment()
{
	return comment;
}

public void setComment(String comment)
{
	this.comment = comment;
}

public float getTokenPerSentence()
{
	return tokenPerSentence;
}

public void setTokenPerSentence(float tokenPerSentence)
{
	this.tokenPerSentence = tokenPerSentence;
}

public float getErrorPerSentence()
{
	return errorPerSentence;
}

public void setErrorPerSentence(float errorPerSentence)
{
	this.errorPerSentence = errorPerSentence;
}

public float getNounPercentage()
{
	return nounPercentage;
}

public void setNounPerventage(float nounPercentage)
{
	 this.nounPercentage =  nounPercentage;
}

public float getVerbPercentage()
{
	return verbPercentage;
}
```

```
public void setVerbPerventage(float verbPercentage)
{
	this.verbPercentage =  verbPercentage;
}

public float getAdverbPercentage()
{
	return adverbPercentage;
}

public void setAdverbPerventage(float adverbPercentage)
{
	this.adverbPercentage =  adverbPercentage;
}

public float getAdjectivePercentage()
{
	return adjectivePercentage;
}

public void setAdjectivePerventage(float adjectivePercentage)
{
	this.adjectivePercentage =  adjectivePercentage;
}

public float getTokenCount()
{
	return tokenCount;
}

public void setTokenCount(float tokenCount)
{
	this.tokenCount = tokenCount;
}

public float getSentenceCount()
{
	return sentenceCount;
}

public void setSentenceCount(float sentenceCount)
{
	this.sentenceCount = sentenceCount;
```

```
}

public String getHelpfulPercentage()
{
	return helpfulPercentage;
}

public void setHelpfulPercentage(String helpfulPercentage)
{
	this.helpfulPercentage = helpfulPercentage;
}

public String getRating()
{
	return rating;
}

public void setRating(String rating)
{
	this.rating = rating;
}

public String getName()
{
	return name;
}

public void setName(String name)
{
	this.name = name;
}

public String getReview()
{
	return review;
}

public void setReview(String review)
{
	this.review = review;
}

public String getAge()
{
```

```
		return age;
	}

	public void setAge(String age)
	{
		this.age = age;
	}

	public String getReviewerNo()
	{
		return reviewerNo;
	}

	public void setReviewerNo(String reviewerNo)
	{
		this.reviewerNo = reviewerNo;
	}

	public String getLabel()
	{
		return label;
	}

	public void setLabel(String label)
	{
		this.label = label;
	}

	public String getReviewerRanking()
	{
		return reviewerRanking;
	}

	public void setReviewerRanking(String reviewerRanking)
	{
		this.reviewerRanking = reviewerRanking;
	}

}
```

TextMining.java

```
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;

import org.languagetool.JLanguageTool;
import org.languagetool.language.AmericanEnglish;
import org.languagetool.rules.RuleMatch;

import au.com.bytecode.opencsv.CSVReader;
import au.com.bytecode.opencsv.CSVWriter;
import au.com.bytecode.opencsv.bean.ColumnPositionMappingStrategy;
import au.com.bytecode.opencsv.bean.CsvToBean;


import opennlp.tools.cmdline.PerformanceMonitor;
import opennlp.tools.cmdline.postag.POSModelLoader;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSSample;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.sentdetect.SentenceDetectorME;
import opennlp.tools.sentdetect.SentenceModel;
import opennlp.tools.tokenize.Tokenizer;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;
import opennlp.tools.tokenize.WhitespaceTokenizer;
import opennlp.tools.util.InvalidFormatException;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;

public class TextMining {

	/**
	 * @param args
	 */
	public void SentenceDetect() throws InvalidFormatException,
			IOException {
```

