Aspect-based opinion summarization of product reviews on social media

Ásgeir Ögmundarson

Kongens Lyngby 2015

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

With the increase of opinions shared by people on social media the importance of algorithms capable of analyzing them for sentiment to produce value has grown. All companies face hidden costs in the form of lost revenue due to unhappy customers not returning and potential customers snubbing a product due to bad publicity. This thesis develops an aspect-based opinion summarization system capable of extracting opinion features from product reviews, determining the sentiment of the reviews the opinion features appeared in and producing a summary of the most frequent features discussed with regards to sentiment. The results from the summary system are promising but could still be improved to produce even more value.

Preface

This thesis was prepared at DTU Compute in fulfillment of the requirements for acquiring an M.Sc. in Software Engineering.

The thesis deals with the task of developing a system capable of aspect-based opinion summarization of product reviews on social media. The system does this by scraping product reviews from Twitter, extracting product features from those reviews, determining the sentiment of each review before finally presenting a summarization for the product.

The thesis consists of an introduction which describes the problem, the goals of this thesis, datasets used and related work (Chapter 1); methods used for extracting product features and for sentiment analysis (Chapter 2); evaluation of these methods and determination of best performing techniques (Chapter 3); results and discussion regarding summarization of features and sentiment (Chapter 4); conclusions and possible future work regarding the summarization system (Chapter 5).

Lyngby, 17-August-2015

Ásgeir Ögmundarson

Acknowledgements

I would like to thank my supervisor, Ole Winther, for his guidance and feedback regarding the direction of this thesis and obstacles faced.

I also wish to thank Henning Christiansen of the DTU Compute IT Support team for all his help on using the DTU cluster.

Contents

Chapter 1

Introduction

The amount of information available on the internet is vast and the development of methods for automatic categorization and organization of this information have been the focus of many researchers. While a lot of the work in this area has focused on topical categorization, such as categorizing news articles by subject[Pan] (e.g. politics, business, sports), the growing amount of discussions of the general public online has increased the usefulness of categorization according to sentiment, e.g. if it is positive or negative. Social media platforms have expanded exponentially for the last few years [oSMFPTtIOI] with ever so many people sharing their opinions on review sites, message forums, blogs, micro-blogs and other publicly open websites. Twitter alone has 302 million monthly active users and 500 million tweets every day [Twi]. This freely available information can be used to produce value, such as predicting the stock market [Si13] [Che11] [Bol], electronics sales [Nas12] and box office revenues [Liu]. However, the use of sentiment analysis on online discussions is not limited to prediction of behavior but can also be used to examine the standing of a company, product, person or any other object or entity with regards to public perception.

All companies face hidden costs, one of which is lost future revenue due to customers not returning because of their unhappiness with aspects of the product purchased. 96% of unhappy customers don't complain to the company so problems with products (and their degree) could go unnoticed [sFTS]. These unhappy customers could however complain on social media, potentially leading to

more lost revenue due to potential customers snubbing said product or company. If companies could identify the aspects of their products frequently discussed in a negative manner on public forums they could better address them and increase customer satisfaction, resulting in reduced hidden costs and increased profits. Additionally, if companies could identify what aspects of their products people like that information could be used to integrate those features into other products or otherwise use for marketing and product development purposes.

Using only sentiment analysis to locate negative or positive discussions regarding a product might not yield comprehensive and informative results as a product might be mentioned in text that is determined to be negative or positive but without knowing what aspects of the product are being discussed the value of the information is significantly diminished. It is therefore necessary to pair sentiment analysis with product feature extraction to obtain the most value from the information.

This thesis aims to design and implement a system capable of extracting product features from reviews (discussions) on social media and producing a summarized report of the most frequently discussed product aspects with regards to the sentiment of the reviews they appear in.

One might wonder about the possibility of the product supplier providing a list of product features to simplify this system. Although that is possible it might be hard for the supplier to provide a list of product features since they may sell a large amount of different products, terminology used might differ between supplier and customers for certain product features, the customers may comment on product features not thought of by the supplier or on features lacking from the product.

1.1 Goals

The problem this thesis will examine is aspect-based opinion summarization of product reviews on social media. For this thesis the term *review* is defined as a comment on a particular product. The root of the aforementioned problem is reduced customer retention- and acquisition rates due to undiscovered customer dissatisfaction and bad publicity on social media regarding specific features of a product.

The process of solving this problem can be split into three parts:

1. Identification of features of a product that people expressed an opinion on

in a review (called *opinion features*)

2. Sentiment determination of each review of that product
3. Summarization of the most frequent opinion features with regards to the number of appearances in positive and negative reviews.

For the first part I will develop an algorithm for finding opinion features in each review by using grammatical analysis, i.e. part-of-speech, before with evaluating the performance of different sets of rules which determine which candidate features are saved as opinion features. Descriptions of the methods used in this part can be found in section 2.1.

For the second part I evaluate different sentiment analysis algorithms to determine the model best suited for reviews from social media. Two frequently used techniques for sentiment analysis are a lexical approach or a machine learning approach. The lexical approach typically uses a list (lexicon) of terms labeled with sentiment rankings to calculate the sentiment of text according to the rankings of each word present in both the text and the lexicon. Machine learning methods use example data to build a model capable making predictions on other data. In this thesis I will evaluate and compare two different lexicons along with two supervised learning algorithms to find the best performing model for sentiment determination of the reviews. Descriptions of these methods can be found in section 2.2.

Let us use an example of a review for a mobile hone to illustrate the concepts of the first two parts:

"The picture quality is terrific on my phone!"

In the first part *"picture quality"* would be identified as an opinion feature with the opinion that modifies the feature as *"terrific"*. The term used as search query, in this case the word *"phone"*, would usually be saved as a candidate feature as it is a noun, but since I am only looking for product features and not the product itself the search query is removed from the collection of candidate features. In the second part this review would be classified as positive, either with a lexical approach where the word *"terrific"* would likely result in a positive determination (it is also possible that the words *"quality"* and the word *"new"* could influence the decision) or with a supervised learning, where reviews in a training dataset with similar wording are labeled as positive, resulting in a positive classification for this review.

Finally, for the third part of the thesis the most frequent opinion features extracted in part one are each labeled with the sentiment of the review they

appeared in (obtained in part two) to produce the summary.

Let us demonstrate an example of a summary of a frequent opinion feature ,*"picture quality"*, displaying how many times it appeared in a positive review and how many times it appeared in a negative review.

Feature : Picture quality
Positive: 14
Negative: 154
Ratio : 92% Negative

For this example the opinion feature *"picture quality"* has appeared in 14 positive reviews and 154 negative reviews. Individual reviews of the opinion feature could then be inspected for further manual analysis.

With an interactive interface it would also be possible to rank opinion features according to highest number of positive or negative occurrences, most controversial opinion features (high number of both positive and negative reviews) or features most unanimously reviewed (biggest different between number of positive and negative reviews).

1.2 Datasets

Four different datasets were obtained and used for this thesis, one for training a classifier for sentiment analysis, one for testing and tuning the opinion-feature miner, one containing tweets labeled with regards to sentiment to test the performance of the classifier on realistic data and one was scraped from Twitter for the purpose of summary demonstration. These datasets are described in individual sections below.

1.2.1 Amazon Review Data for Supervised Learning

The dataset used to train, validate and test the classifier was obtained from Julian McAuley of UCSD [Moh] [McA]. The particular data used for this project was a part of 143,7 million Amazon review dataset spanning May 1996 - July 2014. The part of the dataset obtained for this project was an aggressively de-duplicated subset of the aforementioned dataset which included 83 million reviews. Each review has 8 properties: reviewer ID, reviewer name, helpfulness rating, review body, rating, review summary, review unix time stamp and review

raw time stamp. Meta-data regarding the products reviewed was available in a separate file with the review ID as a primary key but since analysis focused on the review text itself the meta-data was not necessary for this thesis.

1.2.2 Annotated Opinion Feature Mining Dataset

Hu an Liu [Hub] used reviews collected from Amazon.com and Cnet.com where each review contains a title and review text [OFM] [1]. For five products they collected the first 100 reviews and manually read all the reviews and produced a feature list for each sentence in each review. They also included a manual sentiment estimate for each opinion of a feature and characters indicating if the feature was explicitly used in the sentence or if it was referenced or other wise implied (i.e. *"it was great"*). This annotated review set was made available and was used in this project to test the opinion feature mining part of the project.

