Sentiment Classification with Case-Based Approach

Mémoire

Bibizeinab Torabian

Maîtrise en informatique
Maîtrise ès sciences (M.Sc.)

Québec, Canada

© Bibizeinab Torabian, 2016

Sentiment Classification with Case-Based Approach

Mémoire

Bibizeinab Torabian

Sous la direction de :

Directeur de recherche Luc Lamontagne
Codirecteur de recherche François Laviolette

Résumé

L'augmentation de la croissance des réseaux, des blogs et des utilisateurs des sites d'examen sociaux font d'Internet une énorme source de données, en particulier sur la façon dont les gens pensent, sentent et agissent envers différentes questions. Ces jours-ci, les opinions des gens jouent un rôle important dans la politique, l'industrie, l'éducation, etc. Alors, les gouvernements, les grandes et petites industries, les instituts universitaires, les entreprises et les individus cherchent à étudier des techniques automatiques fin d’extraire les informations dont ils ont besoin dans les larges volumes de données. L’analyse des sentiments est une véritable réponse à ce besoin. Elle est une application de traitement du langage naturel et linguistique informatique qui se compose de techniques de pointe telles que l'apprentissage machine et les modèles de langue pour capturer les évaluations positives, négatives ou neutre, avec ou sans leur force, dans des texte brut.

Dans ce mémoire, nous étudions une approche basée sur les cas pour l'analyse des sentiments au niveau des documents. Notre approche basée sur les cas génère un classificateur binaire qui utilise un ensemble de documents classifies, et cinq lexiques de sentiments différents pour extraire la polarité sur les scores correspondants aux commentaires. Puisque l'analyse des sentiments est en soi une tâche dépendante du domaine qui rend le travail difficile et coûteux, nous appliquons une approche «cross domain» en basant notre classificateur sur les six différents domaines au lieu de le limiter à un seul domaine. Pour améliorer la précision de la classification, nous ajoutons la détection de la négation comme une partie de notre algorithme. En outre, pour améliorer la performance de notre approche, quelques modifications innovantes sont appliquées. Il est intéressant de mentionner que notre approche ouvre la voie à nouveaux développements en ajoutant plus de lexiques de sentiment et ensembles de données à l'avenir.

Abstract

Increasing growth of the social networks, blogs, and user review sites make Internet a huge source of data especially about how people think, feel, and act toward different issues. These days, people opinions play an important role in the politic, industry, education, etc. Thus governments, large and small industries, academic institutes, companies, and individuals are looking for investigating automatic techniques to extract their desire information from large amount of data. Sentiment analysis is one true answer to this need. Sentiment analysis is an application of natural language processing and computational linguistic that consists of advanced techniques such as machine learning and language model approaches to capture the evaluative factors such as positive, negative, or neutral, with or without their strength, from plain texts.

In this thesis we study a case-based approach on cross-domain for sentiment analysis on the document level. Our case-based algorithm generates a binary classifier that uses a set of the processed cases, and five different sentiment lexicons to extract the polarity along the corresponding scores from the reviews. Since sentiment analysis inherently is a domain dependent task that makes it problematic and expensive work, we use a cross-domain approach by training our classifier on the six different domains instead of limiting it to one domain. To improve the accuracy of the classifier, we add negation detection as a part of our algorithm. Moreover, to improve the performance of our approach, some innovative modifications are applied. It is worth to mention that our approach allows for further developments by adding more sentiment lexicons and data sets in the future.

Table of Contents

List of Tables

List of Figures

Acknowledgement

First of all, I am deeply grateful to my thesis research director Prof. Luc Lamontagne for accepting me as a Master student in his group, GRAAL, and for his continuous encouragement, patience and support throughout my master period. I would like to express my deepest appreciation to my research co-director Prof. François Laviolette for his support and helpful discussions in different stages of this research work.

Most important my deep gratitude to my dear mother who plays an important role as mother, father, and friend in all my life. My huge love goes to my youngest sister Neda for her amiability and support. Also I am very thankful to my sister Fatima, and my brother Hassan, who have been a constant source of love and consolation despite of being thousand miles away from me.

Besides, I want to thank my dear uncle Joseph and my precious friends Ahdi and Hamid for their kindness and helpful supports during my earliest days in Canada.

Also many thanks to Hani and Sharar who make me beautiful and memorable moments in twenty years of friendship, and my teammates and all beloved friends here, particularly Adeleh, Asra, Amir, and Hamid for sharing their moments and personal experiences of living in Canada with me.

Last but not least, my sincere gratitude goes to Shirley, Paule, René and Chahinez who have always believed in me and supported me with their kindness and valuable advices during writing this thesis.

Finally, I give my personal thanks to anyone reading this thesis.

Chapter 1

Introduction

In recent years, the explosive growth of user-generated content such as Internet forums, discussion groups, blogs, and on-line services allows people share their feelings and opinions in many forms such as reviews, star rating and recommendations toward different products or services [1]. This makes a large set of views and opinions in the form of subjective texts that requires deriving a conclusion whether a product or a service is favorable or unfavorable [1]. Therefore, analyzing and mining these huge collections are necessary and motivate researchers to find out the automatic techniques in order to extract the sentiments from those textual repositories. Sentiment analysis, also called opinion mining, is a computational study of opinions, sentiment and emotions which uses natural language processing (NLP), computational techniques, and text analysis for extracting the polarity of unstructured documents, or textual reviews [1][2]. Complexity of human languages has made sentiment analysis a challenging research area in computer science and computational linguistics.

1.1 Motivation

"*What are people's opinions about...?*" has always been one of the crucial questions for most of us especially at the time of making a decision[1]. Before the Web revolution, when people wanted to buy a product or use a service, many asked their friends or family members for recommendations. However, this approach did not always work because their social network did not have plenary of experiences to cover all of their needs [1]. In addition, if an organization or a company needed to know the public's feelings and judgements about its fabrications or services, it had to manage an opinion polls or take a survey of a target group [3]. With the popularization of Internet usage since 1998, a large repository of textual opinions and reviews has been created. The most popular such sources are Amazon, C-Net, RottenTomatoes, and IMDB. The availability of these reviews has changed the information gathering process[2]. Now, it is possible to read the opinions and experiences of hundreds of people about almost every existing product [1]. However, reading through all this information in order to reach a conclusion on whether a product or a service is good or bad, is a time-consuming task [2]. Moreover, drawing an inference (positive or negative) when there are multiple conflicting opinions is very difficult, even for human. Nonetheless, mining different views and experiences, and determining public opinions about a given service or merchandise, is crucial for commercial companies and individual consumers [2]. Sentiment analysis is a powerful tool that can automatically extract opinions and sentiments from online sources, and classify them as positive, negative or neutral in a structured manner [2].

1.2 Main Contributions

The main objective of this study is to present a case-based reasoning approach for sentiment analysis task. Our approach is inspired by Ohana et al [4]. This approach determines the polarity of the product reviews in document level where out-of-domain labeled training data is used to train a classifier that predicts the orientation of new documents. Case-based reasoning is the act of developing a solution for a new problem based on the pre-existing experiences of similar prior problems [5]. Based on the various researches, beyond the general sentimental words like "good", sentiment analysis is depended on the domain which used for. Since there are vast different domains on the web, making a special classifier for each domain is not an efficient way. Thus, we examine our approach on cross-domain to obtain a method more general. Cross-domain approaches use domain-independent data set instead of domain-specific data to train a classifier that works well on different target domains [6]. In this project, we use a training dataset that contains of six different domains of user-generated products including apparel, book, electronics, hotel, music and movie. Each domain contains 2000 labelled reviews (1000 positive and 1000 negative) in plain text which are extracted from Amazon.com and IMDB database. In our method, each review or document is converted to a case with two essential components: a case description and a case solution. The case description is a document signature that is derived from document and includes two categories: the document statistics features, and the writing style parameters. Case solution is a set of sentiment lexicons yielding correct prediction of the document polarity. A case base includes a set of cases that is used to evaluate the polarity of new unlabeled reviews. To make a prediction, the most similar cases to an unseen review are retrieved, and lexicons of their case solutions are reused [4].

To improve our results, we apply negation detection as an important aspect of the sentiment analysis. For example the sentences "*I like this book*" and "*I don't like this book*" have opposite sentiment orientation, despite the presence of the positive term "*like*" on both [1]. In this regard, firstly we specify the negation clues or keywords. We consider 20 negation markers as: *"no"*, *"not" and "n't", "none", "nobody", "nowhere", "never", "barely", "scarcely", "hardly", "nothing", "neither…nor", "no one", "no person", "anyone", "anybody", "without", "lack", "lacked", "lacking", "lacks"*. Secondly, we determine the scope of the negation terms as an essential parameter in negation detection by defining 90 different patterns based on the *Penn Treebank* of sentences which is inspired by Blanco and Moldovan [7]. For this purpose, each sentence is parsed using *Stanford Parser* [7] and we use *Tregex Pattern Compiler*[1] to define specific macros and patterns for recognizing negation in a review (see Appendix 3). In addition, we find out the order of the patterns is important for a proper detection thus, we adjust their order in the appropriate manner to have more accurate scope.

