NTNU
Norwegian University of
Science and Technology

Sentiment Analysis on User-Based Reviews: Movie Recommendation Case

Aina Elisabeth Thunestveit

Master of Science in Informatics
Submission date: June 2016
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Preface

This master thesis is written as part of the study program Informatics at the Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis is 50% of the required work to achieve the master in artificial intelligence, Informatics. The idea of the thesis was brought up by professor Heri Ramampiaro of NTNU.

Trondheim June 15, 2016

Aina Elisabeth Thunestveit

Acknowledgments

I would like to thank my supervisor Heri Ramampiaro, for invaluable response and guidance in this project. He always had faith in me and urged me to do better. I am very much thankful to my cohabitant Marius for quality assurance and proofreading of the thesis. Finally I would like to express my gratitude to my family and friends for motivation and support.

Samandrag

Mange menneske brukar Internett som ein stad for å leita etter andre sine meiningar om ting. Med den aukande mengda av brukarbasert innhald på nettet, har det vore ei stor framvekst av forskningsfelt som brukar sentiment analyse for å utnytta denne informasjonen. Dette kan resultera i fleire tilfredse kundar, sidan relevant informasjon vil vere enklare å få tak i.

I denne forskninga, presenterar me ein ny metode, basert på å finna og å analysera adjektiv frå brukarbaserte kommentarar på nettet. Me utnyttar ideen om at adjektiv ofte inneheldt ei kjensle, og presenterar eit system som baserar seg på at adjektiv kan bestemma kjensla i eit utsagn.

I eksperimenta våre, brukar me klassiske maskinlæringsteknikkar som Random Forest og Support Vector Machines for å forutsjå positive og negative kjensler i kommentarane. Samanlikna med baseline, har me forbetra resultatet i alle eksperimenta våre. Resultata visar at med ein nøyaktigheit på 94.7%, har systemet vårt eit betre resultat enn state-of-the-art metode på eit stort datasett, beståande av 50 000 filmkommentarar.

Abstract

Many people use the Internet as a place for seeking opinions. With the increasing amount of user-based content on the web, there has been an emergence of research fields that uses sentiment analysis to take advantage of, and process this data. This could result in more satisfied customers, as relevant information will be easier to find.

In this research, we present a new method, based on extracting and analyzing adjectives from user-based reviews. We exploit the idea that adjectives often contain a sentiment, and present a system entirely based on adjectives as sentiment deciders.

Our experiments uses classical machine learning techniques like Random Forest and Support Vector Machines to predict the sentiment orientation of the reviews. Compared to baseline, our system improved results in all our experiments. Results also shows that with an accuracy of 94.7%, our system performs better than state-of-the-art approach on a large dataset consisting of 50 000 movie reviews.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The data available on the web is increasing every day. In 2013, a research report from SINTEF shows that 90% of all the data in the world has been generated over the last two years [1] and the numbers are still increasing. Recent statistics[1] tells us that the World Wide Web contains at least 4.55 billion web pages. The size of the generated data can be overwhelming, and it can be difficult to find the information we are seeking. An important part of gathering information about something new, have always been to find information about what other people think. When decisions must be made, it is human nature to seek advice and recommendations from trusted peers. For example, we ask our friends for recommendations on what phones to buy, and our professors what articles we should read. With a growing availability of opinion-rich resources on the Internet, such as review sites, comment systems and personal blogs, it has become possible to find opinions from people we have never met before.

The explosive growth of online data presents buyers with an increasing range of choices, while sellers meet the challenge of personalizing their advertisement. Recommender systems have evolved to fulfill these natural needs from the buyers and sellers by automating the recommendations based on data analytics [2]. This project will further investigate how we can process natural language text, finding the underlying opinion. For this task, we will use information stored in user-based reviews, more specifically comments about movies. Note that the research can be applied to all sorts of products or services, where a comment/review system is available.

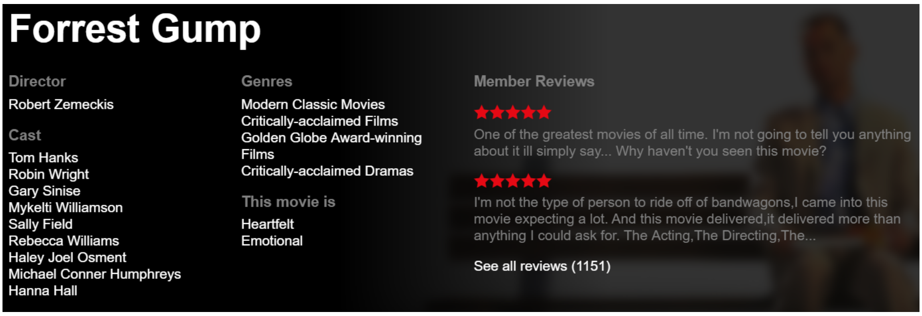
[1] `www.worldwidewebsize.com`. Retrieved May 29th 2016.

1.1 Motivation

According to a survey from 2007 of 2400 adult Americans, by Pew Internet & American Life Project [3]:

- 81% of Internet users have researched information on the Internet about a product they are thinking about buying.
- 20% are doing this on the typical day.
- 79% of Internet users are confident in making the correct decision, as they gather information online in advance of buying something.

As we can see, many people uses Internet as a place for seeking opinions, and a majority is confident in making the correct decision doing so. But, from the same survey [3], 43% of the people say they have been frustrated by the lack of information, 32% have been confused and 30% have felt overwhelmed by the amount of information found, while seeking opinions before a purchase. Although this is a rather old survey, there is no reason to believe that this problem is solved. With the increasing amount of data and comments, there is almost impossible to read it all. Most people are satisfied by reading the first few comments. A tool that can go trough all the comments of a system, rank them positive or negative and summarize them can be extremely powerful in the recommendation area.



Forrest Gump

Director
Robert Zemeckis

Cast
Tom Hanks
Robin Wright
Gary Sinise
Mykelti Williamson
Sally Field
Rebecca Williams
Haley Joel Osment
Michael Conner Humphreys
Hanna Hall

Genres
Modern Classic Movies
Critically-acclaimed Films
Golden Globe Award-winning Films
Critically-acclaimed Dramas

This movie is
Heartfelt
Emotional

Member Reviews

★★★★★
One of the greatest movies of all time. I'm not going to tell you anything about it ill simply say... Why haven't you seen this movie?

★★★★★
I'm not the type of person to ride off of bandwagons,I came into this movie expecting a lot. And this movie delivered,it delivered more than anything I could ask for. The Acting,The Directing,The...

See all reviews (1151)

Figure 1.1: Member reviews for the movie Forrest Gump. Screenshot from Netflix[2].

From the sellers point of view, a good recommender system or a good system for gathering information on products have a big marketing value. The buyers get recommended products they want, which they may not have thought of beforehand or

[2] `http://www.netflix.com/`. Retrieved May 30th 2016.

planned on purchasing. As mentioned earlier, this research will focus on comments about movies. In this case, it can be easier for the user to find movies matching their taste, and it can be easier for the provider to hold on to their customers.

Netflix[3] is an example of a global provider of a streaming service. Their main product is a subscription service that allows members to stream any movie or television show in their collection at any time. In December 2015 they had more than 65 million members, which streamed more than 100 million hours of movies and television shows per day [4]. In October 2006, they opened a competition for the best collaborative filtering algorithm to predict user ratings for movies, based on previous ratings, called the Netflix Prize[4], with a grand prize of $1,000,000. Collaborative filtering can be read more about in section 2.2.1. As long as the competition was still active (nobody won the grand prize), it was also possible to win a progress prize of $50,000. It were given each year, to the best entry thus far, that also improved the previous progress prize winner by at least 1%. 3 years after the start of the competition, the team “BellKor’s Pragmatic Chaos” won the grand prize for improving Netflix’s own algorithm with more than 10%. The fact that Netflix was willing to give away that amount of money, shows the importance of their recommender system, as they said themselves in the rules for the competition[5] “Because, frankly, if there is a much better approach it could make a big difference to our customers and our business”.

An adjective is defined by the Oxford Dictionary as “A word naming an attribute of a noun, such as sweet, red, or technical”. In other words, it is a word describing something. Our theory is that adjectives often contain a sentiment. For example “fantastic”, “happy”, “perfect”, “dull” and “boring”. We believe that if we can extract those adjectives and find their sentiment, we can improve sentiment classification.

1.2 Objective and research questions

Most subscription or e-commerce systems, contain a form of review system, but they are not always exploited to the fullest. Meaning the comments are not extracted, summarized or taken advantage of, beside showing comments that people can read. A limitation of such a system is that a user most likely don’t read more than a couple of the first comments, resulting in a user missing out on relevant information. For example, we are looking for a hotel to spend the night. The tenth comment, that we never read,

[3] http://www.netflix.com/

[4] http://www.netflixprize.com/

[5] http://www.netflixprize.com/rules

told that the beds were really hard. Since we never read that comment, we ordered the hotel and got disappointed of the beds. If the recommender system uses all the comments of a movie or a product, the user may experience better recommendations and more satisfaction in their own choices of products.

The scope of this thesis is limited to testing out theories about how we can extract opinions from natural language text. The scope do not include developing a recommender system. Our idea is that adjectives often determines the sentiment of a sentence.

The main research goal of this thesis is to investigate whether it is possible to improve sentiment analysis techniques on user-based reviews, by the use of adjectives.

To achieve our goal, there are several aspects that must be studied, which can be condensed into the following research questions:

RQ1 What recommender and sentiment analysis approaches exist today?

RQ2 How can adjectives be exploited in the area of sentiment analysis?

RQ3 How can the use of adjectives improve results obtained by classical classification methods?

1.3 Project scope

This section will describe the scope of the project, and briefly outline the preconditions and constraints that narrows down the focus area of this thesis.

1.3.1 Dataset

We experimented with three different datasets in this research. We could use any set of product reviews for the project, but we have chosen and limited it to movie reviews. In other words, we can say that we use movie reviews as an “use case” for our project.

Our first experiments, uses a corpus based on positive and negative movie reviews from IMDb. We gathered 1000 positive and 1000 negative reviews from the dataset “polarity dataset v2.0”[6] first introduced in [5] in 2004. This datasest is often used as a benchmark

[6] https://www.cs.cornell.edu/people/pabo/movie-review-data/

for sentiment analysis.

Further, we combined this first dataset with objective texts. The motivation behind this, was to see how our system performed, when noise were included in the dataset. We gathered objective texts from the "CMU Movie Summary Corpus"[7], introduced in [6] in 2013. It is a corpus of 42 306 movie plot summaries extracted from Wikipedia. We extracted 2000 of them randomly and used them in our experiments.

The third dataset is a larger dataset containing 25 000 positive and 25 000 negative movie reviews gathered from "Large Movie Review Dataset v1.0"[8], first introduced in [7] in 2011. It is intended as a larger benchmark dataset for sentiment classification, and consists of movie reviews, gathered from IMDb.

The age of the datasets, does not concern us, since the theories tested in this thesis, can be applied to newer datasets without any problems. The trustworthiness of the documents and data sources will not be a focus of this thesis.

1.3.2 Classification algorithms

The classification algorithms used in our experiments are limited to the Support vector machines and Random forest algorithms. Support vector machines, are chosen because it is a classification algorithm common to use in the area of sentiment analysis. Random Forest is chosen because it has shown to give good results in other classification areas [8, 9].

1.3.3 Language

This project will only consider the English language. This is mainly because the English language represents the documents in the different corpuses used. The results from this project can be transferred to other languages rather easily.

[7] http://www.cs.cmu.edu/~ark/personas/

[8] http://ai.stanford.edu/~amaas/data/sentiment/

1.4 Approach

This section will briefly describe how we proceeded to achieve our goal. We started with a prestudy to examine the recommender and sentiment analysis approaches that exist today, what their functionality, possibilities and limitations were. To further investigate how adjectives could be exploited and used to improve classification results, we used LingPipe[9] to extract adjectives from our dataset. For each adjective, we found its synonyms and antonyms and their sentiment with the help from WordNet[10] and SentiWordNet[11]. The sentiment of all the adjectives were calculated by the majority vote of the sentiments of the corresponding synonyms together with the opposite sentiment scores of the corresponding antonyms. The number of positive adjectives and negative adjectives were counted for each movie review and used as attributes for the classifier. Finally, the classification algorithms Random Forest and Support Vector Machines were used for classification. See Chapter 5 for a more detailed description of the approach.

1.5 Main contributions

The main contributions of this project include an introduction to the basic principles behind todays recommender systems. We present the different approaches and challenges concerning the area of sentiment analysis. We present machine learning algorithms, including a detailed explanation of the Random Forest and Support Vector Machines. Most important, this thesis presents a system and an approach to how adjectives can be used in the area of sentiment analysis. Based on the results proven in this thesis, our system can be used as a basis for a more powerful and personalized recommendation.

1.6 Structure of report

The remainder of the report is structured as follows: First, **Chapter 2** gives an introduction to the domain knowledge and the background of this thesis. **Chapter 3** presents related work and how this thesis differs from work done by others. In **Chapter 4** you get a detailed description of the classification methods used in this research. **Chapter 5**

[9] http://alias-i.com/lingpipe/

[10] https://wordnet.princeton.edu/

[11] http://sentiwordnet.isti.cnr.it/

describes the dataset used and the preprocessing of the same dataset. It also documents the approach of the research and how the results were collected. **Chapter 6** presents the results from the experiments along with the evaluation metrics used. **Chapter 7** discusses the significance of the results. Lastly, **Chapter 8** concludes with a summary drawn from the discussion and presents improvements for future work.

Chapter 2

Background

This chapter provides the reader with a fundamental understanding of the domain knowledge in the field of study. The first two sections present the historic background of the recommender systems and an introduction to the most common recommender techniques. The last two sections, presents important techniques within the textmining and sentiment analysis area.

2.1 The growth of recommender systems

The first commercial recommender system, called Tapestry [10] introduces the term *Collaborative Filtering*. Tapestry was designed to recommend documents, gathered from newsgroups. The motivation for this product was to prevent users to be overwhelmed with documents.

Later, the interest grew due to the relevance directly to e-commerce. Netflix, an online streaming video service, released a dataset containing 100 million ratings given by half a million users to thousands of movies. With this, they announced an open compition (Netflix Prize) for the best collaborative filtering algorithm in this domain, matrix factorization [11].

The architecture of recommender systems and their evaluation on real-world problems is an active area of research [2]. Applications are released in domains ranging from recommending webpages, music, movies, books and other consumer products. From the users side, it is expected that serious e-commerce systems have some kind of a recommender system.

2.2 What is a recommender system?

The goal of a Recommender System is to generate meaningful recommendations to a collection of users, for items or products that might interest them [2]. Suggestions for movies to watch on Netflix or books to buy on Amazon, is real world examples of the results from recommender systems. How the systems works depends on the domain and the characteristics of the data available. For example, after watching a movie on Netflix you can rate the movie on a scale from 1 (dislike) to 5 (like). In addition, the system may have access to user-specific and item-specific profile attributes, like product description for instance. Recommender systems uses different methods and approaches to analyze these data, to recommend the best product ro the user. Collaborative filtering systems analyze historical interactions alone, while Content-based Filtering systems are based on profile attributes. Hybrid approaches combine these two methods and matrix factorization uses the structures of a matrix to do recommendations. The next subsections, will describe these techniques in detail.

2.2.1 Collaborative filtering

In collaborative filtering systems, a user is recommended items based on the past ratings of all users collectively [2].