1.2.3 Labeled Twitter Dataset for Sentiment Analysis Testing

This dataset was in it self not a dataset of tweets as it contained no tweets. It did however contain 5513 IDs of tweets with a hand-classified sentiment label attached to each tweet ID[2]. The sentiment was classified as one of the following: positive, negative, neutral or irrelevant. The tweet IDs were originally obtained by searching for one of the following terms: Apple, Google, Microsoft and Twitter. For this project each tweet had to be crawled from the ID.

1.2.4 Scraped Tweets for Summarization

To demonstrate the summarization system I scraped tweets using the Twitter API[3]. Only tweets containing no urls, were tagged as being in English and were not retweets were collected to avoid spam and duplicates and irrelevant tweets. The products chosen along with the amount of scraped tweets for each are:

Apple TV : 3941
Chromecast : 2419
Firefox : 8912

[1] `http://www.cs.uic.edu/~liub/FBS/`
[2] `http://www.sananalytics.com/lab/twitter-sentiment/`
[3] `https://dev.twitter.com/rest/public`

Galaxy S5 : 1487
Google Chrome : 4611
HTC Desire : 534
PlayStation 4 : 4100
Windows 10 : 7004
iPhone 6 : 954

Unique products were chosen to increase the value of the information regarding the opinion features. If a product line, such as iPhone, would have been chosen and a feature had been frequently criticized, it would not have been possible to know if the customers were talking about this feature in iPhone 4s, iPhone 5 or iPhone 6 (or any other model), making the information much less valuable.

1.3 Related Work

Work done on this paper is closely related to the work of Hu and Liu [Hub][Hua] on mining and summarizing customer reviews on products sold online. Their objective was to make it easier for potential costumers to research a product by creating a search engine capable of taking a product feature as a search query and displaying a summarization of the sentiment of reviews it had appeared in. The used reviews mined from Amazon.com and Cnet.com which they then manually labeled for features before using the data to test various opinion feature mining techniques. They showed that their techniques perform quite well for both opinion feature mining and sentiment analysis. My work differs from theirs in 2 main aspects: (1) They only use the opinion features mined to determine sentiment of reviews using a sophisticated algorithm, requiring no training corpora. (2) They are focusing on longer reviews (obtained from review websites) that contain multiple sentences, which they split up for further analysis.

The work of Dave, Lawrence and Pennock [Dav] is also very similar to work done in this thesis. They use labeled training corpora available from websites, to train sentiment classifiers. They showed that using their classifiers on test reviews worked well, although the performance when classifying sentences obtained with a search engine with a product name as a search parameter was limited due to the shortness of the sentences. Unlike this thesis they did not mine features from their reviews.

Morinaga *et al.* [Mor02] had the objective of mining product reputations on the web, which is related to the subject of this thesis. They compare different products in the same category to find its reputation but they do however not mine product features nor summarize the results. Even though they don't mine

feature they do perform four types of text mining: extraction of characteristic words, co-occurrence words, typical sentences for specific product categories and correspondence analysis among multiple product categories.

Bermingham and Smeaton [Ber] examined the hypothesis that it is easier to classify according to sentiment for short form documents than for long form documents. They also explore the difference in classification of microblogs vs microreviews. They trained and tested on both long- and short form documents, and had some success classifying microblogs but were unable to improve performance by extending a unigram feature representation for these short form documents. Prediction accuracy for longer texts was worse than for the shorter text.

The rest of the thesis consists of a description of methods used (chapter 2), the evaluation of the performance of these methods (chapter 3), results and discussions of the summarization of opinion features (chapter 4) and finally conclusions and future work (chapter 5).

Chapter 2

Methods

In this chapter I will discuss the methods used in this project to create the review summarization system. The system can be divided into into three steps: Opinion feature mining, sentiment analysis and summarization. Opinion feature mining revolves around finding features of products mentioned in the reviews, finding opinions shared in the reviews and then matching them together according to rules about proximity and order. A match of an opinion and a feature is called an opinion feature.

After determining the opinion features the sentiment of each review is determined. The results of these methods are then processed and presented in chapter 4.

The architectural overview of the system can be seen in Figure 2.1.

2.1 Mining Opinion Features

The techniques described in this section are heavily based on the work of Hu and Liu [Hub], although with some modifications. References in this section and its subsection to Hu's and Liu's work refer to the paper referenced in the

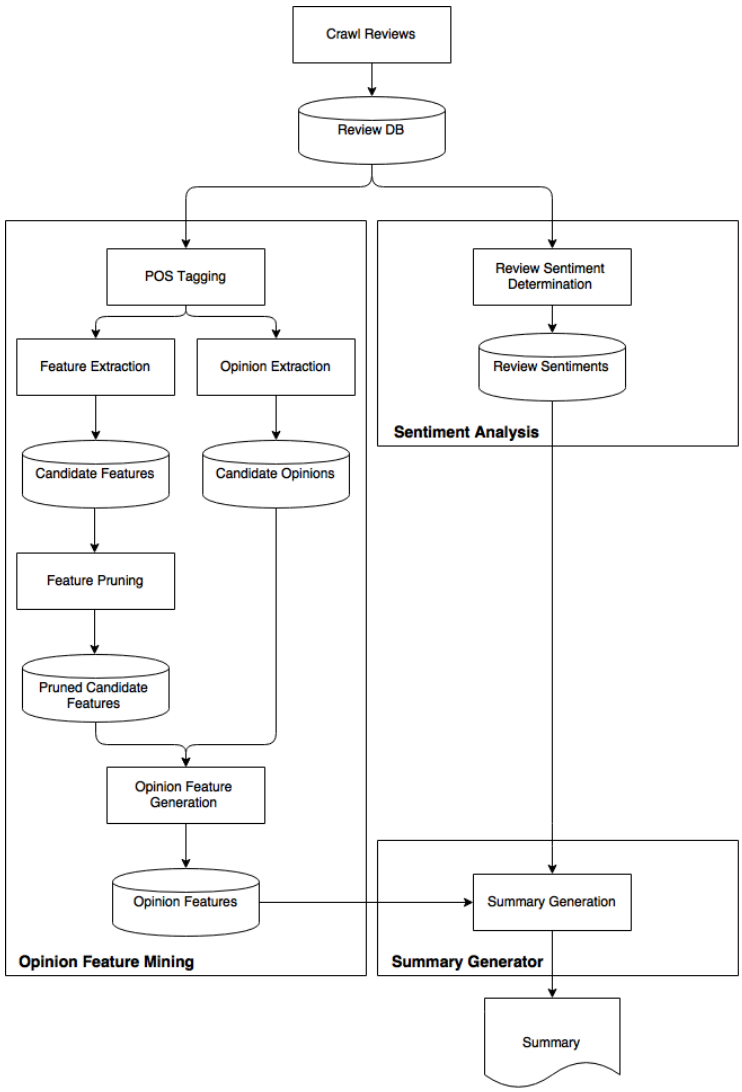


Figure 2.1: System Architecture

previous sentence. The differences in approaches are mentioned in appropriate sections below.

2.1.1 Part-Of-Speech Tagging (POS)

When extracting product features from a review it must be considered what characteristics can be used to identify them. According to Hu and Liu product features are usually nouns or noun phrases in reviews. Part-of-speech tagging must therefore be utilized to identify them. POS tagging is the process of classifying words according to their word class (noun, verb, adjective, etc). It is then possible to further classify those words into sets of words (phrases) according to predefined rules of combining words of certain classes, a process known as chunking. I performed those tasks using the Natural Language Toolkit (NLTK) [Bir09][1]. The following shows an example of a sentence which has been part-of-speech tagged and chunked.

(S This/DT (NP digital/NN camera/NN) is/VBZ (JJP great/JJ) ,/, I/PRP (VBPP really/RB love/VBP) it/PRP)

For instance, /NN indicates a noun and NP indicates a noun phrase. For this project I expanded on Hu and Liu's methods of chunking only noun phrases to find features. I included other chunking patterns to try to capture opinion words that were not adjectives as well as relevant words accompanying opinion words. I created four chunking rules:

Noun Phrases (NP) – Captures simple noun phrases, includes all types of nouns directly adjacent to each other.

Adjective Phrases (JJP) – Captures adjectives and relevant words often used directly in front of adjectives to increase or decrease the significance of the adjective, known as intensifiers. This could for example be *"really great"*, where *"really"* is an intensifier and *"great"* is the adjective the intensifier modifies.