```
List<DataDetails> list = new ArrayList<DataDetails>();
list = parseCSVtoBean("df10.csv");
System.out.println(list.size());
//train model for sentenceCount
InputStream is = new FileInputStream("en-sent.bin");
SentenceModel sModel = new SentenceModel(is);
SentenceDetectorME sdetector = new SentenceDetectorME(sModel);
//train model for tokenCount
is = new FileInputStream("en-token.bin");
TokenizerModel tModel = new TokenizerModel(is);
Tokenizer tokenizer = new TokenizerME(tModel);
//train model for postag
POSModel model = new POSModelLoader().load(new File("en-pos-
maxent.bin"));
POSTaggerME tagger = new POSTaggerME(model);
//train model for error detection
JLanguageTool langTool = new JLanguageTool(new AmericanEnglish());
langTool.activateDefaultPatternRules();


int i = 0;
for(DataDetails d: list)
{
	String sentences[] = sdetector.sentDetect(d.getReview());
	d.setSentenceCount(sentences.length);

	String tokens[] = tokenizer.tokenize(d.getReview());
	d.setTokenCount(tokens.length);

	d.setTokenPerSentence(d.getTokenCount()/d.getSentenceCount());
	i++;

	ObjectStream<String> lineStream = new
PlainTextByLineStream(new StringReader(d.getReview()));

	String line;
	String result = "";
	while ((line = lineStream.read()) != null) {

		String whitespaceTokenizerLine[] =
WhitespaceTokenizer.INSTANCE.tokenize(line);
		String[] tags = tagger.tag(whitespaceTokenizerLine);
```

```
                                POSSample sample = new
POSSample(whitespaceTokenizerLine, tags);
                                System.out.println(sample.toString());
                                result += sample.toString();
                        }
                        d.setAdjectivePerventage(findStrCount(result,
"_JJ")/d.getTokenCount());
                        d.setAdverbPerventage(findStrCount(result,
"_RB")/d.getTokenCount());
                        d.setNounPerventage(findStrCount(result,
"_NN")/d.getTokenCount());
                        d.setVerbPerventage(findStrCount(result,
"_VB")/d.getTokenCount());


                        List<RuleMatch> matches = langTool.check(d.getReview());
                        d.setErrorPerSentence(matches.size()/d.getSentenceCount());

                        d.setComment(labelToComment(d.getLabel()));

                }


                parseBeanToCSV(list);
                is.close();
        }

        public static int findStrCount(String str, String findStr){
                int lastIndex = 0;
                int count =0;

                while(lastIndex != -1){
                     lastIndex = str.indexOf(findStr,lastIndex);
                     if( lastIndex != -1){
                          count ++;
                          lastIndex+=findStr.length();
                     }
                }
                return count;
        }

        private void parseBeanToCSV(List<DataDetails> list) throws IOException{
                FileWriter fw = new FileWriter("test10.csv");
                CSVWriter writer = new CSVWriter(fw);
```

```
		String [] header = {"rating", "name", "review", "age", "reviewerRanking",
"helpfulPercentage", "reviewNo", "comment", "token count", "sentence count", "token
per sentence", "adjective percentage", "noun percentage", "verb percentage", "adverb
percentage", "error per sentence"};
		writer.writeNext(header);
		String [] data;

		for(DataDetails d: list)
		{
			data = new String [] {d.getRating(), d.getName(), d.getReview(),
d.getAge(), d.getReviewerRanking(), d.getHelpfulPercentage(), d.getReviewerNo(),
d.getComment(), String.valueOf(d.getTokenCount()) ,
String.valueOf(d.getSentenceCount()), String.valueOf(d.getTokenPerSentence()),
String.valueOf(d.getAdjectivePercentage()),
					String.valueOf(d.getNounPercentage()),
String.valueOf(d.getVerbPercentage()), String.valueOf(d.getAdverbPercentage()),
String.valueOf(d.getErrorPerSentence()) };
			writer.writeNext(data);
		}
		 writer.close();
	}

	private String labelToComment(String label) {
		String comment = "";
		if(Float.parseFloat(label) >= 0.8)
			comment = "Extremely Helpful";
		if(Float.parseFloat(label) < 0.8 && Float.parseFloat(label) >= 0.6)
			comment = "Very Helpful";
		if(Float.parseFloat(label) < 0.6 && Float.parseFloat(label) >= 0.4)
			comment = "Somewhat Helpful";
		if(Float.parseFloat(label) < 0.4 && Float.parseFloat(label) >= 0.2)
			comment = "Not very Helpful";
		if(Float.parseFloat(label) < 0.2)
			comment = "Not at all Helpful";
		return comment;
		// TODO Auto-generated method stub