User-based collaborative filtering works by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behaviour amongst several users.

Finding a recommendation for an active user is done by choosing a subset of users based on their similarity with the active user. Thereafter a weighted combination of their ratings is used to produce predictions for the active user.

A general approach is defined in the following steps [2]:

1. Assign a weight to all users with respect to similarity with the active user.
2. Select k users that have the highest similarity with the active user - commonly called the *neighbourhood*.

3. Compute a prediction from a weighted combination of the selected neighbor's ratings.

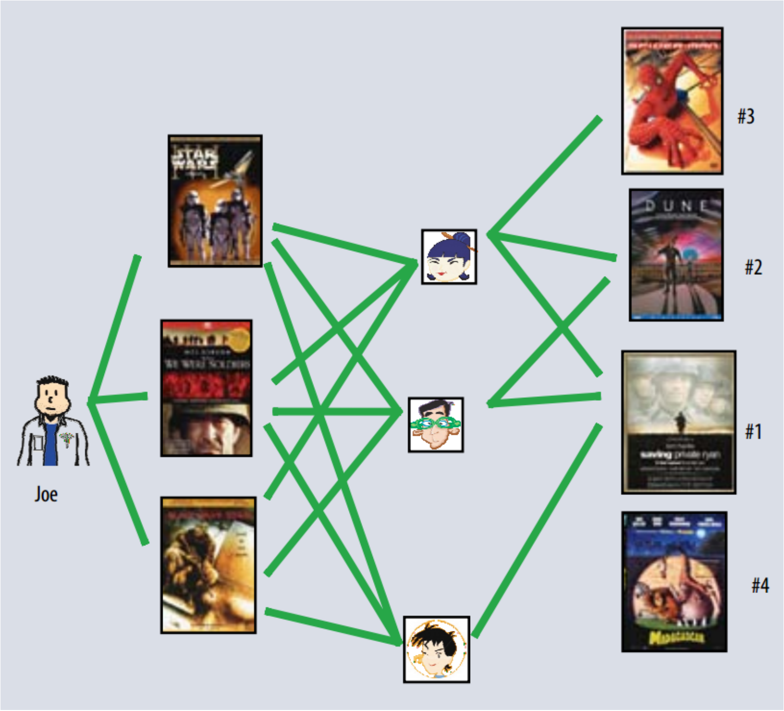


Figure 2.1: User-based filtering. To make a prediction for Joe, we find users that also likes the movies Joe likes, and see what other movies these users like. In this case, all similar users, liked Saving Private Ryan, so that movie is recommended for Joe. Image obtaned from [11]

A very commonly used measure of similarity is the Pearson correlation coefficient between the ratings of two users, a and b. The Pearson correlation coefficient is defined as following:

$$w_{a,b} = \frac{\sum_{i \in I}(r_{a,i} - \bar{r_a})(r_{b,i} - \bar{r_b})}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r_a})^2 \sum_{i \in I}(r_{b,i} - \bar{r_b})^2}} \tag{2.1}$$

where I is the set of items rated by both users, $r_{a,b}$ is the rating given by user a to item i and $\bar{r_a}$ is the mean rating given by user a.

Another way to treat the ratings is as vectors in an n-dimensional space, and compute similarity based on the cosine of the angle between them, given by:

$$w_{a,b} = cos(\vec{r_a}, \vec{r_b}) = \frac{\vec{r_a} \cdot \vec{r_b}}{||\vec{r_a}|| \times ||\vec{r_b}||} = \frac{\sum\limits_{i=1}^{n} r_{a,i} r_{b,i}}{\sqrt{\sum\limits_{i=1}^{n} r_{a,i}^2}\sqrt{\sum\limits_{i=1}^{n} r_{b,i}^2}} \tag{2.2}$$

Important when choosing to compute the cosine similarity is that you cannot use negative ratings and all unrated items should be treated as having a rating of zero.

Empirical studies have found that Pearson correlation generally performs better than the cosine similarity [12].

Predictions are generally computed as the weighted average of deviations from the neigbor's mean, as following:

$$p_{a,i} = \vec{\bar{r_a}} + \frac{\sum_{b \in K}(r_{b,i} - \vec{r_b}) \times w_{a,b}}{\sum_{b \in K} w_{a,b}} \tag{2.3}$$

where p_{a_i} is the prediction for user a for item i, $w_{a,b}$ is the similarity between users a and b, and K is the set of most similar users (neighbourhood).

User-based collaborative filtering do not scale well, when applied to millions of users and items. This is because the user profiles change so quikly and the entire system model needs to be recomputed. Computing similarities between all pairs of users are computational complex and expensive.

As a response to this, **item-based collaborative filtering** was developed. Rather than matching similar users, item-based collaborative filtering match a user's rated items to similar items. This approach has lead us to faster applications and often improved recommendations [13]. Amazon stated that this method powered it's recommender system [14].

Similarities between item i and item j are computed as following:

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r_i})^2 \sum_{u \in U}(r_{u,j} - \bar{r_j})^2}} \tag{2.4}$$

where U is the set of all users who have rated both items i and j, $r_{u,i}$ is the rating of user u on item i and $\vec{r_i}$ is the average rating of the ith item across all users.

Alternatively, one can use an adjusted cosine similarity:

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \vec{r_u})(r_{u,j} - \vec{r_u})}{\sqrt{\sum_{u \in U}(r_{u,i} - \vec{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \vec{r_u})^2}} \tag{2.5}$$

The adjusted cosine similarity takes into account different rating schemes. This means that some users might rate items higher than average and some users give lower ratings in general. Adjusted cosine similarity solves this problem by subtracting the average rating for each user for the pair of items.

The rating can be predicted for item i and user a using a weighted average:

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|} \tag{2.6}$$

where K is the set of the k items rated by a that are most similar to i.

2.2.2 Content based filtering

These approaches recommend items that are similar in content to items the user has liked in the past, or matched to attributes of the user [2].

Collaborative filtering does not need to know anything about the users or the items, except their ratings. Content based filtering can make a better personalized recommendation by knowing more information about the user or the item, such as genre and actors in it [15]. For example if we have actor information and a user likes the movies *Titanic*, *Inception* and *Blood Diamond*, it is a reasonable possibility the user is fan of Leonardo DiCaprio. We can therefore recommend more DiCaprio movies to that user. Similarily if the user likes *Frozen* and *Wreck-It Ralph* we can assume the user are into animated movies. Content-based filtering uses content about the items and compare them to content that interests a user.

Recommending items with associated textual content have been researched a lot. The textual content can for instance be descriptions or user reviews. Some approaches have treated this problem as an information retrieval task, where the user content is treated as a query and items without ratings is being scored with similarity to that query [16].

An alternative to treating the problem as an information retrieval task is to treat it as a classification task. In this case, each example represent the content of an item and the past ratings of a user are used as labels for those examples.

2.2.3 Hybrid approaches

Cold start is a potential problem in computer-based information systems, which involve a degree of automated data modelling. Specifically, it concerns the issue that the system cannot draw any inferences for users or items that it has not yet gathered sufficient information. In our domain, the cold start problem typically involve a new user that has not seen or rated any movie or product. Content-based recommendation systems can provide recommendations for cold start items and users for which little or no training data are available, but typically have lower accuracy than collaborative filtering systems. On the other side, collaborative filtering techniques often provide accurate recommendations, but fail on the cold start problem [17]. Hybrid approaches attempt to combine collaborative filtering, content-based and other recommendation methods to yield better recommendations across the board. We will explain some of the many hybrid approaches that exist.

Weighted hybrid approaches computes the score of a recommended item from the results of all available recommendation techniques present. As an example, we can look at a simple weighted hybrid; a linear combination of recommendation scores. Initially, collaborative and content-based recommenders have equal weight, but the weighting are adjusted as predictions about user ratings are confirmed or disconfirmed. The benefit of a weighted hybrid is that it is straightforward and it is easy to perform post-hoc tasks and adjust the hybrid accordingly. However, the assumption that the relative value of the different techniques is more or less uniform across the space of possible items, is not always true. A collaborative recommender will be weaker for those items with a small number of raters [18].

Switching hybrid systems uses some criterion to switch between recommender techniques. An example of a switching hybrid is the DailyLearner system, presented in [19]. In this system, content-based recommendation (nearest neighbor algorithm) is employed first, and if the content-based system cannot make a recommendation with sufficient confidence, then a collaborative recommendation (naïve bayesian classifier) is used. The benefit of switching hybrids is that the system can exploit the strength and weaknesses of different recommender systems in different situations, but it also introduces more complexity into the recommendation, since a suitable switching criteria must be determined [18].

Mixed hybrid is the process where recommendations from several different recommender systems are presented together at the same time. Therefore, mixed hybrids are very useful in situations where it is practical to do large number of recommendations simultaneously. When conflicts occur between the used recommendation techniques, some kind of combination technique is needed. The mixed hybrid avoids the cold start problem of a new item, because a content-based component can come up with new recommendations based on the description, even if there exist no rating. It does not get around the cold start of a new user, since both content-based and collaborative methods need some data about user preferences to get started [18].

2.2.4 Matrix factorization

While collaborative filtering and content-based filtering are rather simple and intuitive to understand, matrix factorization is usually more effective, because they allow us to discover the latent features hidden in the data. As the name describes, matrix factorization is about factorizing matrices, i.e. find two (or more) matrices that multiplied together, will get back the original matrix. Lets look at an example. Assume we have 5 users and 8 movies, and the rating are integers between 1 and 5. The matrix could look like the one in Table 2.1, where 0 means that the user have not yet rated the movie.

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6
User1	1	4	3	0	0	5
User2	0	0	4	4	0	3
User3	5	2	0	0	1	0
User4	5	0	0	4	0	0
User5	0	0	2	0	2	5

Table 2.1: Matrix describing users and their ratings of different movies

The task of matrix factorization can be seen as a task of predicting the missing ratings (the zero's in the matrix). The idea behind matrix factorization is that there is a latent feature hidden, that determines how a user rates a movie. For example, a user gives two movies directed by Tarantino a high rating. A hidden feature can be that this user likes Tarantino as a director. If we can discover these hidden features, we could be able to predict a rating of a movie, not yet rated by this user.

Matrix factorization can be implemented as follows [20]: For a system with p users and q movies (or items) we have a set $U = \{u_1, u_2, ..., u_p\}$ of users and a set $M = \{m_1, m_2, ..., m_q\}$ for movies. We define the set T as follows, where O is the set of possible ratings, for example $O = \{1, 2, 3, 4, 5\}$:

$$T \subset U \times M \times O \tag{2.7}$$

The set T is called the training set and describes all known ratings by users on different movies. It will then exist a mapping e of a user $u_i \in U$ and a movie $m_j \in M$ to a rating r_{ij}:

$$e : U \times M \to O \qquad e(u_i, m_j) = r_{ij} \tag{2.8}$$

The mapping e is estimated by a model $e\hat{}$ (called an estimator) that predicts ratings between a given user-movie pair. The estimator is linked with the true mapping e, as follows:

$$e\hat{}(u_i, m_j) = e(u_i, m_j) + Z \tag{2.9}$$

where Z is the error between the prediction and the actual rating.

Finally, the rating matrix R given by

$$R = (r_{ij}) \tag{2.10}$$

where i = 1, ..., p and j = 1, ..., q will be a complete rating matrix. This means it has no empty cells, i.e. all user-movie ratings are predicted. The movie with largest predicted rating will be recommended for the user.

The method presented above is not the only matrix factorization method. There exist many different extensions and one of theese is the non-negative matrix factorization (NMF). In NFM all the elements of the factor matrices are non-negative. One advantage of NMF is that it is intuitive to understand, and since no elements are negative, there is no subtraction involved in multiplying the matrices to get back to the original matrix.

2.3 Text mining

Knowledge discovery in databases is defined by Fayyad et al. to be “the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data” [21]. Data mining is essentially concerned with information extraction from structured databases [22].

As an example we can take a look at the structured data in Table 2.2. The data are fictional and serves only the purpose as an example.

Person	**Sex**	**Age**	**Interests**	**Netflix-Customer**
Anna	F	16	Movies, books	yes
Bob	M	53	Boats	yes
Michael	M	27	Computer science	no
John	M	55	Nature, books	yes
Sarah	F	21	Movies	yes
Jack	M	72	Cooking	no

Table 2.2: Netflix Customer Table

From the data in Table 2.2 a textmining approach could produce rules similar to this:

Induced rules
if Age(Person) < 21 **then** Netflix-Customer(Person)
if Interests(Person) = movies **then** Netflix-Customer(Person)

Table 2.3: Induced rules

In reality much information appears in textual and hence unstructured or implicitly structured form. Specialized techniques operating on textual data became necessary to extract information, and the problem of doing this got the name text mining. In other words, text mining refers to the process of deriving useful information from unstructured text. In our research, we want to derive the opinions of the text. This is more known as sentiment analysis or opinion mining.

2.3.1 Natural language processing

Natural language processing (NLP) is concerned with the interaction between computers and human natural language. Humans often use the language in particular situations and therefore natural languages often becomes very context-dependent. As a result, natural languages are ambiguous, complex and difficult to fully understand for a computer. This is the main reason, extracting information from natural language text remains a challenge in Natural Language Processing (NLP) today. Some of the major task of NLP involves:

Automatic summary: Produce a readable summary out of one or more text-documents.

Machine translation Automatically translate a human language to another.

Natural language generation translate information from databases into readable human language.

Part-of-speech Given a sentence, determine the part of speech (adjective, noun, ..) for each word.

2.3.2 Text classification

A classification problem is described as following: Given a set of classes, we seek to determine which class(es) a given object belongs to [23]. The description is very general and has many applications, amongst them sentiment detection. Sentiment detection can be used for classifying product reviews into the classes “positive” and “negative”. An example is an user searching for negative reviews before watching an action movie, to make sure the movie has a desired quality.

Another use for the classification task is an user that wants to pay attention to all new movies containing firearms. One way of doing this is to keep track of all new movies each week, and mark the ones that contain the word “firearm” in the description. This is a rather time consuming and repetitive task. Many systems support standing queries to automate this task. A standing query is a query that is executed on a collection to which new documents are incrementally added over time [23]. If your standing query is “firearm AND gun AND pistol”, you will miss many relevant movies which use other terms such as “revolver”. To achieve good results, the standing queries need to be continuously redefined over time and will often become very complex [23].

In the example above, our standing query divides new movies into the two classes: movies containing firearms and movies not containing firearms. This is called a two-class classification. Often a class is more general like action or horror. Such general classes are often referred to as topics and the classification task is then called text classification, text categorization, topic classification or topic spotting. Although standing queries and topics are different in their degree of generality, the methods for solving classification using standing queries and text classification are essentially the same [23].

In the following paragraphs, we will see some examples of standard text classification methods.

Naive Bayes classifiers are linear probabilistic classifiers based on the Bayes theorem. The features in the dataset are mutually independent, hence the name “naive” [24]. The strong independence assumption is often violated in reality, but naive Bayes classifiers still often perform well. Naive Bayes classifiers especially performs well on small dataset, where it can outperform the more powerful classifiers like C4.5 (an extension from the ID3 algorithm) and the CN2 induction algorithm [25].

Bayes theorem is given as follows [26]:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \tag{2.11}$$

where $P(h \mid D)$ is the probability that h holds given the observed training data D, $P(D \mid h)$ the probability of observing D given some world where h holds. $P(h)$ is the initial probability that H holds and $P(D)$ is the probability that D holds, given no knowledge of which hypothesis holds.