Verb Base Form Phrases (VBPP) – Initially this was meant to capture phrases including opinions of things that did not include adjectives, for example words like "*love"*, *"hate"*, *"dislike"*, *"adore"*. After preliminary test it was discovered that the chunking rule included too many irrelevant verbs that did not directly imply a positive or a negative opinion, such as *"do"*, *"play"*, *"give"*. This rule was then further narrowed to include only the words *"love"* and *"hate"*. This rule also observes intensifiers in front of the words *"love"* and *"hate"*.

Verb Base Form Determiner Phrases (VBPDTP) – This rule is the same as the one above but includes a determiner appearing directly after the words *"love"* and *"hate"*.

[1] `http://www.nltk.org/`

Stemming was used to reduce words to a common form but stopwords were included.

2.1.2 Feature Extraction

To solve the problem of finding product features in text Hu and Liu used association rule mining [Agr] utilizing the CBA algorithm developed by Liu et al. [Liu98]. The CBA algorithm utilizes the Apriori algorithm which operates in two stages, first generating a complete set of classification association rules by identifying frequent individual items in a document and then extending them to larger item sets one item at a time, as long as they meet standards of minimum support, and in the second step creating a classifier from the rules. They only used the first step as the only needed to find the frequent itemsets for use as their candidate features. The Apriori does not consider word order or distance between words so multiple pruning and combination rules must be applied in order to use the algorithm for this purpose. I decided to develop my own method of extracting the features from the reviews instead of utilizing the CBA algorithm developed by Liu et al.[Liu98] to circumvent the aforementioned procedures. Hu and Liu[Hub] used the Apriori algorithm to find single nouns and check if they met the minimum support and then extended to phrases containing two and three nouns and then checked if they meet the minimum support threshold. They then used two pruning rules, described in section 2.1.3 to remove unwanted items/itemsets in order to increase precision and recall. My algorithm goes through every POS-tagged review and finds all items (nouns or noun phrases) and counts how many times they appear without looking at any minimum support thresholds and without producing a large amount of irrelevant features, as the Apriori algorithm does. I then use two methods, similar to the pruning methods used by Hu and Liu[Hub] to try to increase the number of the most frequent features. These are not necessary, unlike those employed by Hu and Liu[Hub] to remove irrelevant features.

I first collected review parts which were labeled as nouns or noun/phrases from the POS tagging described in section 2.1.1 for each sentence and stored them as candidate features. Candidate opinion words were also collected and stored in parallel to this operation but that process is further explained in section 2.1.4. Each candidate feature stored was labeled with the ID of the review it came from along with the index (location in the review) of each word that occurred in the feature for use in further processing.

In an effort to capture more opinion features I collected all occurrences of the words *"it"* and *"them"* that appeared after a candidate feature in a review with word distance of 4 or less. If the word matched that criteria it is considered to

be a reference to that feature and was stored as a reference-feature. For example,

"The player is great. It can play any disc I put in it."

Here the word *"it"* refers to the player mentioned in the previous sentence. As the word distance is within the limit of 4 *"it"* is saved as a reference-feature, using the nearby feature *"player"* instead of *"it"*, with the index of the original *"it"*.

Another effort to capture more opinion features consisted of collecting all instances of words with the stem *"look"*, e.g. *"look"*, *"looking"* and *"looks"*, to be saved as a candidate feature of type *look-feature* to describe the look of the product.

Not all of these feature types were handled in the same manner when generating opinion features by matching candidate opinion words with candidate features. That procedure is detailed in section 2.1.5.

2.1.3 Feature Pruning

Not all features found in the feature extraction are useful, interesting or are genuine features. Feature pruning attempts to remove those features that are inappropriate. Hu and Liu presented two types of pruning methods, both of which I used (at least to an extent) and are explained below.

2.1.3.1 Compactness Pruning

This method examines features containing at least two words, called feature phrases, and removes those likely to be of no use. Hu and Liu note that their association rule mining algorithm does not consider word position relative to other words. Since in natural language the order and composition of certain words are likely to be meaningful phrases this they attempted to prune meaningless phrases found by the association rule mining. They defined a feature phrase to be compact in a sentence if the word distance between any two words in a feature phrase was no greater than three and that the words were in a correct order. If a phrase is compact in at least two sentences then the feature is called a compact feature phrase. If a feature phrase is not compact in at least two sentences in their review database then the feature phrase is pruned.

While associate rule mining does not consider word (item) order my feature

extraction method does consider that so along with the fact that I am only interested in finding the most frequent features of a review set and not striving to achieve the highest precision implementing their method of compactness pruning for this project would not be useful. I did however implement a modified version because even though my method considers word order it only puts together compact phrases if the words in the phrase appear directly adjacent to each other and are therefore determined to be a noun phrase. It does not consider words that are not directly adjacent to each other but are compact according to the definition. Therefore I compile a list of all feature phrases and check if any sentences contain single word features that appear in the correct order within a feature phrase in the review database with word distance between the words within the acceptable limit. If a compact feature phrase is found from such single word features a new feature phrase is created with word indices saved for the first word and the last word of the single word phrases while deleting the old single word feature phrases from the dataset. This increases the frequency of compact feature phrases, which I is relevant to this thesis since I am looking at the most frequent features in this project.

2.1.3.2 Redundancy Pruning

This method checks features containing one word for redundancy. To determine the redundancy of single word features Hu and Liu introduced the concept of p-support. The p-support of a feature *ftr* is defined as the number of sentences it appears in without the presence of a feature phrase that is a superset of *ftr*. If a feature has p-support of less than three and is a subset of a feature phrase int the review database is it deemed redundant and is pruned.

One might wonder about the usefulness of performing this redundancy pruning for my project since I am only focusing on the most frequent features while the pruning method only removes infrequent, single word features. However, without redundancy pruning an opinion could be matched with a redundant feature instead of a relevant feature since it was nearer to the opinion. It could be argued that the pruned feature was the feature modified by the opinion word since it was nearer to it than the other nearby non-pruned phrase but I make the assumption that the feature pruned is indeed uninteresting and irrelevant and is therefore unlikely to be the feature modified by the opinion, regardless of it's proximity to it.

2.1.4 Opinion Word Extraction

As mentioned in section 2.1.2, opinion extraction is performed in parallel with feature extraction. When parsing through each review all words and phrases labeled as JJP, VBPP or VBPDTP (explained in section 2.1.1) along with their review ID and indices.

Hu and Liu performed the extra step of collecting infrequent features with the use of adjectives and word distance. This step is unnecessary for this project since my method does consider infrequent features when generating opinion features along with the fact that I am only interested in the most frequent features.

2.1.5 Opinion Feature Generation

Six rules were created to control how candidate opinions and candidate features are matched together. These rules are explained below (with abbreviations explained in section 2.1.1)

Rule 0
If a candidate feature appears adjacent to a JJP with word distance $<= 4$ then and JJP is an opinion that modifies the feature. If more than one JJP are within word distance of 4 then the JJP with the shortest word distance is chosen as the opinion that modifies the feature.
Example: *"The camera is great."*

Rule 1
If a candidate reference-feature appears directly after a VBPP then the VBPP is an opinion that modifies the reference-feature.
Example: *"I love them."*

Rule 2
If a candidate reference-feature appears before a JJP with word distance $<= 3$ then the JJP is an opinion that modifies the reference-feature.
Example: *"It is fantastic"*

Rule 3
If a VBPP appears directly in front of a candidate feature then the VBPP is an opinion that modifies the feature.
Example: *"I love Nikon"*

Rule 4

If a VBPDTP appears directly in front of a candidate feature then the VBPDTP is an opinion word that modifies the feature.
Example: *"I love this player"*

Rule 5
If a look-feature appears directly adjacent to a JPP then JPP is an opinion word that modifies the look-feature.
Example: *"The phone is great looking"*

2.2 Sentiment Analysis

As mentioned in section 1.1 this thesis will examine and evaluate two different lexicons and two different machine learning techniques to find the best performing classification method for determining the sentiment of our review dataset. Lexical approaches can be as simple as ranking words in text according to a predefined list of words ranked by sentiment before calculating the sentiment for the text. Lexicons can also be used alongside grammatical analysis like part-of-speech for increased complexity and/or to help train classifiers where only unlabeled data is available so researchers must discover hidden structure in the data for successful classification. This process is called unsupervised learning. For this thesis I will only examine simple lexical methods using labeled lists for word matching.

The machine learning algorithms used is this thesis are Naive Bayes classification and Logistic Regression classification. Both these classifiers will use labeled data in the form of Amazon reviews to learn how to predict a class, a process called supervised learning[Moh].

