Polarity Classification of Reviews by Expanding a Preliminary Lexicon

by

Ali Yadollahi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Polarity classification in text is the problem of automatically detecting the general opinion of textual data. Analyzing the general opinion toward a topic of interest is important for different audiences, such as companies, politicians or even regular users. On the other hand, the availability of opinionated text data has been rapidly increasing in WWW during the past decade. However, most of the available text data is not provided with a polarity label and hence one may not be able to learn from it. Addressing this challenge shapes the main motivation of the research in this field of study.

Previous works in polarity classification suffer from some drawbacks. As the major drawback, most of the research in the literature naively assumes the availability of the labelled data and takes the advantage of supervising their method by the given ground truth. Furthermore, they do not generally perform well in all domains of textual data, since they are specifically trained over one domain. Although there exists a limited number of works proposing an adaptable classification method, their performance is not promising on the unlabelled data.

This thesis tackles the problem of multi-domain unsupervised polarity classification. We propose a hybrid polarity classification technique, which is the combination of a lexical and a learning classification. We also propose a lexicon expansion method, which uses the hybrid classification along with the association rule mining at the sentence level. Our experiments on various datasets show that our hybrid classification method outperforms the previous state-of-the-art unsupervised works. Furthermore, our expansion method is shown to increase the performance of the initial lexicon.

Acknowledgements

I would like to thank my supervisor, Professor Osmar Zaiane for his invaluable guidance, encouragement and patience throughout this thesis. I cannot imagine this project done without his incredible support.

I would also like to thank my parents, who are always supportive of me and make my life as easy as possible. Lastly, I thank my beloved wife, who I can always count on.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Gathering and analyzing opinions toward a topic or entity is a crucial and fundamental task every politician, famous company or even individual person is concerned with. There are many large companies and firms that collect and analyze the reviews and feedback from users. This is also a common interest among many politicians and administrators, who want to know more about opinions and feelings of the general public toward them or their organization. Furthermore, from an individual point of view, realizing about the opinions of other people psychologically helps the instigator to grasp knowledge about an entity and to feel more confident about the future decisions one wants to make.

On the other hand, growth of available data in the World Wide Web has assisted researchers to gain more information about people's opinions regarding various targets of interest. This has made the opportunity for researchers to study people's attitudes about products, companies, politicians, etc. These information are often represented textually in different formats such as reviews of products, opinions inside a weblog, posts in a forum, tweets as short messages and comments in social media such as Facebook, Youtube and Google plus. In general, any type of text, which expresses an opinion (either positive or negative) toward a subject such as an event, issue, service, or any other interest, is called "opinionated text". Figure 1.1 is an example of an opinionated text about a laptop extracted from the "Amazon" online shopping centre.

The positivity/negativity degree of a text unit is called "polarity". In the sample opinion shown in Figure 1.1, one can see the star-based polarity given to the review. In spite of the availability of a huge number of opinions in the WWW, most of them are not provided with a manually labelled polarity, such as posts inside a forum. Hence, devising an automatic labelling method to find the general polarity of a set of textual opinions is fundamental.

Problem Definition: Given an opinionated document d evaluating an entity e, deter-

Figure 1.1: Review of a laptop product from the online shopping centre called "Amazon"

mine the opinion orientation *oo* of *d* on *e* [1].

In order to come up with an automatic solution for the labelling task, diverse techniques from different fields of computing science have been applied. Since the labeller deals with the text, "Natural Language Processing" (NLP) techniques can be used to extract information out of text. Furthermore, since the task in general is a classification task, "Machine Learning" ideas and methods would be of great importance.

Finding the general polarity of a text unit has been done in different granularities. Analysis of polarity can be done at the document level, sentence level or phrase level. Furthermore, polarity can be categorized in binary classes of positive/negative or in ternary classes of positive/neutral/negative or it can be considered as a regression problem in which the task is to assign a rate (for example from one star to five stars) to a text.

This work is set up under the set of assumptions as below:

1. Text units are documents. We create our classifier over documents, instead of other text units such as sentences, since they represent the general opinion of the users, while sentences would express users' opinions toward a specific target.

2. The classification task is binary, i.e. documents are considered to be either positive or negative (due to lack of available data with the additional label "neutral").

3. We assume that we do not have the Labels of the documents in order to simulate the unsupervised environment. However, the datasets that we use in this work are provided with the labels, which we use for evaluating our methods.

4. We study the performance of the classification over a mixture of reviews subject to various types of entities, gathered from different online media. Our motivation to work on such a mixed data is that we desire a generic and robust polarity classifier, which is adaptable to new domains of discussions. Furthermore, since reviews are

meant to bear a general opinion toward a subject, they become an interesting type of opinionated text to be investigated.

1.1 Thesis questions

This research mainly attempts to approach the problem of polarity classification of texts. We manage to run experiments and observations with the purpose of answering the following questions:

1. Would it be possible to build a polarity classifier with a high level of performance, without having any knowledge on the polarity of text documents?
2. Can we build a robust polarity classifier, adaptable to be applied in multiple domains of text?
3. What are the discriminative words inside a corpus of text documents, which would determine the polarity of the text? Can we expand an initialized lexicon by extracting those words from an unlabelled corpus?
4. Can we apply users' knowledge in our classification system to enhance its quality?

1.2 Thesis contribution

In this work, we contribute to the problem of polarity classification through various dimensions:

1. A lexicon expansion method is proposed, which is capable of mining the informative and dominant rules from a set of multi-domain unlabelled text data. This expansion technique is also able to apply feedback at different levels of text, including sentence and word to modify its behaviour. The lexicon resulted from the expansion method is shown to improve the initial lexicon.
2. A hybrid method is proposed for the scoring of documents and sentences, which is a combination of a lexical and a machine learning scoring method. This method is shown to outperform the state-of-the-art unsupervised methods.
3. A new feature selection method is proposed, which is capable of ranking the features without having any labels from the instances. The resulted features are shown to

represent more accurately the polarity of the documents and to boost performance of the applied learning methods.

4. Based on the techniques proposed in this work, a sentiment analysis tool has been implemented and embedded in a community mining package called “Meerkat”. This tool provides polarity classification and polarity visualization of documents, words and sentences. Furthermore, it enables the user to provide feedback about the polarity of text units and exploit this feedback to improve performance.

1.3 Thesis organization

The content of this manuscript is structured as following: Chapter 2 introduces the previous methods and techniques in polarity classification; Chapter 3 discusses the techniques we have used for preprocessing; in Chapter 4, the scoring methods we have applied in our system are examined; Chapter 5 describes our proposed lexicon expansion method; Chapter 6 discusses the feedback system and finally Chapter 7 is dedicated to the conclusion of our work and the future research directions.

Chapter 2

Background and Related Work

2.1 Polarity Classification Methodologies

Polarity classification is the task of classifying the opinion of a given text as falling under one of two opposing sentiment polarities, the most famous of which are "like" vs "dislike" [1]. Although much of the work in this area has been done on products and services reviews which mostly hold positive or negative opinions, there are other problems where "like" or "dislike" are interpreted as other concepts such as different political views [1].

Automatic classification of polarity can be categorized from different aspects. Different media can be used to express opinions, among which we only focus on text. For more information about other types of polarity classification, one can refer to [2].

Polarity classification in a text can be done either at the document or sentence level. At the document level, the whole document, whether short or long, is the atomic unit of input to the problem and what matters is the polarity of the whole document, while at the sentence level, each sentence is expected to have a polarity independently. As noted in [3], a challenge in labelling at the sentence level is the influence of the surrounding context on the sentence. For example, depending in what context it is used, the sentence "*I can't give a better score to this product.*" can have different polarities. For instance, if there are other sentences before this sentence stating the positive features of the product, the sentence is a complement. But, if it is followed by a set of drawbacks, it would be a negative sentence.

With respect to the nature of the data, there are two important modes of the problem. Some datasets benefit from being annotated by a human, while there are many unlabelled datasets of reviews and posts. Methods working with labelled data often show better results, nevertheless they require manual labelling which is an expensive process in terms of both time and money. In the following two subsections, we discuss previous methods on annotated and unannotated text data, respectively.

2.1.1 Works on annotated data

The algorithms that deal with labelled data are called "supervised methods". Supervised methods apply some machine learning algorithms on a set of training data, in order to be able to predict the label of unseen test data. They need an annotated dataset of texts for the task of training, which creates a model to discriminate between polarities.

In order to apply machine learning methods, one should represent the text by means of descriptive features. After that, some techniques should be used to train a polarity classifier. Most solutions introduced in the literature are general purpose machine learning techniques, while some of them are sentiment specific. Sebastiani was the first to apply general text categorization algorithms on the field of sentiment detection [4]. Later, Pang et al. [5] compared performance of SVM and Naïve Bayes for classifying polarity of the movie reviews.

A large part of the literature is dedicated to finding out usefulness of many features and techniques in learning. Most common types of those features, which have been also applied in other areas of text mining, are as follows:

Presence-based and frequency-based features

The most common way to describe a piece of text is by using a binary vector, in which each element corresponds to one term from a dictionary. The element at index i in the vector is set to 1 if the term i is present in the text and is 0 otherwise. Likewise, one may describe the text as a vector representing the number of times individual terms have been repeated. The former is called the presence-based and the latter is named the frequency-based type of features. Although term frequency is a popular feature in information retrieval, Pang et al. [5] obtain better performance when using presence-based features.

Unigram and n-grams features

A unigram refers to one single word in a text and an n-gram represents a group of adjacent words in a sentence, preserving the order. For instance, in a sentence such as "I ate a delicious apple", delicious would be a unigram and "delicious apple" would be an n-gram, more exactly a bigram in this case. Although n-grams have more information than unigram features, concerning the position of words in the sentence and being used as a group, their effectiveness in increasing the performance is a matter of some debate. For instance, Pang et al. [5] report that unigrams are more effective than n-grams; however, some other research

such as the work of Dave et al. [6] indicate better results for the combination of bigrams and trigrams.

Part of speech

A part of speech is any grammatical category, which represents syntactic functions of a word in a sentence. For example, nouns, verbs and adjectives are three most common parts of speech. Some types of words are more probable to carry information about the polarity of a sentence or document, hence part of speech can be a crucial discriminator to detect such words. It is indicated in previous works that adjectives are very important in determining sense of the text. In fact, adjectives can be used both as main features such as in works like [7] and [8] and as filters for selecting other features. For instance, Turney uses adjectives to detect a set of phrases as features and then determines the polarity of documents based on those features [9].

In addition to adjectives, other part of speech tags such as nouns like "gem" or verbs such as "love" can improve performance of the task [1]. Some previous works focus on comparison of adjectives, adverbs, verbs and nouns like [10], [11], and [12].

Syntax

Several researchers investigate usage of word dependency-based features by using dependency trees [13]. There are contradicting results regarding the effectiveness of dependencies in text in previous works. Slight improvements in performance are reported in [6] and [14], while Ng et al. [15] conclude that addition of dependency-based features does not offer any improvements over the simple n-gram-based classifier.

Negation

The use of negating words in a sentence may totally flip the polarity of that sentence. For instance, ignoring "not" in "He does not like the color blue" results in a false positive. Attaching "not" to the words occurring near the negating words is one of the elementary techniques done for the first time by Das et al. [16]. Although the naïve assumption that each negation word flips the polarity of a window of following words is working in many cases, it is not a general rule. Later works try to optimize this technique by reversing the phrases based on the part of speech tag patterns [17].

Besides the explicit negation words, there are other terms which may negate a sentence. For instance the verb "prevent" in the sentence "They prevent keeping unhealthy foods in the

store" and the verb "deny" in "She denies admiring the brand" are implicitly reversing the polarity.

Topic-oriented features

Sentiment of a given sentence may be topic specific. For instance, the word "fast" in the context of car reviews is considered as positive, while it may be considered as negative in movie reviews. Different features have been investigated based on topic in the literature, especially in the work of Mullen et al. [7].

2.1.2 Works on unannotated data

It is obvious that providing a solution for unannotated data is always harder because of the lack of labels comparing to annotated ones. In fact, most of the informative and also subjective text formats, such as comments, reviews and news, are left unlabelled and hence it is useless using them for the purpose of training a classifier. Also, annotating the data is very expensive in terms of time, money, and human effort.

Researchers try to tackle the problem of unlabelled data from a wide range of perspectives. We have categorized the related methods in three different groups: the first group belongs to the solutions whose aim is to expand a lexicon of words which contains words and their prior polarity, so called the "Lexicon expansion" methods; the second group is concerned with "domain adaptation". Most of the works on unannotated data would fall in one of these two categories. However, there are also other types of works in this scope, all of which are described in the third category referred to as "other methods".

Lexicon expansion

A very basic and simple idea to build a classifier for unannotated data is to use a lexicon of words. A lexicon is a dictionary of words, each associated with a score showing its degree of polarity. While classifying the test instances, polarity scores of each word contained in the test sample are fetched and processed for the sake of predicting the polarity of the whole text. Processing of these scores could be done in different ways, including summing up, taking the average, etc. This generic solution is called "lexical based" method. Currently existing lexicons can be used for this purpose; however, to have higher performance, one may need to create his/her own lexicon of words suitable for the domain of data. Since manually building a lexicon is a tedious and time consuming task, automatic solutions, called "lexicon expansion" methods are suggested. Researchers apply different methods

for automatic creation of the lexicon from the information lying in the data. The first works belong to Hatzivassiloglou et al. [18], Pang et al. [5], and also Yu et al. [19]. They approached the problem by making simple assumptions about the occurrences of words. For instance, Pang et al. [5] assumed that words present near the word "excellent" could be counted as positive while words adjacent to the word "poor" can be negative.

Further attempts to create a useful lexicon were concerned with clustering of words or phrases in sentiment clusters including [20], [21], [22], [23], [24], [25], and [26]. One of the valuable attempts in this set of works was the work of Hatzivassiloglou et al. [18]. They created a lexicon by using "opposition constraints" such as "but" and "and" between pairs of words and separating the words into two categories with opposite polarities. After finding these two groups of opposing words, in order to assign sentiment orientation (or degree of polarity) to each category, different techniques have been proposed. For instance, Hatzivassiloglou et al. [18] simply assumed that the words with more frequency seem to be positive. For instance, their method takes the sentence "I think this book is well-written but controversial", puts "well-written" on one category and "controversial" on the other one. Then by using the frequencies of the words in each of those categories, they find the general polarity of each category and assign it to all of the words inside it. The larger category is simply and naively assigned a positive polarity.

Another technique, popular in previous works, is to have a set of "seed" words with known polarity and then to assign the polarity of new words with respect to their relationship to the seed words, i.e. polarity of new words are assigned by propagating the polarity of seed words to other words such as [20], [27], [21], [22], and [28]. For instance, they take a dictionary and find the synonyms and antonyms of the seed words and assign the same polarities and the opposite polarities of the seed words to those related words respectively. Note that most of the mentioned methods try to find a "prior polarity" of words. "Prior polarity" of a word is the polarity that it invokes no matter what context that word is occurring in while "contextual polarity" is the polarity of the word with respect to the context. For instance, since the word "security" bears a positive polarity in general, we can assume a positive prior polarity for it. However, if it occurs inside a sentence like "There are three living former Secretaries of Homeland Security.", it does not infer any positive or negative polarity since it is part of a referring expression. Therefore, it has a neutral "contextual polarity". Prior polarity should be further applied to determine the "contextual polarity" of the words with respect to the concept and domain such as in the work of Wilson et al. [29].