	}

	private List<DataDetails> parseCSVtoBean(String filename) {
		try {
			// To ignore Processing of 1st row
			CSVReader reader = new CSVReader(new FileReader(filename),
','
```

```
                                        '\"', 1);

                        ColumnPositionMappingStrategy<DataDetails> mappingStrategy
= new ColumnPositionMappingStrategy<DataDetails>();
                        mappingStrategy.setType(DataDetails.class);

                        String[] columns = new String[] { "rating", "name", "review",
"age", "reviewerRanking", "helpfulPercentage", "reviewerNo", "label"};
                        mappingStrategy.setColumnMapping(columns);

                        CsvToBean<DataDetails> csv = new CsvToBean<DataDetails>();
                        List<DataDetails> dataList = csv.parse(mappingStrategy, reader);

                        for (int i = 0; i < dataList.size(); i++) {
                                DataDetails dataDetails = dataList.get(i);
                        }
                        return dataList;

                } catch (FileNotFoundException e) {
                        System.err.println(e.getMessage());
                        return null;
                }
        }


}
```

Appendix C: Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 1 -V -1 -W 1
-K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Relation:     df_sentiment_c
Instances:    2276
Attributes:   19
              rating
              age
              reviewerRanking
              helpfulPercentage
              reviewNo
              comment
              token.count
              sentence.count
              token.per.sentence
              adjective.percentage
              noun.percentage
              verb.percentage
              adverb.percentage
              error.per.sentence
              polarity
              subjectivity
              pos_refs_per_ref
              neg_refs_per_ref
              senti_diffs_per_ref
Test mode:    split 66.0% train, remainder test

=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: Not at all Helpful, Not very Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

        -0.3764 * (standardized) rating
```

+ -0.2674 * (standardized) age
+ 0.1331 * (standardized) reviewerRanking
+ 1.4137 * (standardized) helpfulPercentage
+ -2.4454 * (standardized) reviewNo
+ -0.3168 * (standardized) token.count
+ 0.7389 * (standardized) sentence.count
+ -0.0113 * (standardized) token.per.sentence
+ 0.0846 * (standardized) adjective.percentage
+ 0.0499 * (standardized) noun.percentage
+ -0.085 * (standardized) verb.percentage
+ 0.0845 * (standardized) adverb.percentage
+ -0.0127 * (standardized) error.per.sentence
+ 0.0103 * (standardized) polarity
+ 0.0224 * (standardized) subjectivity
+ 0.0074 * (standardized) pos_refs_per_ref
+ 0.0301 * (standardized) neg_refs_per_ref
+ -0.0108 * (standardized) senti_diffs_per_ref
+ 0.6714

Number of kernel evaluations: 244601 (95.75% cached)

Classifier for classes: Not at all Helpful, Extremely Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

0.501 * (standardized) rating
+ 0.0165 * (standardized) age
+ -0.6432 * (standardized) reviewerRanking
+ 1.8289 * (standardized) helpfulPercentage
+ -0.1416 * (standardized) reviewNo
+ 0.161 * (standardized) token.count
+ -0.1141 * (standardized) sentence.count
+ -0.1626 * (standardized) token.per.sentence
+ 0.1605 * (standardized) adjective.percentage
+ 0.0345 * (standardized) noun.percentage
+ 0.0051 * (standardized) verb.percentage
+ 0.0713 * (standardized) adverb.percentage
+ -0.0007 * (standardized) error.per.sentence
+ 0.2876 * (standardized) polarity
+ 0.0029 * (standardized) subjectivity
+ -0.1369 * (standardized) pos_refs_per_ref
+ 0.2103 * (standardized) neg_refs_per_ref