[1] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/trees/tregex/TregexPattern.html

1.3 Thesis outline

This thesis is organized as follows. In Chapter 2, we conduct a literature review and present some works that have been done in the field of sentiment analysis (SA) and also a brief background on some of the most important concepts that will be used in the later chapters such as artificial intelligence, machine learning, and different classification methods. Chapter 3 provides our case-based approach on SA that is inspired by Ohana et al in [4]. In this regard, the concept of case-based reasoning and its components are explained. Then our classifier and its associated issues are discussed. Negation detection as a necessary aspect of SA is expressed in Chapter 4. A bag of word method is used to identify negation keywords. To find out the negation scope, we apply *Tregex Pattern Compiler* and *Penn Treebank* to documents. In the following, we present our experiment results obtained from the negation detection. In this chapter, the performance of two classifiers, i.e. case-based classifier with and without negation identification, are evaluated and compared using the accuracy measure. Finally, we formulate conclusions and some proposals for future research avenues in Chapter 5.

Chapter 2

Literature review

This chapter provides a brief background on the concept of learning in an intelligent system, and then gives an overview on various models of machine learning approaches. In following sections, we describe the meaning of sentiment analysis along its tasks, features and evaluation measures. Finally, we introduce some of the most popular sentiment analysis tools and end by mentioning some research efforts that have done in the field of sentiment analysis.

2.1 Artificial intelligence overview

How human can think and learn is an important question for many scientists. They have tried to build intelligent entities which can think, make right decisions, solve problems, and more importantly learn. The quest for creating the intelligent machines direct them to a field of study that called *artificial intelligence*, or AI [8].

In the 1940s, the innovation of the programmable digital computer, which was built based on the abstract theories of mathematical reasoning, made great strides in the field of artificial intelligence. Between 1940s and 50s, a handful of scientists from various domains such as mathematics, psychology, engineering, economics, political science and so on, started seriously to research about the possibility of designing and building an electronic or artificial brain [9].

Alan Turing, in 1950s, published a landmark paper [10] with purpose of the possibility of making machines with thinking ability. In his famous test, called *Turing Test*, he marked although "thinking" is a difficult task to define, we could consider that if a machine could handle a conversation and not be differentiated from a human being, then we could claim that the machine has some ability to "think" [8].

The golden years for AI were 1956 to 1974, when the computers were able to prove geometry theorems, solve word problems in algebra, and learn how to speak in English. In that period, many successful and influential works had been accomplished such as reasoning in order to achieve special goals like winning a game or prove a theory. Natural language processing is one of the most important goals of AI researches. Its goal is to give communication abilities in a human language to the computers in which they can understand and respond to an individual's questions, or participate in a discussion with some humans [8].

Nowadays, AI is a multidisciplinary field of study whose aim is creating an intelligent machine or software with ability to think, learn and act to the level of human beings. Since AI is an interdisciplinary field, it requires knowledge of different domains such as computer science, linguistics, psychology, biology, philosophy and so on [8].

An intelligent machine or software (called an agent) needs to possess the various capabilities to make it able to act rationality for the purpose of achieving the best outcome even in uncertainty conditions. Some fields that need be considered at the time of designing an intelligent agent are [8]:

- Natural language processing to give the ability to communicate and understand human's language successfully;
- Knowledge representation to save the agent's information and data;
- Machine learning to learn and adapt to new circumstances.

2.1.1 Learning in intelligent systems

The process of *learning* in an intelligent system is the ability of an agent to improve its performance on future tasks by observing its world and earning experiences. In a learning process, any component of an agent can be improved by using a set of data or examples. The level of improvement and techniques which are used for an intelligent agent are depending on four main factors [8]:

- Which *component* is considered to improve (such as the conditions, states, actions, relevant world's properties and etc.)
- What *prior knowledge* of the world does an agent already have
- What kind of *representation* is used to display the data set and/or the components (e.g. exhibiting the data as a vector of continues or discrete numerical values, or Boolean values).
- What type of *feedback* is available for learning. The feedback can be expressed implicitly or explicitly, or by using positive points as rewards or negative ones as punishment. Based on which form of feedback is selected, there are different models of learning algorithms. We will explain them in following sections.

The critical question here is if the performance of an agent can be improved, why wouldn't the system designers just provide that improvement to start with? There are three main reasons to answer to this question. First, the designers cannot estimate all possible situations that the agent might be faced with. Second, anticipating the changes over time is one of the most complicated tasks. Third, sometimes system designers do not have any idea how they can develop a solution. Thus, using learning methods is a critical factor in the intelligent systems to improve their efficiency and reliability [8].

Another important distinctive feature of learning is, as far as possible, decreasing the humans' direct involvement. Indeed, in the learning phase, the human interaction part is limited to give some data (called training data) to the learning algorithm (the learner). This data can be some set of solved examples of the same type of problems that the agent will face or some observations obtained by supervising the agent's interactions with its environment [8].

2.2 Machine learning paradigms

Generally speaking, Machine Learning (ML) is a scientific discipline that designs or studies the algorithms that can learn from data. It is a subfield of computer science and statistics which has a strong relation to the artificial intelligence. There are several definitions for ML. However, the most formal and popular ones is provided by Tom M. Mitchell in 1997 [11]. According to him, "a computer program (as an agent) is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*". The experience *E* is a set of data that can be observed gradually by the agent via interacting with its environment. The task *T* is the agent's behavior related to its prediction ability. The performance measure P can be either accuracy or precision/recall and etc.

As mentioned in section 2.1.1, anticipating all possible behaviors given all possible inputs is too complicated to be considered by programmers. Hence, the learning algorithm must have the ability to generalize its observations in such a way that it will continue to act correctly, with high probability, even when confronted with unseen situations. Thus, the learning algorithms are designed to generate a "predictor". Given a situation, this predictor can be [8]:

- An action or behavior's policy that is chosen based on some perceptions of the agent's environment, or
- A model to predict the chances of a new observation to come, or
- A function to categorize the inputs (classification).

Based on the types of environment feedback, the machine learning algorithms are organized into several taxonomies. We explain some in below without plunging into details [8]:

1. **Supervised learning:** in this type of learning, we have a set of examples that consists of input-output pairs. The desired predictor is a function that maps an input to a relevant output or label. This set of examples is divided into two distinct subsets: a training set and a test set. The training set is a bunch of correctly labeled examples while the test set contains unseen or new input data that is labeled by the predictor.
2. **Unsupervised learning:** the predictor in this model is a function that detects the patterns in the input data even though no explicit feedback is provided. There is no training set, and no labeled data are involved. The function groups all inputs into several sub-groups based on their common patterns. This task is called clustering.
3. **Semi-supervised learning:** we use this learning method when there are a few labeled examples and more unlabeled ones. The algorithm generates an appropriate predictor to label the new data using knowledge of both labeled data and the clustering function.
4. **Reinforcement learning:** in this method, the objective of the agent is to learn how to take an action based on the environment's feedback. Every action of the agent has some positive or negative impacts in the state of the environment, and the environment provides the feedback in the form of rewards or punishments. In the absent of the training set, the goal of the model is to maximize the rewards in the long term.
5. **Probabilistic model estimation:** the target is to achieve a prediction function as accurately as possible to model the underlying environment especially in the prevalence of uncertainty in the real world (we assume this model is probabilistic).

Note that for items 4, and possibly 5, in some environments, when time goes on, the agent gathers some new information that has to be considered in its task. In these cases, the structure of the environment and the associated learning data are sequential. In contrast, for the rest of items, the agent learns from some examples that are given for once, and provide a function (predictor) that will not consider the environment any more. The structure of this environment is non-sequential or *asynchronous* [8].

2.3 Natural language processing

Natural language processing (NLP) also called as computational linguistics is an interdisciplinary field of computer science, artificial intelligence, and linguistics which is dedicated to the interaction between computers and human via a natural language. The goal of this research area is to give the ability of understanding human languages to computers in order to derive the meanings from natural language inputs either on textual materials or from vocals exchanges [12].

The history of NLP generally began in the 1950s, when Alan Turing published his most cited article titled *"Computing Machinery and Intelligence"* [10] which is known as a measure of the intelligence for machines. Up to 1980, scientists used very complex hand-written rules for NLP systems. In the late 1980s, by introducing machine learning concepts, a significant revolution has come in NLP research area and language processing field. Modern NLP systems are using various paradigms of machine learning to drive the hidden rules and structures behind the textual or vocal human languages by analysis of large sources (corpora) of typical real-word examples [12].

There are several works in NLP which serve direct real-world applications, while others commonly are used as sub-tasks of larger tasks. For instance, machine translation (an application that automatically translates a text from one human language to another) is a direct application example. At the opposite end of the spectrum, part-of-speech tagging which is used to identify the grammatical role of each word in a given sentence or phrase, is considered as a sub-task of many applications such as question answering (determining the answer of a given question) or sentiment analysis [12].

2.4 Classification and regression

In supervised learning given a training set of N examples, we can present each example as a pair of input-output like $(x_1, y_1), (x_2, y_2)\dots(x_N, y_N)$ where y is an unknown function of x ($y = f(x)$). In this case, x and y can be any value and need not be only numbers.

Depending on the type of y there are two different prediction functions: the classification, and the regression.

If y consists of a finite set of discrete numbers like *{1, 2,…,m}* where *m>=2,* or Boolean values (i.e. *true* or *false)* the learning function is named *classification* [8]. If *m=2* or limited to Boolean values then the classification is called *Boolean* or *binary*, otherwise is called *multiclass* [13].

In the case that y is a set of continuous numbers such as weather temperatures or real numbers, the function is called *regression*. Regression can be linear or polynomial depending on $f(x)$ [13].