Domain adaptation

One idea to produce a generic classification method which is adaptable to any kind of data on any domain and extremely useful for unannotated data, is training a classifier over a labelled dataset from one domain or topic, called the "source" and use it to label the unlabelled data from another domain, called "target". For instance, if one has the polarity labels of the instances in a dataset in the domain "movies" and wants to do the classification in the domain "cars", the source domain would be the movie data and the target would be that of the cars. However, the results of such a process in various domains have been shown to be unsatisfactory in [30]. This is expected since the keywords and phrases used in one domain may differ totally from the keywords in another one. Furthermore, one word in a domain may bear a different sentiment from what it does in another domain. For instance, sophisticated may be interpreted positively in domain of the movie, while it might bear a negative meaning in domain of the cars. Therefore, adapting the classifier trained over the source to be useful for the target is an essential step. This procedure is called "domain adaptation".

According to [31], domain adaptation is done in two distinct ways, namely "labelling adaptation" and "instance adaptation". In labelling adaptation, the labelling function is adapted since some features (words in opinion mining) may differ in polarity between source and target domains. In instance adaptation, the probability function of features is adjusted, for instance the changes of word frequency from one domain to another one are modelled.

Early attempts to approach the problem relates to the work of Aue et al. [32], which evaluates the performance of four rudimentary approaches to somehow adapt a classifier to be useful for the target domain. Those approaches include: training over all possible domains, limiting features to those observed in the target domain, ensemble of classifiers, and using a small set of labelled in-domain data.

Further simplistic attempts are the work of Yang et al. [33], which ranked features of the two labelled datasets by running logistic regression over the sentences and selecting the highly ranked features as the ones that are most common in all domains.

Label transferring is another method used in some of the previous works for domain adaptation. The basic idea of label transferring is to find the most informative samples of the target domain by means of a classifier that is trained over the source domain and then label those informative instances to train a brand new classifier over the instances.

The first work that exploited this idea belongs to [34]. Later, the same team improved the performance of their system with selecting "generalizable features" by means of a measure they named "Frequently Co-occurring Entropy" in [35]. This measure includes two criteria which are frequently occurring in both of the domains and also having almost the same probability in the two domains. Recently, Li et al. [36] applied the same idea by finding the most informative instances in the target domain using classifiers with a query by committee (QBC) strategy.

A very common technique, used in different schemes in previous works, is to categorize the features in every domain into two groups. The first group belongs to features, which happen frequently regardless of the domain, called "domain-independent" features. The second group, called "domain-specific" features, are common just inside their belonging domain. For instance, assume that we have two sets of reviews belonging to the domains "food" and "movie". We can categorize all of the words of these two datasets into three different categories. One of them would be the set of the words specific to the domain "movie" such as the words "actor", "screen" and "perform". The second category would be the set of the words specific to the domain "foods" such as "pot", "spicy" and "bake" and the last category would be the set of the words in common between the two domains, which would occur independently of the domain of the text such as "good", "bad", etc. The reason of applying such a categorization over the features is to align domain-specific features of the source domain to those of the target domain and then adapt the trained classifier in the source domain.

Based on the explanation above two steps should be followed:

Clustering features: There are methods suggested to distinguish the two types of features. The idea to recognize domain-independent features is to find features which occur more than a threshold in any domain. To find domain-specific ones, the degree of dependency of each feature to each domain should be calculated. In information theory this can be done by using "mutual information" between the feature and domain.

Alignment: Alignment is a step in which each domain-specific feature in the target domain is mapped to one or more domain-specific features in the source domain. In the literature this is done in various ways. In the first attempts, Blitzer et al. [30] approached the problem by using an algorithm, called "structural correspondence learning (SCL)". SCL tries to find the domain-independent features (pivot features) as most frequent ones and finds the correspondence model between pivot features and all other features by training linear pivot trainers.

Li et al. [37] try to approach the alignment problem by using non-negative matrix tri-factorization of term-document matrix. Basically, they factorize the term-document matrix in the source domain and then by means of a matrix (which expresses if each of the words is occurred in both domains or not) they estimated the factors of the term-document matrix in the target domain.

Pan et al. [38] aimed to cluster domain-specific words in both domains by means of a spectral feature alignment (SFA) algorithm. This work is promising to exploit all the relationships between the domain-specific and domain-independent words in spite of SCL. Basically, they create a bipartite graph of features that consists of two clusters of domain specific and domain independent features. Then if there exists two domain specific features from two domains that have many common domain-independent features they align them to be in correspondence to each other.

In addition to clustering-alignment method, there exists another group of solutions to the problem of adaptation which is based on feature selection in both of the source and target domains. This approach tries to find a feature space in which the gap between distribution of source and target domains is minimum, comparing to other spaces. Features of both domains are transferred to this new feature space and then a classifier over the source domain in the new feature set is trained. This classifier can be guaranteed to be working with higher performance over the target domain.

Other methods

Some other methods appropriate for unannotated data include but are not limited to:

Bootstrapping: The general idea is to use an initial pre-trained classifier on another dataset to label the target dataset and then using this newly labelled dataset to train a new classifier. Kaji et al. [39] used this method to label a set of HTML documents with the positive/negative polarities.

Belief network modelling: One of the recent usage of belief networks is on training a model for the task of sentiment classification. Lin et al. [40] add a layer of sentiment to the structure of a famous probabilistic document model called "Latent Dirichlet Allocation (LDA)" [41] and created a new model called "joint sentiment topic" (JST) to find the polarity of the words inside a set of documents with respect to each topic. The same group conducted a comparative study of Bayesian models for unsupervised sentiment analysis in [42]. They also examined the performance of the JST model over different domains of datasets in the same work.

Geometric structure: There exists a set of works, which attempt to find the polarity of the data points (either documents, sentences or words) by using the assumption that the points sufficiently close to each other prefer to have the same polarities. The works of Zhou et al. [43] and also that of Cai et al. [44] are examples of this category.

Combining lexical and machine learning methods: Lexical and learning methods can be combined to compensate the disadvantages and drawbacks of each other. In order to optimize the performance of an initially trained classifier (over a different domain), Qiu et al. [45] use a lexicon-based classifier in which in each step firstly the lexical classifier labels the data and then the learning classifier is trained over the labelled dataset. Operations continue until results of the two datasets have the minimum distance. In another work, Prabowo et al. [46] try to build a semi-supervised hybrid classifier by using both rule-based classifiers and SVM classification.

It is worth to note that there exists another group of works, which exploit a small set of labelled data to train a classifier with a low level of accuracy. Then, by using this classifier to label the unlabelled data points, they attempt to expand the training data. One of the recent works in this category belongs to the work of Dasgupta et al. [47], which uses spectral clustering to find the unambiguous reviews in a corpus and then to apply those reviews to classify the ambiguous ones.

Some of the previous works employ other types of information of the corpus, besides their lingual content, to enhance their classification. For instance, Hu et al. [48] use emoticons to find the sentiment of a given comment in social media. Strong analytical references exist for a broader study of all the previous methods and their details in the area of the polarity classification, such as the survey published by Liu et al. [49], the book [50] published by the same author and the surveys [51] and [52].

2.2 Polarity Related Resources

Building lexicons and collecting datasets is hard and time consuming. This is why most researchers prefer to take advantage of current resources. Here we introduce some of the most well-known lexicons and datasets for polarity mining. Note that getting to know the process for the creation of these resources helps if one desires to build his/her own lexicon or dataset.

2.2.1 Lexicons

There are many publicly available lexicons which are results of lexicon creation and expansion of the previous sentiment analysis works. Among these lexicons the following ones are known to be the most frequently used and effective in the literature (A summary for the following lexicons can be seen in Table 2.1):

Harvard General Inquirer

This lexicon [1] is the result of one of the first attempts [53] in compiling a list of words for sentiment analysis. The lexicon contains syntactic, semantic and pragmatic information of its words. Among the information provided for each word, the one that is of our interest is "positive" and "negative". The lexicon includes $11,790$ words. The score of each word in this lexicon would be 1, 0 or -1 respectively meaning that the word is positive, neutral or negative.

Opinion Lexicon

This lexicon [2] which is an outcome of Bing Liu's research in sentiment analysis [54, 55] consists of $6,786$ words, among which $2,009$ of them are positive and the rest are negative. The corpus from which they have extracted the words, includes customers' opinions about various features of products. They have extracted the words of their lexicon by finding sentences which include a frequent feature and pulling adjectives out of those sentences. Afterwards they have separated those extracted words into two clusters of positive and negative ones based on the score of their synonyms and antonyms using a dictionary of words. The score of each word is defined in a similar way to the scoring of the Harvard General Inquirer lexicon.

MPQA

"MPQA" lexicon [3] [29] consists of $8,222$ words, each of which is provided with a set of information including how much subjective the word is, how strong its subjectivity is, prior polarity of the word which can be positive, negative, both or neutral, and whether the word is stemmed or not.

This lexicon is built on a subjectivity lexicon that resulted from the works of the same team [56]. In the first step, annotators are given a set of instructions and also an annotating

[1] http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm
[2] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon
[3] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

scheme to annotate phrases and words as to be either positive, negative, both or neutral. In the second step, they measured agreement between annotations of two annotators to evaluate the lexicon. Based on their annotation scheme, annotators' decisions depend mostly on the emotion of the sentences inside their corpus.

WPARD

Using an online form, [57] collected information from 342 undergraduate students. Participants were asked to rate how negative or positive were the emotions they associate with each word, using a scale from -6 (very negative) to $+6$ (very positive). They built the lexicon Wisconsin Perceptual Attribute Rating Database (WPARD) [4] from this data such that each word has a corresponding polarity and a real number showing the strength of that polarity.

SentiWordNet 3.0

"SentiWordNet 3.0" [5] is a lexical resource provided by [58]. This lexicon is built on top of its previous version, SentiWordNet 1.0. It contains $155,287$ words and is providing them with a polarity degree. In a nutshell, their method to create SentiWordNet 3.0 consists of 5 steps including starting from a seed set of positive and negative words and applying synonyms and antonyms to expand the lexicon, adding objective words as a new cluster, training a community of ternary classifiers for the glosses of the words, classifying clusters of words with the classifiers and finally running a random walk on the graph of words to make their scores converge to a final state. In this lexicon each word is provided with a decimal signed polarity score.

NRC

Since 2009, S. M. Mohammad has compiled several word-sentiment lexicons [6] from different corpora including Twitter and customer reviews of Yelp and Amazon. In some cases, labelling is done manually and in other cases it is done automatically, such as using hashtag of a tweet as its label. For more elaboration on each of them, one can refer to [59, 60, 61, 62]

[4] http://www.neuro.mcw.edu/ratings/

[5] http://sentiwordnet.isti.cnr.it/download.php

[6] http://saifmohammad.com/WebPages/lexicons.html

Table 2.1: Summary of polarity related lexicons

Name	Author	Year	Size (words)	Set of polarities
Harvard General Inquirer	P. Stone	1968	11,790	positive, neutral and negative
Opinion Lexicon	B. Liu	2005	6,786	positive, negative
MPQA	T. Wilson	2005	8,222	positive, neutral, negative and both
WPARD	D. A. Medler	2005	1,400	positive, negative
SentiWordNet 3.0	S. Baccianella	2010	155,287	degree of polarity
NRC	S. M. Mohammad	2009+	various sizes	positive, negative

2.2.2 Datasets

Compared to other areas of text categorization, polarity classification benefits from a larger number of well-annotated datasets. Because of this, here we only point to the benchmark datasets which have been very commonly exploited in the literature for experiments.

Amazon

"Amazon" [7] is the result of the work of Blitzer et al. [30]. It is a dataset of product reviews constructed from the website of an online shopping center with the same name. It includes 4 different domains of product reviews, including DVDs, books, electronics and kitchen items, each of which has $2,000$ reviews. The reviews of each domain are half positive and half negative making this dataset balanced. Each instance in this dataset includes detailed information of a review consisting of its rating which is from 0 to 5 stars, review title and date, and the review content. They have crawled the data from the Amazon website, annotated reviews with the rating of greater than 3 stars as to be positive, and those with rating less than 3 stars as to be negative and discarded the ones with 3 stars since those reviews are more likely to have ambiguous sentiments. In addition to the labelled data, this dataset includes $3,685$ unlabelled instances in the DVD domain and $5,945$ unlabelled instances of kitchen reviews. This part of the dataset is created by selecting an equal number of positive and negative reviews from a set of labelled data and discarding the labels.

[7] `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

Movie datasets

Various versions of datasets [8] have been extracted from the movie reviews of famous online movie databases all of which are built by Pang et al. Here is a summary of the details for each of these versions:

- Pool of HTML files: This data consists of $27,886$ HTML files which are unprocessed and unlabelled. Files consist of reviews crawled from an online database called "Internet Movie Database (IMDB)". This version is the raw version of the next labelled one (Polarity dataset).

- Polarity dataset: This version of the data includes four different minor versions. The minor versions 0.9 and 1 [5] consist of 700 positive and 700 negative processed reviews, version 1.1 is slightly modified by removing a few non-English/incomplete reviews and correcting some mislabelled reviews. Finally, the last minor version 2 consists of $1,000$ reviews for each class of polarities [63]. Since not all the reviews in the raw version have the same format of rating, labelling them is done differently based on the format of rating. First of all, only those reviews whose author has explicitly declared the rating for them are classified. With a five-star system, reviews with 3.5 stars and up are labelled as positive and reviews below or equal to 2.5 are counted as negatives. With a four-star system, reviews higher or equal to 3 stars are labelled as positive and the ones with 1.5 stars or lower are labelled as negative. Finally, with a letter grade system, B or above is considered positive and C or below is considered negative.

- Sentence polarity dataset: This version of the data includes $5,331$ positive and $5,331$ negative processed sentences and snippets provided by Pang et al. [64]. All of the instances are downloaded from a movie review database called "rottentomatoes" which classifies reviews either as fresh (meaning positive) or rotten (meaning negative).

Blogs

This dataset which is provided by Melville et al. [65] includes two different sets of blog posts, one of which is concerned with technology blogs and the other one is related to political blogs. The first set, named "lotus blogs", is a set of posts corresponding IBM Lotus collaborative software gathered from 14 blogs, 4 of which have posted mostly negative

[8] http://www.cs.cornell.edu/people/pabo/movie-review-data/

Table 2.2: Summary of polarity related datasets

Name	Author	Year	Size	Type of Data
Movie	B. Pang	2004	29,419	processed labelled IMDB movie reviews (document-level)
		2004	2,000	raw unlabelled IMDB movie reviews (document-level)
		2005	10,662	processed rottentomatoes movie reviews (sentence-level)
Amazon	J. Blitzer	2007	8,000	reviews of products
Blogs	P. Melville	2009	252	product review and political posts

comments about the product and the others have provided positive posts. The data have been provided by downloading either the latest posts of each blogger's RSS feed or the archived posts of that blog. Afterwards, they extracted text from those parts of the HTML files in which the ratio of tags to words is above a minimal threshold. Then all the posts were read and labelled manually to either positive, negative, neutral or irrelevant. There exist 34 positive and 111 negative instances in this set.

The second part of this dataset consists of political posts regarding two candidates of the United States presidential election in 2008 namely "Barak Obama" and "Hillary Clinton". The posts were taken from a set of $16,741$ blogs, filtered based on whether they have the words "Obama" and "Clinton" and randomly selected for the manual labelling. Based on [65], labelling political posts is much more difficult than that of product reviews because posts are more emotional, mostly mention implicit comments and judgements about the candidate and may apply cultural references to make a point. Therefore, they have labelled those posts which have explicitly mentioned an opinion about one of the two candidates as positive or negative. Hence there are no neutral or irrelevant posts in this set. The "Politic" dataset includes 49 positive and 58 negative posts.