These aim of these sentiment analysis methods is to accurately predict the sentiment of reviews for the summary of opinion features.

2.2.1 Lexical Analysis

Two different lexicons were used to classify reviews with regards to sentiment. The method used is very similar to the techniques implemented by Hu and Liu[Hua] in that it aims to use only opinions modifying opinion features to determine the sentiment of the sentence.

The first lexicon, obtained from Dobbs *et al.*[Dod][2], is called labMT 1.0 and contains 10.222 words evaluated for happiness by users on Mechanical Turk [3]. Each word is rated on a scale of 1-9, 9 being happiest. The second lexicon, obtained from Nielsen[Nie11], is called AFINN-111[4] and contains 2477 words evaluated by Nielsen for valence (sentiment) between -5 and +5, with +5 being the most positive. The lexicon was developed for sentiment analysis of microblogs such as Twitter and includes internet slang acronyms, such as *"WTF"* and *"LOL"*.

I ran the method twice on each lexicons, first it starts by collecting the opinion words (those that modify opinion features) of each review. It then tries to find a match for each opinion word in the lexicon. If a match is found for one or more of the opinion words the average sentiment rating per opinion word matched is calculated to determine the sentiment of the review. If no match is found for opinion words in a review it shifts to the traditional way of finding matches for all words in the review. This differs from the method of Hu and Liu[Hua] as they use the sentiment obtained for the previous sentence in the same multi-sentence review if sentiment can not be found for the opinion words. Fore my method, if one or more match is found the average sentiments for matched words is calculated to determine the sentiment of the review. If still no match is found the review is determined to be neutral for lack of sentiment indicators. If no opinion word is found in a review then the review is discarded, as only reviews that have opinion features in them are relevant when it comes to review summarization.

I then ran it again with the addition of intensifier considerations. Intensifiers are words that increase or decrease the significance of an opinion, e.g. *"very"* and *"extremely"*. This might b useful to increase the probability of a correct prediction if a review includes both a positive and a negative opinion word, but with one of these having an intensifier attached to it, giving it stronger sentiment than the other opinion word. Let's look at an example:

"The phone is extremely good with the exception of horrible color choices"

Here the review is more positive than negative, but without intensifier consideration the review would likely be classified as negative as "horrible" is stronger than "good". A list of 7 intensifiers was created and given a coefficient according to their strength. The list and ratings are as follows:

Extremely : 2.5
Very : 2
Really : 2

[2] http://s3-eu-west-1.amazonaws.com/files.figshare.com/360592/Data_Set_S1.txt
[3] https://www.mturk.com/mturk/welcome
[4] http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6010/zip/imm6010.zip

Quite : 1.3
Pretty : 0.8
Fairly : 0.7
Somewhat : 0.5

If these words appeared in front of an opinion word their sentiment ranking was multiplied by the intensifier's coefficient. For example, if the opinion phrase *"very good"* appeared in a review, the sentiment ranking for the opinion word *"good"* would be (very)*(good) = 2*3 = 6.

The first method was developed for a positive scale of sentiment ratings of words, i.e. from 1-9, like the labMT 1.0 lexicon[Dod] so the AFINN-111 lexicon[Nie11] had to normalized to that form for use with that method. The second method was developed for a sentiment scale centered around zero, i.e. negative words have negative numbers for ranking while positive words have positive numbers like the AFINN-111 lexicon, so the labMT 1.0 lexicon had to be normalized for that form for use with that method.

2.2.2 Machine Learning

The two algorithms used for supervised learning in this thesis are Naive Bayes and Logistic Regression. The philosophies behind these algorithms are quite different but they have both proven to be effective when it comes to text classification. When comparing these algorithms the concepts *generative* and *discriminative* are important. Naive Bayes is a generative model, meaning that it models the distribution of each class, i.e. how the data is generated, to categorize a signal. Logistic Regression is a discriminative model, meaning it models the decision boundary between classes and uses that information to categorize a signal[Ngb].

The research of Ng and Jordan [Ngb] showed when comparing the performance of these two models that discriminative models have a lower asymptotic error but generative models reach their asymptotic error faster. This means that for smaller training datasets generative models, such as Naive Bayes, will perform better but as the number of training examples grow discriminative models, such as Logistic Regression, will catch up and likely eventually perform better.

For this thesis these models will be trained on a *bag-of-words*, where each document/text is represented by a set of features $\{f_1, ..., f_m\}$. Examples of features are the word *"good"* and the bigram *"really bad"*. One way of constructing a feature vector for a document would be to let $n_i(d)$ be the number of times feature f_i appears in document d so that each document d is then represented by the

vector $\vec{d} := (n_1(d), ..., n_m(d))$. For this thesis however I will use Scikit-learn's [5] TdfidfVectorizer module to create a feature vector $\vec{d}$ for document d using tf-idf (term-frequency times inverse-document frequency) normalization. This re-weights feature counts into floating point values with the objective of giving frequent, yet less meaningful words, decreased significance. These words are for example *"the"*, *"a"* and *"is"*. The vector is ordered from highest frequency to lowest and if a limit is put on the size of the vector (i.e. number of features) used for a classifier less frequent features are ignored up to a predefined limit.

2.2.2.1 Naive Bayes

Generative classifiers such as Naive Bayes learn a model of the joint probability distribution $P(d, c)$, where d is the input data and c is the label/class. To assign label c^* to document d it must evaluate

$$c^* = arg\ max_c P(c|d)$$

where $P(c|d)$ is the probability of label c being assigned given document d. To derive this classifier it is observed that by Bayes' rule,

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

where $P(c)$ and $P(d)$ are the probabilities of c and d independent from each other and $P(d|c)$ is the probability of document d given a label c. In order to estimate $P(d|c)$ Naive Bayes makes the assumption that the f_i's are conditionally independent given label c:

$$P_{NB}(c|d) := \frac{P(c)(\prod_{i=1}^{m} P(f_i|c)^{n_i(d)})}{P(d)}$$

This assumption is said to be *naive*, the term that the approach draws its name from. This thesis will evaluate the two classic Naive Bayes variants used in text classification, Multinomial Naive Bayes and Bernoulli Naive Bayes. The former will estimate $P(c)$ and $P(d|c)$ by using relative frequency counting with Laplace smoothing while the latter uses a decision rule that explicitly penalizes the non-occurrence of a feature f_i that is an indicator for class c (while Multinomial Naive Bayes simply ignores a non-occurring feature).

Multinomial Naive Bayes implements the Naive Bayes algorithm for multinomially distributed data, typically represented as feature vector counts or td-ifd vectors.

[5] `http://scikit-learn.org/`

Bernoulli Naive Bayes requires samples to be binary values, i.e. $n_i(d)$ takes the value 1 for if feature f_i occurs in document d or the value 0 if it doesn't[SKL]. The Bernoulli Naive Bayes classifier module from Scikit-learn has a built in binarizer that maps sample features to boolean values if the input feature vector includes counts.

2.2.2.2 Logistic Regression

Despite its name, Logistic Regression is a linear model used to solve classification problems but not regression problems. Regression problems involve the predicting of a continuous target variable while classifications problems predict for only a small number of discrete values.[Nga]

Discriminative classifiers such as Logistic Regression learn a model of the conditional probability distribution $P(c|d)$. Logistic regression models the probabilities of possible outcomes of a single trial by using the logistic function[SKL]. This function is useful since its output always takes values between zero and one regardless of the size of positive or negative values of the input[Hos00]. The logistic function is defined as follows:

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

where t is the linear function of the exploratory variable. To prevent overfitting Logistic Regression uses regularization, with the most common methods being L1 and L2. For this thesis I will use L2 as it is expected to perform better than L1 on large datasets such as ours [AN].

Note that the definition of the Logistic Regression classifier reflects only on the occurrence/non-occurrence of a feature rather than directly incorporating feature frequency.

Chapter 3

Evaluation of Methods

In this chapter the methods detailed in chapter 2 will be evaluated for performance to determine the best performing techniques to be used for the third part of this project, the aspect-based review summarization. The evaluations will also be discussed and analyzed in this chapter along with descriptions of dataset processing.

3.1 Pre-processing the Datasets

Before using the datasets described in section 1.2 for the evaluation of the methods they needed to be pre-processed. This section details the pre-processing of each dataset.