Sentiment analysis is an example of classification because we have discrete values as output. It can be a binary classification if our outputs are just limited to positive and negative classes, or can be triple classification if there is neutral class as well.

2.5 Sentiment Analysis

Generally, textual data can be categorized into two main types: facts and sentiments. Facts are objective statements about persons or/and actions and their properties. For example the sentence "*I bought an iPhone a few days ago.*" is a factual sentence. Sentimental texts are mostly subjective statements that express people's feelings, opinions or assessments toward persons, actions or organizations and their properties[3] such as this sentence "*I really like iPhone!*". The meaning of sentiment is comprehensive, ranging from emotions to personal experience and can be positive or negative. If the positivity sense of a statement is equal to its negativity, then it is considered as neutral text. The general feeling of a document, a sentence or a word is called *polarity*.

Nowadays, the dramatic growth of user-generated contents such as Internet forums, blogs, discussion groups, and social media networks change the people's manner to express their opinions and experience about products or any other subjects by importing them on the Web. This on-line vast pool of data provides a valuable and a measurable source of information that can be used by any industry and business. Presently a person or an organization is no longer limited to ask someone or conduct a survey to collect the sentiment information. Such a repository of available data gives users an opportunity to exploit useful information, and can lead them to make better decisions [3].

However, finding and monitoring sentiment sources on the Web is not an easy task because there are many different sources of opinions, and each source may have a large number of opinionated texts. Obviously, finding the most relevant sources, extract related phrases or sentences with sentiment, summarize and organize them into appropriate forms are difficult tasks for humans. Moreover, opinions are expressed in different ways, formal or informal. Thus, we need automated sentiment identification and summarization systems. *Sentiment analysis* has arisen for answering these needs [3].

Sentiment analysis (also known as opinion mining, review mining, appraisal extraction, subjectivity analysis and polarity classification [2]) is an application of natural language processing (NLP) [12], computational linguistic [14] and text analytic [15]. It is made up of several computational techniques such as detecting, extracting and classifying the sentimental statements expressed towards different events, entities and etc. [16]. In sentiment analysis, researchers attempt to give the ability to computers to determine whether a particular text involves positive or negative opinions. For this purpose, there are several tasks to address this goal as presented in section 2.5.3.

It is important for most organizations to know how their customers feel about their products and services and what are their complaints or issues. They look for audiences' suggestions and opinions. They can also find out what people think about their competitions. Sentiment analysis also is a powerful benchmark of the marketing programs that enables managers to track changes in overall customers' feeling over time [17].

In general, sentiment analysis contains of four major research areas: text subjectivity classification, sentiment classification including word-level classification or document-level classification, opinion extraction, and opinion summarization. In this thesis, our focus is on document-level sentiment classification.

2.5.1 Basic concepts

In sentiment analysis, like all scientific subjects, there are some basic definitions that we express some as following:

Object: An object *O* can be a service, a product, an individual, an organization, an event or any other topic. Each object *O* is represented with a pair as *(T, A)*, where *T* is a set of hierarchical components (or parts), and *A* is a set of attributes (or component properties). Each set *T* may consist of its own subset of sub-components which each may be accompanied with some attributes. For example, a particular brand of camera is an object while its different parts such as lens or battery are its components. The camera itself, as an object, has some attributes such as the picture quality or size. Also its components can have one or more attributes. For example, consider properties such as zoom or focus for the lens (see Figure 2.1). As illustrated in Figure 2.1, an object and its components and sub-components can be represented as a tree while the object itself is the root, and a component or a sub-component is a non-root node. Each node associates with a set of attributes. Each link between nodes shows a relation. Based on this definition, an opinion can be expressed on any node or any attribute [3].

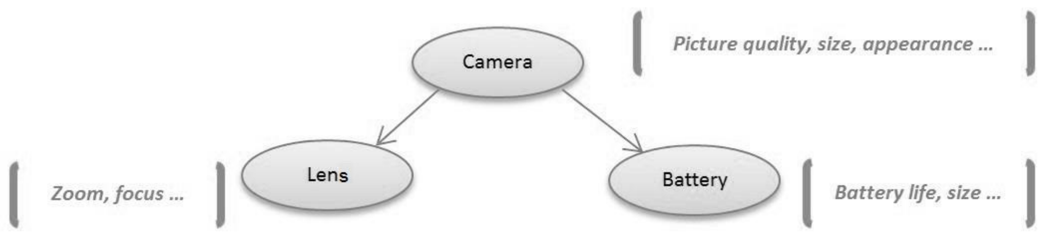


Figure 2.1: Example of an object and its components

In practice, hierarchical representation of opinions on an object or its components or/and attributes are complex to understand for ordinary users. Thus, a flatten architecture is considered for an object and used the term *feature* instead of both components and attributes [5].

Feature: A feature can represent a component, a sub-component or an attribute of an object. There are two ways to express a feature: using an explicit expression or an implicit description. A feature *f* can be explicit if *f* or any of its synonyms appears in an opinionated sentence *s*. For example, "*The battery life is too short*". The "*battery life*" is an explicit feature. In contrast, a feature *f* can be implicit if neither *f* nor any of its synonyms come into a subjective sentence *s*. For instance, "*The battery is large*". Here "*large*" points out to the size of the battery as an implicit feature. The adjective "*large"* that indicates to the battery size called feature indicator [3].

Opinion holder: Also known as *opinion source* is a person or an organization that holds the opinions toward an object. In review sites and blogs, writers of the posts are opinion holder (e.g. "*Jack told that this is a good apartment*", Jack is the opinion holder) [3].

Opinionated document: A group of consecutive sentences that can be either positive or negative is called an opinionated document or review. We can present a document as $d = \{s_1, s_2, s_3 ...\}$ where s_i is a sentence. The document *d* can consist of sentiment expressions on a set of objects like $\{o_1, o_2, o_3 ...\}$ that comes from opinion holders $\{h_1, h_2, h_3 ...\}$, where each o_j involves a set of features $\{f_{j1}, f_{j2}, f_{j3} ...\}$ [3].

Polarity or **orientation**: It indicates whether an opinion is positive, negative or neutral [3].

Opinion: An opinion is a positive or negative view, emotion, feeling, sentiment or appraisal toward an object *O* itself, or on its features $\{f_1, f_2, f_3 ...\}$ which is expressed by opinion holder *h* in a subjective document *d* (see Figure 2.2). Generally, there are two types of an opinion: direct opinion and comparative opinion. A direct opinion is expressed on one object and comes from one opinion holder such as "*The iPhone is a nice phone*". A comparative opinion compares two or more objects together. This comparison can be about their similarities or their differences. For example, "*Camera X is better than camera Y.*" [3].

Sentiment: Sentiments describe emotions, feeling, judgements, ideas and opinions toward an object. In sentiment analysis the word "sentiment" is used instead of word "opinion" [3].

Opinion words and **phrases**: Opinion words are terms that are specifically used to explain a positive or negative sentiment. For instance, *good*, *beautiful*, and *kind* are positive opinion words while *bad*, *poor*, and *horrible* are negative opinion words. These words can have different grammatical role such as adjective (e.g. *bad*), adverb (e.g. *kindly*), noun (e.g. *junk*), and verb (e.g. *like* or *dislike*). Apart from opinion words, there are opinion phrases and idioms. For example, consider the phrase "*cost someone an arm and a leg*". Opinion words and phrases play an important role in sentiment classification [3].

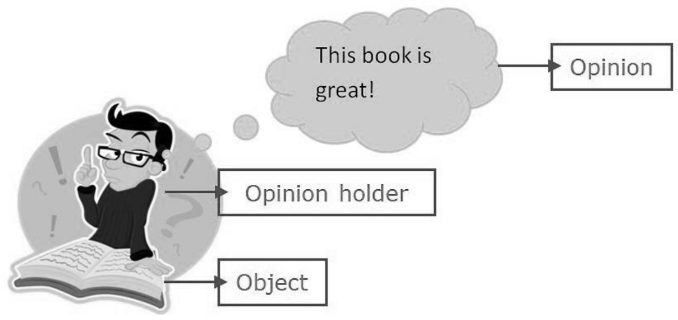


Figure 2.2: An example of an object and its opinion holder

Term: A term can be an individual word such as "*beautiful*" or word n-grams (a sequence of *n* contiguous words) like "*beautiful house*" [3].

Part-of-speech (POS) tagging: POS tagging is the process of assigning a syntactic class or grammatical role (such as noun, verb, adjective, etc.) to all words of a statement [12].

2.5.2 Sentiment analysis application areas

In addition to the individuals and organizations, advert placement systems are another sector that uses sentiment analysis. An advertisement is placed in a user-generated content when one praises some products, or another ad from a competitor is put if one criticizes some items [18].

Another application area is opinion search and retrieval that provides the general opinions toward a specific subjects [18]. For example, in politic politicians need to know voters' views and the voters are interested to know politicians' point of views, and who else supports them. Also it is useful in social media networks when one wants to find like-minded individuals or communities [3].

Conducting a survey is an old fashion to gather information, i.e. quality control, for market researches. Using sentiment analysis tools can make it much easier and more accurate for trend watchers and intelligent marketing [19][2].

Recommendation system is another area that SA applications can play very serious role since they should only recommend the objects which are received positive remarks [2].