Lets look at an example presented in [27]; a fruit may be considered to be an apple if it is red, round and about 4 cm in radius. A naive Bayes classifier considers all of these features to contribute independently to the probability that this fruit is an apple, without considering the existence of the other attributes.

To classify our text, we want to consider a set of hypotheses H and are interested in finding the most probable hypothesis $h \in H$ given the observed data D. A maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis and is calculated as follows [26]:

$$h_{MAP} = argmax_{h \in H} P(h \mid D) \tag{2.12}$$

using Bayes theorem we get

$$= argmax_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} \tag{2.13}$$

We can drop the term $P(D)$ because it is a constant independent of h.

$$= argmax_{h \in H} P(D \mid h)P(h) \tag{2.14}$$

Assuming all hypotheses have the same probability $P(h_i) = P(h_j)$ for all $h_i, h_j \in H$ it is possible to further simplify the calculation. Any hypothesis that maximizes $P(D \mid h)$ is called a maximum likelihood (ML) hypothesis h_{ML}.

$$h_{ML} = argmax_{h \in H} P(D \mid h) \tag{2.15}$$

Neural network are nodes put together to form a network which mimics a biological neural network, like the human brain. Normally in the case of classification, the feature weights are sent into the input nodes and the output nodes produce the classification based on the links between. These links represent dependence relations in the network. The neural networks are trained by back propagation. This means that if a misclassification occurs, the error is propagated back from the output nodes, through the network, modifying the link weights to minimize the error. Because of this, neural network require a large set of training data for complex classification tasks. The simplest kind of a neural network is called a perceptron and is equivalent to a linear classifier[1]. A perceptron only contains the input nodes and the output nodes. In most cases, the neural network contain one or more hidden layers between the input and output layer [28].

[1] Linear classifier: divides the classes by a linear separator in the feature space. In a n-dimensional space, the linear clsasification is a $(n-1)$-dimensional hyperplane. For example in two dimensions, a linear classifier is a line.

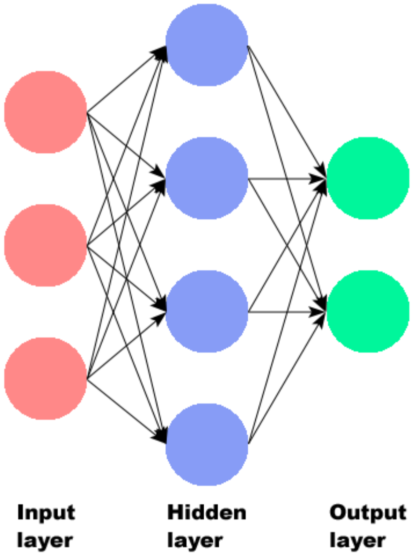


Figure 2.2: Neural net with one hidden layer.

Deep learning (sometimes called feature learning or representation learning) is a set of machine learning algorithms which attempt to learn multiple-layered models of inputs, commonly neural networks [29]. The brain recognizes positive and negative sentences and deep learning could therefore be used in sentiment analysis and similar contexts. However, deep learning is beyond the scope of this thesis.

Decision tree classifiers are a tree, where the internal nodes are represented as features, the edges are represented by tests on the features weights and the leaves are represented as categories. Many categorization methods cannot be easily understood by humans, but that is not the case of Decision trees. A decision tree categorizes a document by starting at the root of the tree and moving downwards via the edges whose conditions are true by the document, until a leaf node is reached. The document are then assigned the category that represents that leaf node [28].

A tree is typically built recursively by picking a feature at each step and divide the trainingset into two subsets, one subset containing the feature and the other not. This is done until a single category remains and a leaf node is created. One problem with

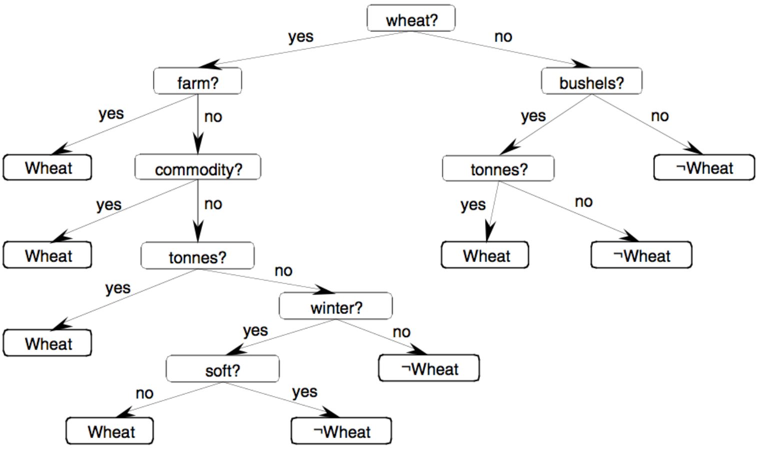


Figure 2.3: Decision tree classifier. Image obtained from [28]

decision trees created like this are that they are prone to overfit the training data. A solution to this include pruning, removing the too specific parts of the tree, for a better result [28]. Random Forest is an example of an decision tree algorithm and will be discussed further in Section 4.2.

Support vector machine (SVM) is a vector-based classification method. It is known as a very fast and effective algorithm for classification problems. We can think of a SVM as a hyperplane in the feature space. The unique hyperplane chosen during training, is the hyperplane that separates the positive from the negative instances with the maximal margin. The margin is defined as the distance from the hyperplane to the nearest point from the positive and negative sets. SVM hyperplanes are fully determined by a relatively small subset of the training instances, called the support vectors. The rest of the training data have no influence on the trained classifier [28]. Figure 2.4 shows a maximal margin hyperplane in two dimensions.

In our experiments, SVM, along with Random Forest are chosen as our classification algorithms, and will be discussed further in Chapter 4.

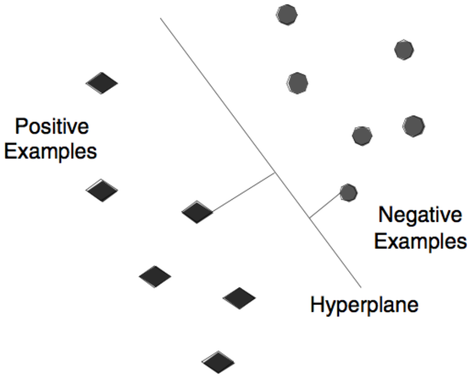


Figure 2.4: Linear SVM classifier in two dimensions. Image obtained from [28]

2.3.3 Information extraction

In a relational database, data is stored in tables. If the names of the tables, rows and columns are known, a structured query can easily extract the information needed. In the case of unstructured data, it is not a trivial task to extract such information from the text, because there exist no reference to the location of the information needed. Information extraction is defined as the process of extracting those important paths of information from the text [30]. This can for instance be names of people, organizations, locations or movies.

The information that is extracted depends on the purpose and context of the process. The elements of the textual data, i.e. tokens, terms and separators, play an important role when defining the scope of the information extraction. Tokens can be considered to be a series of characters without separators. Separators can be a special character such as a blank or a punctuation mark and a term is a token with a specific semantic purpose [30].

2.3.4 Clustering

Clustering split records in a data set into non overlapping groups, such that the subjects within a group are similar and the subjects between the groups are dissimilar [30]. An example of this can be seen in Figure 2.5. In reality overlapping is often the case, but that is not a problem since each document can overlap in topic areas after classification.

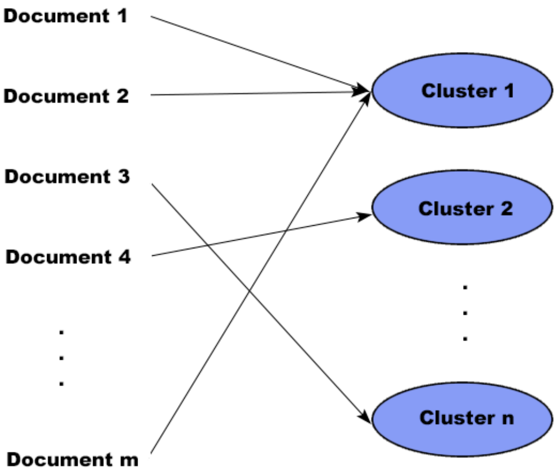


Figure 2.5: Clustering with non overlapping groups.

The non overlapping groups are based on the presence of similar themes. These themes may vary, based on the task that are being solved. Often there are large amounts of textual data and difficult to find the theme by manually reading all the text in a cluster. Instead, a set of descriptive terms of each cluster identifies the theme. A vector of the terms then represent the weights, measuring how the document fits into each cluster [30].

There are two main types of clustering, flat clustering and hierarchical clustering. Flat clustering creates a flat set of clusters without any explicit structure that relate clusters to each other. It is efficient and conceptually simple, but require a prespecified number of clusters as input and are nondeterministic. Examples of flat clustering algorithms are K-means and k-nearest neighbors. Hierarchical clustering on the other hand, creates a hierarchy of clusters that are more informative than the unstructured set of clusters returned by flat clusters. Hierarchical clustering does not require a prespecified number of clusters and are often deterministic, but also less efficient in most cases [23]. Since we are working with short texts in this thesis, it is most common and more efficient to use the flat clustering algorithms.

For example, consider these documents (comments) about a movie.

1. The actors and actresses were great.
2. Loved the plot of the movie.
3. A movie classic.
4. Highly recommended.
5. This is a feel-good movie I would watch again.
6. Heartwarming and beautiful story.
7. The actors were awesome.

The results from the clustering could look similar to Table 2.4. Each cluster is described by a set of terms that identifies the theme of the cluster. This type of analysis can help in correctly classifying comments based on topics.

Cluster	**Document**	**Key words**
1	1,7	Actors, actresses, great, awesome
2	2,5,6	Loved, plot, feel-good, heart-warming, beautiful, story
3	3,4	classic, recommended

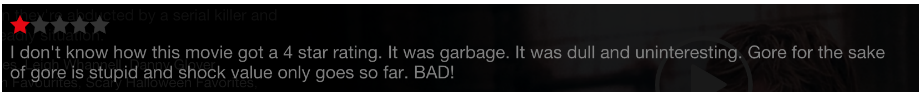
Table 2.4: Clustering results from text mining

2.4 Sentiment analysis

Sentiment analysis (also known as opinion mining) deals with the computational treatment of opinion, sentiment, and subjectivity in unstructured text [31]. In other words this means that sentiment analysis tries to identify and extract subjective information from the text. This is a complex process, even for humans. Consider the statement

“’Quentin Tarantino directed this movie”. Is that a neutral, negative or positive sentence? The answer is probably different for different people. For Tarantino fans this is an positive sentence, for people that haven’t seeen Tarantino movies it is an neutral sentence and for people that doesn’t like Tarantino movies this is a negative sentence.

Suppose we want to classify an opinionated text as either positive or negative.



★☆☆☆☆

I don't know how this movie got a 4 star rating. It was garbage. It was dull and uninteresting. Gore for the sake of gore is stupid and shock value only goes so far. BAD!

Figure 2.6: Negative comment using negative keywords. Screenshot from Netflix[2].

This seems easy, as you can see in Figure 2.6, it is an extremely negative review of the movie Saw, with lots of negative keywords like “garbage”, “dull”, “uninteresting”, “stupid” and “BAD”. But the results of an early study by Pang et al. [32] shows that it is not as trivial as one might think. Two graduate students in computer science were asked to (independently) pick keywords that they considered to be good indicators of positive and negative sentiments in movie reviews.

As shown in Table 2.5, the list of words made by the humans achieves about 60% accuracy when used within a straightforward classification policy on a dataset with 700 positive and 700 negative reviews. In comparison a list with an equal amount of words were made based on a very preliminary examination of frequency counts in the entire corpus (including the test data) plus introspection, raised the accuracy to 69%.

Just because it can be hard for humans to choose the best keywords, does not necessarily mean that it is impossible for the computer to make such a list. Applying machine learning techniques based on unigram models can achieve an accuracy up to 82,9% [32]. This is way better than the keywords reported above, but there are still room for improvements.

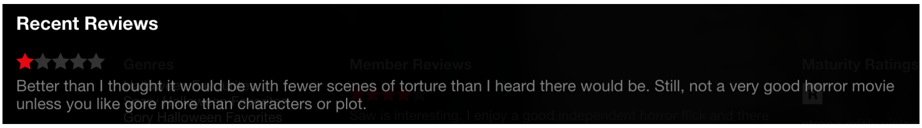
An example of why this is not working perfectly is that many comments that are negative do not use negative keywords. As you can see in Figure 2.7 the comment on the movie Saw are obviously bad (the user gave 1 star in rating), but the comments only contain positive keywords like “better” and “very good‘”.

[2] `http://www.netflix.com/`. Retrieved December 11th 2015.

[3] `http://www.netflix.com/`. Retrieved December 11th 2015.

	Keywords	Accuracy	Ties
Human 1	**Positive**: *dazzling, brilliant, phenomenal, excellent, fantastic* **Negative**: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	**Positive**: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* **Negative**: *bad, cliched, sucks, boring, stupid, slow*	64%	39%
Statistic of the test data	**Positive**: *love, wonderful, best, great, superb, still, beautiful* **Negative**: *bad, worst, stupid, waste, boring, ?, !*	69%	16%

Table 2.5: Sentiment classification using keyword lists created by humans, compared with using keywords selected via examination of simple statistics of the test data. Tie rates is the percentage of documents where the two sentiments were rated equally likely. This table are adopted from Figure 1 and 2 in Pang et al. [32]



Recent Reviews

★☆☆☆☆

Better than I thought it would be with fewer scenes of torture than I heard there would be. Still, not a very good horror movie unless you like gore more than characters or plot.

Genres
Member Reviews
Maturity Ratings
Gory Halloween Favorites
Saw is interesting. I enjoy a good independent horror flick and there

Figure 2.7: Negative comment using positive keywords. Screenshot from Netflix[3].

2.4.1 Objective facts versus subjective opinions

When working with sentiment analysis, we want to find the opinion in the text and detect if the opinion is “positive”, “neutral” or “negative”. When doing this, it is important that we detect if a given text contains subjective or objective information, or identify which parts of the text that are subjective. The objective sentence “Quentin Tarantino directed this movie” has no opinion and it can mean different things to different people. Tarantino fans will maybe be adding this movie to their que, while people who do not like Tarantino will go look for another movie. In contrast the subjective sentence “I love the work Tarantino did on this movie", have an opinion that we want to extract. The opinion says accurately that Tarantino did a good job on this movie. This may imply that the movie is interesting for more people than just the Tarantino fans.

Distinguishing objectivity from subjectivity is not an easy task. Mihalcea et al. [33] summarize several studies on the topic as follows: "the problem of distinguishing subjective versus objective instances has often proved to be more difficult than subsequent polarity classification, so improvements in subjectivity classification promise to positively impact sentiment classification".

Subjectivity detection is nearly connected to text categorization. Text categorization is the problem of automatically assigning predefined categories to free text documents [34]. Studies done by Yu and Hatzivassiloglou achieved an accuracy of 97% with a Naive Bayes classifier on a corpus consisting of Wall Street Journal articles, when separating articles under *News and Business* (objective facts) from articles under *Editorial and Letter to the Editor* (subjective opinions) [35]. The study separate positive, negative and neutral opinions into three classes, based on the number and strength of semantically oriented words (either positive and negative) in the sentence.