Chapter 3

Processing the Data

3.1 Data

Since the main purpose of this work is to have a sentiment analysis tool on different domains, the data that we used includes five different domains. Four of the domains are provided from the "Amazon" website, which (as discussed in Section 2.2.2) are reviews of some products encompassing four types of $\{$"$books$", "$dvds$", "$electronics$", "$kitchen$"$\}$. The main purpose of publishing these reviews by Blitzer et al. [30] was to evaluate their proposed method, which transfers knowledge from a labelled source domain to an unlabelled target domain. We add the domain of "movie" as the fifth one to their dataset by combining the IMDB movie dataset with their data.

The purpose of using two sets of data from two different websites is to have a mixture of reviews, which include various types and formats of the language. The audience of Amazon website consists of regular people who use simple and easy-to-understand words and formats of writings to write a review about a product that they have purchased, had or seen. On the other hand, reviews of the IMDB dataset are written by different types of writers varying from normal users to experts and critics. They contain a wide range of writings, from simple formats to sophisticated words and grammatical structures. This yields a valuable set of data to evaluate our work and to check whether it can boost the performance, regardless of the constraints the test data might have.

In order to simulate the aforementioned challenges in our dataset, we select all of the domains of the Amazon dataset as the development set and apply the IMDB movie review dataset as the test set. We use the development set to refine any parameter, or to decide among any techniques, which we may have in our method. After setting all the parameters and finalizing the classification model, we classify the reviews in the movie dataset by means of our model and get the accuracy as the performance measure of our method.

3.2 Preprocessing

In general, for every classification problem before tackling the main subtasks, such as feature extraction, feature selection or learning, one needs to process the raw input data to provide a set of meaningful and informative instances. Preprocessing the data is a crucial step, since real world raw input includes different types of noise and irregularities, which need to be removed before the data can be used in one's application.

Ever since textual information are one of the main components of real world data, they suffer from noise, variations and abnormalities. The format of the text, the language in which the text is written in, the level of formality of the language, symbols, abbreviations and misspellings are some of the many parameters which create many possible formats and types of writings. This pushes the necessity of preprocessing the text input, before applying it in the next steps. In this chapter, we explain the preprocessing techniques and steps involved in our project. Two of the main steps in preprocessing (including tokenizing, part of speech tagging), we used a powerful tool named "Stanford NLP Toolbox", which is a collection of different natural language processing tools, developed in java, by Manning et al. [66] at Stanford university.

3.2.1 Tokenizing

Text consists of different units and segments. A text document can include multiple sentences. Also, every sentence can be interpreted as a sequence of words. In order to analyze a document, collecting information with respect to all the units is necessary. Segmenting a text document into sentences and words is called "tokenizing". There are two different kinds of tokenization applied in our work: sentence tokenization and word tokenization.

Sentence and word Tokenizing

We used one of the tokenizers implemented inside the Stanford NLP toolbox called "PTBtokenizer", to segment each review to its corresponding sentences and then to tokenize each sentence to its words. "This tokenizer mimics the behaviour of Penn Treebank 3 (PTB) tokenization" according to documentation of the toolbox. It is a fast, deterministic rule-based tokenizer, which also benefits from a set of heuristics to decide when a symbol (like a period) represents the end of a sentence and when it does not.

Although, PTBtokenizer tokenizes simple sentences with a high degree of accuracy, it lacks the ability to segment different clauses of the complex sentences. For instance, if we

give the below review to the PTB tokenizer as the input, it will provide us with the two mentioned output sentences. This can lead to a major drawback in our work which we will discuss later in detail. Also, as can be seen in this example, punctuation marks such as dot or question mark are also considered as tokens.

- **Input (Review):** This is a good movie. Although the movie was too lengthy it had a very novel and new point of view toward life.
- **Output (Sentences):**
  1. [This, is, a, good, movie, .]
  2. [Although, the, movie, was, too, lengthy, it, had, a, very, novel, and, new, point, of, view, toward, life, .]

3.2.2 Part of Speech Tagging

After segmenting the document to sentences and words, we need to know the role of each word within its sentence. The reason to perform such a process is that there are many words in English, which have the same exact writing but they bear different meanings according to their role in the sentence. For instance, a word like "will" occurs with two different meanings according to online dictionary "Merriam-Webster" [67] which are mentioned below. The first one is completely neutral as it does not indicate any feelings, emotions or opinions. However, the second one can show a strong desire toward something which bears a considerable degree of subjectivity.

- *will(auxiliary verb)*: used to express futurity.
- *will(noun)*: a strong desire or determination to do something

"In grammar, a part of speech (POS) is a linguistic category of words (or more precisely lexical items), which is generally defined by the syntactic or morphological behaviour of the lexical item in question" [68]. For instance, verbs, nouns and adjectives are the most common parts of speech in every language.

In our work we tag every word resulting from the tokenization step by its part of speech. The method we used belongs to the Stanford NLP tool called "MaxentTagger". This class includes two different models of POS tagging: the first model uses a bidirectional dependency network while the second one only applies left second-order sequence information (left 3 words). Although the accuracy for the second model is a little bit lower than the first

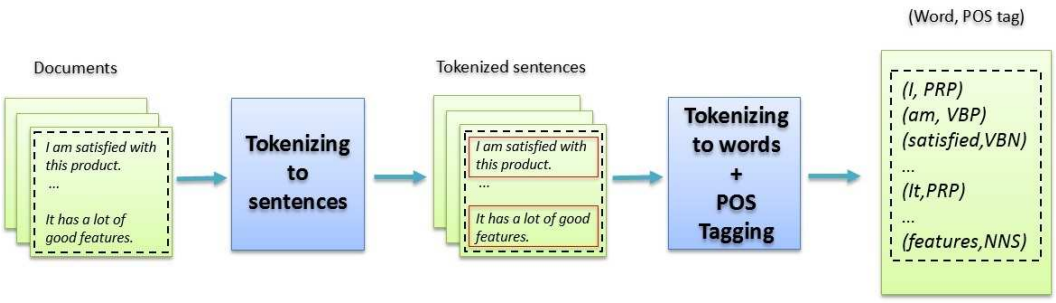

Figure 3.1: Symbolic procedure of tokenizing an input document review to its sentences and words

one (96.92% versus 97.32%) it runs much faster and is recommended for general use. [69] points to some of the example of the failures for these tagging methods.

Figure 3.1 illustrates the symbolic procedure of tokenizing and POS tagging of a document to words with a simple example. In this figure, results of word tokenization and POS tagging steps are shown together for better illustration though the word tokenization is done in the same step as the sentence tokenization.

3.2.3 Word cleaning and Preprocessing

Main Steps

1. **Lower case conversion:** In order to look up a word in a lexicon or to create an inverted index of words (we will talk about this later), we need to use comparison between words. Therefore, all the letters should be converted to lower case so that different models of writing the same word would fall into the same category. For instance, two words such as “Hello” and “hello” could be converted to the same shape as “hello”.

2. **Omitting meaningless words:** Different deformations of words can be found in various types of writing such as spelling the words differently, using combination of numbers or punctuation marks with letters or separating words with underscore instead of spaces or tabs. These changes in the shape of words can make the analysis harder since they are linguistically meaningless and cannot be understood by lingual analysis tools. Therefore, we filter all the words with characters other than letters to discard these deformations and to make the analysis simpler.

3. **Stemming:** Stemming is an NLP technique which attempts to find the root or base form of any inflected or derived word. Applying this procedure over the words enables us to treat all the words with the similar base form in the same way. For instance, the word "liked" has the same polarity of its root form, which is "like". We use the morphology package from Stanford NLP toolbox for this step. This package finds the root form of only inflected forms (not derivational). In other words, it only stems plural nouns, verb endings and the pronouns.

4. **Comparative and superlative adjective conversion:** Since in the stemming step we do not stem the simple form of adjectives and also because there is no certain rule in the English grammar for stemming adjectives, we undertake a simple procedure to estimate the root form of some adjectives.

   The comparative and superlative forms of a simple adjective can be made either regularly or irregularly. In the regular forms, either one of the two postfixes of "er" and "est" is added to the end of the word or the keywords "more" or "most" is used before the word to express the type of adjective. In this case, the ending letter of the main root of the word may or may not change. For instance, comparative adjective of the word "neat" is "neater", in which the simple form is not changed, while the root form of the word "easy" is changed in its comparative form "easier".

   For some adjectives, the comparative and superlative forms are totally different from the root form. For instance, adjectives "good" and "bad" have the comparative forms of "better" and "worse" respectively. In order to find the root form of an adjective from its comparative form we run the procedure shown in Algorithm 1:

   In the pseudo code below, $exception$ is a list of exceptional comparative adjectives which are shown in Table 3.1. The base form of the adjective and its comparative and superlative forms can be found in this table. Also, in order to perform a sanity check we look up the resulting base forms in a dictionary to see if they are valid English words. If the procedure is not able to find a base form of an adjective, it will return the input as the output without any changes.

   The procedure of converting superlative adjectives to their base form is very similar to the above procedure with only two slight changes. Firstly, we add "est" or "iest" postfixes to the end of the base form of an adjective (instead of "er" and "ier") to convert it to its superlative form. Furthermore, since the constructing postfixes of the superlative forms have one more letter compared to that of the comparative ones,

Algorithm 1 Comparative to base conversion

```
1:  function ComparativeToBinary(comparative)
2:      exception                          ▷ list of all irregular comparative adjectives
3:      base                               ▷ list of the base form for all exceptional adjectives
4:      if comparative ∈ exception then
5:          index = search(exception, comparative)
6:          return base[index]
7:      else
8:          if comparative ends with "er" then
9:              result = Substring(comparative, 0, length(comparative) − 2)
10:             if result is in dictionary then
11:                 return result
12:             end if
13:         else if comparative ends with "ier" then
14:             result = Substring(comparative, 0, length(comparative) − 3) + 'y'
15:             if result is in dictionary then
16:                 return result
17:             end if
18:         end if
19:     end if
20:     return comparative
21: end function
```

we need to deduct the ending indices in lines 9 and 14 of Algorithm 1 by one, so that it finds the proper base form from the comparative form. For instance, if we pass "cleverer" to the comparative conversion procedure it finds "er" at the end of the word and returns the substring of the word starting from the index 0 and ending at the index $n-2$, in which n is the number of letters of the word. However, if we pass "cleverest" to the superlative conversion procedure it will find "est" at the end of the word and returns the substring starting from index 0 and ending at index $n-3$.

It is worth mentioning that the aforementioned procedures are not working for all the cases since we just consider most frequent patterns. For instance, we do not handle cases such as the word "nicer", which is converted to "nic" when we omit "er" from its end and it is not a meaningful, valid word.

Note that we use these two procedures for those words whose part of speech indicate the type of adjective. Then we replace the word with the output of the functions.

5. **Stop word omission:** In the English language, there exists a set of words which although occur in the text very frequently, they do not bear significantly any valuable information. Articles such as "a", "an" and "the" or auxiliary verbs like "be" and

Base form	Comparative form	Superlative form
good	better	best
bad	worse	worst
far	farther	farthest
far	further	furthest
late	later	latest
late	latter	last
little	less	least
many	more	most
old	older	oldest
old	elder	eldest

Table 3.1: Adjectives with exceptional superlatives and comparatives

"would" are examples of these words so called as "stop words". The set of the stop words used in a text processing problem can be defined differently depending on the task and the information it requires. For instance, the word "good" can be considered as a stop word for the task of "topic mining" , while it carries crucial information for sentiment analysis. Stop words act as noise in decision making since they are very common and may decrease the impact of other more important words, which have less frequency. Hence we need to omit the stop words in order to lower the noise in the system. We use the list of "stop words" from one of the "machine learning" packages written in java environment, called "Weka".

Other Steps

So far in this chapter, we have discussed about the main preprocessing steps that are applied in our work. However, by taking a look at the results of the preprocessing steps, we find many other insufficient and deformed words which need to be processed mostly because of the following reasons:

- **Misspellings:** There exists a considerable number of misspellings and dropped letters from words in the corpus, which cannot be corrected by any of the mentioned preprocessing steps. Therefore, the essence of an automatic correction step is sensed.

  The general method applied for the auto-correction task, is exploiting a dictionary of words as the ground truth and finding the nearest word to the misspelled one by means of a distance measure (such as the Euclidean distance). For instance, if the word "cure" is incorrectly written as "curr", by finding the nearest word in the dictionary to "curr" we may be able to correct this word. However, based on the used dictionary

and the distance measure, the output is not always guaranteed to be correct. For instance, in the mentioned example, if the word "curl" exists in the dictionary it may be a better candidate for us, which is not the right word. Furthermore, depending on the number of words in the dictionary this task may be too much time consuming since for every word in the corpus, the auto-correction method needs to search over many words to find the correct one. Therefore, we decide not to run such a procedure.

- **compound words:** During the next steps such as sentence scoring, we need to look up the words in a lexicon of words. While looking up words, one may not find the exact word in the lexicon; though the base form of the word may exist. As an example, consider the word "hopelessly". If the lexicon does not include this word, by looking up "hopelessly", we cannot find the polarity of the word. This word is a combination of three parts, called "lemmas". In this example, the lemmas are "hope", "less" and "ly". The first lemma is a meaningful word in a lexicon with positive polarity. The second one is a suffix which converts a noun to an adjective form with an opposite polarity and the last lemma converts an adjective to an adverb.

  The process of decomposing a word into its lemmas is called "lemmatization". For each word that is not found in the lexicon we may be able to find the polarity of the whole word by lemmatizing it, finding the polarity of the base lemma and applying the flipping effect of possible prefixes and the suffixes. However, sometimes the base form of a word does not exist in the lexicon but a subset of its adjacent lemmas may shape a word existing in that lexicon. For instance, in the word "usefulness", "use" and "ful" consist a word, which exists in the lexicon and bears a positive polarity. But the word "use" by itself may not exist inside that lexicon.

  Lemmatization needs a broad knowledge of lemmas which is not publicly available. Furthermore, even by having the lemmas of a word, considering different combinations of adjacent lemmas to look up a part of the word in the lexicon takes a long time and is beyond the scope of this research.

As in each of the above cases the reason not to apply such a method was mentioned, we do not use these steps in our models despite of the potential advantages they might have.