Sentiment analysis can also be used to detect the “flames” (antagonistic phrases) on social media networks or in emails. Automatic and quick monitoring forums and news web sites to discover the flames, is an important task area in SA [2][19].

Opinion spam identification, also called bogus or fake opinions, is another application of SA. Generally, opinion spam is an undeserving positive or negative remark toward an object in order to, respectively, promote the object or damage its reputation for misleading the readers or sentiment mining systems [2]. Since email and Web spam are quite similar, opinion spam detection can be consider as a classification problem to classify contents into spam and non-spam. The importance of this problem will become more essential when individuals and organizations using the Internet dramatically as a source of opinions [2].

The other notable areas for the sentiment analysis are question answering, following the opinion timeline in online newsgroups or forums, text semantic analysis, and automatic text-to-speech synthesis [2].

2.5.3 Sentiment analysis tasks

Watching "*How the people think?*" leads the governments and companies to understand "*What the people need*" and it always helps them to improve the quality of their products and services [20]. These improvements impact directly on people’s life style. Today, there are several sources of people’s views on the Web, and there are several techniques and methods to understand, categorize, and analyses the opinions.

These techniques lead organizations and individuals to gain the necessary information easily and rapidly. Some of the important tasks in sentiment analysis are presented in the following.

Subjectivity classification

The goal of this task is to identify whether a sentence is subjective (opinionated) or objective (factual). This task attempts to categorize the sentences into two classes (or categories) under the name of subjective class and objective class. For example, "*The weather is cold.*" is determined as objective sentence, and "*I like cold weather.*" is identified as a subjective sentence [3].

Opinion extraction

This approach detects the opinions that are embedded in sentences or documents [20]. An opinionated sentence is the smallest complete unit that sentiments can be extracted from. Thus, to extract opinions and determine their polarities, the sentiment words, the opinion holders, and the contextual information are considered as indications (or clues) [20]. To date, in most existing works, whole documents are considered to identify the polarity. In contrast, opinion extraction focuses on sentences and product features [21]. Therefore sentiment words are identified in the first phase. Then the polarity of the sentences is detected. Finally the document orientation is identified [20].

Opinion tracking

The goal of this approach is to find how the public change their views or opinions over time. Tracking opinion systems can track the opinions in different sources according to various requests. These sources can be news articles, agencies, or any review sites. The results of this approach are very useful for companies, institutes, concerned public and especially for governments [20].

Opinion summarization

Unlike traditional summarization algorithms which assign confidence to important facts of documents and eliminate superfluous information, opinion summarization algorithms focus on two factors: sentiment degree and correlated events [11]. The four steps of opinion summarization consist of: detecting subject (major topic, product features ...), retrieving relevant sentences, identifying the opinion-oriented sentences, and summarizing [20]. The opinion summarization divides the opinionated documents in two categories: positive and negative.

Among these categories, summarizing can be applied as a brief or a detailed opinion summary. For brief summary, the documents with the greater number of negative or positive sentences are selected and their headlines used as overall summary (e.g. "*UK Government Stops Funding for Sheep Cloning Team*") [20].

In detailed summary, positive-topical or negative-topical sentences (opinionated sentences that are expressed on a considered subject) with higher sentiment degree (the degree that mentioned positive or negative rate of a sentence) are listed (e.g. "*Dolly the cloned sheep is only 3, but her genes are already showing signs of wear and she may be susceptible to premature aging and disease all because she was copied from a 6-year-old animal, Scottish researchers say.*") [20].

Sentiment classification

Sentiment classification is an area in sentiment analysis which determines the overall polarity of a text, or a sentence or a feature. It categorizes the opinionated statements into different classes by using a function that is called classifier. Since opinions are mostly expressed in subjective format, thus it is assumed that the opinionated documents involve subjective information. There are different classification methods. In binary classification, each opinion belongs to one of a positive or a negative class and it can be a feedback about a favorable or unfavorable product or a service. In triple-classification, an opinion can belong to a positive, a negative or a neutral class. It is necessary that we mention sentiment classification is not related to the topic a text is about, but is about the opinion that is expressed [1].

In this thesis, we study a binary sentiment classification as the main task and we compare the performance of two approaches.

2.6 Evaluation measures

Efficiency of sentiment analysis applications is calculated through experiments in the form of test data. For binary classifiers there are different metrics to measure the performance as following.

Before embarking, it is necessary to explain some parameters that are used in the following metrics. If a binary classifier is applied on a test data set, the results will be like Figure 2.3 (see Table 1 for more details).

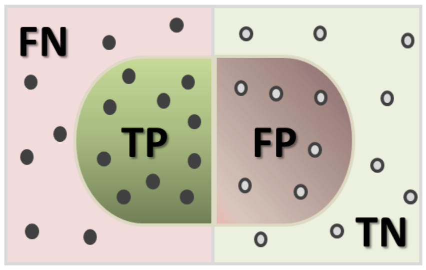


Figure 2.3: Precision and recall

The words positive and negative refer to the polarity of sentiment reviews, or more general, classifier's prediction or expectations. The terms true and false show whether the classifier prediction or expectation corresponds to real observations or not, thus [22]:

- True positive (TP) is the number of positive examples that are predicted as positive
- False positive (FP) is the number of negative examples that are labeled as positive
- True negative (TN) is the number of negative examples that are classified as negative
- False negative (FN)is the number of positive examples that are tagged as negative

Table 1: Contingency table

	Actual positive	**Actual negative**
Test outcome positive	True positive (TP)	False positive (FP)
Test outcome negative	False negative (FN)	True negative (TN)

2.6.1 Precision

In binary classification precision (also called positive predictive value) is the fraction of the number of true predicted documents to total number of predicted documents. In the other words precision is quotient of TP and total of TP and FP as following [22]:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Generally, high precision means that the classifier performs well to correctly classify or predict the true polarity of test data set.

2.6.2 Recall

Recall or sensitivity is defined as the number of true predicted document to all existing documents including true and false predicted as follow [22]:

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

High recall means that the classifier can predict correctly most of the documents.

2.6.3 F-measure

F-measure also called harmonic means is the combination of precision and recall that also is called balanced F-score and is calculated as follow [26]:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

In this formula, precision and recall are weighted evenly. If we want to emphasize on precision or recall, the F-measure is computed as bellow:

$$F_\beta = (1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * Precision + Recall} \tag{2.4}$$

In this case to put emphasis on precision more than recall, β is set to < 1.0 while, when recall is weighted higher than precision, $\beta > 1.0$ [26].

2.6.4 Accuracy

Another statistical metric of how well a binary classifier performs correctly on test data is accuracy. To calculate the accuracy both true positive and true negative values among the total number of examined cases are considered. The accuracy formula is [8]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

2.7 Sentiment analysis tools

Nowadays, there are various sentiment analysis tools available on the Web. Organizations choose different software depend on their level of needs and priorities. Some of these tools are stand-alone, and some are related to a specific social media network [2]. While a company starts searching for sentiment analysis tools, there are some points that needs to be considered to find an appropriate tool. One of the issues is data type. There are two ways to gather data: real-time and aggregated data. Some companies' customer services are interested to track all conversations and comments of their customers by pushing the alerts on certain keywords. For example, if a customer cannot find what he wants after waiting in a long line, the customer service can receive a notification as soon as he posts his disappointment note online. However, most of large businesses use aggregated data, and analyze them in a specific time period. For example once every quarter of a year by creating reports [17].

The next issue is data processing method. Most of the software relies on language algorithms and sentiment keywords. To show the analyzed results, different tools use different manner such as graphical gauge, score, scale or grade that help to better and quick understand the states [17]. Some tools even let companies see the top positive and top negative comments or posts [2].

It is worth noting that some of the open-source text analysis tools for information extraction or classification such as NLTK [2] (Natural Language Toolkit), WEKA data-mining workbench[3], RTM[4] (Text mining), RapidMiner[5], Gate[6] and StanfordCoreNLP[7] can be used for sentiment analysis[8]. In the following, we describe some of the most popular tools for SA.

[2] http://www.nltk.org/
[3] http://www.cs.waikato.ac.nz/ml/weka/
[4] https://cran.r-project.org/web/packages/tm/index.html
[5] https://rapidminer.com/
[6] https://gate.ac.uk/sentiment/
[7] http://nlp.stanford.edu/software/corenlp.shtml
[8] http://www.quora.com/What-are-the-most-powerful-open-source-sentiment-analysis-tools

Semantria: It is a solution for text and sentiment analysis for Excel. This tool is user friendly and yet powerful to visualize and monitor social networks with unstructured data like Twitter or Facebook. It can be used for survey as well[9].

Trackur: This tool is a monitoring software for social media networks that offers powerful measurement features for the company reputation. Trakur is designed to track customers' opinions and comments [2] that can help organizations to know how they can keep themselves on top[10].

Social Mention: This brand tracker is a free tool that is equivalent to Google Alert, but as a social media. It allows brand companies scan pre-defining relevant keywords in comments and mentions of videos, blogs, images, news, social media, events and even podcasts, and indicates if these posts are positive, negative or neutral. This application has the ability to send notifications when someone spots a mention about the brand or relevant persons, and can make daily summary alerts for particular day or over a period of time[11].

Google Alert: this application tracks user search queries and content marketing. Also it monitors the influencers, sentiments and competitors[12].