+ -0.2358 * (standardized) senti_diffs_per_ref
+ 0.1261

Number of kernel evaluations: 239518 (87.116% cached)

Classifier for classes: Not at all Helpful, Very Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

0.2437 * (standardized) rating
+ 0.2302 * (standardized) age
+ -0.5686 * (standardized) reviewerRanking
+ 1.2016 * (standardized) helpfulPercentage
+ -0.1716 * (standardized) reviewNo
+ -0.4674 * (standardized) token.count
+ 0.6034 * (standardized) sentence.count
+ -0.0404 * (standardized) token.per.sentence
+ 0.1423 * (standardized) adjective.percentage
+ -0.074 * (standardized) noun.percentage
+ 0.0301 * (standardized) verb.percentage
+ 0.0924 * (standardized) adverb.percentage
+ 0.0604 * (standardized) error.per.sentence
+ 0.0416 * (standardized) polarity
+ -0.0551 * (standardized) subjectivity
+ -0.0581 * (standardized) pos_refs_per_ref
+ -0.0144 * (standardized) neg_refs_per_ref
+ -0.0412 * (standardized) senti_diffs_per_ref
+ 0.9475

Number of kernel evaluations: 284051 (89.435% cached)

Classifier for classes: Not at all Helpful, Somewhat Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

-0.356 * (standardized) rating
+ 0.2093 * (standardized) age
+ -0.3516 * (standardized) reviewerRanking
+ 1.2055 * (standardized) helpfulPercentage
+ -0.509 * (standardized) reviewNo

+ -0.3441 * (standardized) token.count
+ 0.7224 * (standardized) sentence.count
+ 0.0598 * (standardized) token.per.sentence
+ 0.1073 * (standardized) adjective.percentage
+ -0.0231 * (standardized) noun.percentage
+ -0.0441 * (standardized) verb.percentage
+ 0.0469 * (standardized) adverb.percentage
+ -0.0548 * (standardized) error.per.sentence
+ 0.1634 * (standardized) polarity
+ -0.0063 * (standardized) subjectivity
+ -0.0353 * (standardized) pos_refs_per_ref
+ 0.0413 * (standardized) neg_refs_per_ref
+ -0.0534 * (standardized) senti_diffs_per_ref
+ 0.8028

Number of kernel evaluations: 195985 (93.686% cached)

Classifier for classes: Not very Helpful, Extremely Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

0.7185 * (standardized) rating
+ 0.2701 * (standardized) age
+ -0.3208 * (standardized) reviewerRanking
+ 2.3232 * (standardized) helpfulPercentage
+ -0.0833 * (standardized) reviewNo
+ 0.1662 * (standardized) token.count
+ -0.3694 * (standardized) sentence.count
+ -0.1044 * (standardized) token.per.sentence
+ 0.11 * (standardized) adjective.percentage
+ 0.0545 * (standardized) noun.percentage
+ 0.0317 * (standardized) verb.percentage
+ 0.0816 * (standardized) adverb.percentage
+ -0.0806 * (standardized) error.per.sentence
+ -0.3428 * (standardized) polarity
+ -0.1885 * (standardized) subjectivity
+ -0.0634 * (standardized) pos_refs_per_ref
+ -0.2518 * (standardized) neg_refs_per_ref
+ 0.0891 * (standardized) senti_diffs_per_ref
+ 0.0667

Number of kernel evaluations: 166797 (86.455% cached)