3.2.4 Inverted Indexing

Given a document and a word, if we need to find the number of occurrences of the word inside the document, we need to search the word inside that document and find out the an-

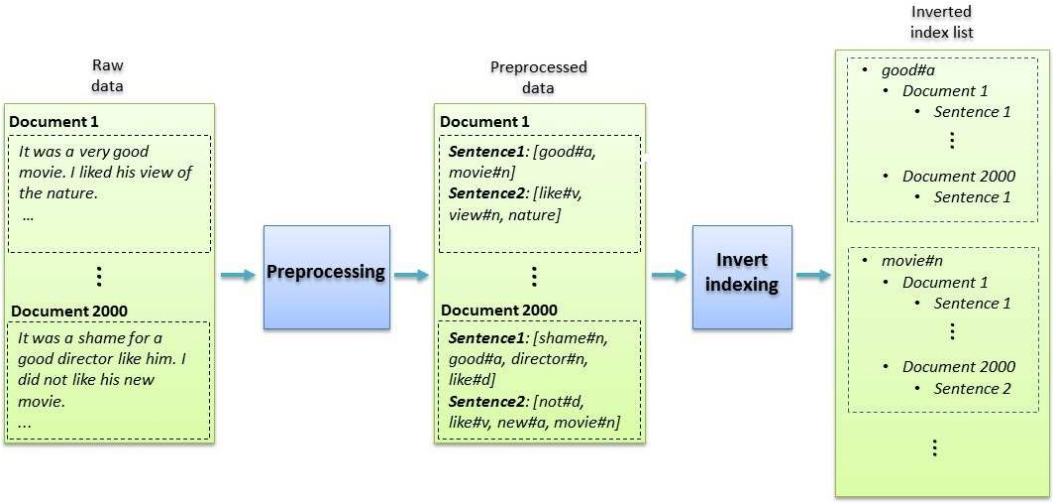

Figure 3.2: Preprocessing and inverted indexing of the words

swer. However, naively searching the document whenever a query arrives would be highly time consuming if the number of queries increases. For each query, the running time of this search would be $O(N)$, in which N is the number of words in the given document. Since the number of these queries is significantly large in our application, we need to exploit a data structure, which decreases the running time of response for queries. Furthermore, since we also need to have the sentences of the given document, in which the given word occurred in, that data structure must also encapsulate information about those sentences. To solve this problem we use the "inverted index" technique:

Inverted index: "In computer science, an inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents." [70]

The way we use inverted indexing is by mapping each word to the list of all the documents it has occurred in. Furthermore, inside this list, we map each document to the list of all of its sentences, in which the given word has occurred. Then, by means of this data structure, we can find the answer in $O(1)$ for each query (regardless of the number of words of the document).

Figure 3.2 summarizes all the preprocessing steps we discussed in this chapter. The first step shows the result of applying all of the techniques except inverted indexing and the second step shows the result of the inverted index. After the data is preprocessed, each word is tokenized, simplified to its most possible base form and the result and its part of speech tag are attached together with a # sign in between. The tag, which is added to the

end of the word, is not the main tag determined by Stanford POS MaxentTagger. This tag is determined as below:

- Simple, comparative and superlative forms of adjectives and adverbs are given the general tag of “a”
- Singular and plural nouns and proper nouns are given the general tag of “n”.
- Base form, past tense, present participle, past participle and singular present forms of the verbs are given the general tag of “v”.
- Other parts of speech not included above are given the tag “d”.

For instance, the output of the preprocessing step for an adjective such as “Good” is “good# a”. Furthermore, as can be seen in Figure 3.2 the inverted index of documents and sentences, in which each word is occurred, is mentioned in the inverted index list.

Chapter 4

Scoring Methods

After preprocessing the text and creating all the necessary units for fast and optimized computations, we need to invest on designing and building a **scoring unit**. The scoring unit is a module which gets a preprocessed text unit as the input and outputs a number as a measure of how much that text unit is positive or negative. As we discussed in the "background" chapter, there are two general types of techniques for creating a scoring unit. The first type, called "lexical method", uses a lexicon of words while the other category of techniques, named "learning methods", trains a classifier by means of machine learning approaches. Both types of methods have some advantages and disadvantages.

Lexical methods need a strong lexicon to have a reasonable performance. If we use general lexicons which are not specific to the domains of our datasets, we may face poor performance, since information included for the words in those lexicons may be generally true but not specifically correct for our domains. For instance, the word "fast" may be interpreted positively in general but it may be negative in the domain of "teaching". On the other hand, manually creating a lexicon, which includes polarity of the words in our corpus, requires a considerable amount of effort and cost. However, lexical methods are easy to use, relatively fast and not computationally intense. Furthermore, if we need to provide the user with the clues of why we pick a label for a certain sentence (or document), we can easily highlight words of the lexicon which exist in that sentence (or document) and give the user the total score. Ultimately, although machine learning methods perform better compared to lexical methods, they need a set of training sentences (or documents), which should be large enough to provide a classifier with a decent discriminative knowledge.

It is important to note that we can use these two types of scoring methods in any level of granularity of text. In other words, both of them could be applied for scoring either phrases, sentences or documents. In this chapter, we proceed to explain how to apply these

methods at the document level, since our task is to label document reviews. However, later in the next chapters we will apply them for scoring sentences. The following three sections explain respectively the lexical methods, learning methods and a hybrid method combining the two.

4.1 Lexical method

Assuming that we can infer the polarity score of every word from a lexicon, we apply the following process to a text unit to assign a polarity score to it:

1. **Preprocessing:** As explained in Chapter 3, we tokenize each document to sentences and then tokenize each sentence to its constituent words. Next, we provide all the words with their POS tags and apply the preprocessing filters to get to the base and clean forms of the words.

2. **Word look up:** We look up each preprocessed word in the lexicon and assign a score to that word based on the information provided in the lexicon. If the word does not exist in the lexicon we assign 0 to it. The scoring function differs among the lexicons, since the information each of them provides, varies. In Section 4.1.1, we will explain the scoring function for each of the lexicons on which we run our experiments.

3. **Negation:** We make a simple assumption for applying the effect of occurring negation words in the polarity score of the text unit. Having a list of negation words, we look up all the occurrences of the negation words in the text unit. Then for each of the occurred negation words, we flip the polarity score of all the words which fall into the adjacency interval of the negation word.

4. **Summation:** We assign the score for each text unit as the sum of all the scores of all of its words. The resulting score would be our measure of its polarity degree. The sign of the score shows the polarity of the text unit and its absolute value represents the strength of that polarity.

5. **Normalization:** In order to normalize the polarity score for the given text unit based on the number of its words, we calculate the mean score as:

$$S_{u_i} = \sum_{j=1}^{N} \frac{S_{w_j}}{N}. \tag{4.1}$$

in which S_{w_j} is the score for the word w_j and N is the number of the preprocessed words resulted from the text unit.

Table 4.1: Evaluation of the lexicons based on the accuracy measure

	Books	**Dvd**	**Electronics**	**Kitchen**	**Average**
Opinion Lexicon	71.70	73.55	76.00	74.95	**74.05 ± 1.86**
MPQA	69.45	70.45	72.15	73.85	**71.50 ± 1.93**
General inquirer	67.4	66.55	67.20	69.7	67.71 ± 1.37
SentiWordNet	62.40	62.25	64.05	66.35	63.76 ± 1.90

4.1.1 Comparison between lexicons

We have introduced some of the most famous and publicly available lexicons in Section 2.2.1. We select four of those lexicons: Harvard General Inquirer, Opinion Lexicon, MPQA and SentiWordNet 3.0, to apply the lexical method and to investigate their performance over the polarity classification task. The word scoring function described in step 2 of the lexical method, i.e. word look up, should be defined for each of these lexicons. In Harvard General Inquirer and Opinion Lexicon, the score of each word would be 1, 0 or -1 respectively if that word is positive, neutral or negative. In SentiWordNet 3.0, all the words are provided with a decimal signed polarity score. However, there is no explicit score assigned to each word in MPQA lexicon. According to the information provided in this lexicon, we calculate the score of each word as to be $S_w = S * P$ in which S is 2 if the word is strongly subjective or 1 if it is weakly subjective. Also, P is $\{-1, 0, 1\}$ if the word is $\{negative, neutral, positive\}$ accordingly. Therefore, each word has a score between -2 and 2.

Table 4.1 shows the summary of the results of the aforementioned experiment. Accuracy of classification, using every lexicon over each of the domains in our dataset, is mentioned in its corresponding cell in the table. As can be observed from the results, "Opinion Lexicon" and "MPQA" have the best performances compared to the other lexicons. Interpretation of this observation could be that, both of these two lexicons have a manually annotated source of data engaged in their creation. "MPQA" has been annotated by human annotators. Moreover, "Opinion Lexicon" is built automatically on top of a dataset of reviews, labelled manually by the costumers. Therefore, it also has a set of manually labelled data engaged in its generation. In another sense, the method used in building the "Opinion Lexicon" is similar to what we will explain about our method in Chapter 5. These are the motivations for us to use these two lexicons for the next steps.

4.2 Learning method

Assuming that we have labels for the data, one idea to build a scoring module is to find out the correlation between text units and their label. Assume that we can represent a dataset of text units as a matrix of numbers called X, in which each row represents an instance of the whole data and each column represents a numerical measure of an instance. Furthermore, assume that we have the labels of the whole dataset in another numerical column matrix, called Y. If we can find a function F, such that $F(X) = Y$, whenever any unlabelled text unit arrives, we can represent that instance as a row matrix of numbers T and assign $F(T)$ as the score of that unit. Finding the function F of raw text reviews to their labels is the main purpose of "machine learning" techniques. Applying a machine learning technique to create a scoring module requires one to take multiple steps as follows:

1. **Feature extraction:** In order to represent a text unit in the form of a matrix, we need numerical metrics which can represent the information in the text. These metrics are called "features" in machine learning.

2. **Feature selection:** If the number of features, which are extracted in the previous step, is large, then we should select the best features and discard useless ones to decrease computational cost of dealing with a huge feature space. In Section 4.2.1 we define some of the quality measures for the features. Furthermore, we introduce some types of dimensionality reduction and feature selection methods that we applied in the process.

3. **Learning:** The process of training the classifier on the labelled data to find the correlation between text units and their labels is called "learning". In Section 4.2.3 we talk about two different types of learning techniques, which are very effective in text processing.

4. **Classification and score assignment:** After running the learning step, we have a trained classifier which is capable of getting a numerical representation of an instance as input and returning the predicted class of that instance. Depending on the learning technique, the classifier returns also meta information for each instance, such as the likelihood of the assigned label to the instance, distance from a hyperplane that separates two classes etc. By having this information, we can assign a score to each instance as the polarity strength of that instance. Since this depends on the learn-

ing method, we will explain the score assignment for each method separately in its corresponding section.

4.2.1 Feature extraction

The feature space is a crucial matter in the concept of learning. Features based on which the classifier is trained, play a very effective role in making a discriminative classifier. An informative and high quality set of features would include the following characteristics:

1. **Frequent features:** If the features are not likely to be happening in instances, they do not bear much information about them. Therefore, frequency of the features is one of the main criteria.

2. **Discriminative features:** If the features are equally distributed among various classes and do not differentiate among them very well, they do not help making a proper decision. Therefore, they should be discriminative enough to make any classification task effective.

3. **Independent features:** If distribution of occurrence in one feature obeys that of another feature in some way, those two features are dependent on each other. If two features in a feature set are dependent to each other, their information would provide for the task of classification is similar. Therefore, keeping both in the feature set does not add value. Thus, extracting independent features would increase efficiency of the learning process.

In text classification problems, the common feature sets used to train a classifier are constructed from different tokens such as unigrams, bigrams and trigrams. It is proven empirically that unigrams provide more independent features, since each word of the text appears only in one feature instead of multiple features. The common raw feature sets (without using dimensionality reduction techniques) for textual instances are presence-based and frequency-based features, explained in Chapter 2. From the experience of previous works, we realize that frequency-based features do not bear much more useful information than presence-based features. This is due to the fact that, in the context of sentiment analysis, there are many sentimental words which occur rarely in the text, while there are also many neutral words which are frequent in the text. Therefore, frequency of a feature does not make it more important than the other features.

4.2.2 Feature selection

Based on what we discussed about criteria of a high-quality feature set, for our work we decide to have a presence-based feature set of unigram tokens. To find frequent features for our feature set we must engage frequency of the words in the selection process. However, naively selecting words with high frequencies in all the documents could result in picking less informative words such as frequent neutral words. Therefore, we need to have a better measure of the frequency of each word in the dataset. This measure must favour the words which are regularly frequent while penalizing words with too much frequency. In text mining and information retrieval, a measure called "tf-idf" is defined to have the desired behaviour.

Term Frequency Inverse Document Frequency (tf-idf): " is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus." [71]

Based on the definition, tf-idf increases with proportion to the number of times a term occurs in a document. Furthermore it decreases, if the total number of occurrences of that term in the whole corpus is getting higher. This measure represents a proper intuition of the frequency of a word.

There exist various versions of calculation for the tf-idf. We pick a normalized version of tf-idf in order to prevent bias toward the longer documents, with the formulation as follows:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{max\left\{f(w,d) : w \in d\right\}} \tag{4.2}$$

$$idf(t,D) = log\frac{N}{|\left\{d \in D : t \in d\right\}|} \tag{4.3}$$

$$tfidf = tf(t,d) \times idf(t,D) \tag{4.4}$$

in which $tf(t,d)$ is the augmented frequency of the term t inside the document d, $f(t,d)$ is the raw frequency of a term inside a document; $idf(t,D)$ is the measure of how many times the term t is repeated in all documents (represented with D) and N is the total number of documents in the corpus. As can be seen, the augmented frequency is calculated by finding the ratio of the frequency of the term in a document against the frequency of the most frequent item in that document.

We use tf-idf in order to rank the tokens as features to figure out how frequent and also informative they are compared to each other. After ranking them, we pick the top P_t percent of the tokens (where P_t is a parameter) and construct a feature vector based on them.

Although $tfidf$ represents a useful measure of frequency in text for each token, it may skip a set of informative words which are not commonly used in the text and yet are considerable in changing the total polarity of the text unit. For instance, a word like "gracious" bears a very positive meaning, though it is rarely used in a corpus with informal language.

4.2.3 Learning

Many learning methods have been tried in the literature such as SVM, Naïve Bayes, logistic regression and maximum entropy, each with different setups and parameters. However, according to the results of previous works, SVM and Naïve Bayes are shown to have better performances in text categorization. Here, we discuss the details of these two learning methods and explain how we calculate the score of each instance based on the output of their classifier:

Naïve Bayes

Naïve Bayes is a machine learning algorithm from the family of Bayesian network classifiers. It takes the word "Bayes" from the fact that Bayes theorem is applied when constructing its probabilistic model and takes the word "Naïve" because it naively assumes that, given the label of the instance as the condition, all features are statistically independent from each other. Suppose X is a document consisting of m n-grams $x_1, ..., x_m$, then:

$$y = \underset{k \in \{pos,neg\}}{argmax}\ p(C_k) \prod_{i=1}^{m} p(x_i|C_k) \tag{4.5}$$

where C_k can be either the positive class or the negative class and y is the predicted label for X.

SVM

Support Vector Machine (SVM) [72] is a two class classification method, which is commonly used in different applications of machine learning. The main idea behind SVM is to separate two different classes of instances by a hyperplane, given a non-linear set of vectors as input. The main criterion for SVM is to maximize the "margin" of hyperplane. The margin is defined as the distance of the plane from the nearest data point of any class. This is due to the fact that, by having a greater margin for the plane we can decrease the generalization error. However, it is not always possible to find a hyperplane which completely separates instances of the two classes in the feature space. There is another version of SVM so called "soft margin" which also allows for instance the number of mislabelled examples

as a measure which should be minimized while still keeping the margin maximum. Thus, the soft margin will treat the classification task as a min-max problem, which can be solved by linear programming. Although SVM is mainly designed as a linear classification module, by using different kernels and applying the "kernel trick" one can investigate non-linear behaviours with SVM.