SAS Sentiment Analysis: This analyzer extracts automatically sentiments using NLP techniques. The software evaluates and manages sentiments and their changes over a period of time or in real time for a source company. This tool identifies and sends customers' feedback to the company in order to improve its competitive positions and define new targets[13].

[9] https://semantria.com/excel
[10] http://www.trackur.com/
[11] http://www.smartinsights.com/social-media-marketing/social-media-listening/managing-online-brand-sentiment/
[12] http://www.iprospect.com/en/ca/blog/10-sentiment-analysis-tools-track-social-marketing-success/
[13] http://www.sas.com/en_us/software/analytics/sentiment-analysis.html

2.8 Related works

In this section we present various techniques that are more used in sentiment analysis along some works done in this area. These works are consistent and varied representation of some of the existing solutions emerging in this field.

2.8.1 Bag of words

The back-of-words (BOW) is a simple method used in NLP and information retrieval (IR). In this technique, a text is considered as a bag or a multi-set of its words. Each word is assigned to its frequencies as its weight, disregarding grammar and word order in the text [12]. This model is generally used in text classification and is one of the most widely used approach for sentiment analysis [23].

2.8.2 Semantic relations

Semantic relation approaches in sentiment analysis investigate a new set of features branched from the semantic conceptual meaning of the entities in polar texts. The semantic features can include the general semantic concepts such as “person”, “country” or “company” that are corresponding with the entities like “Bill Gates”, “Canada” or “Apple”, or semantically relationships, i.e. synonym, antonyms or hyponyms, that are extracted from sentiment corpus. Since some entities will have more stable correlation with positive or negative sentiment, adding and using the semantic features are introduced. Having pre-knowledge about these correlations can be helpful to determine the orientation of similar or semantically relevant objects, and therefore the accuracy of the sentiment analysis systems is increased [24].

One notable work with this technique is the research that has been done by Saif et al. [24]. They use some semantic conceptual features and their incorporation into their sentiment analysis approach for Twitter. For each extracted object from tweets, they add a semantic concept with a given positive or negative orientation to measure the overall correlation of a general group of objects. For example the objects like “iPad” and “iPhone” are appearing in more positive tweets (70%) than negative ones (30%). So their classifier maps them to a semantic concept, e.g. “Apple product”, with positive polarity. The results show an increment of F-measure score (see section 2.6.3) for recognizing both positive and negative tweets over the baseline approach [24].

Another example of semantic approach in sentiment analysis is the work of Andreevskais and Bergler [25]. They propose an algorithm called STEP (Semantic Tag Extraction Program) that uses a small set of seed words with known positive, negative or neutral polarity, and extends the set by adding their synonyms, antonyms and hyponyms extracted from WordNet dictionary[14]. Then the algorithm identifies words that contain the extended seed list in their definitions by going through all WordNet glosses. These head words are added to the seed list based on their corresponding semantic orientations [25][2].

2.8.3 Lexicon-based classification

A sentiment lexicon is a list of opinion words and phrases associated with their polarity, e.g. (“*good*”, positive), (“*bad*”, negative), or (“*high quality*”, positive) etc. In the recent decades, a lot of studies have investigated how to build sentiment lexicons automatically. Most of these researches have relied on unsupervised or semi-supervised learning approaches. So far, two approaches are more proposed to address this problem: the thesaurus-based methods and the corpus-based fashion. The first method uses the synonyms or glosses of a thesaurus in order to determine the orientation of words like WordNet lexicon. The second approach uses the concept of co-occurrence in a raw corpus which is based on this hypothesis that polar phrases usually carrying the same polarity when co-occur with each other. In this method, a small set of polar phrases, as seeds, are prepared, so new polar phrases are found based on the strength of co-occurrence with these seeds [26] [27]. An instance of this method is MSOL lexicon (see section 3.5.2).

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

The lexicon-based model for SA relies on using external lexical resources that consist of words, and/or phrases along corresponded polarity categories or scores. In fact, this classification method maps a word of a text to a positive, negative or neutral category or a real numerical sentiment score. To obtain the overall orientation of the text, there are several methods including aggregation all achieved scores while the highest amount define the final polarity of the text [28]. There are different sentimental lexicons available on the Web for academic research such as *SentiWordNet*, *MPQA*, *General Inquirer* and etc. (see section 3.5.2 for more information).

Chaumartin [29] designs and develops a rule-based algorithm to detect emotions and valence tagger in news headline using WordNet as a semantic lexicon, SentiWordNet and WordNet-Affect as sentiment lexical resources. The algorithm identifies polar words and annotates them by using a pre-defined list containing *surprise, joy, fear, disgust, anger* and *sadness* emotions. Then it classifies them into positive or negative categories.

Taboada et al. [30] propose a lexicon-based binary classifier to extract the semantic polarity from a text called *semantic orientation calculator* (SO-CAL). In their work, they create a sentiment lexicon that consists of words along their polarity and strength. In order to improve the performance of their classifier, this lexicon incorporates with negation keywords (e.g. *not*), intensifiers (e.g. *very*), and diminishers (e.g. *little*). The classifier captures expressed opinions toward a main subject matter. The SO-CAL approach shows a acceptable performance across different domains especially on movie reviews [30].

Musto et al. [28] study the role of four sentimental lexicons: SentiWordNet, MPQA, WordNet-Affect, and SenticNet, to compute the polarity of microblog posts on Twitter. Their results show the two first lexicons perform better than the two last lexicons.

2.8.4 Language model

The language model is a probability distribution that assigns a probability over sequences of words. In sentiment analysis, the language model is implemented by n-grams methods. In this classification manner, presence or frequency of the sequence words is used and converted to TF-IDF evaluation measure [31] [12].

Pang et al. [32] and Dave et al. [33] show that bi-grams and tri-grams perform better than unigram in sentiment analysis task.

Liu et al. [34] investigate new language model to handle sentiment classification for Twitter posts. Their model called *emoticon smoothed language model* (ESLAM) and uses unigram model to train their classifier on either labeled or noisy tweets. For labeled data, they concatenate all positive tweets together to form one synthetic positive document, and do same for all negative tweets. Then they train the classifier on two language models and use the emoticon unlabeled data for smoothing. To evaluate their approach, they use likelihood measure to rank the classes. The highest likelihood indicates the tweets' class [34].

2.8.5 Cross-domain approach

In different domains where sentiment is conveyed differently, classification methods require in-domain labeled data for training the classifiers [35]. To do this, sentiment analysis applications need collecting and compiling labeled data in order to train the classifiers for each target domain and it can be prohibitively expensive. In addition, the sentiment terms derived from classification methods are strongly dependent on the source domains, and thus not efficiently reusable on a different domain [36]. For example, "*unpredictable*" is used to express positive sentiment in the movie domain, whereas it has negative sense in the automobile domain [35][37]. Furthermore, in some domains some words have positive or negative orientations that have no senses in another domain. For instance, "*short battery life*" indicates negative sense in the electronic domain, while does not carry any sense (neutral orientation) in the book domain [35]. Lack of learning unseen sentiment words is the another reason that a classifier trained on particular domains might not well perform on different domains [35]. Therefore, domain dependency is an inherent problem of sentiment analysis [6].This downside of such models has encouraged researchers to find out more flexible manners that need the minimum requirements to in-domain labeled training data; *cross-domain approach*. Cross-domain approaches use out-of-domain data set to construct learning models (here supervised learning models) that accomplish well on different target domains.

One considerable approach in this context is cross-domain contextualized sentiment lexicons that are proposed by Gindel et al. [6]. They create a generic domain-independent sentiment dictionary that can work on different target domains without the need of specific training labeled data. For this purpose, they distinguish stable (domain-independent) from unstable (in-domain) contextualized sentiment terms. They apply this method on two knowledge based repositories: Amazon and TripAdvisor. They keep only stable sentimental terms. The results show higher accuracy compare to similar methods [6].

Bollegala et al. [38] use multiple source domains to create a semantic sensitive thesaurus for training a cross-domain binary classifier. The thesaurus terms are derived from two data sets: a labeled data set (from source domains), and an unlabeled data set (from source and target domains). These sets are extracted from Amazon user review repository. Their approach uses unigram (one word) and bigram (two consecutive words) as lexical elements. Since a word can appear in different forms, they use lemmatization technique to reduce the data sparseness. Each lexical element in the thesaurus contains a feature vector that represents its distribution in data set along a value that indicates its polarity.

2.8.6 Machine learning approaches

Out of five mentioned machine learning approaches (section 2.2), supervised learning techniques have been the most common approaches in sentiment analysis field. In contrast reinforcement learning methods have only been applied in recent researches. Among the different supervised learning approaches, the most predominant techniques are Support vector machine (SVM), Naïve Bayes classification, and maximum entropy classifier. Each one of these approaches can be incorporated with the models presented in section 2.8.1 to 2.8.5 [33].

SVM inherently is a kernel based method that produces a binary predictor to predict the polarity of a text in SA. It is a linear classifier in a feature space that looks for the maximal margin. The margin of SVM classifier is described as the minimal distance between a training example and the hyperplane associate to the classifier in the feature space as depicted in Figure 2.4 [8].