2.4.2 Degrees of positivity

Problems related to determine the degree of positivity includes analysis of comparative sentences[4] [36]. The idea is that sentences like "This movie contains more action than the previous I saw" and "I prefer this movie to the previous movie I saw" are important information, when trying to find the degree of positivity in the sentences.

Suppose we want to split comments into the classes "positive", "neutral" and "negative". A challenge arises when analysing the neutral class. Comments with a equally mixture of positive and negative language will be placed in this class amongst the comments with neutral words like "mediocre", "meh" and "so-so". Also in some domains "neutral" is used when there is a lack of opinion. Studies from eBay found that the information contained in neutral ratings is interpreted by users to be much closer to negative feedback than positive [37]. Another important aspect is the difference between the strength of an opinion and the positivity. It is possible to feel real strong about a movie being mediocre.

[4] Comparative: expressing a higher degree of a quality, but not the highest possible (e.g. braver; more fiercely).

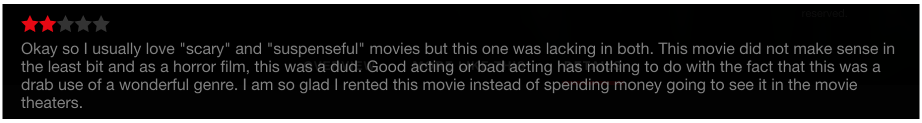
2.4.3 Feature vectors

In information retrieval tasks, text is often represented as a feature vector where the entries correspond to individual terms. Term frequencies often defines these entries. In sentiment analysis, the idea is that features that often appear in one class (e.g. "good" in the positive set) can be used to predict the class of a new document, containing the term "good". Topic-based text categorization also uses the same approach [31], because the topic is most likely mentioned many times in the text. In contrast Pang et al. [32] obtained better results gathering opinions using presence instead of frequency. Presence where gathered as feature vectors containing binary values (1 if the term occurs, 0 otherwise) versus frequency, where the entry value increases with the frequency of the word in the text. Subjectivity can therefore not be gathered the same way as topics/objective information. Not only are presence better than frequency, it has been found that unique words, words that only appear once in a text, are precise indicators of subjectivity in the same text [38].

2.4.4 Negation

Negation play an important role in sentiment analysis. Sentences like "I like this movie" and "I don't like this movie" are very similar, but because of one negation term they are supposed to be classified into opposite classes. However, due to similarity in the sentences and especially in the positive word "like", they are not. One possibly way to handle this is to handle the negations indirectly. An example of this is to use a second-order feature of a text segment, where an initial representation, such as a feature vector ignores the negation. That representation is then converted into a different representation that takes negation into account [31]. Alternatively, negation can be encoded directly into the definitions of the initial features. A study by Das and Chen suggest attaching "NOT" to words close to the negation terms such as "no" or "don't" [39]. An example of this is the sentence "I don't like this movie", where the word "like" is converted into "like-NOT". This seems quite easy, but not all negation terms reverse the polarity of the sentence. Consider the sentence "No wonder this is considered one of the best movies", in this case, it is incorrect to attach "NOT" to the word "best".

Another difficulty with negation is that negation often can be expressed in ways that are difficult to detect. Irony and sarcasm are examples of this and will be discussed further in the next section. Another example can be seen in the first sentence in Figure 2.8, where the word "lacking" becomes a polarity reverser.



Okay so I usually love "scary" and "suspenseful" movies but this one was lacking in both. This movie did not make sense in the least bit and as a horror film, this was a dud. Good acting or bad acting has nothing to do with the fact that this was a drab use of a wonderful genre. I am so glad I rented this movie instead of spending money going to see it in the movie theaters.

Figure 2.8: Negation. Screenshot from Netflix[5].

2.4.5 Dealing with irony

The word irony is defined as following, as specified in the English dictionary[6]:

> **irony**
>
> **Noun**
>
> 1. *The expression of one's meaning by using language that normally signifies the opposite, typically for humorous or emphatic effect.*
> 2. *A state of affairs or an event that seems deliberately contrary to what one expects and is often wryly amusing as a result.*

The universal definition of irony is somewhat vague, and does not address some of the challenges that emerge when trying to detect irony in a given text. For people, detecting irony can be hard enough, but for machines even harder. The lack of a formal definition for the structure of irony makes it hard to train agents to recognize irony [40]. Many believe irony changes over time, there is regional differences, age and gender differences, that makes this formal structure definition hard to find.

Another problem arises when the context is not available. In many cases a stand alone text (for instance a sentence) cannot be reliably judged as ironic without the surrounding context [40].

When classifying opinion in text, correct identification of irony and sarcasm is important. In worst case, with an unsuccessful identification, the text can be classified as the opposite of the correct class [41]. However, studying irony in text is a research area on it's own and beyond the scope of this thesis.

[5] `http://www.netflix.com/`. Retrieved December 11th 2015.

[6] `http://www.oxforddictionaries.com/definition/english/irony`. Retrieved November 8th 2015.

Chapter 3

Related work

Identifying the sentiment of texts is a key component in many research fields. This chapter will review some of the related work where others try to solve the sentiment challenge and briefly present the approach of these researches. The first section will describe three researches working on sentiment analysis in product reviews, an area very similar to movie reviews. The following two sections will describe research within the field of micro-blogging and longer text. The fourth section describes related work, using adjectives as sentiment deciders, very similar to the research questions of this thesis. The last section will provide a research using sentiment analysis in the area of the financial market, focusing on saving time during classification.

3.1 Sentiment analysis in product reviews

User-generated content has become a huge part of the Web. With the rapid growth of such content, many organizations are using sentiment analysis on online review postings. Analysing opinions expressed on the Web has become increasingly important for effective organizational decision making [42]. Several works address the sentiment analysis on product reviews, such as the ones proposed in [43, 44]

The work of Dave et al. in [43], described a tool for sifting through and summarizing product reviews, automating the sort of work done by aggregation sites or clipping services. Their approach began with training a classifier, using a corpus of self-tagged reviews available from C|net and Amazon. Starting with the raw document, HTML tags were stripped out and the document was split into sentences. The sentences were then passed through a parser before being split into single-word tokens. Statistical substitutions were made and the document was passed through a linguistic parser, outputting the part of speech of each word and the relationship between parts of the sentences. Words were then passed through WordNet, a database for finding similarities of meaning. Features were selected and their probabilities were smoothed. An algorithm from information retrieval were further used for assigning scores. The scores of the words in an unknown document were summarized and the sign of the total (+/-) were used to determine the class.

In [44], Thet, Na and Khoo presented a method for automatic sentiment analysis of movie reviews. The proposed method in this work was to perform fine-grained analysis to determine the strength of the sentiment, in addition to the sentiment orientation. A dependency tree was constructed from a set of grammatical dependencies between the words in a given sentence, based on the syntactic structure. Further, the tree was divided into dependency sub-trees, each representing a single clause focusing on just one aspect of the movie. After dividing the sentence into separate clauses, a contextual sentiment score towards each movie aspect was calculated. Prior sentiment score to each word in a sentence were derived from both a domain-specific lexicon and a generic opinion lexicon, derived from SentiWordNet, in addition to a subjectivity lexicon. Based on these scores for each clause and the contextual sentiment score for each review aspect, the overall sentiment score for the whole sentence was calculated. The output sentiment scores, were used to identify the most positive and negative clauses or sentences with respect to particular movie aspects.

3.2 Sentiment analysis in micro-blogging

Social micro-blog sites, such as Twitter, has become a very popular communication tool among Internet users. Twitter grows in user base and has become an attractive platform for companies, politicians, marketeers, and others wishing to share information and/or opinions. In [45], Pak and Pasoubek used a so-called TreeTagger[1] to analyze the corpus and make it ready for the classifier, in this case the multinomial Naïve Bayes classifier. They used the presence of n-grams as a binary feature and applied normal text processing methods for processing the text. The results of Pak and Paroubek, shows

[1] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

that the best performance for detecting sentiments in Twitter is achieved when attaching negation words to bigrams. System performance can be increased by an increased sample size, measured by F-measure, together with the salience strategy (n-grams with a low salience[2] are discarded) for filtering out common n-grams (n-grams that do not strongly indicate any sentiment or indicate objectivity).

3.3 Sentiment analysis in longer texts

Longer texts like newspapers and blogs express opinion of news entities, e.g. people, places and things, while reporting on recent events. In [46], Godbole, Srinivasaiah and Skiena presented a system that assigned scores indicating positive or negative opinion to each entity in the text corpus. The system was built on top of the Lydia text analysis system [47]. Their approach used a sentiment word generation algorithm that expanded a set of seed words using synonym[3] and antonym[4] queries. Polarity were associated with each word, synonyms inherited the polarity from the parent and antonyms got the opposite polarity. The research based itself on words having paths to each other, for example, one path from good to bad could be: $Good \rightarrow Serious \rightarrow Hard \rightarrow Bad$. The significance of a path decreased as a function of its length or depth from a seed word. The final score of each word was the sum of the scores received over all paths. For paths that alternate between positive and negative terms, the number of apparent sentiment alternations called flips were calculated. The final score only took the paths whose flip value was within a threshold into account. Many words lying in the middle of the distribution were ambiguous, in other words they could not be classified as either positive or negative. Since the assigned scores followed a normal distribution, ambiguous words were discarded by only taking the top $X\%$ words from both extremes of the curve into account.

3.4 Related work concerning adjectives

The idea that adjectives often describe the sentiment of sentences, has led to several researches studying adjectives [48, 49, 50].

[2] Salience: The quality of being particularly noticeable or important

[3] Synonym: A word that means exactly or nearly the same as another word

[4] Antonym: A word opposite in meaning to another.

In [48], Hatzivassiloglou and McKeown presented a technique for finding the semantic orientation of adjectives from constraints of conjunctions[5]. First, they extracted all conjunctions of adjectives from the corpus. Then a log-linear regression model was used to combine the information from different conjunctions to determine if each pair of conjoined adjectives were of the same or different orientation. The adjectives were further separated into two sets of different orientation, by a clustering algorithm. For each set, the average frequency was calculated, and the set with the highest frequency was labeled positive. They obtained 90% precision for adjectives that occurred in a modest number of conjunctions in the corpus and a 82% accuracy when each conjunction was considered independently. However, their system relies on a large corpus and need a large amount of manually tagged training data to perform optimally. This research differentiates from mine because it does not perform any kind of sentiment analysis, it rather decides the sentiment of adjectives.

In [49], Whitelaw et al. presents a method for sentiment classification based on extracting and analysing appraisal groups such as “very good” and “not terribly funny”. More specifically, they investigated and built a lexicon of appraising adjectives and their modifiers. The lexicon was built, using a semi-automatic system that heuristically extracted adjectival appraisal groups from texts, and computed their attribute values according to this lexicon. Documents were further represented as vectors of relative frequency features computed over these groups. Finally, the positive documents were separated from the negative documents by a Support Vector Machine algorithm. They obtained a 78% performance, when adjectival appraisal group features were used alone and increased their accuracy to 90%, when they in addition included standard bag of words algorithms. In our research we will investigate stand-alone adjectives and not appraisal groups.

Our research use an approach, not that different from the approach proposed by Hu and Liu in [50]. In their research, they aimed to mine and to summarize all opinions in customer reviews of a product. Their focus was on summarizing the opinions for each product feature and not for the product as a whole. Given the inputs, their system first downloaded all the reviews, and put them into the review database. It then found frequent features that many people had expressed their opinions on. For each frequent feature, the adjective words were identified using a natural language processing method. The infrequent features were found using the extracted adjective words. For each of the adjective words, they decided the semantic orientation based on a bootstrapping technique and by the use of synonyms and antonyms in WordNet. Depending

[5] Conjunction: A word used to connect clauses or sentences or to coordinate words in the same clause (e.g. and, but, if).

on the result they obtained from this step, the opinion orientation of the whole sentence was decided. The goal of Hu and Liu in this research was to summarize the opinions on the product features, slightly different from the goal of this thesis.

3.5 Other related approaches

Financial markets react to news very quickly and it is therefore necessary to react quickly in order to make money. Normal text data has a very high dimensionality. Therefore reducing the number of features needed to classify a document reduces the time spent on doing so. Simen Kind Gulbrandsen has looked into Conditional Random Fields and how that can reduce the feature space in his article “Improving News Trades using CRF” [51]. He used a dataset of mandatory stock messages released on the Oslo Stock Exchange. These messages were further on combined with financial data on all trades completed in a three-year period. Important features were extracted by a trained Conditional Random Field on the textual data. The features were further on used to train a Support Vector Machine and a Random Forest classifier. The research found that reducing the number of features, resulted in a 4% point reduction in accuracy and a 81,25% reduction in run-time. In other words, there is possible to reduce the feature space without significant reduction in accuracy, but Gulbrandsen also concludes that this method is not good enough for making significant profit on the financial market.

Chapter 4

Classification

This chapter will give a detailed description of the classification methods used in this project. Those are Support Vector Machines and Random Forest. Some other approaches and concepts are also presented, to make the presentation of those algorithms easier.

4.1 Support Vector Machines

Support vector machines (SVM) is a vector based machine learning algorithm, where the goal is to find a decision boundary, called a hyperplane, between two or more classes. The hyperplane is created so that the distance to any point in the training data is as large as possible. This is an easy task in a 2-dimensional system, where the hyperplane will be a straight line.

The margin of the classifier is defined as the distance from the hyperplane to the closest data point. This means that the decision function for the SVM classifier (position of the hyperplane) is fully determined by, often a small number of, data points. These points are referred to as the support vectors. Figure 4.1 shows the hyperplane, the margin and the support vectors.

Minimizing the distance between the hyperplane and the data points, means that the margin get maximized. This works well because points near the hyperplane represents very uncertain classification decisions. With two classes, it is almost a 50% chance

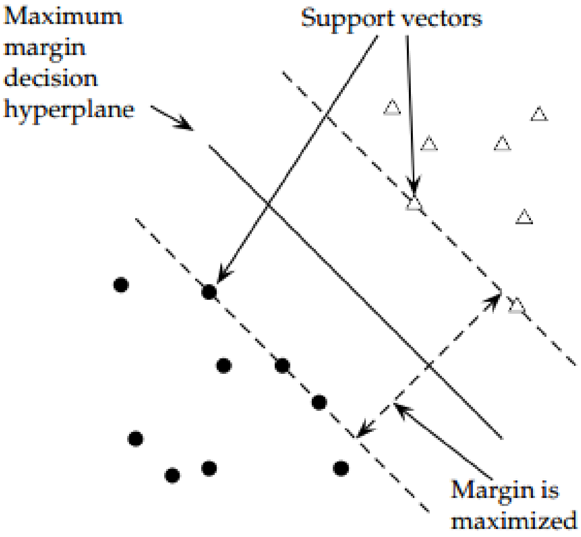


Figure 4.1: Illustration of the main components of the SVM algorithm. Image obtained from [23].

the classifier decides for either class. A classifier with a large margin does not have to make these classification decisions based on a low certainty. This gives the classification a safety margin, i.e. a misclassification will not be caused by a small error in measurements or a small error in document variation [23].