In order to customize SVM for polarity classification, after running experiments on different kernels, we decide to apply SVM with "radial basis function" as the kernel. In order to assign a score to a testing text unit, we need SVM to predict how likely the text unit belongs to each class. This can be predicted by fitting a "logistic regression" learner on the output of SVM and by using the learner to calculate the distribution of the text unit over the two classes. Thereafter by having the distribution we simply set the score as:

$$S_w = \begin{cases} P(t, positive) & \text{if } P(t, positive) \geq P(t, negative) \\ -P(t, negative) & otherwise \end{cases} \tag{4.6}$$

in which $P(t, positive)$ and $P(t, negative)$ are the likelihoods of the instance t belonging to each class respectively. These two values are the results of SVM and running logistic regression.

4.2.4 Experiments

We classify the polarity of the documents by means of different combinations of the feature sets and the two types of learning methods, explained above, to find a high-performance pair of feature selection and learning method. We use presence-based features with two different sets of words. The first one is the set of all the words existing in the MPQA lexicon and the second set includes the top $4,000$ words of the documents ranked based on tf-idf (which is referred to as TFIDF4000 set of features in the results). The reason for choosing the words included in the MPQA lexicon is that, it is a manually annotated dataset. Hence, the certainty of selecting informative words as the features increases.

Table 4.2 shows performance results of all the combinations over all domains of the development dataset. As can be seen, using the Naïve Bayes method with presence-based features of MPQA words leads to a better performance on average compared to the other methods. However, results of all combinations are close to each other. This shows the fact that both Naïve Bayes and SVM classifiers are performing promisingly in our field of study. The results also present better performance of MPQA words over our set of highly ranked tf-idf features. However, it is important to note that MPQA words are labelled manually, while the TFIDF4000 features are extracted automatically, which again shows the power of

Table 4.2: Results of different combinations of two learning methods and two feature vectors

	Books	**Dvd**	**Electronics**	**Kitchen**	**Average**
NB & MPQA	75	**78.95**	78.55	**81.35**	**78.46 ± 2.61**
SVM & MPQA	73.35	76.9	78.2	79.85	77.07 ± 2.76
NB & TFIDF4000	**77.25**	72.35	**79.15**	80.05	77.20 ± 3.44
SVM & TFIDF4000	72.7	73.55	76.2	77.55	75.00 ± 2.26

ranking the words based on tf-idf.

4.3 Hybrid method

Since both of the lexical and learning methods have their own benefits and drawbacks, we decide to investigate the effectiveness of a hybrid method built by combining a lexical and a learning method. In an unsupervised fashion, we do not have the labels to provide the learning component with the ground truth. However, we can estimate the labels by applying the lexical labelling system in order to provide a set of preliminary weakly labelled instances with normalized scores. The magnitude of the score for each instance could represent a certainty, to which that instance is positive or negative. We rank all the instances based on this measure to find the most certain ones, to later use them as training feeds for the learning classifier.

Although the lexical method would provide a set of information for the learning counterpart, erroneous preliminary labels can lead to a classifier biased toward wrong information. In order to increase the certainty of instances participating in the training set, we can pick the top instances, ranked based on the absolute amount of score. On the other hand, we need to be careful of making our training dataset balanced, in order not to bias the classifier toward a particular class. Therefore, if we need to select the top P percent of instances, we should pick top $P_{normalized}$ percent of positives and also top $P_{normalized}$ percent of negatives, which is calculated as follows:

$$P_{normalized} = \frac{min(N_p, N_n, N * P/2)}{N} \tag{4.7}$$

In the above formula, N_p and N_n are the number of positive and negative instances.

4.3.1 Experiments

We configure the hybrid method to use Naïve Bayes classification as the learning algorithm fed by presence-based features of the first $4,000$ words with highest TFDFT scores. Fur-

Table 4.3: Experiment over how much percent of the weakly scored data to select for training

P	**Books**	**Dvd**	**Electronics**	**Kitchen**	**Average**
0.2	69.7	67.1	69.3	71.85	69.48 ± 1.94
0.4	69.15	68.1	72	75.4	71.16 ± 3.27
0.6	67.05	70.35	74.7	77.15	$\mathbf{72.31 \pm 4.49}$
0.8	67.35	68.45	75.85	77.5	72.28 ± 5.13
1	63.95	67.25	75.95	75.7	69.05 ± 6.05

Table 4.4: Evaluation of the hybrid method at document level against MPQA lexicon

	Movie
MPQA	70.90
Hybrid(MPQA)	78.65
Opinion Lexicon	71.05
Hybrid(Opinion Lexicon)	78.8

thermore, in order to experiment the efficacy of the hybrid method over various lexicons, we use both MPQA and "opinion" lexicons for the preliminary lexical scoring. These lexicons are the best two lexical scoring resources, based on the results of Table 4.1.

In order to study the effect of the parameter P over the performance of the hybrid method, we run the hybrid method with 5 different values of P from 0.2 to 1 with equal steps. Table 4.3 shows the accuracy of the methods over each dataset in the development set. As can be seen in this table, the maximum average accuracy of all datasets is achieved when $P = 0.6$. This observation would be expected, since increasing P results to a trade-off between the certainty of the labels and amount of information handed over to the learning phase. In other words, having lower values of P, we increase the minimum certainty of the training instances and lose true information at the same time and vice versa by assigning larger amounts to P. Therefore, by setting P to a value close to the middle of the range $[0, 1]$, we can gain a dataset with both enough instances and acceptable certainty.

Note that, our hybrid method and the lexical methods are designed to be used in unsupervised classification tasks. Therefore, comparing their results with those of the learning methods would be unfair, since learning methods take the advantage of the ground truth to learn from. Based on this reasoning, here we only compare performance of running the hybrid method over each dataset against applying the MPQA and "opinion" lexicons.

According to Table 4.4, the hybrid method increases the performance of each of the MPQA or "opinion" lexicons by about 7 percent, which shows a great improvement of the classification. We also measure the significance of this improvement over each of the lexi-

cons by means of the McNemar's test [73], which has the chi-squared distribution. Comparing the classification results of the hybrid method using the Opinion lexicon with its lexical counterpart leads to $\chi^2 = 21.09$ which strongly rejects the null hypothesis ($p < 0.001$). Moreover the same comparison for the MPQA lexicon leads to $\chi^2 = 8.5$ which also rejects the null hypothesis($p < 0.001$). This means that the hybrid method has a statistically significant improvement over the lexical method. Furthermore, the hybrid method which is fed by "opinion" lexicon scores performs slightly better than that of the MPQA. However, the improvement over the MPQA lexicon is more pronounced, compared to that of the "opinion" lexicon (7.25 versus 6.75). This can be the effect of engaging manual scores of the words in the MPQA lexicon. This makes the MPQA lexicon a more interesting candidate compared to the "opinion" lexicon, which can be employed in the next experiments (as will be discussed in Chapter 5).

Chapter 5

Lexicon Expansion

As has been discussed in the previous chapters, the goal is to expand the initial lexicon based on the distribution of words in the target domain, since it lacks domain specific words and their score. According to what we discussed in the "previous works" chapter, expanding a lexicon consists of two major steps, which are finding the most informative words and then assigning polarity scores to them by means of a scoring function. For the first step, one can choose different metrics to rank the words from most informative to least informative. The problem of finding informative words in the text can be represented as a rule mining problem.

- **Rule:** a rule is a logical clause which consists of a set of conditions and a result such as $p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q$

In general, the task of automatically extracting rules from a database is called rule mining. Every p_i in the antecedent of the rule can be one of the words in our text and q would be the polarity to which the co-occurrence of the words in the antecedent set leads. Therefore, if we have a set of these rules extracted from our textual data using rule mining techniques, we can use the words in their antecedents and the label in their consequent to expand our lexicon.

For example, given a sentence like "I watched a very good movie" with a positive label in our corpus, we could extract one or some of the following rules:

1. "$good$" $\rightarrow +$
2. "$good$" $\wedge$ "$movie$" $\rightarrow +$
3. "$very$" $\wedge$ "$good$" $\wedge$ "$movie$" $\rightarrow +$

Therefore, our lexicon expansion problem would be reduced to the classic problem of automatic rule mining. For this work, we use "Association rule mining" as a fast mining method to discover dominant and informative polarity rules.

5.1 Association rule mining

"Association rule mining" is intended to identify strong rules discovered in transactional databases using different measures of interestingness. This method is a simple rule mining method, which searches for the rules that satisfy both a minimum degree of "confidence" and "support". Support of a rule represents the frequency of occurrence of the elements of the rule in the dataset, while confidence shows the conditional probability of the elements of the rule given its consequence, which is the polarity of that rule in our problem. These two metrics are calculated as follows:

$$supp(w) = \frac{N_w}{N_T} \tag{5.1}$$

$$Conf(w, C_i) = \frac{supp(C_i \cup w)}{supp(w)} \tag{5.2}$$

in which N_w and N_T are frequency of the word w and sum of frequencies of all the words in total, respectively. For instance, if the word "nice" has occurred 20 time in negative documents and 30 times in positives ones and also all the words of all the documents are $50,000$, $supp(w = nice) = \frac{20+30}{50000}$, $Conf(w = nice, C_i = positive) = \frac{30}{20+30}$ and $Conf(w = nice, C_i = negative) = \frac{20}{20+30}$. Since our goal is to expand a lexicon of single words, we focus on those rules which have a single word in their antecedents. Extracting the rules with a set of co-occurred words as their antecedents would be a promising future work direction for this scope of the research.

In summary, we need to find frequency of word occurrences, rank them based on that and pick a set of them which satisfy the minimum support. The resulting group of words are the candidates to update the lexicon with the condition to satisfy also the minimum confidence.

For the purpose of measuring confidence of the words we need to find their co-occurrence with the positive and negative labels. These labels could be the labels of the documents that contain the words. But, since we do not have labelled documents in our dataset, we need to consider of an alternative set of labels to find the label/word co-occurrence.

The next unit of the text to consider, as an alternative to documents, is the sentence. Since we also do not have the labels of the sentences, they equally have the same drawback

in our application. However, from the semantic point of view, considering the label of a word as the same as the label of its document could be wrong for many more cases than assigning the label of its sentence to it. This claim can be backed up by the fact that, there may be many positive and negative sentences in a document. For instance, words of a positive sentence which exists in a negative document may weigh the sentence to be positive. But, they do not lead the document to be positive.

Although the sentence is a better text unit than a document to judge the polarity of words, words inside a sentence do not always have the same polarity as itself. There are various reasons supporting this fact including negating words, irony and sarcasm. Since dealing with sarcasm and sense of irony is beyond the scope of this work we just consider negation as a factor which may reflect the reverse polarity of the words to the sentence.

5.2 Providing an Annotated Sentence Dataset

Since we do not have the labels for sentences we should consider a way to provide labels for our sentences. As discussed in the "scoring methods" chapter, we can apply either lexical methods or learning methods or even a combination of them. We plan to try all three of these methods to experiment which one is more powerful and more accurate in labelling the data.

In order to compare the performance of the scoring methods at the sentence level we conduct an experiment over a dataset of sentences extracted from Amazon reviews and labelled manually. Since this dataset is extracted from the same resource that includes our document-level reviews, it would be reasonable to run experiments over it and to analyze the results for later decisions and justifications at the document level.

We pick a combination of Naïve Bayes and TFIDI features as the learning method, MPQA lexicon as the resource of our lexical method and their combination as the configuration of our hybrid method. The reason for selecting these methods is that they were shown to have the best performances among other methods based on the experimental results ran at the document level described in Chapter 4. In Table 5.1, the performance of the lexical and the hybrid methods are compared on three domains at the sentence level. There are no available datasets at the sentence level for the domain of kitchen items from Amazon. Therefore, we only consider the domains of Books, Dvd and Electronics for setting up this experiment. Furthermore, the reason not to consider the results of the learning method in this experiment is that it applies labels of the sentences to build its model, while lexical and

Table 5.1: Evaluation of lexical and hybrid methods at the sentence level based on the accuracy measure

	Books	**Dvd**	**Electronics**	**Average**
Lexical	57.95	50	52.94	53.63 ± 4.01
hline **Hybrid**	72.72	69.67	69.8	$\mathbf{70.73 \pm 1.72}$

hybrid methods are scoring sentences with an unsupervised manner.

As can be observed in Table 5.1, the hybrid method performs with a higher degree of accuracy compared to the lexical method. This is due to the advantage that the hybrid method gets by applying learning and analysing correlation between sentences instead of just applying the domain independent knowledge existing inside a lexicon. Based on this observation, we decide to pick the hybrid scoring at the sentence level to estimate the polarity score of sentences in our work.

5.3 Scoring words based on scores of sentences

Now that we have the preliminary estimated scores for the sentences in our corpus we are able to calculate both the confidence and support of each word occurring in those sentences. Afterwards, we filter the words which are satisfying the minimum thresholds for confidence and support to have the words discriminative and frequent enough for updating the lexicon.

In order to assign a score to each word based on scores of sentences, we need to decide between two options. The first is to simply set the score of the word based on the confidence level of the class for that word such that:

$$S_w = \begin{cases} Conf(w,P) & \text{if } Conf(w,P) \geq Conf(w,N) \\ -Conf(w,N) & \text{if } Conf(w,P) < Conf(w,N) \end{cases} \tag{5.3}$$

where $Conf(w,N)$ and $Conf(w,P)$ are the confidences of the word w belonging to the classes of negative and positive, respectively. For example, if for the word "sad" we have $Conf(w = sad, P) = 0.4, Conf(w = sad, N) = 0.6$, the score of the word would be -0.6.

The second option is to use the following formula as the scoring function. For future references we call this scoring function the "average difference":

$$S_w = \frac{score(w,N) + score(w,P)}{|score(w,N)| + |score(w,P)|} \tag{5.4}$$

$$score(word, Class) = \sum_{word \in sentence_i, sentence_i \in Class} \left(\frac{score(sentence_i)}{num(sentence_i)}\right) \tag{5.5}$$

In the above formula, $score(sentence_i)$ is the score that we provide by means of applying the previous scoring steps, $num(sentence_i)$ is the number of the words in $sentence_i$ and $score(word, Class)$ is a function, which gets the $word$ and $Class$ as the input and returns a score as the output. This score represents the degree to which the $word$ is correlated to $Class$, based on the scores of the sentences it has occurred in. For example, if the average score of all the positive sentences containing the word "beauty" is 24.5 and that of the negative sentences is -5, the score for the word would be $S_w = \frac{24.5+(-5)}{|24.5|+|-5|} = 0.66$.

We need to have a summary of scores of all the sentences in which $word$ has occurred and also those sentences belonging to the class $Class$. In order to obtain this measure, we simply divide the score of each sentence with the number of words of that sentence and sum all of these average scores to assign the score to the $word$. The longer the sentence is, the lesser probable the label of the sentence is the same as the label of its words. This fact would be the justification for taking the average score of the sentence.

In order to experiment on the effectiveness of these two different scoring options we decide to update the preliminary lexicon by using the results of each of these scoring methods and then to apply the new lexicons as the lexical resources to find the polarities of documents. We can then compare the performance of the two scoring methods based on their accuracies for scoring documents. After we explain how to update the lexicon in the next section, we will demonstrate results of running experiments over our word scoring methods.

5.4 Updating the lexicon

The whole procedure of finding frequent and discriminative rules (which are unigrams in our application) by means of applying association rule mining, is to achieve two main purposes:

1. **Extracting and adding new words:** The initial lexicon we start the iteration with, includes mostly domain-independent words that are frequently used in many domains. In order to extend the capability of the lexicon toward classifying the polarity in our target domain, we need to add unigrams which are dominant and also effective in terms of determining polarity.