Classifier for classes: Not very Helpful, Very Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

```
       0.5656 * (standardized) rating
 +     0.2424 * (standardized) age
 +    -0.4362 * (standardized) reviewerRanking
 +     1.8005 * (standardized) helpfulPercentage
 +    -0.2048 * (standardized) reviewNo
 +    -0.217  * (standardized) token.count
 +    -0.0231 * (standardized) sentence.count
 +     0.0166 * (standardized) token.per.sentence
 +     0.1442 * (standardized) adjective.percentage
 +     0.0732 * (standardized) noun.percentage
 +     0.1519 * (standardized) verb.percentage
 +     0.0377 * (standardized) adverb.percentage
 +    -0.0263 * (standardized) error.per.sentence
 +    -0.1246 * (standardized) polarity
 +    -0.1187 * (standardized) subjectivity
 +    -0.0419 * (standardized) pos_refs_per_ref
 +    -0.1556 * (standardized) neg_refs_per_ref
 +     0.0527 * (standardized) senti_diffs_per_ref
 +     0.7899
```

Number of kernel evaluations: 279302 (91.089% cached)

Classifier for classes: Not very Helpful, Somewhat Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

```
      -0.1626 * (standardized) rating
 +     0.3964 * (standardized) age
 +    -0.5872 * (standardized) reviewerRanking
 +     0.6311 * (standardized) helpfulPercentage
 +    -0.032  * (standardized) reviewNo
 +     0.1374 * (standardized) token.count
 +    -0.0189 * (standardized) sentence.count
 +     0.0653 * (standardized) token.per.sentence
 +     0.0868 * (standardized) adjective.percentage
```

+ 0.1021 * (standardized) noun.percentage
+ 0.0897 * (standardized) verb.percentage
+ 0.072 * (standardized) adverb.percentage
+ -0.0876 * (standardized) error.per.sentence
+ -0.029 * (standardized) polarity
+ -0.0162 * (standardized) subjectivity
+ 0.0136 * (standardized) pos_refs_per_ref
+ -0.0502 * (standardized) neg_refs_per_ref
+ 0.04 * (standardized) senti_diffs_per_ref
+ 0.4726

Number of kernel evaluations: 124038 (96.541% cached)

Classifier for classes: Extremely Helpful, Very Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

-0.0016 * (standardized) rating
+ 0.1031 * (standardized) age
+ -0.252 * (standardized) reviewerRanking
+ -3.481 * (standardized) helpfulPercentage
+ 0.0721 * (standardized) reviewNo
+ -0.1798 * (standardized) token.count
+ 0.2306 * (standardized) sentence.count
+ 0.2411 * (standardized) token.per.sentence
+ 0.0307 * (standardized) adjective.percentage
+ -0.0493 * (standardized) noun.percentage
+ -0.0014 * (standardized) verb.percentage
+ 0.0581 * (standardized) adverb.percentage
+ -0.3054 * (standardized) error.per.sentence
+ 0.1105 * (standardized) polarity
+ 0.0038 * (standardized) subjectivity
+ 0.0034 * (standardized) pos_refs_per_ref
+ 0.0019 * (standardized) neg_refs_per_ref
+ 0.0018 * (standardized) senti_diffs_per_ref
+ 1.8902

Number of kernel evaluations: 975094 (95.225% cached)

Classifier for classes: Extremely Helpful, Somewhat Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

```
      -0.65   * (standardized) rating
+      -0.0249 * (standardized) age
+       0.2741 * (standardized) reviewerRanking
+      -2.5503 * (standardized) helpfulPercentage
+       0.1473 * (standardized) reviewNo
+      -0.1532 * (standardized) token.count
+       0.3467 * (standardized) sentence.count
+      -0.0457 * (standardized) token.per.sentence
+       0.0117 * (standardized) adjective.percentage
+      -0.0291 * (standardized) noun.percentage
+       0.1013 * (standardized) verb.percentage
+       0.1062 * (standardized) adverb.percentage
+       0.114  * (standardized) error.per.sentence
+      -0.2839 * (standardized) polarity
+       0.0254 * (standardized) subjectivity
+       0.1137 * (standardized) pos_refs_per_ref
+      -0.1235 * (standardized) neg_refs_per_ref
+       0.1669 * (standardized) senti_diffs_per_ref
+       0.4407
```

Number of kernel evaluations: 339525 (90.593% cached)