The following presentation of SVM, follows the definitions and presentation of SVM in Chapter 15 in the book "Introduction to Information Retrieval" by Manning et al. [23]:

The hyperplane, can be defined by an intercept term b and a hyperplane normal vector $\vec{w}$, often called weight vector, which is perpendicular to the hyperplane. This means that all points $\vec{x}$ on the hyperplane satisfies $\vec{w}^\mathsf{T}\vec{x} = -b$. Suppose we have a set of training data points $\mathbb{D} = (\vec{x}_i, y_i)$, where $\vec{x}_i$ is the point in space and y_i is the class label. For a two class system, one class is denoted as $+1$ and the other as -1. The class is determined by which side of the hyperplane the point is located. The linear classifier is then defined as:

$$f(\vec{x}) = sign(\vec{w}^\mathsf{T}\vec{x} + b) \tag{4.1}$$

The $sign$ operator outputs whether the argument is greater or lower than zero, giving an output of either $+1$ or -1. This is used as the prediction. An output of -1 means the classifier predicts one class, and an output of $+1$ means the classifier predicts the other class. The distance from the hyperplane is used as a measure of how good the classification is. If the classification of a point is far away from the hyperplane, there is a big chance the classification of that point is correct.

The distance from the hyperplane, called the functional margin, of the i'th example $\vec{x}_i$ with respect to a hyperplane $< \vec{w}, b >$ is defined by Manning et al. [23] as the quantity $y_i(\vec{w}^\intercal \vec{x}_i + b)$. The functional margin of the hole data set is then defined as twice the functional margin of the point(s) in the data set with minimal functional margin.

There is one problem with this definition; the value is underconstrained. This means that we can make the functional margin as wide as we want just by upscaling $\vec{w}$ and b. For example by replacing $\vec{w}$ with $5\vec{w}$ and b with $5b$, the functional margin $y_i(5\vec{w}^\intercal \vec{x}_i + 5b) = 5y_i(\vec{w}^\intercal \vec{x}_i + b)$ gets five times as large. The solution to this problem is to set a constraint on the size of the $\vec{w}$ vector. Let $\vec{x}'$ be the point on the hyperplane closest to $\vec{x}$, r the distance from $\vec{x}$ to the hyperplane and y a factor, so that when multiplied with, it only changes the sign of the $\vec{x}$ on either side of the hyperplane:

$$\vec{x}' = \vec{x} - yr\frac{\vec{w}}{|\vec{w}|} \tag{4.2}$$

Since $\vec{x}'$ lies on the hyperplane, it satisfies $\vec{w}^\intercal \vec{x}' + b = 0$, giving us:

$$\vec{w}^\intercal(\vec{x} - yr\frac{\vec{w}}{|\vec{w}|}) + b = 0 \tag{4.3}$$

Solving for r gives us:

$$r = y\frac{\vec{w}^\intercal \vec{x} + b}{|\vec{w}|} \tag{4.4}$$

As mentioned before, the closest points to the hyperplane are called the support vectors. Manning et al. [23] defines the geometric margin of the classifier as twice the minimum value over data points for r given in Equation 4.4. You can think of it as “the maximum width of the band that can be drawn separating the support vectors of two classes” [23]. The geometric margin is invariant to scaling parameters because it is normalized by the length of $\vec{w}$. This means that we can scale $\vec{w}$ as much as we want, without changing the geometric margin. Let us scale $\vec{w}$ so that the functional margins are at least 1 and for some data vectors are equal to 1. Thus, for all vectors in the set we have:

$$y_i(\vec{w}^\intercal \vec{x}_i + b) \geq 1 \tag{4.5}$$

Our goal is to maximize the geometric margin. From Equation 4.4 we know that all distances from the hyperplane is given as $r_i = y_i \frac{\vec{w}^\intercal \vec{x} + b}{|\vec{w}|}$. Knowing this, we know that the geometric margin is given as $\rho = \frac{2}{|\vec{w}|}$. Maximizing $\rho = \frac{2}{|\vec{w}|}$ is the same as minimizing $\rho = \frac{1}{2}\vec{w}^\intercal \vec{w}$. Maximizing the geometric margin is therefore the same as finding $\vec{w}$ and b so that:

- $\rho = \frac{1}{2}\vec{w}^\intercal \vec{w}$ are minimized and
- for all $(\vec{x}_i, y_i), y_i(\vec{w}^\intercal \vec{x}_i + b) \geq 1$

This is now an optimization problem of a quadratic function. The solution to such a problem, is to transform it into a dual problem, where a Lagrange multiplier α_i is associated with each constraint $y_i(\vec{w}^\intercal \vec{x}_i + b) \geq 1$:

Find $\alpha_1, ..., \alpha_N$ such that $\sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i^\intercal \vec{x}_j$ is maximized and

- $\sum_i \alpha_i y_i = 0$
- $\alpha_i \geq 0$ for all $1 \leq i \leq N$

The solution is then:

- $\vec{w} = \sum \alpha_i y_i \vec{x}_i$
- $b = y_k - \vec{w}^\intercal \vec{x}_k$ for any $\vec{x}_k$ such that $\alpha_k \neq 0$

Each non-zero α_i (most of them will be zero) indicates that the corresponding $\vec{x}_i$ is a support vector. Finally, we get the classification function defined as:

$$f(\vec{x}) = sign(\sum_i \alpha_i y_i \vec{x}_i^\intercal \vec{x} + b) \tag{4.6}$$

The assigned class is determined by the sign of this function. A benefit with the algorithm is that if a point is inside the margin of the classifier, or another threshold t, the classifier can return “not sure” rather than one of the classes.

A problem with this solution is that it assumes the whole dataset is separable with no noise and that makes the solution prone to overfitting. Overfitting occurs if a model "memorize" the training data instead of "learning" to generalize from trend. This causes the model to describe random error or noise, instead of the underlying relationship. An overfitted model has poor predicitve performance, because it often overreacts to small variations in the training data, like you can see illustrated in Figure 4.2.

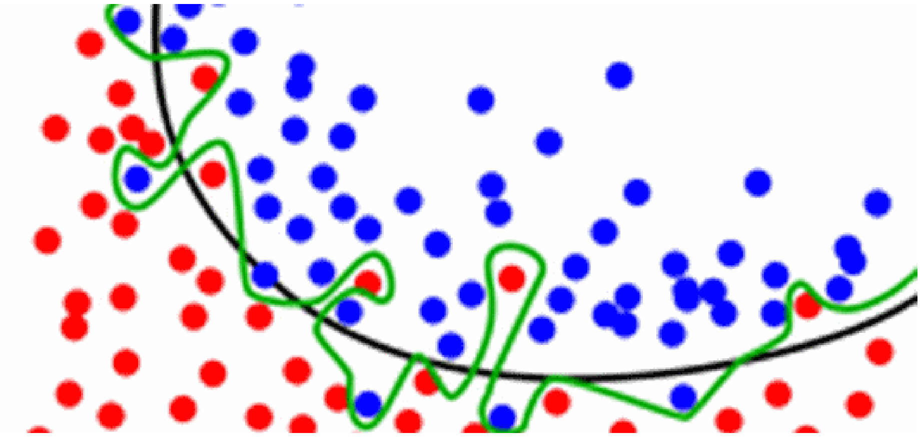


Figure 4.2: The green line illustrates an overfitted model. Instead of describing the underlying relationship, it describes random noise confusing the results. Image obtained from the Kaggle competition: Don't Overfit![1]

A solution to this problem is called a **soft margin classification**. It allows that some of the data points are present on the wrong side of the hyperplane, by paying a cost for each misclassified point. The algorithm introduces slack variables ξ_i that allows $\vec{x}_i$ to not meet the margin requirement at a cost proportional to the value of ξ_i. The formula for the optimization problem in SVM with the introduction of the slack variables then becomes [23]:

Find $\vec{w}, b$ and $\xi_i \geq 0$ such that:

- $\frac{1}{2}\vec{w}^\intercal\vec{w} + C\sum_i \xi_i$ is minimized and
- for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^\intercal\vec{x}_i + b) \geq 1 - \xi_i$

Here, the paramter C is a regularization term set beforehand, that deals with errors in training. A large C indicate a high cost for errors in classification, making the hyperplane narrower. A small C on the opposite, indicates a smaller cost for errors, making the hyperplane wider. The dual problem for soft margin classification is given as [23]:

[1] https://www.kaggle.com/c/overfitting

Find $\alpha_1, ..., \alpha_N$ such that $\sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i^\intercal \vec{x}_j$ is maximized, and

- $\sum_i \alpha_i y_i = 0$
- $0 \leq \alpha_i \leq C$ for all $1 \leq i \leq N$

And the solution is given as [23]:

- $\vec{w} = \sum \alpha y_i \vec{x}_i$
- $b = y_k(1 - \xi_k) - \vec{w}^\intercal \vec{x}_k$ for $k = \arg\max_k \alpha_k$

The classification can be done in terms of dot products with the data points, as in Equation 4.6. The only difference is that in the standard SVM approach, the non-zero α_i indicates a support vector, and in the soft margin approach it can also be a data point inside the margin. Notice that this could lead to a problem with computational complexity, making the classification slow, for large data sets [23].

We have now presented the SVM method for a two-class classification. In many cases, we have more than two classes. **Multiclass SVM** solves this by splitting the problem into multiple binary classification problems. Two simple approaches are the "one versus all" and the "one versus one" approaches. The first approach chooses the class which classifies the input with the greatest margin and the second approach chooses the class selected by the most classifiers. More complex solutions exists, but we will not go in detail on those methods.

We have not yet looked at data sets that are not linearly separable, meaning there does not exist a line that split the classes without errors. A solution to this, is the **nonlinear SVM** that map all data points into a higher dimension and use a linear separator in this dimension, as showed in Figure 4.3. This is done by the use of the "kernel trick", a trick that enable the method to operate in a high-dimensional space, without computing the coordinates of the data in that space. Kernel functions are functions where the value equals the dot product of two points in a arbitrarily high dimension. Let K be the kernel function and let $K(\vec{x}_i, \vec{x}_j) = \vec{x}_i^\intercal \vec{x}_j$. The classifier is then defined as [23]:

$$f(\vec{x}) = sign(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b) \tag{4.7}$$

The kernel function can be computed as: $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^\intercal \phi(\vec{x}_j)$, where ϕ is the transformation formula. The use of this "kernel trick" gives us the power to raise the data set to a higher dimension, computationally cheaper than explicit computation of the coordinates.

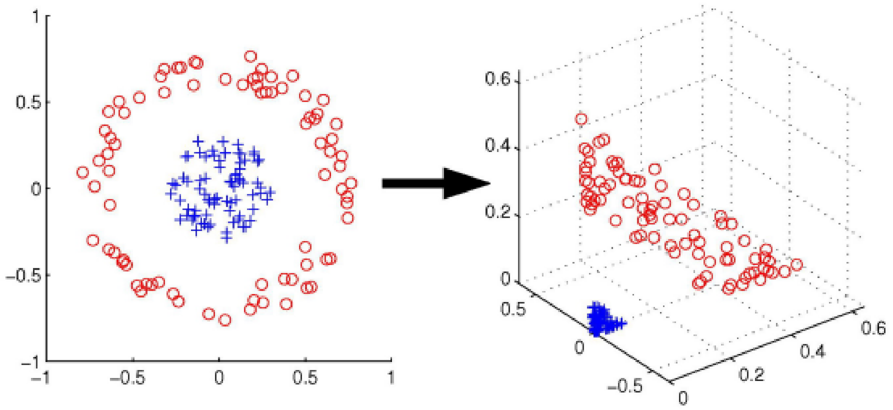


Figure 4.3: Transforming a dataset to be linearly separable. The dataset in $\mathbb{R}^2$ (to the left) are not linear separable. To the right, the dataset is transformed into $\mathbb{R}^3$ by the transformation: $[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$. In $\mathbb{R}^3$ this set is linear separable by a plane. Image obtained from [52]

4.2 Random Forest

Random Forest is a classification algorithm, based on decision trees. The difference is that instead of training one tree, the Random Forest algorithm creates many trees to make it's decisions.

Before we can explain the Random Forest classifier, we need to take a look at bootstrap aggregating, also called bagging. It is a technique for reducing the variance of an estimated prediction function, as well as it helps to avoid overfitting. It is designed to improve stability and accuracy of machine learning algorithms used in classification and regression. Since the focus of this research is classification, we will only explain how bagging and random forest are used for classification.

The bagging technique works as follows [53]:

Given a set of N instances, each belonging to one of K classes. For each repetition $t = 1, ..., T$ a training set, called a bootstrap sample, are sampled with replacement from the original instances. These sets are the same size as the original set, but they are not equal. Since the original set is sampled with replacement, some instances may not appear, and others appear more than once. A learning system, that constructs a classifier from the instances, then generates a classifier C^t from each bootstrap sample. To classify an instance x, each classifier C^t calculates the class k, such that $C^t = k$ and gives a vote for that class. The class k that got the most votes, are used as the final answer.

Random Forests [54] differ from bagging in only one way. A random subset of the features, are selected at each candidate split in the learning process, by a modified tree learning algorithm. The reason behind this, is the correlation between the trees in the original bagging method; if one or a few of the features are very strong predictors for the classification, these features will be selected in many of the trees, causing the trees to become correlated. Reducing this correlation will reduce the total error rate of random forests.

A proof that the Random Forest algorithm does not overfit, is shown by Breiman [54], by the use of the Strong law of Large Numbers. This means that the error rate will converge to an asymptotic value, no matter how many trees created and how many variables chosen.

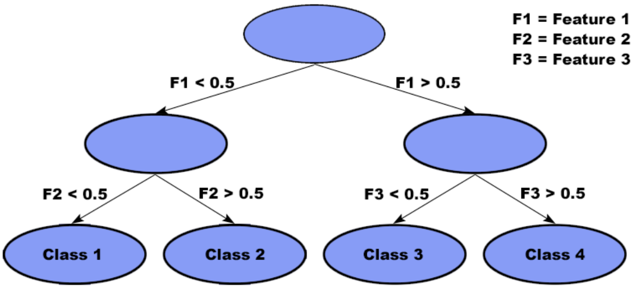


Figure 4.4: A single decision tree with four classes, evaluating three features. An instance with the feature vector (0.2,0.5,0.0) would be classified as Class 2, and an instance with feature vector (0.9,0.6.0.2) would be classified as Class 3.

Each tree in the Random Forest algorithm is constructed as follows:

- Sample with replacement, as many cases in the training set as there are instances.
- Choose the number N of variables to be evaluated in each tree node. For each node, pick N variables at random. Calculate and perform the best split based on these variables in the training set.
- Grow the tree as large as possible, without pruning.

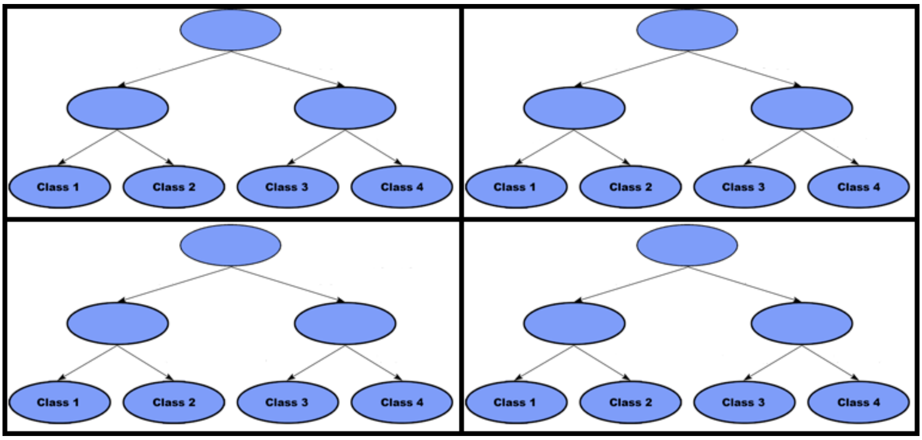


Figure 4.5: Random Forest generates multiple trees. The majority vote from all the trees are used as the prediction for the instance. For example if 3 of the trees votes for class 2 and one tree vote for class 1, the class is predicted to be 2.