3.1.1 Amazon Review Data for Supervised Learning

Each of the 83 million reviews obtained for this thesis have a rating from 1-5, 1 being most negative and 5 being most positive. To collect positive and negative reviews all reviews with ratings 1 and 2 were saved as negative and all who had a rating of 4 or 5 were positive while reviews with a rating of 3 were

left out as neutral. The result from this categorization were 69.152.357 positive reviews, 8.436.508 negative and 5.412.140 neutral. In order to have a random baseline classifier with 50% accuracy the number of positive and negative reviews included in the final dataset had to be balanced. After trimming symbols and punctuation the data was balanced, resulting in 8.435.324 positive reviews and 8.435.324 negative reviews.

3.1.2 Annotated Opinion Feature Mining Dataset

For this dataset, each review had been split into sentences. Each line in a review file contained a sentence from a review. If a feature was found in a sentence it would be listed in front of the sentence with sentiment and feature types in brackets. For this project I needed to parse through these reviews and scrape these features from each sentence along with the review text to save for later use.

3.1.3 Sentiment Labeled Twitter Dataset for Testing

As some tweets had been deleted from the time this list of IDs was compiled the result of this crawl was only 4620 tweets of the possible 5513 tweets. This included 324 positive tweets, 408 negative tweets, 972 neutral tweets and 897 irrelevant tweets. Tweets including links were dismissed due to likelihood of being spam along with retweets.

Tweets that were shorter than 4 words were also removed, as they are less likely to contain relevant information. User handles, hashtags, punctuation, numbers and symbols were removed from the remaining tweets. This resulted in 242 positive reviews and 274 negative reviews. They were then balanced, resulting in a final dataset of 242 positive tweets and 242 negative tweets.

3.1.4 Scraped Tweets for Summarization

This dataset was subject to the same trimming procedure as described in the second paragraph of section 3.1.3, sans balancing. This resulted in the following number of tweets for each queried product:

Apple TV : 3701
Chromecast : 2319

Firefox : 8658
Galaxy S5 : 1419
Google Chrome : 4521
HTC Desire : 520
PlayStation 4 : 4047
Windows 10 : 6819
iPhone 6 : 901

3.2 Experimental Set-up

After pre-processing of the required datasets the methods ready to be evaluated for performance. The evaluation of methods pertaining to opinion feature mining and sentiment analysis are described below.

3.2.1 Mining Opinion Features

To find the best performing technique for opinion feature mining experiments were conducted on the dataset described in section 1.2.2. The five products reviewed were two digital cameras, one mp3 player, one cellular phone and one DVD player. First every review were part-of-speech tagged. After that the pruning methods were applied and all combinations of the matching rules described in section 2.1.5 were executed to see what combination produced the best results. Note that rule 0 was always implemented so only the presence/absence of rules 1-5 was experimented with. The results of Hu and Liu[Hub] focused on examining the precision and recall for all features using different techniques. Those results are not directly relevant to this thesis but the manual feature count for each review can be used to find their top ten most frequent features. One problem did however arise when comparing results with the manual feature count obtained from Hu and Liu[Hub]. In a table displaying their results they claimed a certain number of features that were manually found for each product. However, when parsing through the review files and counting features manually labeled in the data the results were very different. Below is their count for each product followed by a dash and then my count from their annotation:

Digital camera 1 : 79 - 105
Digital camera 2 : 96 - 75
Cellular Phone : 67 - 115

Table 3.1: Comparison of top 10 most frequent features found by the best performing method from this system and those found manually by Hu and Liu [Hub]

Product	Matches	Matched Count	True Matched Count	Matched Count Accuracy	Top 10 Accuracy	Pruning	Rule
Apex	5	95	162	58.6	49.6	Redundancy	01101
Canon	4	111	84	132.1	95.71	Redundancy	01010
Creative Labs	4	145	222	65.3	55.07	Redundancy	00010
Nikon	4	80	70	114.3	79.09	Redundancy	00010
Nokia	4	117	84	139.3	81.21	None	01010

Mp3 player : 57 - 188
DVD player : 49 - 116

It appears that they processed the labels after annotating them, probably grouping them together or otherwise modified them. Attempts to manually replicate grouping of these features were unsuccessful. This discrepancy effects the evaluation of my methods as it is not possible to get an accurate estimate on how they have performed. I will however compare my results to the ones discussed above to get a comparison, however flawed it may be.

The objective of this part of the thesis is to find the most frequently occurring opinion features. Table 3.1 shows comparison of top 10 most frequent features found by this system and the ones manually found by Hu and Liu[Hub]. The column *Matches* shows how many of the top ten features from their manual list was found, *Matched Count* shows a sum of how often these matched features were found, *True Matched Count* show how often they found the matched features, *Matched Count Accuracy* shows *Matched Count* divided by *True Matched Count* as a percentage, *Top 10 Accuracy* shows my total count of the *True* top 10 features divided by the *True* total count of the *True* top 10 features, *Pruning* shows which pruning method was used to obtain the best result and *Rule* shows the (fewest) rules needed to obtain the best result. The best result was defined as the highest *True Matched Accuracy*. The binary representation indicates whether a rule was included or not, where the first digit represents the presence/absence of rule 1 and so on.

When comparing with the (flawed) manual feature count of Hu and Liu[Hub] it can be seen that the opinion feature mining algorithm does not perform particularly well, finding only four or five features from their top ten. The *Matched Count Accuracy* ranges from 58.6% to 139.3%, which is perhaps not a very telling statistic since having accuracy of 139.3% implies that there opinion features being found where they shouldn't be found. It can then be seen that the *Top 10 Accuracy* ranges from 49.6% to 95.71%, a more reasonable range. It must however be considered that the high accuracy of 95.71% is likely not

accurate as some opinion feature count accuracies are low while some are way too high (e.g. 139.3%), giving the average accuracy of 95.71%.

It is clear that without having the accurate manual feature count for these reviews significantly impairs any estimation of the opinion feature mining algorithm's true performance.

The minimal combination of rules needed to obtain all the results in table 3.1 is *01101*, i.e. rules 2, 3 and 5 are included (along with the ever present rule 0) while rules 1 and 4 are not used. I assume that the inclusion of a rule can not harm results. This combination along with redundancy pruning will be used to mine opinion features for the review summary.

3.2.2 Sentiment Analysis

Lexical analysis
To test the lexical methods the best performing opinion feature mining algorithm is used on the labeled Twitter dataset described in section 1.2.3 to obtain opinion features for those reviews. The sentiment of the reviews were then calculated according to the descriptions of the lexical approaches from section 2.2.1. The result can be seen in table 3.2 where row (1) and (2) shows how many times sentiment was determined from opinion words and from the text in the review, respectively, row(3) shows how many times the algorithm was unable to obtain sentiment from either opinion words or text, row (4) and row (5) show the sentiment prediction accuracy when only opinions are used and when only text was used, respectively, row (6) shows the sentiment prediction accuracy when opinions and text is used and row (5) shows the accuracy when all reviews are included. *Std.* stands for the standard method while *Int.* stands for the method including intensifier consideration while labMT and AFINN are short for labMT 1.0 and AFINN-111, respectively.

It can be seen that using the intensifiers did not improve accuracy but rather decreased it. The explanation could be that for this particular dataset there are multiple negations, effectively increasing the probability of an inaccurate prediction by taking a strong opinion and categorizing it as having the opposite sentiment. It can be seen that the large size of the labMT 1.0 lexicon has resulted in no undetermined reviews. However, the AFINN-111 lexicon performs much better than the labMT 1.0, which may be caused in part by the micro-blog vocabulary focus of AFINN-111. The low accuracy of the labMT 1.0 lexicon stems from a high *False Positive* count, for which reasons or not clear. While the accuracy of the AFINN-111 lexicon is considerably better than that of labMT 1.0 it is still not what would be called good when it comes to senti-

Table 3.2: Lexical analysis results

		Std. labMT	Std. AFINN	Int. labMT	Int. AFINN
(1)	**Opinions**	275	117	275	117
(2)	**Text**	27	143	27	143
(3)	**Neither**	0	42	0	42
(4)	**O accuracy [%]**	63.3	75.2	63.1	75.0
(5)	**T accuracy [%]**	40.7	70.6	39.3	70.3
(6)	**O+T accuracy [%]**	61.3	72.7	60.9	72.3
(7)	**O+T+N accuracy [%]**	61.3	62.6	60.9	62.3

ment classification of text. It can be seen that for the AFINN-111 lexicon that the relatively high number of unpredicted reviews strongly affects the results when they are included. Leaving them out in order to include only sentiment predicted reviews would, in this case, reduce the size of the dataset by 13.9%. The best accuracy, 75,2%, was achieved using the standard method with the AFINN-111 lexicon and only including reviews which yielded a sentiment classification through the use of opinion words. However, to achieve that accuracy 61,3% of the dataset would have to be discarded, which is not ideal. The optimal way to solve this problem would be to improve the opinion feature mining algorithm, possibly resulting in more opinion features which can be matched to the lexicon. The improvement could include an increased sophistication and complexity in how it analyses words and word relationships (e.g. prepositions), using other lexicons, combining lexicons and/or combing the lexical approach with a machine learning approach. Other ways to improve results could be to try the inclusion of stemming, more intricate intensifier consideration and/or negation consideration.