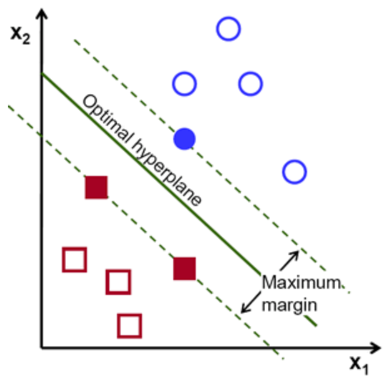


Figure 2.4: Support vector machines

The assumption in Naïve Bayes classification is that given a class variable, the presence or absent of a specific feature of the class is not related to the existence or non-existence of any other feature. The features are often indicated by a random variable of the method. Even if in particular case, the features are depend on each other or upon the presence of the other properties, a Naïve Bayes classifier assumes all of these features contribute independently. The researches show Naïve Bayes classifiers can be trained with high efficiency because of the accurate nature of the probability model. In practice, Naïve Bayes classification method can use maximum likelihood estimates, that mean the Naïve Bayes model can be applied without considering the Bayesian method probabilities [8].

The basic idea of maximum entropy (MaxEnt) is to select the most uniform models which can satisfy any given restrictions simultaneously with a few assumptions about training data to have a good grade of generalization on new data. MaxEnt as a probabilistic classifier belongs to exponential model classes. In contrast of Naïve Bayes classification method, MaxEnt assumes that features have conditional dependency together. Through all models that fit the training data, MaxEnt classifier chooses the one with largest entropy[15].

[15] http://blog.datumbox.com/machine-learning-tutorial-the-max-entropy-text-classifier/

Pang et al. [32] show for movie reviews classifying task at document level, SVM achieves the best performance while the worst performance is obtained by Naïve Bayes classifier in comparison with SVM and maximum entropy approaches.

Mullen and Collier [39] introduce an approach that assign semantic scores to words or phrases within movie reviews from www.imdb.com. Their approach uses these semantic values as features for a feature-based SVM classifier that is combined with another SVM classifier which is based on unigrams and lemmatized unigrams model. The outcomes reveal their approach outperforms the one presented in [32].

Pang and Lee [40] examine the relation between subjectivity identification and sentiment classification by applying the Naïve Bayes classifier on two different training date categories: 5000 movie reviews from www.rottentomatoes.com , and 5000 plot summary sentences from www.imdb.com which are most objective. They show although the subjectivity detection makes reviews shorter and cleaner, the obtained accuracy is still comparable with using original reviews [40].

Prabowo and Thelwall [41] combine a rule-based classifier with some machine learning approaches such as SVM into a new classifier. They evaluate their method on movie reviews, customer feedbacks, and MPQA corpus[16]. Their results show using a multiple classifier in a hybrid way can achieve better performance and efficiency in terms of F-measure compared with using any individual classifier.

Das and Chen[23] extract sentiments from stock market message boards using five different classifiers: Naïve Bayes classifier, vector distance classifier (VDC), discriminant-based classifier (DBC), adjective-adverb phrase classifier (AAPC) and Bayesian classifier (BC). These algorithms consist of different techniques such as part-of-speech tagging, and traditional statistical methods which categorize documents in three types as optimistic or bullish, pessimistic or bearish, and neutral. The neutral class can be either spam or

[16] http://mpqa.cs.pitt.edu/

messages which are neither optimistic nor pessimistic. These classifiers are based on a lexicon to determine the positive and negative words and are tuned on a small subset of pre-classified messages as training set in order to extract the rules. These rules apply on out-of-sample messages. The results show an improvement in accuracy level for their classifier specially when there are noisy stock massage boards [23].

Chapter 3

Case-based reasoning approach

In this chapter, we provide a case-based reasoning approach (CBR) on cross-domain for sentiment classification that is inspired from Ohana et al in [4] where out-of-domain labeled training data is used to predict the polarity of unseen documents. The majority of the work expressed in this chapter focuses on our project. Section 3.1 and 3.2 consist of the definition and a brief history of CBR. Sections 3.3 and 3.4 present CBR working cycle and its knowledge types. Following sections describe our case-based classifier and evaluating it on six different domains included 12000 reviews in details. This chapter finished by our experiment methodology and discussion sections.

3.1 Case-based reasoning

In everyday life, when people face a new problem, in order to achieve more complete understanding and to begin developing a solution strategy, they naturally recall the similar problems that have encountered before. For example, the lawyers use similar cases as legal precedents for creating and justifying arguments in new cases, in the trial, to advocate their clients. The act of developing a solution for a new problem based on the pre-existing solutions of similar prior problems is named *Case-based reasoning* (CBR). Case-based reasoning is a subfield of artificial intelligence and a method of computer reasoning (also called as automated reasoning)[17] which is the process of solving unseen problems based on a library of past cases (case base) rather than being grounded on classical rules [42]. A case base is a set of cases where each case includes a prior similar problem along its successful solution, and the annotations which describe how the solution is derived. Broadly construed, CBR means using old cases and solutions to meet new demands, to explain new situations or critique new solutions based on the reasoning from precedents (as lawyers do) [5].

3.2 Brief history

During the 70s and 80s, Rule-based Expert Systems (RBES) were one of the most dominating developments in AI researches. This approach is applied to problems that require extensive domain knowledge such as hardware troubleshooting, medical diagnosis, or geological explorations. In general, The RBES methods need a deep and explicit model of the problem domain knowledge that enable them to reason by using first principles [42]. RBES programmers must implement an explicit model of the domain whether the knowledge was shallow or deep. Hence, despite the success obtained with these approaches in many research areas, developers have met several serious problems as summarized by Schank [43], and [42] as following:

- Constructing an intended knowledge based method is difficult and time-consuming due to this fact that eliciting the expert knowledge is complex and requires taking a lot of time specially when the problem domains cover a wide range of science.
- The rule-based expert systems are incapable to deal with problems that are not explicitly covered by the appropriated rule bases. Generally, the RBES are fit to the problems with domains that have well formalized, stable and limited built-in knowledge.

[17] http://plato.stanford.edu/entries/reasoning-automated/

- Any additional options to the existing RBES programs need reprogramming, or creating new methods by programmers if no learning facility is considered in these systems.

As solution to these problems, researchers introduce CBR approach as an alternative reasoning paradigm and computational problem-solving method.

However, fundamentally, CBR is different in many respects from other major AI methods. Instead of depending on particularly general domain knowledge of a problem, or building associations between problem identifiers and conclusions, it is able to employ the specific knowledge of pre-existing similar experience (here training data or a case base) to create new solutions. Another important difference is that CBR is a progressive approach, and continuing learning. Whenever a new problem is solved, it is retrained as a new experience and CBR methods make it immediately available to reuse for the future problems[44]. Over the last few decades, CBR has grown rapidly as a field of widespread interest by drawing attention of computer scientists because [42]:

- Having an explicit domain model is not required in CBR, so extracting the solutions is a task of collecting case histories.
- In CBR, the implementation is reduced to the distinguishing important features that explain a case. Therefore it is an easier task than building an explicit model.
- Developing the database management techniques makes the maintenance of the large volume of cases easier.

3.3 The case-based working cycle

According to Aamodt and Plaza [45] CBR working cycle comprises the four steps as following:

1- **Retrieve:** given a new case, retrieve the most similar case(s) from the case base.
2- **Reuse:** map the solutions from the retrieved case(s) to the target problem. This may need to adapt the retrieved solutions to fit better the new problem.

3- **Revise:** an evaluation can be held either before or after applying the solution(s) to the target situation in order to examine the results. If unsatisfactory results are performed, more cases should be retrieved or the retrieved solutions must be adapted again.

4- **Retain:** If the proposed solutions lead to appropriate results, the problem along with its solutions and annotations are added to the case base.

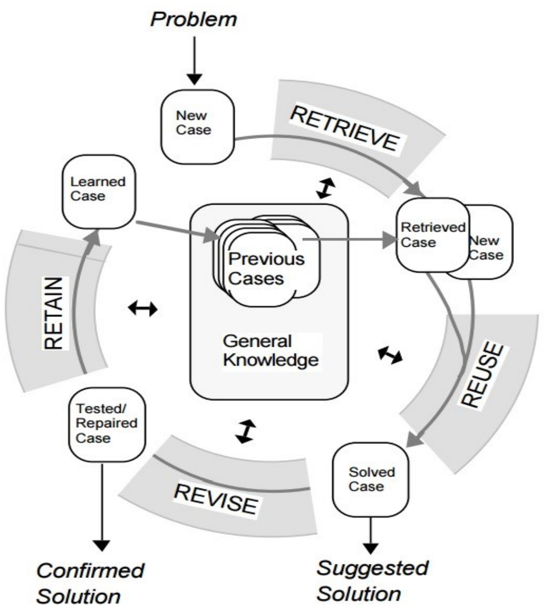


Figure 3.1: CBR cycle diagram[44]

These four steps are shown in Figure 3.1. A new problem is forwarded to the CBR algorithm, the closest cases to it are retrieved and solutions of these cases are reused to solve it. If the retrieved solutions are not fit to the problem, depending on the revised policy, they are adapted again, or more cases are retrieved. After finding and applying the appropriate solution, the new problem and its solution are retained for the future and the case base is updated [45].

3.4 CBR knowledge types

Using several types of knowledge about the problem domains are one of the CBR system goals. Richter [46] categorizes the knowledge container in CBR systems to four categories:

the vocabulary, similarity measures, cases themselves, and adaptation knowledge. The first three categories generally indicate common knowledge about the domain of a problem. The fourth category, the case base, commonly handles any exceptions from these knowledge containers [42].