2. **Modifying scores of the previously existing words:** In the English language there exist many words whose polarity changes depending on the the context and domain of the text. For instance, the adjective "fast" may be positive in the context of vehicles while it is a negative feature in context of teaching. Thus, it is essential to modify

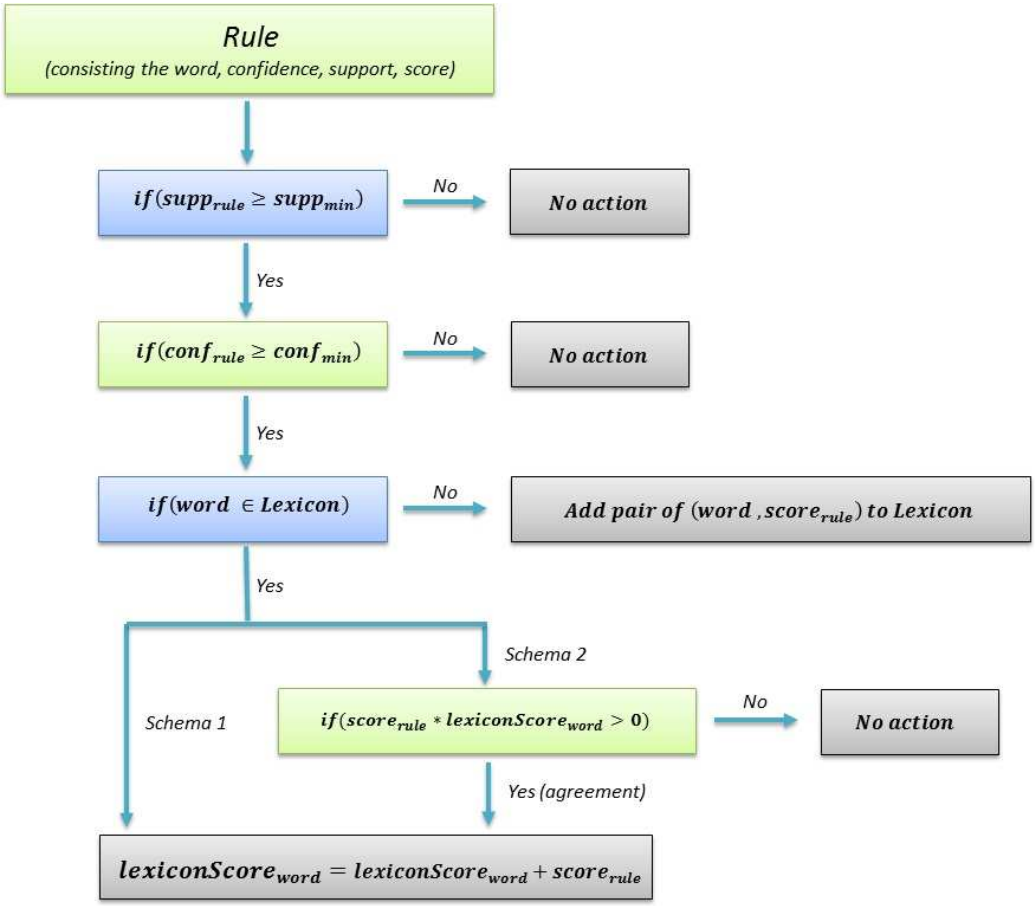

Figure 5.1: Behaviour of the two proposed lexicon update procedures

scores of these types of words in the lexicon based on the new domain in order to boost the performance of the lexicon.

In each iteration of finding dominant words in the documents and scoring them there might exist false positive and false negative scores for the words. Therefore, applying these scores to the lexicon may lead to losing true information and replacing with wrong information instead, which weakens the performance of the lexicon.

In order to achieve the aforementioned two goals without running into the problem of missing information from the lexicon, we propose two versions of updating procedures which are almost the same with just a small difference. These procedures will receive a rule containing a word, its confidence, support and score calculated based on the previous steps and update the lexicon according to their behaviour.

As the steps involved in the two procedures are almost the same, their behaviour are shown simultaneously in Figure 5.1. As can be seen in this figure, a series of conditions and steps are involved in both of the procedures which check boundaries of confidence, support and score of the rule. The only difference between the two procedures is the last step. While the first schema (so called as "Careless modification") does not check if the estimated score of the word matches the score inside the lexicon, the second schema (called "Agreeing modification) ignores whatever rules that disagree with the lexicon. For instance, given the input of the procedure such that the rule is "$essential$" $\rightarrow +$ and its score is equal to 1.5 and also assuming that the word "essential" already exists in our initial lexicon with the score of -1.0, "Careless modification" changes the score of the word in the lexicon to be $(-1.0) + (1.5) = 0.5$, while "Agreeing modification" ends the updating procedure without changing anything in the lexicon.

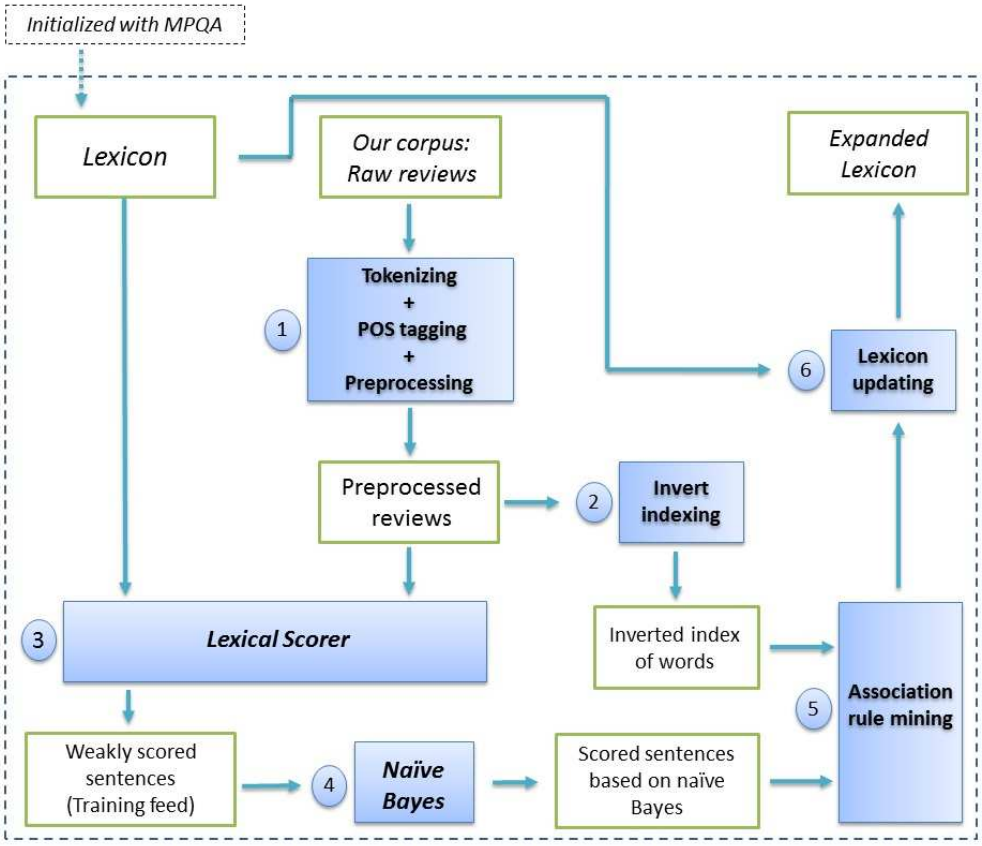

Figure 5.2: Big picture of the lexicon expansion model including all the components

The updating procedures described in Figure 5.1 inject two parameters to the method which belong to association rule mining algorithm. These parameters, called $supp_{min}$ and

$conf_{min}$, are relatively the minimum thresholds of support and confidence to keep a rule as a legitimate one. We would show results of running experiments to fine tune these parameters as well.

By having the lexicon updating module, we create all the essential components of our expansion system. Figure 5.2 represents the big picture of the whole process of the lexicon expansion.

5.5 Experiments on lexicon expansion

Now that we have put all pieces of the chain together for our lexicon expansion method, we can run experiments to fine tune the parameters and decide among different settings of the method.

The evaluation can be done by iterating over different sets of parameter values, expanding the lexicon based on each set and applying the expanded lexicon in each iteration to measure accuracy of the polarity classification over our documents. The performance measure for the expanded lexicon would represent how much efficient the setting of its parameters would be.

Since there exist multiple parameters in different parts of the method, running optimization over all possible sets of values for them would be significantly time consuming. Therefore, we choose to act greedily to find a set of parameters which would result to a close to optimal solution. In order to fine tune each parameter, we fix all the other parameters to certain values and iterate the parameter value over a certain interval to search for the best performance. When we find the best value for this parameter, we fix it to that value and apply the same action for the rest of the parameters.

- **Minimum confidence and minimum support:** These two parameters belong to the association rule mining and determine the minimum level of quality for a rule to be considered for the lexicon expansion procedure. By increasing these thresholds, the minimum desired confidence and frequency of a word will be increased which makes it harder for the words to be qualified. On the other hand, increasing them too much leads to losing informative words. For instance, there are words which are not very frequent (hence have a low support) but are effective in the polarity classification. This behaviour is observed also in experiments, which are shown in Figure 5.3 and 5.4. In Figure 5.3, we observe that the performance tends to get to the maximum value with a minimum confidence of 0.65. Also, as can be seen in Figure 5.4, performance

reaches maximum with a minimum support of 0.00008.

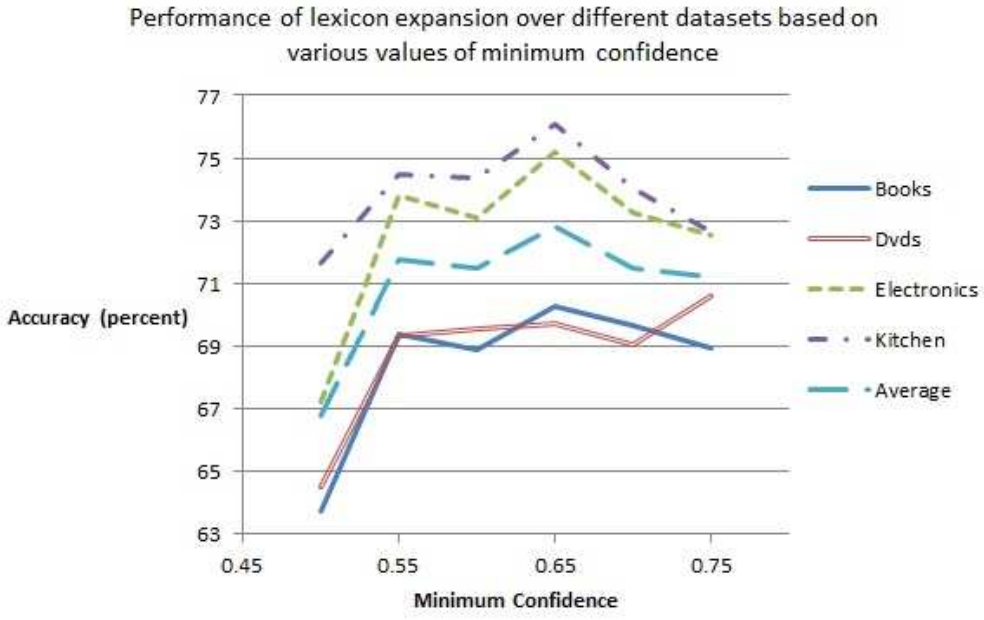

Figure 5.3: Performance of the lexicon expansion over different datasets based on the various values of minimum confidence

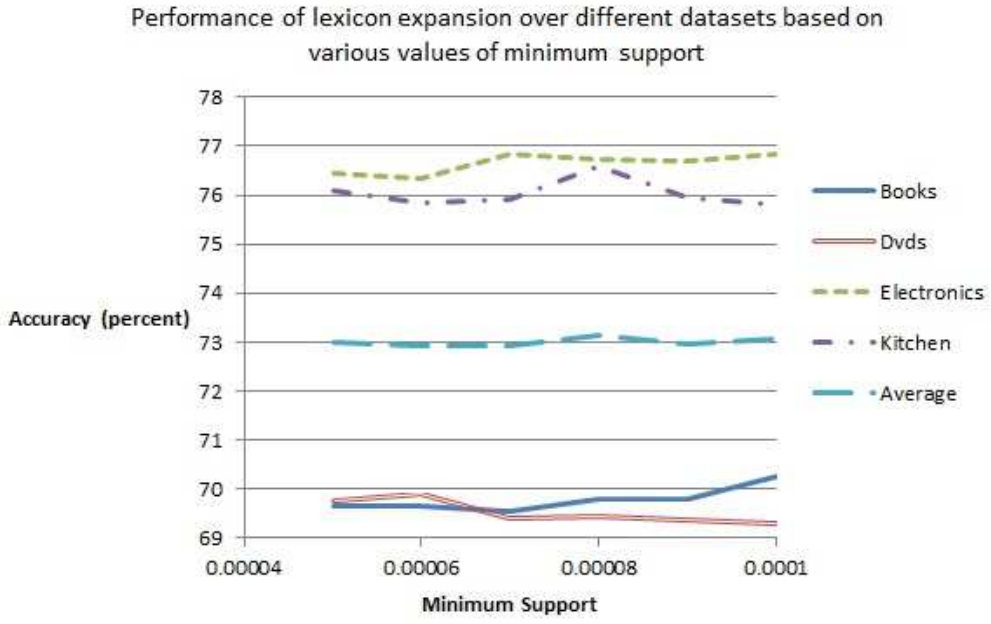

Figure 5.4: Performance of lexicon expansion over different datasets based on various values of minimum support

Table 5.2: Performance comparison between the two word scoring methods

	Books	**Dvd**	**Electronics**	**Kitchen**	**Average**
Confidence	73.1	72.05	79.4	78.85	75.85 ± 3.81
Average difference	69.25	71.2	72	73.55	71.5 ± 1.78

Table 5.3: Performance comparison between the two lexicon updating procedures

	Books	**Dvd**	**Electronics**	**Kitchen**	**Average**
MPQA (before expansion)	69.45	70.45	72.15	73.85	71.47 ± 1.93
Careless modification	70.3	69.45	75.3	75.9	72.73 ± 3.33
Agreeing modification	73.1	72.05	79.4	78.85	75.85 ± 3.81

- **Word scoring method:** In Section 5.3, we have proposed two different scoring methods for words. Here we compare the performance of running the lexicon expansion by means of applying each of these methods. As can be seen in Table 5.2, assigning the confidence as the score would lead to a better set of results in most cases. Thus, we pick this method as the word scoring module.

- **Updating lexicon procedures:** In Table 5.3, the performance of "Careless modification" and "Agreeing modification" procedures are shown. As can be seen, the "Agreeing modification" schema performs better. This leads to the fact that, scores of the words in MPQA lexicon are generally true in all tested domains, which by modification based on poorly scored data may flip to be wrong.

5.5.1 Analysis of results

As shown in the experiments over the development set, the performance of the initial lexicon will increase by using our expansion method. By analyzing the results of these experiments we can summarize the behaviour of the expansion model and discuss about its pros and cons. The expansion model attempts to apply two types of changes in the lexicon: updating the previous words and adding new words. As shown in Table 5.3, if we apply the "Agreeing modification" schema, it would result a better performance, compared to the "Careless modification". Based on this, we can infer that the expansion method does not update the previously existing words properly in the lexicon. However, according to the same experiment, even "Careless modification" would increase the performance of the initial lexicon. This means that the second type of changes applied by the expansion model, which is addition, compensates the improper effect of updating the scores.