Machine Learning

To test and train the two machine learning algorithms the Amazon review dataset described in section 1.2.1 is split in to two parts, 75% for training and validation and 25% reserved for testing. The first part is used for 3-fold cross-validation, where the data is split into three folds and then iteratively trained on two folds and validated on one until all three combinations have been executed. The result is the average accuracy from these three runs. This is done to minimize variance from irregularities in data. It is not uncommon to use more than 3 folds for cross-validation, with 10 being the most popular, but for this thesis I only used three due to time constrictions. The test set is kept com-

pletely unused during the tuning of the classifier. This is done so that when the classifier is used on the test set it gives a better estimate on how the classifier will perform on unseen data.

The classifiers used the bag-of-words framework for training and predicting. The features focused on were ngrams of sizes one and two, also known as unigrams and bigrams[Bro]. When training classifiers it is important to find the correct fit for the data, i.e. to avoid under- or overfitting. Underfitting in when the classifier is not complex enough to accurately predict the classes of the data input, e.g. using too few features. Increasing the amount of data will likely not improve results for a model that underfits. Overfitting is when the model is too complex and is fitting noise rather than the real tendencies of the data. The training error of such a model is likely to be low but it will likely perform poorly on new data[JV].

To find the amount of features best suited for this dataset I used cross-validation for the Multinomial Naive Bayes classifier on the data for various amount of unigram features. The results of this evaluation can be see in Figure 3.1. On the left side of the graph where there are few features it can be seen that the model is underfitting since it does equally poorly on both unknown and known values. On the right side it is starting to overfit, as can be seen by the divergence of training- and cross-validation errors. The optimal model is where the cross-validation error is minimized[JV]. For this model the cross-validation error reaches similarly low values from around 30.000 features to around 300.000 features. I will choose the number of features to be the 29.958 most common features from the tf-idt vector, which are features that appeared at least 700 times, to minimize computational time and memory usage.

To verify the that the chosen feature amount is a good fit the learning curve of the Multinomial Naive Bayes classifier is plotted using 29.958 features. The learning curve, seen in figure 3.2, shows the training and cross-validation accuracy as a function of training data size. It can be seen that, as was expected, that the model is underfitting on the left side of the graph but does not overfit as the training data size is increased so the amount of features chosen fits this model well. It can however be seen that at around 5 million reviews the model has reached the point where more data does not improve the results. It would therefore be possible to train this model on only 5 million reviews and still get very similar results as with more reviews.

These 29.958 features were used for all classifiers for both unigrams and bigrams. It's worth noting that when bigrams are used as features each bigram counts as only one feature.

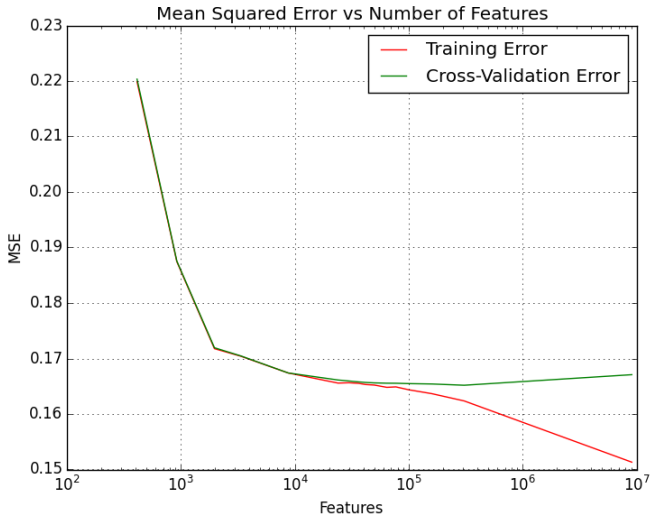


Figure 3.1: Mean Squared Error vs Number of Features for Multinomial Naive Bayes

Figure 3.2: Learning curve of Multinomial Naive Bayes for 29.958 features

As stated in chapter 2 I used L2 regularization with the Logistic Regression classifier in this thesis. A parameter that needed to determined before using the classifier is C, the inverse of regularization strength. C controls how strong the regularization should be, with smaller values specifying stronger regularization. To find the optimal C value for this model using 29.958 unigrams I again use cross validation to test and plot for different Cs. The result of this evaluation can be seen in figure 3.3. A low C value produces a poorer result than higher values, with higher values having no hint of overfitting. As before when optimizing the feature count the optimal value for C is found where the cross-validation error is lowest. For this model this happens when C is 10^2, or 100. The improved performance for larger Cs was expected as strong regularization was not needed since the amount of features had already been chosen, reducing the risk of overfitting.

To verify this result the learning curve of the Logistic Regression classifier is plotted using 29.958 unigrams as before with C set to 100. The result can be seen in figure 3.4. The curves are confirm our assumptions of a good fit and show that for C = 100 the accuracy looks to be near its maximum possible value for this model when all the training data is used. I will therefore set C to 100 when using the Logistic Regression classifier in this thesis.

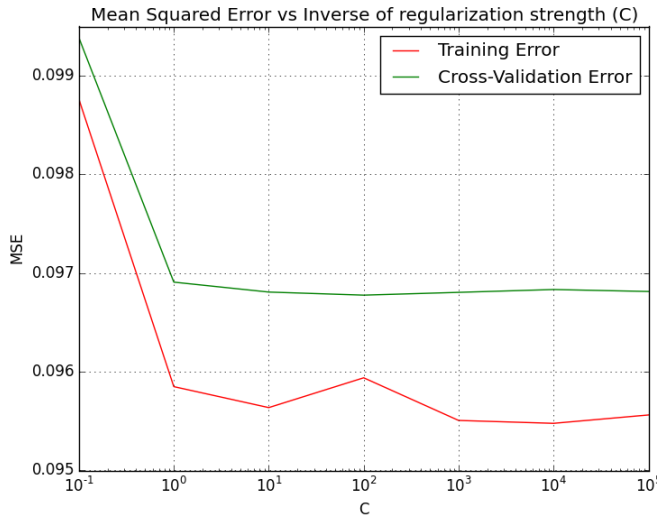


Figure 3.3: Mean Squared Error vs C for Logistic Regression

The cross-validation error of the classifiers, using the optimal parameters determined above, are displayed in table 3.3, where it can be seen that Logistic Regression performs better than the Naive Bayes classifiers, reaching 93.1%

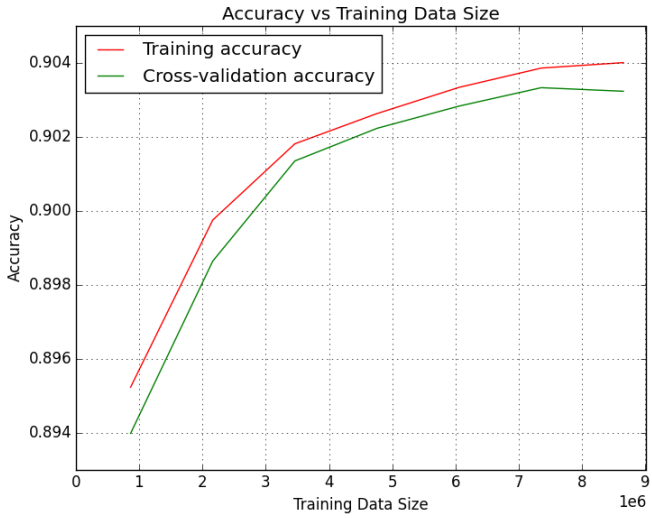


Figure 3.4: Learning curve of Logistic Regression for 29.958 features and C value of 100

accuracy when using bigrams as features. The unigram-based Logistic Regression classifier performs a little better than the bigram-based Multinomial Naive Bayes classifier with 90.3% accuracy versus 88.9%. The rest of the classifiers all performed considerably worse, with the poorest performer being the unigram-based Bernoulli Naive Bayes classifier.