An important advantage of the Random Forest algorithm is its use of out-of-bag (OOB). OOB is estimated internally, during the run as follows: Approximately one third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree. Each of these cases are then runned down the kth tree to get a classification. Let j be the class that got the most votes every time case n was OOB. The OOB error is then defined as the proportion of times that j is not equal to the true class of n, averaged over all cases. In Figure 4.6, we provide an small example computing the OOB error.

In this way, a test set classification is obtained for each case in approximately one-third of the trees and it proves to be almost identical to what is obtained by N-fold crossvalidation[3] [55]. It also gives Random Forest the ability to rank the importance of each feature. The features that has the biggest impact on the OOB error, are the features most important to the classification. This method can therefore be used to reduce the number of features needed for classification.

[2] http://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf

[3] N-fold crossvalidation: The data is randomly split into N equal sized parts. One of these parts is retained as the validation data for testing the model, while the remaining $N - 1$ parts are used as training data. This is repeated N times, with each of the N parts used exactly once as the validation data.

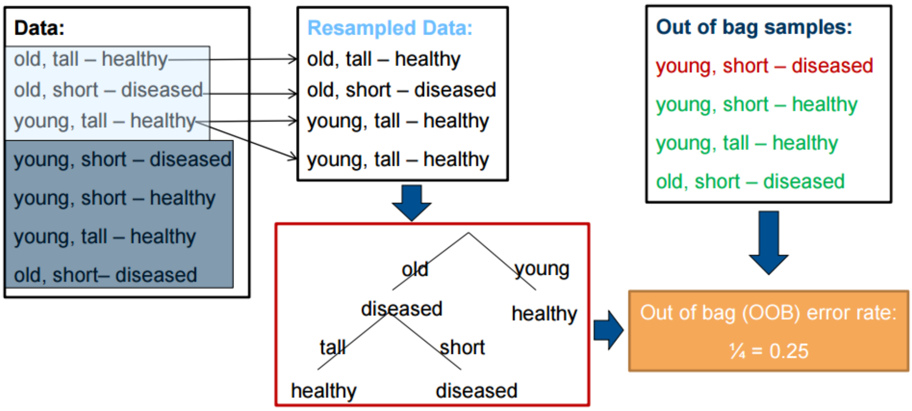


Figure 4.6: An example calculating oob-error. Image obtained from[2].

Chapter 5

Approach

This chapter describes our system in detail. The first section presents an overview of the system flow. The second section describes the details of the data collection and the third section explains the preprocessing of the datasets. The following section provides a detailed explanation of the approach and the system components. In the two last sections we explain how we prepared the data for classification along with the classification itself. The implementation of the system is mainly performed in Python, with an exception of the POS-tagging, performed in Java.

5.1 System flow

In this section we presents a high-level description of how our system is constructed: A collection of movie reviews, are preprocessed, involving part-of-speech tagging. Further, we locate the adjectives, find their sentiment and count them. The count will be used as attributes in the classification process. Before we can start the classification, we need to transform the results into a format that is ready for classification. Finally the classification is performed by Weka[1]. The following sections will describe each of these components in greater detail. The system flow of this project is divided into the following steps:

[1] http://www.cs.waikato.ac.nz/ml/weka

1. **Obtaining the dataset**
2. **Preprocessing of the data**
   (a) Part-of-speech tagging
   (b) Feature selection
3. **Finding sentiment of adjective**
   (a) Find the adjectives
   (b) Find synonyms and antonyms of these adjectives
   (c) Decide sentiment of the synonym and antonyms
   (d) Use results from last step to find sentiment of adjective
   (e) Count the positive and negative adjectives for each file and use it as attributes for classification
4. **Prepare the dataset for Weka**
5. **Classification**
   (a) Random Forest
   (b) Support Vector Machine

5.2 Collection of the data

We used three different datasets for our experiments. To minimize confusion, we will refer to them as the "small dataset", the "small + objective dataset" and the "large dataset", in the remainder of the report. The following paragraphs will give a detailed description of the datasets.

Small dataset
The first dataset consists of 1000 positive and 1000 negative movie reviews, gathered from the dataset "polarity dataset v2.0"[2], first introduced in [5]. This dataset has often been used as a benchmark for sentiment classification and consists of a subset of the movie reviews found in the IMDb archive of the rec.arts.movies.reviews newsgroup[3]. The "true sentiment" are based on the rating of the review.

[2] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] http://reviews.imdb.com/Reviews

Small + objective dataset

In our second dataset, we tried to combine the first dataset with objective texts. In the research [56], Koppel and Schler shows how important a neural class is, in the area of sentiment analysis and that the introduction of the neutral category can even improve the overall accuracy. We intended to include objective texts as noise in our dataset, but without success.

First, we gathered movie synopsises from IMDb , with help from the Python package IMDbPY[4]. A problem occured, when many synopsises returned empty. We think the reason for this, is that the synopsises were locked by IMDb. According to their SynopsisHelp page, a synopsis can be locked for three reasons; permanently for policy pages, temporarily, in the case of an edit war or temporarily, in the case of frequent vandalism. This problem caused it to be really time consuming, to substitute the empty synopsises with new ones.

Instead of using this python package, we found the "CMU Movie Summary Corpus"[5] introduced in [6]. This is a corpus of 42 306 movie plot summaries extracted from Wikipedia. We extracted 2000 of them and split them into single files. When we tried to classify this dataset we encountered a new problem. The classification algorithms classified the objective texts perfectly or nearly perfect (without including our system). This is probably because the two datasets are so different from each other, making them easy to classify. The dataset did therefore not function as noise in our dataset, as intended.

In the belief that this problem was caused by a much more formal language in the objective texts than in the reviews, we tried to extract objective texts from forum posts. We extracted posts[6] from Apple Discussion, Google Earth, CNET and objective review data from Amazon MP3 Data Set. We tested the datasets seperately and together in a mix. All resulting in the same problem as before.

None of these datasets functioned as noise in the data as intended. However, we will present the results using the combination of the small dataset and the 2000 reviews extracted from the "CMU Movie Summary Corpus", even though the results will be unrealistic high.

Large dataset

The third dataset is a larger dataset containing 25 000 positive and 25 000 negative

[4] Downloaded from: `http://imdbpy.sourceforge.net/`

[5] `http://www.cs.cmu.edu/~ark/personas/`

[6] Gathered from `http://times.cs.uiuc.edu/~wang296/Data/`

movie reviews gathered from “Large Movie Review Dataset v1.0”[7], first introduced in [7]. It is intended as a larger benchmark dataset for sentiment classification, and consists of movie reviews, gathered from IMDb. The “true sentiment” is also here, gathered from the rating given by the writer of the review. The negative reviews have a rating 4 or below and the positive reviews have a rating 7 or higher from a maximum rating score of 10.

5.3 Preprocessing of the data

Before information can be extracted from the data, it must be preprocessed to prepare it for further processing. There are many operations that can be applied for this task, including elimination of stopwords, stemming, lemmatization and part-of-speech tagging. This section will explain the techniques we find relevant for this research.

5.3.1 Part-of-Speech tagging

A Part-of-Speech tag or PoS tag is a tag describing the lexical group the word is a part of. Examples of important PoS tags are verb, adjective and noun. The process are based on both the word’s definition and the context of the word, i.e. the word’s relationship with adjacent and related words in a phrase. To illustrate how PoS tags are used in practice, look at this example:

This	movie	was	fantastic	.
DT	NN	VBD	JJ	.

The JJ PoS tag is used to mark an adjective. In addition to the JJ adjective tag, the JJR tag marks adjectives with the comparative ending -er, and the JJS tag marks adjectives with the superlative ending -est. In the example above, we can see that the adjective is determining the sentiment of the sentence. How we have used adjectives as sentiment deciders will be further explained in Section 5.4. A complete overview of standard English POS tags and their expanded forms [57] are given in Appendix A.

[7] `http://ai.stanford.edu/~amaas/data/sentiment/`

For this project, we used an open source natural language processing (NLP) package called LingPipe[8] for performing the PoS tagging. Lingpipe is a Java package that provides an API for text processing using computational linguistics. The implementation follow the Lingpipe tutorial[9], with small adjustments; The input are read in as files, and the output saves each processed files into a new file.

5.3.2 Feature selection

A standard approach to represent documents is by use of vectors, where each dimension represents one feature. The terms are often weighted and in this research, we used the TF-IDF weighting method. TF stands for term frequency and is the number of times the term occurs in the specific document. In other words, TF tells us how widespread the term is in the document and the more often a term occurs in in a document, the better the term describes the document. DF is the number of documents in the whole collection that contains the term. A term that occurs in almost every document is not helpful in discriminating between classes. Typical words can be “the”, “and”, “as” and so on. We want terms that occur in fewer documents to be weighted higher and this is what inverted document frequency IDF does. The inverted document frequency is defined as follows [58]:

$$IDF = \log \frac{|N|}{DF} \tag{5.1}$$

where N is the total umber of documents in the collection and DF is the document frequency. By multiplying TF with the IDF we get a weight that increases with many occurences of a term within a document and decreases with the number of documents that contain the term. The TF-IDF is defined as TF multiplied with IDF as follows [58]:

$$TF\text{-}IDF = TF * \log \frac{|N|}{DF} \tag{5.2}$$

5.3.3 Stemming and lemmatization

Stemming is a technique for reducing the feature numbers [59]. The technique count all variations of the words as a single feature, by only using the stem of the word. An

[8] http://alias-i.com/lingpipe/

[9] http://alias-i.com/lingpipe/demos/tutorial/posTags/read-me.html

example is the words “walk”, “walks”, “walking”. They would all be instances of the stem “walk”. Lemmatization is very similar to stemming, but instead of finding the stem of the word, if finds the normalized form of the word. An example is the words “compute”, “computing” and “computer”. The stem would be “comput”, but the normalized form of the word (lemma) is “compute”. Another example is the word “better”. The lemma of the word is “good”. This is missed by the stemming, because it requires a dictionary look-up. We did not use stemming or lemmatization in our experiments, as both stemming and lemmatization has shown to not increase performance [60].

5.4 Finding the sentiment of adjectives

In general we can say that adjectives have the same sentiment as their synonyms and opposite sentiment as their antonyms. We use this fact, to find the sentiment of the extracted adjectives. The procedure is divided into three subtasks:

1. Identify all synonyms and antonyms of the adjectives.
2. Determine the sentiment of the synonyms and antonyms.
3. Determine the sentiment of the adjectives, based on the results from step 2.

The synonyms and antonyms for the given adjective are found using WordNet[10] [61]. WordNet is a lexical database for the English language, where adjectives are organized into a bipolar cluster system. As you can see illustrated in Figure 5.1, the cluster for the antonym pair wet/dry, consist of two half clusters, one for synonyms of wet and one for synonyms of dry [62].

Both the synonym list and the antonym list returned from WordNet consists of several duplicates. These duplicates are removed, before we proceed to the second step. Unfortunately, WordNet do not include information about the semantic orientation for given words [62]. Because of this, we used SentiWordNet[11] for the task of finding the sentiment of the synonyms and antonyms. SentiWordNet is a dictionary of 117 569 english words, assigned with a neutral, negative and a positive weight.

The sentiment of each adjective is calculated based on the sentiment of the synonyms and the opposite sentiment of the antonyms. The majority vote of this result (positive or negative), decide the sentiment of the adjective. For each review, we count the total

[10] Wordnet can be downloaded from `https://wordnet.princeton.edu/`

[11] SentiWordNet can be downloaded from `http://sentiwordnet.isti.cnr.it/`

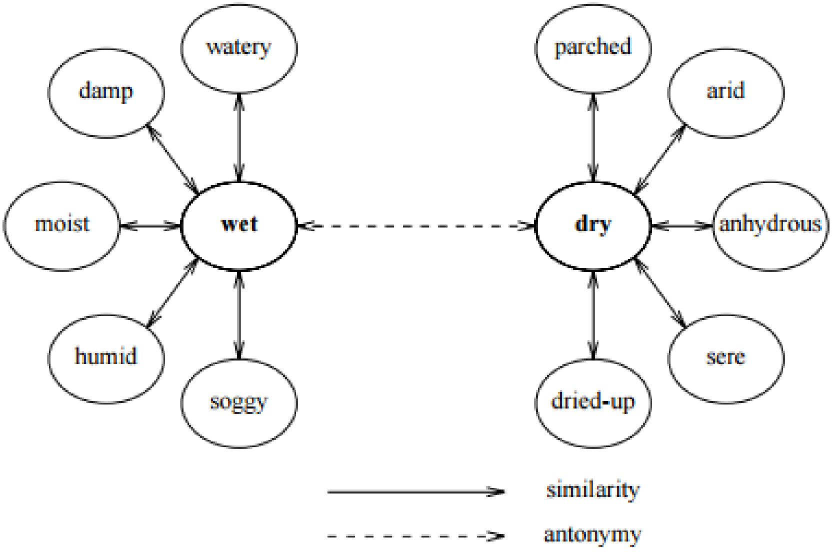


Figure 5.1: Adjective structure in Wordnet. Image obtained from [62].

of positive and negative adjectives. In the next section, we will explain how we further prepare the adjective counts and the reviews for classification in Weka.

5.5 Preparing the data for Weka

Before the data can be classified, we need to prepare the dataset for Weka. Weka's native storage method is the ARFF file format. A small example of an ARFF file can be viewed in Figure 5.2. The ARFF file starts with defining the name of the dataset (relation), followed by a block of attributes. There are 5 kinds of attributes accepted by the ARFF format: nominal, numeric, string, date and relation-valued attributes. In our experiment, we use three of those attributes. String attributes are followed by the keyword string. Nominal attributes are followed by two curly braces, containing the set of values the attribute can have and numeric values are followed by the keyword numeric. The ARFF file does not say which class you want to predict. This means that if you want to investigate more than one attribute and how well that attribute can be predicted from the others, the same file can be used several times, without changing

```
@relation MovieSentiment

@attribute text string
@attribute posAdj numeric
@attribute negAdj numeric
@attribute sentiment {pos,neg,obj}

@data
"This movie was fantastic" , 1 , 0 , pos
"This movie was awful and terrible" , 0 , 2 , neg
"The movie was about a family living in the forest" , 0 , 0, obj
```

Figure 5.2: ARFF file for the MovieSentiment data

it. In our experiment, we just want to predict one class; the sentiment. After the attribute block, follows a @data line that signals the start of the data set. Instances of the data set appear as one per line, with the attribute values separated by commas. If a value is missing, it is represented with a question mark. Notice that the string instance is represented within quotation marks. This means that character escaping on question marks and quotation marks is needed inside the string (replace the symbol with \symbol).

In our case the ARFF file look similar to the one presented in Figure 5.2. Each instance contain the text, the positive adjective count, the negative adjective count and the true sentiment. Note that the strings presented in this example are not collected from the corpus, but made up, for easier readability. Also note that the adjective counts do not represent realistic values. Most of the files, contain both positive and negative adjectives.