The proposed expansion algorithm encapsulates many parameters, which need to be

Table 5.4: Comparison with the previous methods

	Movie
MPQA(Baseline)	70.90
Expansion method	71.2
Hybrid method	**78.65**
G. Zhou [43]	73.6
S. Dasgupta [47]	70.9
Y. He [40]	74.1

fine tuned to increase its performance. As can be seen in Figure 5.3 and Figure 5.4, the sensitivity of the algorithm is high with respect to the changes of parameter values. However, results of running the fine tunings over various domains show a similar behaviour among all the domains, which makes the model robust and adaptable to be used in any domain of data.

Table 5.4 shows the performance results of the state-of-the-art unsupervised polarity classification methods, along with the results of our two proposed methods, the hybrid method at the document level and the lexicon expansion method. The previous works, compared to our methods in this table, include the work of Zhou et al. [43], Dasgupta et al. [47] and He et al. [40], which have been discussed in Chapter 2. As can be seen in this table, the hybrid method has the best performance by about 5 percent with the second best method, which is the work of Y. He.

The result of the expansion model is not competitive, compared to the best methods. Although a slight improvement can be observed by using the expansion method compared to the baseline, this improvement is not statistically significant. The number of the words existing in the MPQA lexicon is 14000. By comparing the words of the expanded lexicon with those of the initial lexicon, we realized that the number of the words updated in the lexicon is 2469, while the number of the new words added to the lexicon is 813, in which case the size of the initial lexicon increases by only 5.8 percent. This is the effect of increasing the quality measures such as the minimum confidence and minimum support thresholds, which decreases the freedom of the expansion model and makes it biased toward a result close to that of the baseline. Furthermore, all of the domains in the development set belong to a different website than that of the test set. Since, all the parameter fine tunings have been done in those domains, this result would be expected for the movie domain.

Chapter 6

Applying Users' Feedback

The procedure of updating a lexicon without having any set of knowledge regarding the ground truth may still be error-prone. We estimate the scores for sentences and used these scores for scoring the words which may propagate the error from the former step to the latter. Also, we update the lexicon based on the estimated scores for the words which (if wrongly scored) may decrease the truly scored weight of a word. In all of these steps, the essence of a knowledge set about the polarity of text units (either sentences or words) to reduce the error in the scoring tasks is felt.

We decide to take advantage of the users' feedback in order to boost the performance of our classifiers. A "feedback" occurs when the output of a system happens to be fed back as the input of the same system. In our work we need to use the opinion of the users about either the results of our classifiers or scorers as the feedback to our system. As we discussed above, errors could happen either in the sentence scoring procedure or word scoring procedure in our system. Therefore, we aim to gather users' feedback about degree of polarity, both in sentence and word level.

6.1 Experiments

Before we proceed to design and implement the feedback collection mechanism, we need to investigate efficacy of gathering and applying feedback over our classifiers and lexicon expansion method. In this section we conduct a series of experiments and analyze the effect of applying feedback over both words and sentences:

6.1.1 Feedback to polarity of words

Since we do not have any set of feedback gathered from actual users we need to simulate a set of feedback for words by means of a manually labelled lexicon. Although it is almost

Table 6.1: Effect of feedback for words over performance of lexicon expansion method

	Books	**Dvd**	**Electronics**	**Kitchen**	**Average**
Before applying feedback	71.35	67.4	74.5	74	71.81 ± 3.25
After applying feedback	72.05	65.75	75.95	78.1	72.96 ± 5.42

trivial that having a set of labelled words would increase the power of the lexicon, we would need to check if this trend exists over our method too. For the purpose of simulating feedback, we use the MPQA lexicon. We separate 20% of the words inside the lexicon and feed our expansion method with the rest of the words inside it. After running the expansion method, we apply the 20% separated part of the lexicon, as feedback to the expanded lexicon.

Table 6.1 shows results of running the aforementioned experiment over all domains of the development set. Except the Dvd domain, applying the feedback has increased the performance of the expanded lexicon. On average the accuracy has been improved about 1 percent. Since the amount of the words applied as feedback are not too much, performance improvement is not considerable. However, this improvement could be promising if we could apply more feedback.

6.1.2 Feedback to polarity of sentences

We need to provide the system with a set of labelled sentences to imitate users' feedback for it. Since, in all domains of our datasets, labels are just provided for documents and not for sentences, we are not able to provide the feedback for sentences of the documents in our data. As an alternative, we use other sources of labelled sentences as our feedback. We need to investigate feedback of sentences over a certain domain of data. Thus, it is necessary to have feedback sentences in the same domain as that of our experimented data. But, we cannot find a set of those sentences for all domains of our data. Therefore, we are able to show results of our experiment for those domains which are publicly provided with labelled sentences.

The datasets we use for sentence feedback for the domains of Books, Dvd and Electronics are provided from the Amazon, which is the same website that instances of our original datasets of reviews are pulled from. The corresponding dataset for each domain includes 200 sentences. We do not include the domain of the Kitchen, since there are no available datasets of annotated sentences for this domain. Furthermore, since the purpose of running

Table 6.2: Effect of feedback for sentences over performance of lexicon expansion method

	Books	**Dvd**	**Electronics**	**Movie**	**Average**
Before applying feedback	70.95	71.85	77.55	71.2	72.89 ± 3.13
After applying feedback	71.9	72.6	77.75	70.7	73.24 ± 3.11

this experiment is solely studying the effect of feedback and also we do not compare its results with the other methods, we include domain of the movies in this experiment as well. There are no available annotated data for the sentences of the IMDB movie website. However, there exists a balanced dataset of the labelled sentences in the domain of the movies from another website called "rottentomatoes", which consists of $10,662$ sentences. We use this dataset as the feedback for the domain of the movies.

Table 6.2 shows the results of applying these feedback. Similar to the results of applying words as feedback, we can observe a slight improvement over the average accuracy of the expanded lexicon. Since the sentences are not originally from our documents, we cannot expect a large progress by applying an external set of feedback. However, we can guarantee the advantage that having feedback on sentences of the same domain could increase accuracy of the method. Furthermore, in spite of the other domains, a slight decrease of the accuracy can be observed in the domain of the movies. This result could represent the fact that, applying the existing labelled instances from another website would not necessarily enhance the performance, even if the domain of that website is the same as the target dataset.

6.2 User interface

According to the results of Section 6.1, having feedback to the output of our lexicon would boost performance of our method. Therefore, we decide to design and build a feedback collector, both at the word and the sentence level. A well designed system with capability of gathering feedback from users should abide by the following criteria:

1. **User friendly:** The more complex the system is, the less the user is willing to enter his thoughts and opinions. Therefore, the system needs to be simple and also attracting enough to bring users' attention to itself.

2. **Applying users' feedback in real-time:** The user expects to see his opinion being applied to the result of classifier immediately, so that he could feel his contribution.

Thus, considering feedback in batch processing and not applying users' feedback instantaneously may not be of users interests.

3. **Aggregation of feedback:** In spite of the fact that users prefer to see modifications instantly as they give feedback, it might not be legitimate for the system to apply each feedback immediately. This is the case since either the user may submit a wrong feedback or the feedback is subjective and needs to be compared against opinions of other users. Therefore, summarizing all the opinions and applying the aggregated result would be necessary.

In order to apply users' feedback instantaneously and to apply aggregated feedback at the same time, we need to personalize our lexicon for each user. In other words, every user needs to have his own version of the lexicon. Also, there should be a general version of the lexicon, in which aggregation of feedback will be applied by means of batch processing. Thus, whenever the user inserts a feedback to the system, his version of the lexicon will be updated immediately and scores of words, sentences and documents are getting modified correspondingly. Furthermore, whenever the number of user's feedback reaches a certain threshold, all of his feedback would be sent to the general lexicon. In the general lexicon, user's feedback for each sentence or word are compared to the other users' feedback and an average of all of their feedback would be applied to that lexicon.

6.3 Meerkat

Our lexicon expansion method along with the feedback collector system is embedded within a public social network analysis tool called Meerkat. Developed in the *Alberta Innovates Centre for Machine Learning (AICML)*, Meerkat offers a wide range of visualization and analyzing algorithms as well as a set of text analysis tools for datasets consisting a group of messages such as forums, discussions, and so forth. Figure 6.1 shows a snapshot of the message panel inside Meerkat, including the forums, discussions and messages inside them. By means of applying our lexicon as the lexical resource of polarity classification, we provide the following functionalities for Meerkat messages:

- **Polarity classification:** By applying our lexicon as the source of the lexical method demonstrated in Section 4.1 we provide the polarity label for each message. Figure 6.2 shows the message window in Meerkat. One can open up this window for any of the message in Meerkat, which includes general polarity of the message, its meta

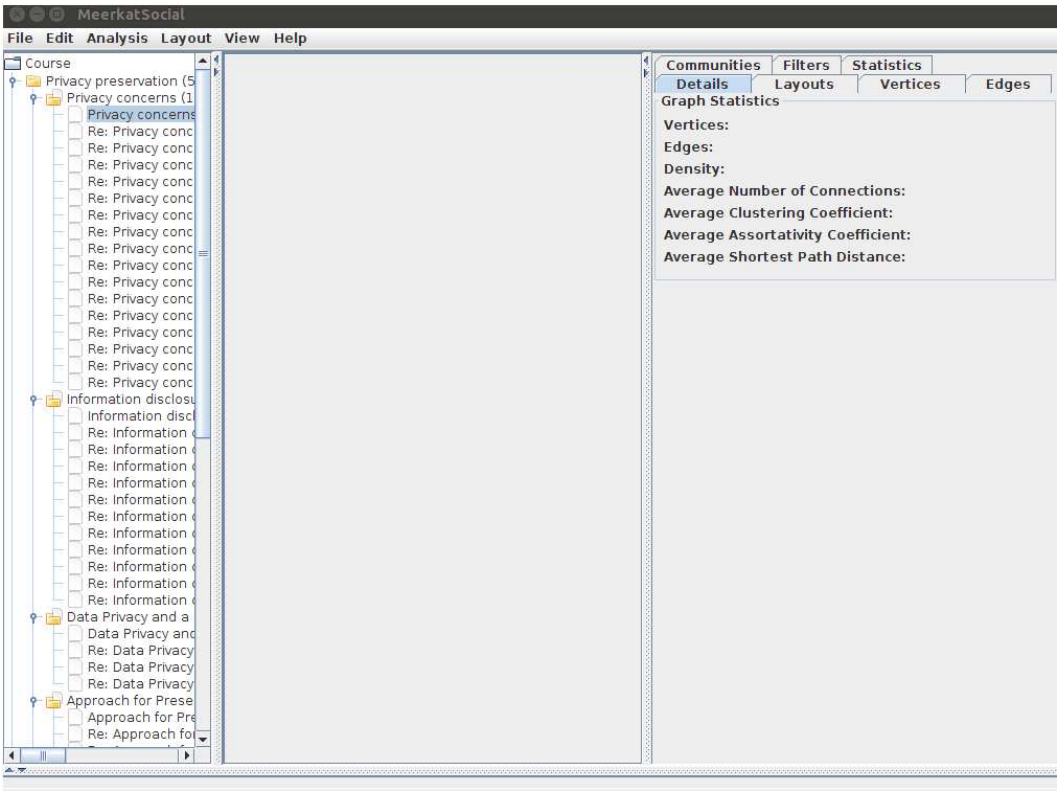

Figure 6.1: A snapshot of message panel in Meerkat (the tree structure shown on the left side of the image)

data and content. Furthermore, the words inside content of the message are coloured with respect to their polarity inside the lexicon. There exists an "edit" button at the left bottom of this window which enables the feedback mode for the message. With this method of visualization, the user would have the clues of our decision about general polarity of the message.

- **Feedback collection:** In order to make an easy-to-use and user friendly interface for feedback collection, we have provided a feedback mode for the message window as shown in Figure 6.3. In this mode, the content of the message is tokenized into its sentences and the sentences are shown separately. Each sentence has a slide bar, whose default value is the polarity assigned by our sentence classifier. This value can change from -1, which is the left side of the slide bar shown as dark red, to 1, which is dark green on the right side.

  Furthermore, if the user clicks on any of the words in the content, he can change the colour of the word, which represents polarity of that word. There is a legend on the

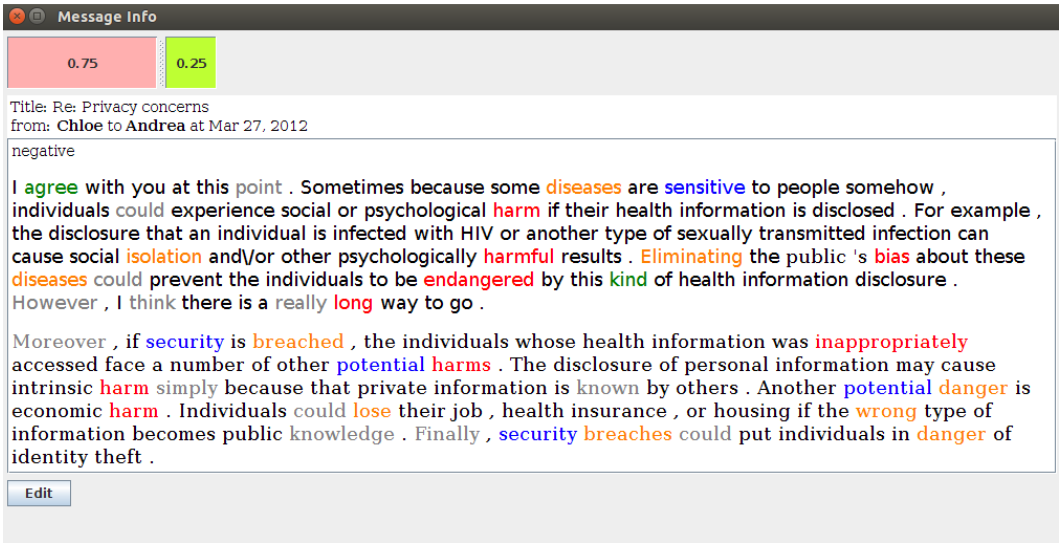

Figure 6.2: Message window, polarity of the message at the top left of the window and coloured words

top left of the window, which shows the polarity that each of the colours represents. Furthermore, the order of colours can help the user to figure out what colour the word will change into, if he clicks on it. The default colour for each word is correlated to the initial polarity of the word in the lexicon. After the user finishes applying the feedback, he can submit the results by clicking on the "confirm" button on the bottom left of the window.

- **Analysis of polarity:** By having the general polarity for each message, we can visualize interesting statistics about polarity of a thread or a forum. The user is able to take a look at the trend of polarity of messages in a thread over time, as shown in the temporal diagram in Figure 6.4. He is also capable of observing the histogram of polarities of a thread or a forum as shown in Figure 6.4. These types of statistics would help the user to have a better idea of the community's characteristics.