Next the performance of the best performing classifiers on the test- and labeled Twitter set is examined. The results from using the bigram-based Logistic Regression classifier can be seen in line (1) of table 3.4. It can be seen that even though the accuracy is 93.4% for the test set it is only 72.8% for the Twitter set. Lines (2) and (3) show that the unigram-based Logistic Regression classifier and

Table 3.3: Average accuracies from 3-fold cross-validation in percentages for all classifiers and ngrams

	Features	**Number of features**	**Frequency/ Presence**	**NB**	**LR**
(1)	unigrams	29958	Frequency	83.4	N/A
(2)	bigrams	29958	Frequency	88.9	N/A
(3)	unigrams	29958	Presence	80.8	90.3
(4)	bigrams	29958	Presence	86.0	93.1

Table 3.4: Accuracy results for test- and Twitter set by best performing classifiers

	Classifier	Features	Frequency/ Presence	Test set	Twitter set
(1)	LR	bigrams	Presence	93.4	72.8
(2)	LR	unigrams	Presence	90.4	75.0
(3)	NB	bigrams	Frequency	89.0	72.5

the bigram-based Multinomial Naive Bayes classifier perform well on the test set, albeit not as well as the bigram-based Logistic Regression classifier while the unigram-based Logistic Regression classifier improves the accuracy of the Twitter set. to 75% accuracy. It seems that since the real review data used for summarization in this thesis will be more like the Twitter set I chose to continue with the unigram-based Logistic Regression classifier ahead of the its bigram-based counterpart. It can be seen that the Logistic Regression classifier produces better results using bigram features on the test set while producing better results on the labeled Twitter set using unigram features. A possible explanation is that using ngrams of higher degree tends to work better on longer text[Ber]. Since the test set includes longer reviews than the tweets, which by design are limited to 140 characters, it could be expected that using bigram features produces better results on the test set while using unigram features produces better results for the labeled Twitter set.

I am interested in trying to increase the accuracy of the unigram-based Logistic Regression classifier for the final summarization task since 75.0% accuracy is not particularly high. Since it is possible to retrieve the prediction probability from the Logistic Regression classifier for each document classified that information can be used to inspect the feasibility of increasing prediction accuracy by increasing the probability confidence needed to include the prediction in our results. In other words, instead of including all reviews determined to be either positive or negative I only include reviews that the classifier deems is more than $X\%$ probable of being either positive or negative, with X being some predefined number. Figure 3.5 shows the trade-off between accuracy and amount of data discarded vs. prediction confidence. It can be seen that by increasing the probability confidence the accuracy is simultaneously increased and while the amount of data that is used is decreased. The objective here is optimizing the trade-off between increased accuracy and decreased data size. Accuracy of 85% was deemed to be acceptable considering the trade-off in data size decrease of 40%. This means that all predictions that have less than a 78.5% probability of being either positive or negative are discarded.
Finally the mean squared error of the test set and the Twitter set is evaluated as a function of training data size for the unigram-based Logistic Regression

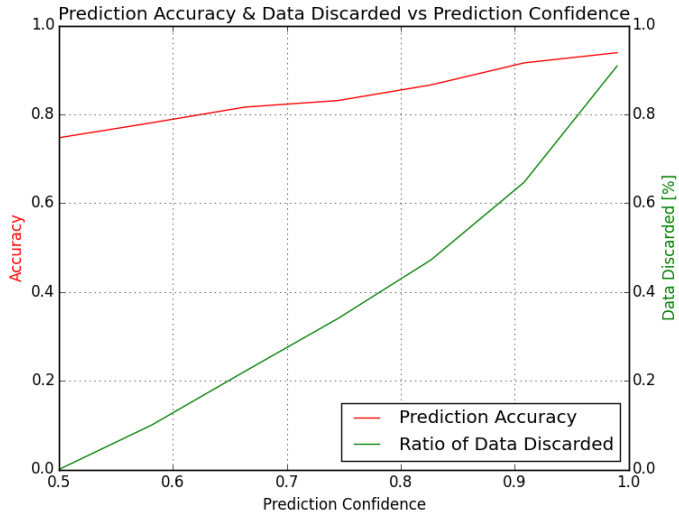


Figure 3.5: Accuracy and Data Discarded vs Prediction Confidence for the unigram-based Logistic Regression classifier

classifier. The result of this evaluation can be seen in Figure 3.6. It can be seen that even though the error is much lower for the test set than for the Twitter set the error behaves very similarly with increased training data size. The graph supports the earlier observations of stagnating accuracy for this model with increased training data size.

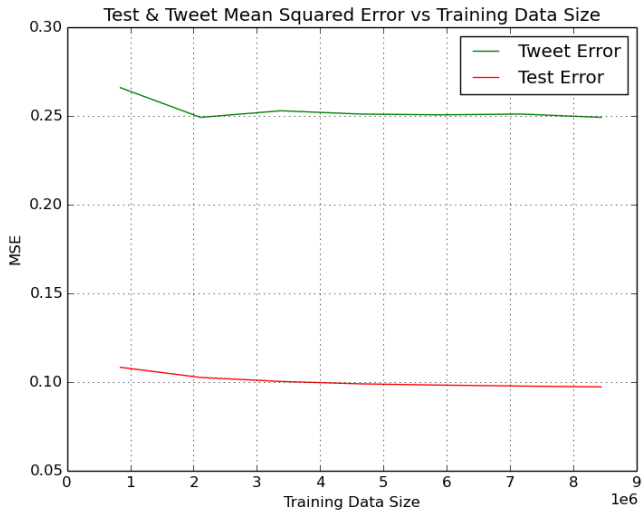


Figure 3.6: Mead Squared Error of the test- and Twitter set for the unigram-based Logistic Regression classifier

Chapter 4

Summarization Results and Discussions

The first step in creating a sentiment summary of the most frequent opinion features is finding the opinion features from all reviews of a product using the best performing opinion mining techniques, redundancy pruning with rules 0, 2, 3 & 5. Then the sentiment of each review is determined using the best performing classifier for our data, Logistic Regression with 29.958 unigram features and C = 100 with a 78.5% prediction confidence level. Each opinion feature is labeled with the sentiment of the review they appeared in. If the prediction confidence is below 78.5% the review is not given a sentiment rating. Next the unique opinion features found in the reviews are counted to find the ten most frequent opinion features. When they have been found the positive and negative occurrences of each opinion feature are counted. Reviews with no sentiment rating due to low prediction confidence are ignored and as a result the opinion features that appear in that review are also ignored. The result is an aspect-based opinion summary of reviews of that product on social media.

Experimental results using this setup on the scraped tweet dataset described in section 1.2.4 showed that a few irrelevant words were slipping into the top ten features for some of the products, such as *"im"*, *"yeah"* and *"hello"*. This is the result of the part-of-speech tagger incorrectly tagging these words as nouns or noun phrases because of word placement, context or inaccuracies in the tagger. To solve this a list of banwords was created to keep these irrelevant words

from being a top ten most frequent feature.

A proof of concept for the aspect-based opinion summarization system can be seen in figure 4.1, demonstrating a summary of the ten most frequent opinion features in reviews regarding the streaming product Chromecast. The summary was created automatically from the data obtained with the methods described in this thesis. The picture of the product was scraped from the first image result of Google Images for the search query *"Chromecast"*. It can be seen that the opinions regarding the top ten features are more negative than positive, with 60.3% of reviews being negative.

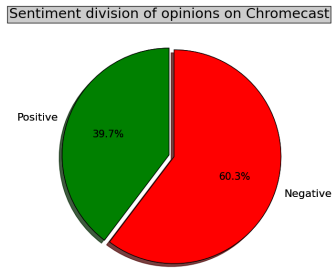
Not all opinion features are what might be called useful, for example the opinion feature *"way"* has little or nothing to do with the product. The report shown in figure 4.1 was chosen as the best result from the small collection of products summarized. Review summaries of the other products mentioned in section 1.2.4 can be seen in Appendix A. They show that the summarization system does include several words as opinion features that have little or nothing to do with the products. It is possible that some products, such as web-browsers Firefox and Google Chrome, are not as well suited with this kind of feature summarization as other products that have more core features that are likely to be the topic of conversation. Overall the opinion feature mining could be improved by making the opinion feature mining process more complex, for example using association rule mining, and doing a thorough analysis of prepositions.