For each of our experiments, we created two ARFF files, one for training and one for testing. The ARFF files was constructed by a small Python script, taking the source files, the “true sentiment“ and the adjective counts as input. In the test files, all the “true sentiment” instances were replaced with question marks. This assures us, the classification is not, in any way, confused by the “true sentiment“ scores. However, this leads to Weka not being able to compute classification results, because Weka has no idea what results are correct and what results are wrong. Therefore, all the results calculated in the next chapter, are done so manually. We chose to compute the exact same evaluation metrics that Weka originally computes (if “true sentiment“ were provided in the test set), with one exception; the results for the ROC Area.

5.6 Classification

For our classification, we used the program Weka[12] (Waikato Environment for Knowledge Analysis), developed by the University of Waikato in New Zealand [63]. It is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. In our experiments, Weka were only used for the classification task.

As mentioned in the previous section, we chose to split the dataset into one set for training and one set for testing at a predefined ratio. This means that the classification model is based on the training set alone and the testset is never seen before testing. For the small dataset and the small + objective dataset, we choose to use 66% for training and 33% for testing. The large dataset was already split into a training set and a test set at a 50% ratio. We did not change this ratio for our experiments.

We chose to use the support vector machine and the random forest algorithms for classification. Weka have a built in support for both algorithms. Both algorithms are used as blackbox algorithms, meaning they are used with default settings. We did in total 12 experiments, and information about each run is presented in the next chapter, together with the evaluation metrics.

[12] Weka can be downloaded from `http://www.cs.waikato.ac.nz/ml/weka/`

Chapter 6

Evaluation and results

Before presenting the results, this chapter describes the evaluation metrics of the experiments. The results include both the confusion matrices and the detailed accuracy of the classification. The results are presented as they are and will be further discussed in Chapter 7.

6.1 Evaluation metrics

This section will explain the confusion matrix along with the evaluation metrics: “TP Rate/Recall”, “FP Rate”, “Precision”, “F-Measure” and “Accuracy”. To be able to understand the results presented later in this chapter, these concepts are important.

A **Confusion matrix**, also called an error matrix, is a matrix visualizing the performance of an algorithm. Each column of the matrix represents the instances in a predicted class and each row represents the instances in an actual class. Look at Table 6.1 to see how the confusion matrix can be read in our case. The correct classifications are marked with bold text.

a	b	c	<- Classified as
True class: a **Predicted class: a**	True class: a Predicted class: b	True class: a Predicted class: c	a = pos
True class: b Predicted class: a	**True class: b** **Predicted class: b**	True class: b Predicted class: c	b = neg
True class: c Predicted class: a	True class: c Predicted class: b	**True class: c** **Predicted class: c**	c = obj

Table 6.1: Example of a confusion matrix

TP stands for True Positive. A true positive means a correctly classified positive document. For example, if a document is classified positive and the "true sentiment" also is positive, the document is counted as a True Positive. The **TP Rate**, also called **Recall**, measures the proportion of positives that are correctly identified by the classifier, and is calculated as follows:

$$TPRate = Recall = \frac{TP}{P} = \frac{TP}{TP + FP} \tag{6.1}$$

Here FN means a False Negative. A False Negative is a document that is returned as negative by the classification, but in reality is positive. Opposite of the FN is the False Positive FP. FP occurs if a document is negative and the classification returns a positive result. In other words, the **FP Rate** defines how many incorrect positive results that occur among all the negative samples, and can be calculated as:

$$FPRate = \frac{FN}{N} = \frac{FN}{FN + TN} \tag{6.2}$$

TN represents a true negative; the classification correctly classifies a negative document. A summary of what TP, FP, FN and TN means, is presented in Table 6.2.

	True sentiment positive	True sentiment negative
Classified positive	TP	FP
Classified negative	FN	TN

Table 6.2: Description of “True Positive”, “False Positive”, “False Negative” and “True Negative”

An often used method to evaluate classification tasks is **precision**. Precision tells us which fraction of the documents that are correctly classified:

$$Precision = \frac{TP}{TP + FN} \tag{6.3}$$

F-measure is a measure of a test's accuracy and is the harmonic mean of precision and recall. It has a parameter that sets the tradeoff between recall and precision. The standard F-measure F_1 assigns equal importance to recall and precision, and is defined as follows:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FN + FP} \tag{6.4}$$

The F_1 score can be viewed as the weighted average of the precision and recall. The F_1 score therefore reaches its best value at 1 and its worst value at 0. The general formula of a real positive number β is defined as:

$$F_\beta = (1 + \beta^2)\frac{Precision * Recall}{(\beta^2 * Precision) + Recall} \tag{6.5}$$

By adjusting β we can adjust the importance of precision and recall.

Accuracy is the proportion of the total number of predictions that were correct. Be aware that if the dataset is very unbalanced, accuracy is not a reliable metric for the real performance of a classifier. For example, if we have 95% positive reviews and 5% negative reviews and the classifier predicts all the instances to be positive, we will get an overall accuracy of 95%. In practice the classifier have a 100% recognition rate for the positive reviews and 0% recognition rate for the negative reviews. If this problem should occur, it would be easily detected by the calculation of the other metrics. The accuracy is determined using the equation:

$$Accuracy = \frac{TP + TN}{Total} = \frac{TP + TN}{TP + FP + FN + TN} \tag{6.6}$$

6.2 Results by Random Forest on the small dataset

Table 6.3: Confusion matrices - Random Forest - Small dataset

a	b	<- Classified as
265	75	a = pos
63	277	b = neg

(a) without adjectives

a	b	<- Classified as
289	51	a = pos
40	300	b = neg

(b) with adjectives

	TP Rate	FP Rate	Precision	F_1	Accuracy	Class
	0.779	0.185	0.808	0.793	0.797	pos
	0.814	0.221	0.787	0.801	0.797	neg
Weighted Avg.	0.797	0.203	0.798	0.797	0.797	

Table 6.4: Detailed accuracy by class - RF - small dataset - without adjectives

	TP Rate	FP Rate	Precision	F_1	Accuracy	Class
	0.850	0.118	0.878	0.864	0.866	pos
	0.882	0.150	0.885	0.868	0.866	neg
Weighted Avg.	0.866	0.134	0.882	0.866	0.866	

Table 6.5: Detailed accuracy by class - RF - small dataset - with adjectives

Most important to notice here, is that the accuracy has improved from 0.797 to 0.866.

6.3 Results by Support Vector Machines on the small dataset

Table 6.6: Confusion matrices - SVM - Small dataset

a	b	<- Classified as
272	68	a = pos
76	264	b = neg

(a) without adjectives

a	b	<- Classified as
287	53	a = pos
58	282	b = neg

(b) with adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.800	0.224	0.782	0.791	0.788	pos
	0.776	0.200	0.795	0.786	0.788	neg
Weighted Avg.	0.788	0.212	0.789	0.789	0.788	

Table 6.7: Detailed accuracy by class - SVM - small dataset - without adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.844	0.171	0.832	0.838	0.837	pos
	0.829	0.156	0.842	0.836	0.837	neg
Weighted Avg.	0.837	0.164	0.837	0.837	0.837	

Table 6.8: Detailed accuracy by class - SVM - small dataset - with adjectives

SVM increases the accuracy from 0.788 to 0.837 by the use of adjectives. Compared to Random Forest, Random Forest has the best accuracy with 0.866.

6.4 Results by Random Forest on the small + objective dataset

Table 6.9: Confusion matrices - Random Forest - Small + objective dataset

a	b	c	<- Classified as
209	118	13	a = pos
80	250	10	b = neg
0	0	680	c = obj

(a) without adjectives

a	b	c	<- Classified as
247	80	13	a = pos
42	285	13	b = neg
0	0	680	c = obj

(b) with adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.614	0.079	0.723	0.665	0.844	pos
	0.735	0.117	0.679	0.706	0.846	neg
	1.000	0.048	0.967	0.983	0.980	obj
Weighted Avg.	0.783	0.081	0.790	0.785	0.890	

Table 6.10: Detailed accuracy by class - RF - small + objective dataset - without adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.726	0.042	0.855	0.785	0.900	pos
	0.838	0.079	0.781	0.809	0.900	neg
	1.000	0.047	0.963	0.981	0.979	obj
Weighted Avg.	0.855	0.056	0.866	0.858	0.926	

Table 6.11: Detailed accuracy by class - RF - small + objective dataset - with adjectives

Also here we see that the adjectives improve the results, but the results are unrealistic high in both cases, because the objective texts were classified 100% correct.

6.5 Results by Support Vector Machines on the small + objective dataset

Table 6.12: Confusion matrices - SVM - Small + objective dataset

a	b	c	<- Classified as
263	73	4	a = pos
90	249	1	b = neg
1	0	679	c = obj

(a) without adjectives

a	b	c	<- Classified as
278	59	3	a = pos
58	281	1	b = neg
1	0	679	c = obj

(b) with adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.744	0.089	0.743	0.758	0.876	pos
	0.732	0.072	0.773	0.752	0.879	neg
	0.999	0.001	0.993	0.996	0.995	obj
Weighted Avg.	0.825	0.054	0.836	0.835	0.917	

Table 6.13: Detailed accuracy by class - SVM - small + objective dataset - without adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.818	0.058	0.825	0.821	0.911	pos
	0.826	0.058	0.826	0.826	0.913	neg
	0.999	0.001	0.994	0.996	0.996	obj
Weighted Avg.	0.881	0.039	0.882	0.881	0.940	

Table 6.14: Detailed accuracy by class - SVM - small + objective dataset - with adjectives

SVM did not classify the objective texts 100% correct, but so close that also here the results are unrelistic high.

6.6 Results by Random Forest on the large dataset

Table 6.15: Confusion matrices - Random Forest - Large dataset

a	b	<- Classified as
10 270	2 230	a = pos
2 253	10 247	b = neg

(a) without adjectives

a	b	<- Classified as
11 819	681	a = pos
787	11 713	b = neg

(b) with adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.822	0.180	0.820	0.821	0.821	pos
	0.820	0.178	0.821	0.821	0.821	neg
Weighted Avg.	0.821	0.179	0.821	0.821	0.821	

Table 6.16: Detailed accuracy by class - RF - large dataset - without adjectives

	TP Rate	**FP Rate**	**Precision**	**F_1**	**Accuracy**	**Class**
	0.946	0.063	0.938	0.942	0.941	pos
	0.937	0.054	0.945	0.941	0.941	neg
Weighted Avg.	0.942	0.059	0.942	0.942	0.941	

Table 6.17: Detailed accuracy by class - RF - large dataset - with adjectives

On the large dataset, the baseline Random Forest have a good accuracy of 0.821. When we include adjective, it increases to 0.941.

6.7 Results by Support Vector Machines on the large dataset

Table 6.18: Confusion matrices - SVM - Large dataset

a	b	<- Classified as
10 787	1 713	a = pos
1 854	10 646	b = neg

(a) without adjectives

a	b	<- Classified as
11 870	630	a = pos
683	11 817	b = neg

(b) with adjectives

	TP Rate	FP Rate	Precision	F_1	Accuracy	Class
	0.863	0.148	0.853	0.869	0.857	pos
	0.852	0.137	0.861	0.857	0.857	neg
Weighted Avg.	0.858	0.143	0.857	0.863	0.857	

Table 6.19: Detailed accuracy by class - SVM - large dataset - without adjectives

	TP Rate	FP Rate	Precision	F_1	Accuracy	Class
	0.950	0.055	0.946	0.948	0.947	pos
	0.945	0.050	0.949	0.947	0.947	neg
Weighted Avg.	0.948	0.053	0.948	0.948	0.947	

Table 6.20: Detailed accuracy by class - SVM - large dataset - with adjectives

Support Vector Machines beats Random Forest with an increase in accuracy from 0.857 to 0.947.

Chapter 7

Discussion

In this chapter we identify possible sources of errors. We presents a discussion of the results and a comparison between the classification methods. A comparison to other work on the same datasets are provided. The results are discussed in the view of the research questions defined in Section 1.2 and we conclude the discussion in a summary.

7.1 Sources of error

In Section 5.4 we described how we used SentiWordNet to find the sentiment for our synonyms and antonyms. This dictionary contain 117 569 words with a negative and/or a positive weight. Unfortunately, 117 569 words were not enough for our system to work perfectly. We experienced that many words got either a zero sentiment score or no score at all. We treated a zero score as an objective word, and a missing score as an unknown polarity. If we could find the sentiment of these adjectives, the results might differ from those shown in this thesis.

We did not detect or compensate for irony in the reviews. Analysing irony is a research area in itself, and is considered beyond the scope of this thesis. We did neither compensate for negation words, misspellings or abbreviations. This could, and most probably lead to abbreviations being interpreted as words, misspelled adjectives not being considered, and adjectives counted as the opposite sentiment because of a negation word. To include these factors would be a natural extension of this system.

As explained in Section 5.2 we had trouble finding suitable objective texts. We did not manage to find a set of objective texts that the classifiers did not classify perfectly or nearly perfect. This causes the results obtained by the evaluation metrics to be unrealistic high. In other words, the results are non conclusive and we will therefore not discuss them further.

For the other two datasets, when we only use two classes (positive and negative), we force the reviews to be classified as either positive or negative. By doing this, we can meet the problem of an overfitted model and become vulnerable to situations where coincidences makes a particular neutral word occurs more times in the positive or negative reviews. However, we will compare our results (without objective texts) with other researches that also uses only the positive and the negative class.

7.2 General discussion around the results

By observation, we see that adjectives included in the classification process yield better results. The question is whether this approach can yield even better results in general. The adjectives are chosen based on the POS-tagging from LingPipe. This makes a good foundation for the system, but some adjectives may not be found. This can be caused by for example misspellings. There is no guarantee that SentiWordNet assignes the correct sentimentscores to the adjectives. Sentiment are subjective and we believe that if a manually sentiment tagging of the words were performed by more than one person, we would possibly obtain different results. Therefore we believe that it might be impossible to achieve a 100% correct classification, because we cannot achieve a 100% correct classification of adjectives.

We have experimented with a small dataset containing 2000 reviews and a larger dataset containing 25 000 reviews. Both dataset are small considering this system could be used by large companies such as Netflix. We did not study how our approach would scale with real world datasets containing millions of reviews. Scalability is beyond the scope of this thesis. Still, if we look at the results from the datasets used, our approach could be considered feasible. However, the classification takes a long time to complete already at this size, especially Random Forest. Luckily for us, this is not a system that depends on a fast time to run, like for example classification of stock messages used to make money on the stock market. In that case you need to react before the stocks have changed.

Our system predicts sentiment with a good accuracy. As we can see in Table 7.1, the accuracy increased in all our experiments, by using the adjectives as an explicit feature.

Dataset	Algorithm	Accuracy without adjectives	Accuracy with adjectives
Small	RF	79.7%	86.6%
Small	SVM	78.8%	83.7%
Small + objective	RF	89.9%	92.6%
Small + objective	SVM	91.7%	94.0%
Large	RF	82.1%	94.1%
Large	SVM	85.7%	94.7%

Table 7.1: Accuracy in all our experiments

The best results on the small dataset have an accuracy of 0.866 (Random Forest). The best results on the small + objective dataset are obtained by Support Vector Machines and is equal to 0.940. Recall however that this result can be misleading due to the aforementioned issues with the objective texts. With the large dataset the best accuracy is as high as 0.947 obtained by Support Vector Machines, with Random Forest just behind with 0.941. This shows that our method is highly effective and that adjectives can be considered as important factors for sentiment analysis.