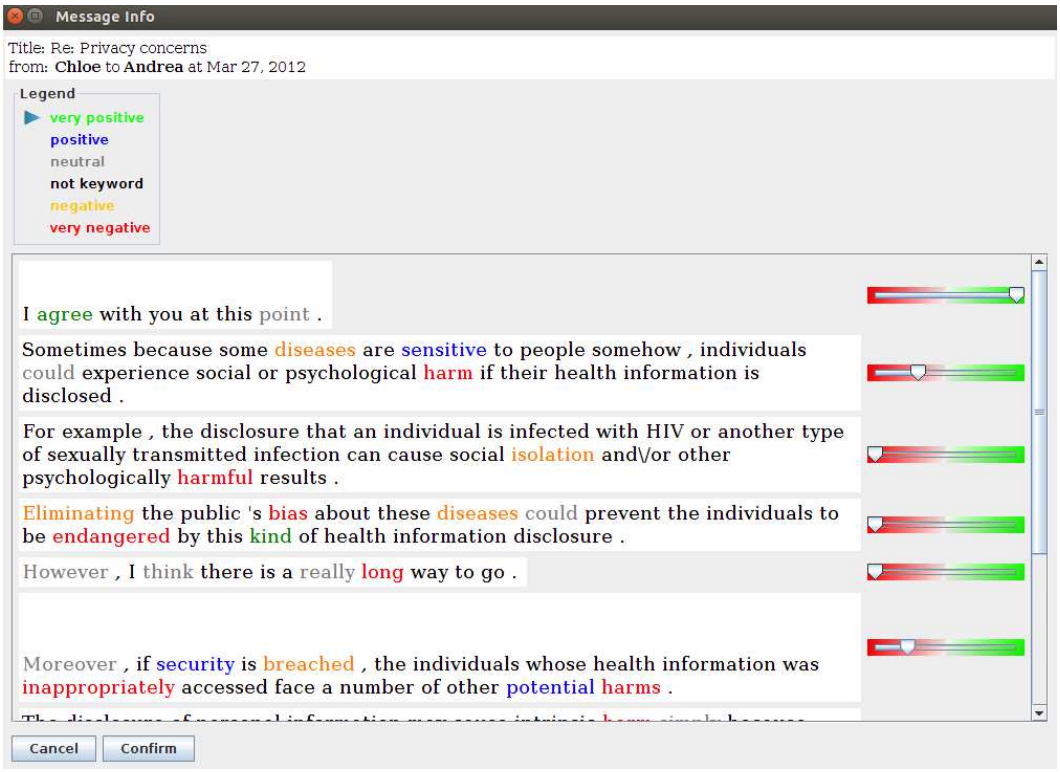

Figure 6.3: Feedback mode in the message window

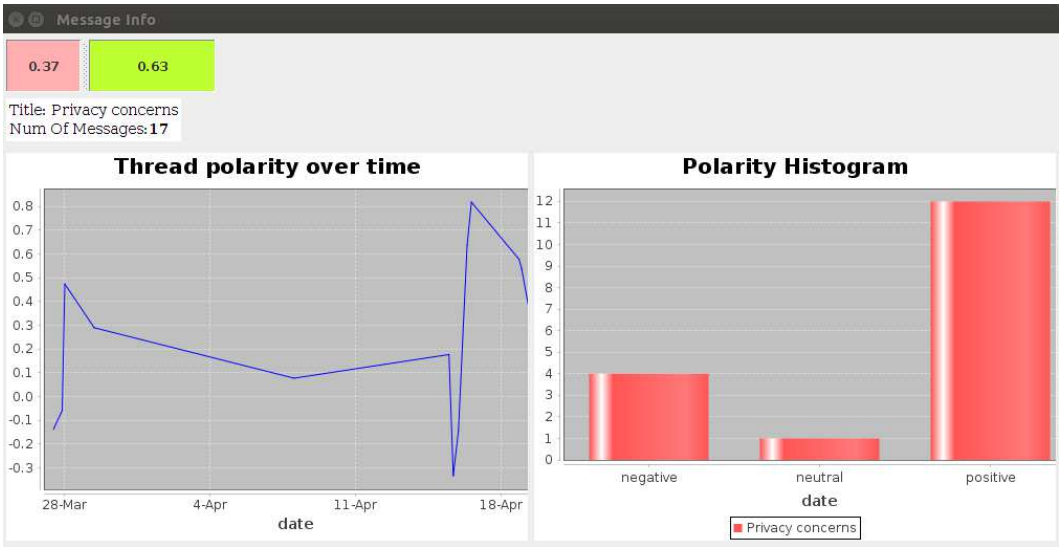

Figure 6.4: Diagrams window for a thread of messages in Meerkat. The left diagram is the temporal trend of polarity and the right one is the histogram of polarities inside that thread

Chapter 7

Conclusions and Future Work

In this work, we tackled the prime problem of polarity classification to build a classifier for the messages inside our educational forum. We needed our classifier to be intuitive and explicit enough, so that the user could get a general understanding of the polarity of the messages inside each thread of discussion. Furthermore, the user could observe the justifications of the results of our classification as intuitive clues and provide feedback to help our system improve.

One of the main challenges we had in our data was the lack of polarity labels for the messages. The other main issue we faced on this data was diversity of domains of topics in discussions. Hence, we needed to have a robust classifier, which does not have any set of information about polarity of data. Furthermore, our classifier must be adaptable to new domains of discussions. We tried to address these issues by creating a lexicon expansion method. Our expansion method uses hybrid scoring, which is a combination of learning and lexical scoring methods. We showed that our proposed hybrid method outperforms both of the lexical and learning methods. We then fed an association rule miner with results of our hybrid scorer, extracted rules out of our messages and expanded our initial lexicon with these rules.

Thereafter, by using the expanded lexicon we classified our messages and showed the polarity of the words inside message by colouring them in order to give clues of our decisions to the user. Furthermore, we created a polarity feedback collector and embedded it inside Meerkat, to gather users' knowledge of polarity both at the word and the sentence level. Finally, we extracted some of the statistics of the threads such as histograms of messages and trend of polarity of messages inside a thread over time.

In summary, we came to a conclusion that we can use the unstructured knowledge, hidden in a set of messages, for the task of classifying them into polarity categories, even

if those messages are not taking the advantage of having polarity labels. Furthermore, our experiments led to the result that having users' feedback would boost performance of our classifiers, though the users may have non-expert opinions about different topics.

In spite of the fact that, our lexicon expansion method improves performance of the initial lexicon, it does not insert a large set of words compared to the size of the initial lexicon. This is due to the fact that the confidence level of many of the rules extracted from the association rule mining is not enough to make them qualified for selection. One of our future research directions would be strengthening the scoring methods to create better scores for the sentences. Furthermore, we could apply other rule mining approaches to investigate their efficacy in our method. In addition, we can extend the rule mining by extracting rules with more than one antecedent as a future direction. Finally, we can apply the results of the classification module to extract more information about the characteristics of our communities and its members, such as users' behaviour mining or monitoring the trends of emotions within the forum.

Bibliography

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[2] L. P. Morency, R. Mihalcea, and P. Doshi, "Towards multimodal sentiment analysis: Harvesting opinions from the web," in *Proceedings of the 13th international conference on multimodal interfaces*, pp. 169–176, ACM, 2011.

[3] A. Neviarouskaya, H. Prendinger, and M. Ishizuka, "Textual affect sensing for sociable and expressive online communication," in *Affective Computing and Intelligent Interaction*, pp. 218–229, Springer, 2007.

[4] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[5] 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*, vol. 10, pp. 79–86, Association for Computational Linguistics, 2002.

[6] 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*, pp. 519–528, ACM, 2003.

[7] T. Mullen and N. Collier, "Sentiment analysis using support vector machines with diverse information sources.," in *EMNLP*, vol. 4, pp. 412–418, 2004.

[8] 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.

[9] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 417–424, Association for Computational Linguistics, 2002.

[10] F. Benamara, C. Cesarano, A. Picariello, D. R. Recupero, and V. S. Subrahmanian, "Sentiment analysis: Adjectives and adverbs are better than adjectives alone.," in *ICWSM*, 2007.

[11] T. Nasukawa and J. Yi, "Sentiment analysis: Capturing favorability using natural language processing," in *Proceedings of the 2nd international conference on Knowledge capture*, pp. 70–77, ACM, 2003.

[12] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, "Learning subjective language," *Computational linguistics*, vol. 30, no. 3, pp. 277–308, 2004.

[13] B. Liu, "Opinion mining and sentiment analysis," in *Web Data Mining*, pp. 459–526, Springer, 2011.

[14] M. Gamon, "Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis," in *Proceedings of the 20th international conference on Computational Linguistics*, p. 841, Association for Computational Linguistics, 2004.

[15] V. Ng, S. Dasgupta, and S. Arifin, "Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews," in *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 611–618, Association for Computational Linguistics, 2006.

[16] S. Das and M. Chen, "Yahoo! for amazon: Extracting market sentiment from stock message boards," in *Proceedings of the Asia Pacific finance association annual conference (APFA)*, vol. 35, p. 43, Bangkok, Thailand, 2001.

[17] J. C. Na, H. Sui, C. Khoo, S. Chan, and Y. Zhou, "Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews," *Advances in Knowledge Organization*, vol. 9, pp. 49–54, 2004.

[18] V. Hatzivassiloglou and J. M. Wiebe, "Effects of adjective orientation and gradability on sentence subjectivity," in *Proceedings of the 18th conference on Computational linguistics*, vol. 1, pp. 299–305, Association for Computational Linguistics, 2000.

[19] 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*, pp. 129–136, Association for Computational Linguistics, 2003.

[20] A. Andreevskaia and S. Bergler, "Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses.," in *EACL*, vol. 6, pp. 209–215, 2006.

[21] A. Esuli and F. Sebastiani, "Determining the semantic orientation of terms through gloss classification," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 617–624, ACM, 2005.

[22] A. Esuli and F. Sebastiani, "Determining term subjectivity and term orientation for opinion mining.," in *EACL*, vol. 6, p. 2006, 2006.

[23] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *Proceedings of LREC*, vol. 6, pp. 417–422, 2006.

[24] A. Finn and N. Kushmerick, "Learning to classify documents according to genre," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 11, pp. 1506–1518, 2006.

[25] H. Takamura, T. Inui, and M. Okumura, "Extracting semantic orientations of phrases from dictionary.," in *HLT-NAACL*, vol. 2007, pp. 292–299, 2007.

[26] N. Kaji and M. Kitsuregawa, "Building lexicon for sentiment analysis from massive collection of html documents.," in *EMNLP-CoNLL*, pp. 1075–1083, Citeseer, 2007.

[27] M. Gamon and A. Aue, "Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms," in *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pp. 57–64, Association for Computational Linguistics, 2005.

[28] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, "Using wordnet to measure semantic orientations of adjectives.," in *LREC*, vol. 4, pp. 1115–1118, Citeseer, 2004.

[29] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pp. 347–354, Association for Computational Linguistics, 2005.

[30] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *ACL*, vol. 7, pp. 440–447, Citeseer, 2007.

[31] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *ACL*, vol. 7, pp. 264–271, Citeseer, 2007.

[32] A. Aue and M. Gamon, "Customizing sentiment classifiers to new domains: A case study," in *Proceedings of recent advances in natural language processing (RANLP)*, vol. 1, Citeseer, 2005.

[33] H. Yang, J. Callan, and L. Si, "Knowledge transfer and opinion detection in the trec 2006 blog track.," in *TREC*, 2006.

[34] S. Tan, G. Wu, H. Tang, and X. Cheng, "A novel scheme for domain-transfer problem in the context of sentiment analysis," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 979–982, ACM, 2007.

[35] S. Tan, X. Cheng, Y. Wang, and H. Xu, "Adapting naive bayes to domain adaptation for sentiment analysis," in *Advances in Information Retrieval*, pp. 337–349, Springer, 2009.

[36] S. Li, Y. Xue, Z. Wang, and G. Zhou, "Active learning for cross-domain sentiment classification," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 2127–2133, AAAI Press, 2013.

[37] T. Li, V. Sindhwani, C. Ding, and Y. Zhang, "Knowledge transformation for cross-domain sentiment classification," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 716–717, ACM, 2009.

[38] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proceedings of the 19th international conference on World Wide Web*, pp. 751–760, ACM, 2010.

[39] N. Kaji and M. Kitsuregawa, "Automatic construction of polarity-tagged corpus from html documents," in *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 452–459, Association for Computational Linguistics, 2006.

[40] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 375–384, ACM, 2009.

[41] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[42] C. Lin, Y. He, and R. Everson, "A comparative study of bayesian models for unsupervised sentiment detection," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pp. 144–152, Association for computational linguistics, 2010.

[43] G. Zhou, J. Zhao, and D. Zeng, "Sentiment classification with graph co-regularization," in *Proceedings of COLING*, pp. 1331–1340, 2014.

[44] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.

[45] L. Qiu, W. Zhang, C. Hu, and K. Zhao, "Selc: a self-supervised model for sentiment classification," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 929–936, ACM, 2009.

[46] R. Prabowo and M. Thelwall, "Sentiment analysis: A combined approach," *Journal of Informetrics*, vol. 3, no. 2, pp. 143–157, 2009.

[47] S. Dasgupta and V. Ng, "Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification," in *Proceedings of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, vol. 2, pp. 701–709, Association for Computational Linguistics, 2009.

[48] X. Hu, J. Tang, H. Gao, and H. Liu, "Unsupervised sentiment analysis with emotional signals," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 607–618, International World Wide Web Conferences Steering Committee, 2013.

[49] B. Liu and L. Zhang, "A survey of opinion mining and sentiment analysis," in *Mining Text Data*, pp. 415–463, Springer, 2012.

[50] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.

[51] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Systems*, 2015.

[52] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[53] P. Stone, D. C. Dunphy, M. S. Smith, and D. Ogilvie, "The general inquirer: A computer approach to content analysis," *Journal of Regional Science*, vol. 8, no. 1, pp. 113–116, 1968.

[54] 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*, pp. 168–177, ACM, 2004.

[55] B. Liu, M. Hu, and J. Cheng, "Opinion observer: analyzing and comparing opinions on the web," in *Proceedings of the 14th international conference on World Wide Web*, pp. 342–351, ACM, 2005.

[56] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan, "Opinionfinder: A system for subjectivity analysis," in *Proceedings of hlt/emnlp on interactive demonstrations*, pp. 34–35, Association for Computational Linguistics, 2005.

[57] D. A. Medler, A. Arnoldussen, J. Binder, and M. Seidenberg, "The wisconsin perceptual attribute ratings database." `http://www.neuro.mcw.edu/ratings/`, 2005.

[58] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.," in *LREC*, vol. 10, pp. 2200–2204, 2010.

[59] X. Z. Svetlana Kiritchenko and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research (JAIR)*, vol. 50, pp. 723–762, 2014.

[60] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," in *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, June 2013.

[61] X. Zhu, S. Kiritchenko, and S. M. Mohammad, "Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets," *SemEval 2014*, p. 443, 2014.

[62] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "Nrc-canada-2014: Detecting aspects and sentiment in customer reviews," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 437–442, Association for Computational Linguistics and Dublin City University, 2014.

[63] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, p. 271, Association for Computational Linguistics, 2004.

[64] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 115–124, Association for Computational Linguistics, 2005.

[65] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1275–1284, ACM, 2009.

[66] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[67] "will." `http://www.merriam-webster.com/dictionary/will`. Accessed: 2014-11-12.

[68] "Part Of Speech." `http://en.wikipedia.org/wiki/Part_of_speech`. Accessed: 2014-11-12.

[69] C. D. Manning, "Part-of-speech tagging from 97% to 100%: is it time for some linguistics?," in *Computational Linguistics and Intelligent Text Processing*, pp. 171–189, Springer, 2011.

[70] "Inverted index." `http://en.wikipedia.org/wiki/Inverted_index`. Accessed: 2014-11-20.

[71] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[72] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[73] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.