Vocabulary contains the knowledge necessary to describe a case by choosing proper features. These features include certain problem parameters that are helpful to retrieve the most relevant cases to a new problem, and prevent to select too different cases that can lead to false solutions and accordingly reduce the CBR system performance. Therefore a precise comprehension of the problem domain is necessary[42].

Similarity measures include the knowledge of different available similarity measure methods and the knowledge of selecting the most efficient organization of case base and most proper case-retrieval methods. Since there are many similarity measure formulas, choosing the most appropriate ones for a given problem needs the efficient knowledge of the problem's domain. In particular, for automatic classification tasks, especially when there are complex structure cases, the similarity measure method plays an important role for further retrievals [42].

Adaptation knowledge, which is usually programmed with explicit rules, requires the knowledge about how the solutions are affected by the differences in problems. It includes the essential knowledge for implementing both phases of adaptation and evaluation of the CBR working cycle. Since acquiring this knowledge for many problem domains is the most difficult part, the adaptation phase is frequently left for the end users especially when the systems can make mistakes with damaging consequences. For the evaluation part, it is usually done before applying a new solution to solve a problem. The accuracy or correctness of a new solution has to be evaluated by estimating the importance of similarities and differences between the cases. Thus, this kind of knowledge can be considered as a refinement and development of the knowledge fitting the similarity measures containers [42].

Cases include the information about solved problem instances and the knowledge that the system earns during its working. In many CBR systems, what the cases will contain is generally specified by vocabularies. Sometimes a case base is initialized by cases that are carefully selected to cover the problem domain as much as possible. This is generally the case when essential adaptation phase is to be kept simple in order to generate manageable system maintenance. To have a wise policy for adding new cases to a case base, we need suitable heuristics with the ability to determine the useful cases to be retrained in the case base. Otherwise, by storing all solved problems as cases, a CBR system needs to have high memory requirements. It may impose long retrieval time, thus system's performance is reduced.

3.5 Case representation

For any CBR system, cases are the basis and contain pieces of knowledge representing experiences. In our work, cases are derived from a training set of labeled opinionated documents of user-generated products and film reviews. Each case encompasses of:

- *Case description*, which expresses the state of the world when a case happened [42]. Here a case description is a set of numerical features that are related to a document's characteristics. We use 17 different metrics which are all derived from a document, and stored into a 17-dimensional feature vector as described in Table 2. These features are independent of any domain specific aspects, such as special terms, which attempt to capture a document's indicators like sentences and word-level statistics, part of speech information, punctuations and etc. [4].
- *Case solution*, which determines the derived solutions for a problem [42]. In our work, case solution is a set of sentiment lexicons that made a correct prediction (predict reviews' true polarity). A sentiment lexicon is a language resource that contains a list of terms associated with their sentiment information such as "positive", "negative", or "neutral", or using numeric values, for example 1 or -1 and so on [4].
- *Outcome* is a description of the state of the world after a case happened. For sentiment classification, the outcome can be binary as positive or negative

orientation, or be triple when consider neutral polarity in addition when a case has equaled positive and negative aspects [42].

Although storing all mentioned information is useful to evaluate the outcome of proposed solutions, yet storing all available information makes the CBR systems more complex, and consecutively more difficult to use. Therefore, most of the CBR systems are limited to recording only case descriptions and case solutions [42].

3.5.1 Case description and feature selection for SA

A case description is essentially composed of information about the problem characteristics as necessary for an accurate and efficient case retrieval process. For the purpose of making a cross-domain sentiment classifier, we provide a set of 17 features that are extracted from a document. This set consists of two sub-sets: a set of document statistics parameters, and a set of document writing style elements. Table 2 presents them as following [4]:

- Total numbers of tokens
- Total numbers of words
- The numbers of sentences
- Average of the different sentence sizes to total sentence number
- Frequencies of seven different POS tags including verb, noun, adjective, adverb, conjunction, interjection, and punctuation.
- Ratio of space character to all non-space characters
- Ratio of stop words to the all other words in a document
- Average of syllables counts to the number of words
- Ratio of monosyllable words to the other words
- Ratio of the number of words to tokens
- Ratio of the words that are appearance once to the total words number

For calculating document statistics, we use Stanford Parser to extract the tokens, words, and sentences from a document. In general, a parser or syntactic analyzer is a tool to analyze the grammatical structure of sentences [47]. For example, it determines which word is the object of a particular verb, or which groups of words get together to make a phrase[18].

Table 2: Case description

Category	Metrics
Document statistics	Total numbers of words Total numbers of tokens Numbers of sentences Average sentence size Part of speech frequencies
Writing style	Spacing ratio Stop-words ratio Average syllable count Monosyllable ratio Word to token ratio Unique words ratio

Determining a sentence and distinguishing it from a phrase in a text is problematic task in natural language processing. Based on the Oxford dictionary, a sentence is a set of words that convey complete senses, has a main verb, and starts with a capital letter. While a phrase is a small set of words within a sentence that makes a meaningful unit. We apply Stanford Document Preprocessor that is a part of Stanford Parser, to each document in order to extract their sentences.

After mining the sentences, we use a tokenizer to create a list of tokens. A tokenizer or lexer is a program or function that converts a sequence of characters into a sequence of tokens such that makes a meaningful character strings [12]. In many languages including English, the whitespace and punctuation are used as word or token delimiter and most of the tokenizers use them to split up the strings into tokens.

After extracting tokens, we apply a stemmer, to have the list of words. A stemmer is a program that removes the inflectional or sometimes derivational or endings from words and

[18] http://nlp.stanford.edu/software/lex-parser.shtml#About

converts them to their base or root form, or more formally their word stem. For example the stem of the verb "*goes*" is "*go*" or the root of a plural term like "*children*" is "*child*". The most popular stemmers are Porter stemmer [48] and Snowball stemmer [49]. We apply both and compare the results in section 3.8. To improve the accuracy of our classifier, we use Stanford POS Tagger to mark the grammatical role of each word in documents [4]. Generally, part-of speech tagging (POS tagging or POST) is the process of mapping a word to a particular part of speech category, e.g. adjective or verb, based on both its definition as well as its relationship with other related words in a sentence or a phrase. Part-of-speech tagging is an ambiguous process especially as a task in NLP, because some words can have more than one part of speech in different contexts, and moreover, some part of speech categories are very complex to determine even for human annotators. For example, in the following sentence "*dogs*" is not a plural noun, but is a verb:

Original sentence: "*The sailor dogs the hatch*." [19]

Analysis result: *The/DT sailor/NN dogs/VBZ the/DT hatch/NN. /.*

Additionally, certain opinionated lexicons keep the terms along their part of speech. Therefore, using a part-of-speech tagger in the pre-processing step can help to increase the accuracy of lexicon queries [4].

In order to extract the writing style metrics of a document, the ratio of spacing is the rate of whitespaces to all characters. The stop-words ratio is computed by rating their frequency to all word counts. Stop-words are common function words that tend to bear little meaning such as "*the*" and "*at*". In our application, we decided to filter them out. From among of different available stop-words lists on the Web, we use SMART system [50]. For finding the average syllable count and monosyllable words we use the English Syllable Counter from MorphAdorner project[20]. The word to token ratio is calculated by dividing the word

[19] http://en.wikipedia.org/wiki/Part-of-speech_tagging
[20] http://morphadorner.northwestern.edu/

count to total token count. The ratio of unique word is the rate of words that are appeared once to the total number of words of a document.

All the case features are numeric and normalized according to the min-max normalization method. Min-max normalization is a linear transformation on the real data values for scaling them to fit in a specific range, frequently between 0 and 1, based upon the maximum and minimum value of that data. Suppose max_f and min_f are respectively the maximum and minimum values for the feature f of a case description. Min-max normalization transforms value v of f to v' in the range of $[new\_min_f, new\_max_f]$ as following [51]:

$$v' = \left(\frac{v - min_f}{max_f - min_f}\right) * \left(new\_max_f - new\_min_f\right) + new\_min_f \qquad \textbf{(3.1)}$$

3.5.2 Case solution and lexicon selection for SA

The case solutions in a CBR system can be either atomic or component. Typically atomic solutions are used for diagnosis systems or classifiers, while compound solutions are more useful in planning or designing systems [42].

As mentioned before, we use sentiment lexicons to compute the general polarity of documents. There are large numbers of sentiment lexicons available in literature. In our work, we use the following five sentiment lexicons: General Inquirer lexicon, SentiWordNet, Subjectivity clues lexicon, MSOL, and NRC emotion lexicon.

General Inquirer lexicon (GI) is often considered as a gold standard for English language researches. It is a manually annotated lexicon that consists of four sources[21]:

[21] http://www.wjh.harvard.edu/~inquirer/

- The Harvard IV-4 dictionary developed in 1998 including almost all words,
- The Lasswell dictionary developed by Namenwirth and Weber [52],
- The "marker" categories mainly expanded as a resource for disambiguation[22], but also is accessible for users, and
- Several categories produced based on the work of Semin and Fiedler [53].

The General Inquirer is primarily a mapping tool and contains 182 categories in all. Each category composes of a list of words and their senses. Words in GI are indexed by "LW" (for "Laswell") or "H4" (for Harvard IV-4) to distinguish similar words that come from each dictionary. The name of each category is unique and case sensitive. For example, "*Positiv*" caption is used for positive sentiment words, and "*Negativ*" caption is for negative words. In general, GI includes 1915 positive and 2291 negative words [54]. The rest of words are neutral[23].