To be absolutely sure these results are not obtained by randomness and to prove that the results obtained actually is a legitimate improvement, we would have to perform a statistical significance test[1]. We believe that the differences in the results are so high, that such a test might not be necessary.

7.3 Comparison with other work

When we compare our results with other work, it is important that we compare them to someone that uses the same dataset. Because of this, we cannot compare the results

[1] You can learn more about a statistical significance test and see a tutorial in excel here: http://policeanalyst.com/performing-a-statistical-t-test-in-excel/

obtained from the experiments using the objective texts. This dataset is a randomly selected subset of the “CMU Movie Summary Corpus” and therefore nobody else would have the exact same set.

In [32], Pang et al. compares Naive Bayes, maximum entropy classification, and support vector machines. This is the first paper that introduces the small dataset used in this thesis. But instead of using the whole datset, they randomly select 700 positive and 700 negative documents. These selected documents are further split into three equal sized folds before classification. They achieved a best accuracy of 82.9% by the use of unigram, presence and Support Vector Machines. Unfortunately, their results are not directly comparable with ours, since they split their dataset and therefore did not use the exact same set as we did. However, this result is beaten by both the results in this thesis. We obtained a 86.6% accuracy by random forest and a 83.7% accuracy by support vector machines. Another research using the small dataset is [49] by Whitelaw et al. They obtained an accuracy of 90.1%. This result is obtained by combining two techniques, bag of words and “appraisal group by attitude & orientation”. Bag of words uses relative frequencies of all words in the text. “Appraisal group by attitude & orientation” means that the total frequency of the appraisal groups with each possible combination of attitude and orientation, is normalized by the total number of appraisal groups in the text. An appraisal group is phrases such as “very good” and “not extremely brilliant”. In other words the approach uses adjectives with a preceding list of modifiers to obtain better results. This is the best results we found that used this exact dataset.

The creators of the large dataset we use, Maas et al. presents their own results on both the small dataset and the large dataset in [7]. They presents a model that uses a mix of both unsupervised and supervised techniques to learn word vectors. This results in a system capturing semantic term-document information and rich sentiment content. On the small dataset they obtain a result of 88.90% accuracy, far better than our results. However on the large dataset, our method scored higher. We got a result of 94.1% accuracy by random forest and 94.7% accuracy by support vector machines. Both of our results, beats their accuracy score of 88.89%.

In conclusion, we obtained good results, but we did not obtain the best results on the small dataset. Compared to other work, we see that there exist several approaches that obtained better results. On the other hand, we obtained really good results on the large dataset. We improved the creators algorithm with over 4% in accuracy. With an best accuracy of 94.7% on the large dataset, our system is highly comparable to other methods.

7.4 Research questions revisited

This section will explain how this thesis answers the research questions, defined in Section 1.2.

RQ1 What recommender and sentiment analysis approaches exist today?

To answer RQ1, we have presented todays recommender systems in Section 2.2. This includes the most common approaches; collabortive filtering and content based filtering. In addition, we present the more complex approaches; hybrid methods and matrix factorization. Section 2.3 and 2.4 presents the functionality and challenges behind text mining and sentiment analysis. Chapter 3 presented real studies around sentiment analysis. This include work done in the area of product reviews, micro-blogging, longer text and in the stock market. With this, we have presented the existing techniques that tries to solve the recommendation and sentiment analysis problems.

RQ2 How can adjectives be exploited in the area of sentiment analysis?

To investigate this, we tested the idea that adjectives can be used as sentiment deciders. Adjectives are words that describe something and therefore also often contains a sentiment. In addition to our approach, we presented three techniques in Section 3.4 that also investigated the sentiment of adjectives; Hatzivassiloglou and McKeown [48] presented a technique for deciding the sentiment of adjectives. Whitelaw et al. [49] investigated the idea that appriasal groups of adjectives could improve sentiment analysis and Minqing Hu and Bing Liu [50] developed a system to summarize the opinions for product features of a product. They found frequent features that many people had expressed an opinion on and used adjectives to decide the semantic orientation of the features in each review.

RQ3 How can the use of adjectives improve results obtained by classical classification methods?

Concerning RQ3 we tested our system on two classification algorithms; Random Forest and Support Vector Machines. The results presented in Chapter 6 shows us that the count of positive and negative adjectives improves the baseline algorithms in all our experiments. From the previous section, we see that our results obtained by the small dataset are not comparable to the best algorithms using the same dataset. However our method shows to be highly accurate on the large dataset, compared to others.

7.5 Summary

We can see that our system performs very well. We feel that adjectives as sentiment deciders are a promising technique, but before we can use it in a large-scale real application, we would need to test its scalability. Based on the experiments in this thesis, we see that our system performs better on the large dataset, than the state-of-the-art approach. However the size of that dataset is still too small compared to those being used in real-life applications. The time consumed on classification is already high and it would be many times higher on such a dataset.

Chapter 8

Conclusions

This chapter conclude the work presented in this thesis, including an evaluation of whether the main goal of this thesis presented in Chapter 1 is achieved. Lastly, we will present ideas we think would be an interesting continuation of this project.

8.1 Conclusions

In this thesis we investigated whether it was possible to improve sentiment analysis techniques on user-based reviews, by the use of adjectives.

We developed a system that located all the adjectives in each review. We found the sentiment of the adjectives synonyms and antonyms using a dictionary of positive and negative english words. The sentiment of the adjective were further calculated based on the majority of the sentiment scores of the synonyms together with the opposite majority sentiment scores of the antonyms. The positive adjectives and the negative adjectives were counted for each review and we used these numbers as explicit features in the classification process.

Compared to the baseline algorithms, we increased performance in all our experiments. However, the highest accuracy obtained from the small set 79.7% by Random Forest, showed not to be competitive against similar solutions. Our results on the large set showed to be much more promising. Compared to the state-of-the-art approach on the same dataset by Maas et al. [7] our best accuracy of 94.7% (by Support Vector Machines) showed to increase their results by more than 4%.

We believe that research within sentiment analysis and recommender systems will become more important each day, as the user-based reviews increase in numbers on the Internet. The system presented in this thesis, is a valuable contribution to the area of research and it forms a solid basis for future research on both sentiment analysis and adjectives as sentiment deciders.

8.2 Future work

We experienced trouble including objective text in the dataset. The objective set did not function as noise in the data, as we intended it to. It would be powerful to prove that our system could differentiate objective text from sentiment text. Therefore, we think that this would be a natural extension of the corpus. To test such a hypothesis, we would use a two-step classification. First, we decide if the text contain sentiment or not. Then we classify the sentiment text into whether it is positive or negative.

User-based reviews are informal text. This means that it is highly possible it contains misspellings, slang, abbreviations and/or emojis. Misspelled adjectives are not detected by our part-of-speech tagger and if we could detect those adjectives, the result would might look different. The system is nor trained on slang, abbreviations or emojis. It would be interesting to see if such a system could improve the results. In 2015 Kralj Novak et al. created their own emoji sentiment lexicon [64] and the detection of emojis is already popular within sentiment analysis of the the micro-blogging area.

This thesis shows that adjectives are an important feature to consider when classifying sentiment text. However, compared to real-world applications, the datasets we experiented with are small. It would be interesting to see how adjectives could affect results, when applied to datasets containing millions of reviews. We could further detect what the adjective described (positively or negatively) to create a feature specific summary containing sentiment scores, based on the user-reviews. The movies could in addition to be rated as a whole, be rated on music, actors/actresses, plot, and so on. Making such a system would make the system able to make a strong personalized recommendation for the user, based on the users most important features of a movie.

Appendix A

Part-Of-Speech Tags

Tag	**Description**
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass

Continued on next page

TableA.1 – *Continued from previous page*

Tag	Description
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun

Continued on next page

TableA.1 – *Continued from previous page*

Tag	Description
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Table A.1: Alphabetical list of part-of-speech tags

Bibliography

[1] A. Dragland, "Big data – for better or worse," *SINTEF*, vol. 22, May 2013.

[2] P. Melville and V. Sindhwani, "Recommender systems," in *Encyclopedia of Machine Learning*, pp. 829–838, 2010.

[3] J. A. Horrigan, "Online shopping. pew internet & american life project report," 2008.

[4] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, pp. 13:1–13:19, Dec. 2015.

[5] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the ACL*, 2004.

[6] D. Bamman, B. O'Connor, and N. A. Smith, "Learning latent personas of film characters.," in *ACL (1)*, pp. 352–361, The Association for Computer Linguistics, 2013.

[7] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[8] K. Balog, N. Takhirov, H. Ramampiaro, and K. Norvaag, "Multi-step classification approaches to cumulative citation recommendation," in *Open research Areas in Information Retrieval (OAIR 2013)*, 2013.

[9] M. Ruocco and H. Ramampiaro, "Geo-temporal distribution of tag terms for

event-related image retrieval," *Information Processing & Management (To Appear)*, vol. 51, no. 1, pp. 92–110, 2015.

[10] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, pp. 61–70, Dec. 1992.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, Aug. 2009.

[12] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, (San Francisco, CA, USA), pp. 43–52, Morgan Kaufmann Publishers Inc., 1998.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, (New York, NY, USA), pp. 285–295, ACM, 2001.

[14] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, Jan. 2003.

[15] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Eighteenth National Conference on Artificial Intelligence*, (Menlo Park, CA, USA), pp. 187–192, American Association for Artificial Intelligence, 2002.

[16] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 40, pp. 66–72, Mar. 1997.

[17] A. Gunawardana and C. Meek, "A unified approach to building hybrid recommender systems," in *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, (New York, NY, USA), pp. 117–124, ACM, 2009.

[18] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, pp. 331–370, Nov. 2002.

[19] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User Modeling and User-Adapted Interaction*, vol. 10, pp. 147–180, Feb. 2000.

[20] P. Ott, "Incremental matrix factorization for collaborative filtering."

[21] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996.

[22] M. Rajman, R. BESANÇON, and R. Besancon, “Text mining: Natural language techniques and text mining applications,” in *In Proceedings of the 7 th IFIP Working Conference on Database Semantics (DS-7). Chapam*, pp. 7–10, Hall, 1997.

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[24] S. Raschka, “Naive bayes and text classification I - introduction and theory,” *CoRR*, vol. abs/1410.5329, 2014.

[25] P. Domingos and M. Pazzani, “On the optimality of the simple bayesian classifier under zero-one loss,” *Mach. Learn.*, vol. 29, pp. 103–130, Nov. 1997.

[26] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[27] P. Bhargavi and S. Jyothi, “Applying naive bayes data mining technique for classification of agricultural land soils,” *International journal of computer science and network security*, vol. 9, no. 8, pp. 117–122, 2009.

[28] R. Feldman and J. Sanger, *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York, NY, USA: Cambridge University Press, 2006.

[29] T. Du and V. K. Shanker, “Deep learning for natural language processing,”

[30] G. Chakraborty, M. Pagolu, and S. Garla, *Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS*. Cary, NC, USA: SAS Institute Inc., 2013.

[31] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Found. Trends Inf. Retr.*, vol. 2, pp. 1–135, Jan. 2008.

[32] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: Sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP ’02, (Stroudsburg, PA, USA), pp. 79–86, Association for Computational Linguistics, 2002.

[33] R. Mihalcea, C. Banea, and J. Wiebe, “Learning multilingual subjective language via cross-lingual projections,” in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[34] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *Proceedings of the Fourteenth International Conference on*

Machine Learning, ICML '97, (San Francisco, CA, USA), pp. 412–420, Morgan Kaufmann Publishers Inc., 1997.

[35] H. Yu and V. Hatzivassiloglou, "Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 129–136, Association for Computational Linguistics, 2003.

[36] N. Jindal and B. Liu, "Mining comparative sentences and relations," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pp. 1331–1336, AAAI Press, 2006.

[37] L. Cabral and A. Hortacsu, "The dynamics of seller reputation: Theory and evidence from ebay," Working Paper 10363, National Bureau of Economic Research, March 2004.

[38] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, "Learning subjective language," *Comput. Linguist.*, vol. 30, pp. 277–308, Sept. 2004.

[39] S. Das and M. Chen, "Yahoo! for amazon: Extracting market sentiment from stock message boards," in *In Asia Pacific Finance Association Annual Conf. (APFA)*, 2001.

[40] E. Filatova, "Irony and sarcasm: Corpus generation and analysis using crowdsourcing," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012*, pp. 392–398, 2012.

[41] P. Carvalho, L. Sarmento, M. J. Silva, and E. de Oliveira, "Clues for detecting irony in user-generated contents: Oh...!! it's "so easy" ;-)," in *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, TSA '09, (New York, NY, USA), pp. 53–56, ACM, 2009.

[42] C. Chiu, "Towards a hypermedia-enabled and web-based data analysis framework," *J. Information Science*, vol. 30, no. 1, pp. 60–72, 2004.

[43] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, (New York, NY, USA), pp. 519–528, ACM, 2003.

[44] T. T. Thet, J.-C. Na, and C. S. Khoo, "Aspect-based sentiment analysis of movie reviews on discussion boards," *J. Inf. Sci.*, vol. 36, pp. 823–848, Dec. 2010.

[45] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining,” in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)* (N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, eds.), (Valletta, Malta), European Language Resources Association (ELRA), may 2010.

[46] N. Godbole, M. Srinivasaiah, and S. Skiena, “Large-scale sentiment analysis for news and blogs,” in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.

[47] L. Lloyd, D. Kechagias, and S. Skiena, “Lydia: A system for large-scale news analysis,” in *Proceedings of String Processing and Information Retrieval (SPIRE)*, no. 3772 in Lecture Notes in Computer Science, pp. 161–166, 2005.

[48] V. Hatzivassiloglou and K. R. McKeown, “Predicting the semantic orientation of adjectives,” in *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, pp. 174–181, 1997.

[49] C. Whitelaw, N. Garg, and S. Argamon, “Using appraisal groups for sentiment analysis,” in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pp. 625–631, ACM, 2005.

[50] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’04, (New York, NY, USA), pp. 168–177, ACM, 2004.

[51] S. K. Gulbrandsen, “Improving News Traders using CRF: Using Conditional Random Fields to reduce Feature Space,” Master’s thesis, NTNU, Norway, 2013.

[52] M. I. Jordan and R. Thibaux, “The kernel trick,” *Lecture Notes*, 2013.

[53] J. R. Quinlan, “Bagging, boosting, and c4.s,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1*, AAAI’96, pp. 725–730, AAAI Press, 1996.

[54] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[55] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 ed., 2008.

[56] M. Koppel and J. Schler, “The importance of neutral examples for learning sentiment,” in *Workshop on the Analysis of Informal and Formal Information Exchange During Negotiations (FINEXIN)*, 2005.

[57] B. Santorini, "Part-Of-Speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing)," tech. rep., Department of Linguistics, University of Pennsylvania, Philadelphia, PA, USA, 1990.

[58] J. Ramos, "Using tf-idf to determine word relevance in document queries," 1999.

[59] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[60] D. Harman, "How effective is suffixing," *Journal of the American Society for Information Science*, vol. 42, pp. 7–15, 1991.

[61] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, Nov. 1995.

[62] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: an on-line lexical database," *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, 1990.

[63] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov 2009.

[64] P. Kralj Novak, J. Smailović, B. Sluban, and I. Mozetič, "Sentiment of emojis," *PLoS ONE*, vol. 10, no. 12, p. e0144296, 2015.