Classifier for classes: Very Helpful, Somewhat Helpful

BinarySMO

Machine linear: showing attribute weights, not support vectors.

```
      -0.7216 * (standardized) rating
+       0.0381 * (standardized) age
+       0.0909 * (standardized) reviewerRanking
+      -2.093  * (standardized) helpfulPercentage
+       0.1192 * (standardized) reviewNo
+       0.0426 * (standardized) token.count
+       0.2545 * (standardized) sentence.count
+       0.0254 * (standardized) token.per.sentence
+      -0.113  * (standardized) adjective.percentage
+       0.0413 * (standardized) noun.percentage
+      -0.0861 * (standardized) verb.percentage
+      -0.0006 * (standardized) adverb.percentage
+      -0.0389 * (standardized) error.per.sentence
```

```
+      -0.2491 * (standardized) polarity
+       0.0479 * (standardized) subjectivity
+       0.1111 * (standardized) pos_refs_per_ref
+      -0.0783 * (standardized) neg_refs_per_ref
+       0.139  * (standardized) senti_diffs_per_ref
-      0.6407

Number of kernel evaluations: 326386 (93.028% cached)



Time taken to build model: 3.53 seconds

=== Evaluation on test split ===

Time taken to test model on training split: 0 seconds

=== Summary ===

Correctly Classified Instances         516               66.6667 %
Incorrectly Classified Instances       258               33.3333 %
Kappa statistic                          0.5572
Mean absolute error                      0.2599
Root mean squared error                  0.3459
Relative absolute error                 85.1215 %
Root relative squared error             88.4218 %
Coverage of cases (0.95 level)          98.5788 %
Mean rel. region size (0.95 level)      80.491  %
Total Number of Instances               774

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.845    0.091    0.687      0.845    0.758      0.699    0.915     0.655     Not at all Helpful
               0.273    0.036    0.529      0.273    0.360      0.319    0.813     0.347     Not very Helpful
               0.843    0.144    0.733      0.843    0.784      0.676    0.893     0.704     Extremely Helpful
               0.636    0.123    0.652      0.636    0.644      0.517    0.831     0.558     Very Helpful
               0.329    0.044    0.436      0.329    0.375      0.324    0.699     0.255     Somewhat Helpful
```

Weighted Avg. 0.667 0.105 0.649 0.667 0.649 0.559 0.852 0.568

=== Confusion Matrix ===

```
  a   b   c   d   e   <-- classified as
125  11   5   4   3 |   a = Not at all Helpful
 46  27   2  12  12 |   b = Not very Helpful
  0   1 209  35   3 |   c = Extremely Helpful
  2   0  60 131  13 |   d = Very Helpful
  9  12   9  19  24 |   e = Somewhat Helpful
```

Bibliography

[1] A. Mukherjee, V. Venkataraman, B. Liu and N. Glance, "What Yelp Fake Review Filter Might Be Doing," in *Proceedings of the International Conference on Weblogs and Social Media*, 2013.

[2] "Amazon," [Online]. Available: http://www.amazon.com/.

[3] "Sephora," [Online]. Available: http://www.sephora.com/.

[4] Y. C. C. L. Y. H. M. Z. J. Liu, "Low-quality product review detection in opinion summarization," *EMNLP-CoNLL*, 2007.

[5] Z. Zhang and B. Varadarajan, "Utility scoring of product reviews," in *CIKM '06 Proceedings of the 15th ACM international conference on Information and knowledge management*, New York, NY, USA, 2006.

[6] S.-M. Kim, P. Pantel, T. Chklovski and M. Pennacchiotti, “Automatically assessing review helpfulness,” in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* , Stroudsburg, PA, USA, 2006.

[7] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang and M. Zhou, "Low-Quality Product Review Detection in Opinion Summarization," in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* , Prague, Czech Republic, 2007.

[8] A. Ghose and P. G. Ipeirotis, "Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics," *IEEE Transactions Knowledge and Data Engineering,* vol. 23, no. 10, pp. 1498-1512, 3011.