The amount of opinion features seem low compared to the amount of reviews collected. The result of only including reviews that were predicted with more than 78.5% confidence reduced the number of opinion features by an average of 53.6% per top feature. Recall that the expected decrease in number of reviews given a 78.5% prediction confidence level was 40%. The actual number of reviews discarded is 54.5%, a significantly higher ratio than expected. By increasing the base accuracy of the classifier from the 75% it is currently achieving for reviews from Twitter the amount of data discarded would be lower, resulting in more opinion features for the summarization.

In order to be viable as a business solution the generation of the report would have to focus on making clear and pleasing visualizations of the data. This prove of concept shows that manipulating data for a visual result can be achieved with minimal work but in order to make this tool an attractive option for businesses to use the work on visualization would need to be improved significantly.

Frequent feature report for Chromecast

Tue Aug 04 09:36:13 2015



Here are the most frequent features people expressed their opinion of regarding Chromecast:

Feature	:	tv	Feature	:	app
Positive	:	14	Positive	:	8
Negative	:	12	Negative	:	17
Ratio	:	54% Positive	Ratio	:	68% Negative
Feature	:	support	Feature	:	amp
Positive	:	1	Positive	:	11
Negative	:	19	Negative	:	1
Ratio	:	95% Negative	Ratio	:	92% Positive
Feature	:	way	Feature	:	roku
Positive	:	1	Positive	:	7
Negative	:	8	Negative	:	0
Ratio	:	89% Negative	Ratio	:	100% Positive
Feature	:	work	Feature	:	microsoft
Positive	:	3	Positive	:	0
Negative	:	4	Negative	:	7
Ratio	:	57% Negative	Ratio	:	100% Negative
Feature	:	screen	Feature	:	month
Positive	:	2	Positive	:	3
Negative	:	5	Negative	:	3
Ratio	:	71% Negative	Ratio	:	50% Positive

Figure 4.1: Summarization report for frequent opinion features for Chromecast

Chapter 5

Conclusion and Future Work

This thesis covered the topic of aspect-based opinion summarization system for product reviews on social media and demonstrated with a proof of concept summary report that creating an automatic system capable of scraping reviews off social media and analyzing it to obtain valuable information regarding product features is possible. I was able to automatically create reports for several products using a opinion feature mining algorithm and sentiment analysis. The quality of the reports did however differ, suggesting that some product types might be more suited for this summary system than others.

Although the results from the summarization system are encouraging there is room for improvement. When comparing our results to the manual feature count of Hu and Liu[Hub], even though it is flawed to extent, the results suggested that the accuracy in identifying opinion features in reviews could be improved. A better estimate of the performance of our opinion feature mining algorithm could be obtained by locating another dataset labeled for features or by manually labeling data with regards to features. Future work focusing on improving the opinion feature mining algorithm could include the used of association rule mining and the pruning methods used by Hu and Liu[Hub] with the addition of analysis of prepositions.

The results from the sentiment analysis looked promising as the bigram-based

Logistic Regression classifier achieved a 93% accuracy rate for the test set. It did however achieve much lower accuracy on the labeled Twitter dataset, along with all other classifiers tested (with highest accuracy achieved for the labeled Tweets being 75%). This is likely a result of incompatibility between the training set of Amazon reviews and product reviews in the form of tweets. For the people writing the difference in platform and text length could contribute to the difference in text structure, with tweets assumingly more likely to include internet slang, acronyms, incorrect spelling (intentional or not) and less formal speech than the Amazon reviews. By including only tweets where prediction confidence was over 78.5% the accuracy increased to 85% but resulted in significant loss of data that did not meet the prediction confidence threshold.

Future work might include using these classifiers on training data similar to the tweets used for the summarization reports. This data could be obtained from the web, where there are several available datasets consisting of tweets labeled with regards to sentiment. It must be noted though that manually labeled Twitter sets currently available are quite small while many of the large tweet datasets available have been labeled using emoticons as sentiment indicators, which could influence accuracy since the text structure used by people who use emoticons could possibly be different for those who don't. It would also be possible to obtain better results by using stemming and fuzzy matching to try transform text from tweets to a more standard form.

Another way of improving the accuracy classifiers would be to re-weigh the prior probabilities of the classifiers when training them to include all training data available. For this thesis the training data was balanced before use, ignoring in the process around 60 million positive reviews. It is however possible that re-weighing the prior possibilities of the classifiers could improve prediction accuracy for the test set without improving the accuracy for the tweets.

Bibliography

[Agr] Srikant R. Agrawal, R. Fast algorithms for mining association rules. *VLDB'94.*

[AN] ICML 2004 Andrew Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.9860&rep=rep1&type=pdf`. Accessed: 2015-07-31.

[Ber] Smeaton AF. Bermingham, A. Classifying sentiment in microblogs: is brevity an advantage? *ACM, pages 1833–1836.*

[Bir09] Klein E. Loper E. Bird, S. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit.* 2009.

[Bol] Mao H. Zeng X. Bollen, J. Twitter mood predicts the stock market. *Journal of Computational Science, Volume 2, Issue 1, March 2011, Pages 1–8.*

[Bro] A. Broder. On the resemblance and containment of documents. *Proceedings of Compression and Complexity of Sequences. 21–29.*

[Che11] Lazer M. Chen, R. Sentiment analysis of twitter feeds for the prediction of stock market movement. 2011.

[Dav] Lawrence S. Pennock D. Dave, K. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *WWW '03.*

[Dod] Harris K. Kloumann I. Bliss C. Danforth C. Dodds, P. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PLoS ONE, 6(12):e26752.*

[Hos00] Lemeshow S. Hosmer, D. *Applied Logistic Regression (2nd ed.).* Wiley, Needham MA, USA, 2000.

[Hua] Liu B. Hu, M. Mining and summarizing customer reviews. *Proceeding of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 168–177, 2004.*

[Hub] Liu B. Hu, M. Mining opinion features in customer reviews. *Proceedings of Nineteenth National Conference on Artificial Intellgience.*

[JV] Astro 599 course Jake Vanderplas. Machine learning with scikit-learn: Validation and model selection. `http://www.astro.washington.edu/users/vanderplas/Astr599/notebooks/18_IntermediateSklearn`. Accessed: 2015-07-31.

[Liu] Huang X. An A. Yu X. Liu, T. Arsa: A sentiment-aware model for predicting sales performance using blogs. *SIGIR 2007 Proceedings, Session 25: Combination and Fusion.*

[Liu98] Hsu Y. Wong C. Liu, B. Integrating classification and association rule mining. 1998.

[McA] Pandey R. Leskovec J. McAuley, J. Inferring networks of substitutable and complementary products. *KDD, 2015.*

[Moh] Rostamizadeh A. Talwalkar A. Mohri, M. Foundations of machine learning. *The MIT Press.*

[Mor02] Yamanishi K. Tateishi K. Fukushima T. Morinaga, S. Mining product reputations on the web. *KDD '02 Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining Pages 341-349*, 2002.

[Nas12] Zargham P. Mahalati R. Nassirpour, S. Electronic devices sales prediction using social media sentiment analysis. 2012.

[Nga] Andrew Ng. Cs229 lecture notes, supervised learning. `http://cs229.stanford.edu/notes/cs229-notes1.pdf`. Accessed: 2015-07-31.

[Ngb] Jordan MI. Ng, A. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Adv. Neural Inform. Process. Syst. 14: 605-610.*

[Nie11] F. Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages 718 in CEUR Workshop Proceedings : 93-98. 2011 May.*, 2011.

[OFM] Customer review datasets (5 products). `http://www.cs.uic.edu/~liub/FBS/CustomerReviewData.zip`. Accessed: 2015-06-15.

[oSMFPTtIOI] The Growth of Social Media: From Passing Trend to International Obsession [Infographic]. Kimberlee morrison.

[Pan] Lee L. Vaithyanathan S. Pang, B. Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 79–86.*

[sFTS] 1st Financial Training Service. Trainer's toolkit.

[Si13] Mukherjee. A. Liu B. Li Q. Li H. Deng X. Si, J. Exploiting topic based twitter sentiment for stock prediction. *Proceedings of The 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013, short paper), August 4-9, 2013, Sofia, Bulgaria*, 2013.

[SKL] Scikit-learn, naive bayes. `http://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes`. Accessed: 2015-08-1.

[Twi] Twitter facts from company website. `https://about.twitter.com/company`. Accessed: 2015-05-22.