[9] Y. Liu, X. Huang, A. An and X. Yu, "Modeling and Predicting the Helpfulness of Online Reviews," in *Eighth IEEE International Conference on Data Mining*, Pisa, Italy, 2008.

[10] S. Siersdorfer, S. Chelaru, W. Nejdl and J. S. Pedro, "How useful are your comments?: analyzing and predicting youtube comments and comment ratings," in *Proceedings of the 19th international conference on World wide web* , New York, NY, USA, 2010.

[11] M. P. O'Mahony and B. Smyth, "Learning to recommend helpful hotel reviews," in *Proceedings of the third ACM conference on Recommender systems* , New York, NY, USA, 2009.

[12] R. Zhang and T. Tran, "An Entropy-based model for discovering the usefulness of online product reviews," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, Washington, DC, USA, 2008.

[13] T. L. Ngo and A. P. Sinha, "Analyzing Online Review Helpfulness Using a Regressional ReliefF-Enhanced Text Mining Method," *ACM Transactions on Management Information Systems (TMIS),* vol. 3, no. 2, p. Article No. 10 , July 2012.

[14] Y. Lu, A. Ntoulas and L. Polanyi, “Exploiting social context for review quality prediction,” in *Proceedings of the 19th international conference on World wide web* , New York, NY, USA , 2012.

[15] Y. Pan and J. Q. Zhang, “Born Unequal: A Study of the Helpfulness of User-Generated Product Reviews,” *Journal of Retailing,* vol. 87, no. 4, pp. 598-612, 2011.

[16] Amazon, [Online]. Available: http://www.amazon.com/review/guidelines/top-reviewers.html/ref=cm_pdp_rev_ln_ntr.

[17] "Scrapy | An open source web scraping framework for Python," [Online]. Available: http://scrapy.org/.

[18] M. Luca and G. Zervas, “Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud,” *Working Paper,* p. September, 2013.

[19] "Apache OpenNLP," [Online]. Available: https://opennlp.apache.org/.

[20] "Alphabetical list of part-of-speech tags used in the Penn Treebank Project," [Online]. Available: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[21] "The R Project for Statistical Computing," [Online]. Available: http://www.r-project.org/.

[22] M. Annau, “tm.plugin.sentiment Online Sentiment Analysis using R,” 12 10 2010. [Online]. Available: http://statmath.wu.ac.at/courses/SNLP/Presentations/DA-Sentiment.pdf.

[23] L. S. a. G. Check. [Online]. Available: https://www.languagetool.org/.

[24] “Source Forge,” [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMO.html.

[25] T. Hastie and R. Tibshitani, "Classification by pairwise coupling," *The Annals of Statistics* , vol. 26, no. 2, pp. 451-471, 1998.

[26] [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/supportVector/package-summary.html.

[27] S. C. W. N. J. S. P. S. Siersdorfer, “How useful are your comments?: analyzing and predicting youtube comments and comments rating,” *WWW*, 2010.

[28] T. T. R. Zhang, “An Entropy-based model for discovering the usefulness of online product reviews,” *WI-IAT*, 2008.

[29] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang and M. Zhou, “Low-Quality Product Review Detection in Opinion Summarization,” *EMNLP-CoNLL,* pp. 334-342, 2007.

[30] M. P. O. B. Smyth, “Learning to recommend helpful hotel reviews,” *ACM RecSys,* 2009.

[31] P. P. T. C. M. P. S. Kim, "Automatically assessing review helpfulness," *EMNLP '06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* , pp. 423-430, 2006.

[32] A. P. S. T. L. Ngo-Ye, “Analyzing Online Review Helpfulness Using a Regressional ReliefF-Enhanced Text Mining Method,” *ACM Transactions on Management Information Systems (TMIS)* , vol. 3, no. 2, 2012.

[33] P. T. A. N. L. P. Y. Lu, “Exploiting social context for review quality prediction,” *Proceedings of the 19th international conference on World wide web* , pp. 691-700, 2012.

[34] [Online]. Available: http://en.wikipedia.org/wiki/Polynomial_kernel.