SentiWordNet 3.0 (SWN) is an automatically built lexicon and improved version of SWN 1.0 [55]. SWN is made in 2006 and is based on WordNet lexicon. It uses WordNet terms relationships and glosses information by applying a semi-supervised learning algorithm [56]. In SWN, each term or synset (synonym set) is associated with three sentiment scores: positivity, negativity, objectivity values along with its part of speech tags. For instance, the sentiment score for the word "*good*" as an adjective is *0.481441398*, while its scores as noun and adverb, respectively, are *0.377640037* and *0.75*[24].

Subjectivity clues lexicon (Clues)[25] is distributed under the terms of General Public License (GNU) and used in [57]. This lexicon includes list of subjectivity clues (opinionated words) along their part of speech tag as well as their prior-polarity. Clues is compiled from several sources such as manually developed resources, and automatically approaches using both

[22] http://www.wjh.harvard.edu/~inquirer/kellystone.htm
[23] http://www.wjh.harvard.edu/~inquirer/homecat.htm
[24] http://sentiwordnet.isti.cnr.it/
[25] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

annotated and un-annotated terms. The majority part of Clues is rolled up as a part of the work of Riloff and Wiebe in [58].

MSOL lexicon is a semantic orientation lexicon which composes both individual words and multi-word expressions along with their polarity. MSOL is a thesaurus-based lexicon based on a development of the Macquarie Thesaurus [59] that includes about 100,000 unique words and phrases. To generate this lexicon, Mohammad et al [60] considered a seed set of positive and negative words as basis, and use Roget-like thesaurus to label the words synonymous with the positive seeds as “positive” and words synonymous with the negative seeds as “negative”.

NRC version 0.92, copyright 2011 National Research Council Canada (NRC), is an emotional lexicon. It contains a list of polarized words that each one is associated with eight emotions included *anger*, *fear*, *anticipation*, *trust*, *surprise*, *sadness*, *joy*, and *disgust*, and as well as two senses: positive and negative. This lexicon is created by manually word annotation through Mechanical Turk of Amazon web site [61].

It is worth noting that Ohana et al. [4] use the first four lexicons and Moby lexicon [62] as the fifth lexicon. Since we could not access to Moby lexicon, replace it by NRC emotion lexicon.

For all lexicons except SentiWordNet, we assign value 1 to a positive word, and -1 to a negative word. For neutral word, we consider value 0. To create a case solution, our classifier predicts the polarity of a document by taking a sentiment lexicon and a documents’ words-list as input. For each word in the words-list, the classifier makes a query in the lexicon in order to find the corresponding score. Then all scores are aggregated together to find the total score of the document. If the classifier can predict the document’s polarity correctly, the lexicon is added to the case solution as appropriate lexicon. This process is repeated for all lexicons one by one.

3.6 Making and populating the case base for SA

In this section we explain how our algorithm works. Our approach includes two phases: a training phase, and a test phase.

In training phase, we build a case base included a set of cases. Each case refers to a document or a review. Algorithm 1 describes the steps of making and populating a case base. At first, the case base is empty. For each document in the training set, a words-list is extracted. The words-list consists of a list of words (stemmed tokens) along with their counts (the number of their presence in the document), and their part of speech tag.

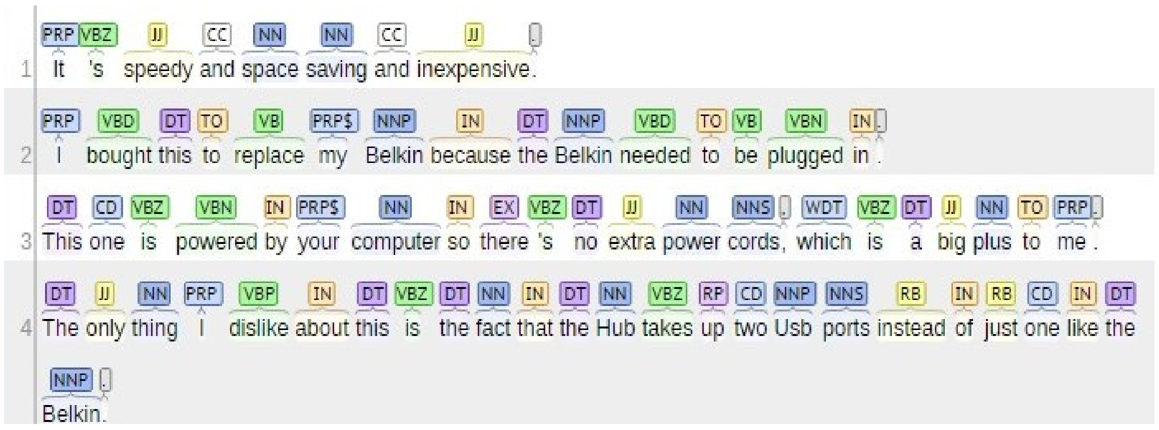
To make a prediction, our program uses a supervised binary classifier as main function. In this regard, the document words-list and one of the sentiment lexicons are given to the classifier as input. For each word in the words-list, our classifier makes a query in the lexicon in order to find its relevant sentiment value. If the lexicon supports the part-of-speech tagging, the tag of the word is considered as well in the querying process. Since all words in a document do not carry sentiment senses, for example stop-words or proper nouns, we consider only words which are tagged as adjectives and verbs [4].

To calculate the total score of a document, the classifier associates each word with a score that is computed by multiplying its frequency and its sentiment value. By convention, the overall sentiment scores of a document are stored in a pair of values (a total positive value, a total negative value). Each total value is generated by summations all scores of each polar word categories separately (means all positive words together and all negative words together) [4]. The general polarity of the document is determined based on the higher value of the mentioned pair. If the predicted polarity matches with the document label, the used sentiment lexicon is added to the case solution. This phase is repeated for all available lexicons. If no lexicon can make correct prediction for a document, the document is discarded. Otherwise, a new case is created where its case solution is a set of all lexicons that yielded a correct prediction, and its case description is extracted from the document as mentioned in section 3.5.1 [4].

To better illustrate the steps of the algorithm we use an example. Consider a sample review for an electronic device as following:

"*It's speedy and space saving and inexpensive. I bought this to replace my Belkin because the Belkin needed to be plugged in. This one is powered by your computer so there's no extra power cords, which is a big plus to me. The only thing I dislike about this is the fact that the Hub takes up two Usb ports instead of just one like the Belkin.*"

The above document is a positive review on a specific Hub. Our algorithm first sends the document to Stanford POS tagger for some pre-processing steps like: sentence segmentation, tokenization all sentences and tagged tokens (see Figure 3.2). In order to extract the document words-list our program builds an array included of all stemmed tokens (the words) along with their frequencies and part of speech tags (see Table 3).



```
  PRP VBZ JJ CC NN NN CC JJ .
1 It 's speedy and space saving and inexpensive .

  PRP VBD DT TO VB PRP$ NNP IN DT NNP VBD TO VB VBN IN .
2 I bought this to replace my Belkin because the Belkin needed to be plugged in .

  DT CD VBZ VBN IN PRP$ NN IN EX VBZ DT JJ NN NNS , WDT VBZ DT JJ NN TO PRP .
3 This one is powered by your computer so there 's no extra power cords , which is a big plus to me .

  DT JJ NN PRP VBP IN DT VBZ DT NN IN DT NN VBZ RP CD NNP NNS RB IN RB CD IN DT
4 The only thing I dislike about this is the fact that the Hub takes up two Usb ports instead of just one like the
  NNP .
  Belkin.
```

Figure 3.2: The pre-processing step of a document using Stanford POS tagger

Table 3: The words-list

Tokens	Word	#Frequency	Tag
bought	buy	1	VBD
speedy	speedy	1	JJ
to	to	3	TO
‘s	is	5	VBZ
…	…	…	…

Algorithm 1: Population of a case base

Input:

- **D**: The training data set
- **L**: The set of all available sentiment lexicons
- **wl**: An extracted words-list of a document
- **F(L,d)**: The supervised binary classifier

Output:

- **CB**: The case base
    - **CS**: A case solution set
    - **CD**: A case description vector
    - **C**: A new case

Process:

```
CB ← {}
for all document d in D do
    CS ← {}
    wl ← extracted words-list of d
    for all lexicon $l_i$ in L do
        predict the general polarity of d using F($l_i$, wl)
        If the predicted orientation is correct then
            CS ←CS ∪ $l_i$
        end if
    end for
    If CS <> {} then
        CD ← extract the case description of d
        create a new case C←(CS, CD)
        CB ←CB ∪ C
    else
        discard d
    end if
    CD ← {}
    wl ← {}
end for
```

After building the words-list, all words which are tagged as verbs and adjectives are sent to a sentiment lexicon in order to extract their polarity scores. The total scores of all positive and negative words are calculated. Based on the higher score, our classifier predicts the document polarity. Since we consider binary classifier, if the score equals to 0 or greater the orientation of the document is positive, otherwise is negative. If the lexicon can correctly predict the polarity of the document, it will be added to the case solution. After repeating this process for all lexicons if case solution is not empty, the case description of the document is extracted and computed. A new case that consists of the case description and the case solution is created and added to the case base. If no lexicon can predict true polarity of the document, it will be discarded.

Our program goes through all training data sets and builds the case base. In test phase, this case base is reused to predict the orientation of unseen documents. One of the strength points of this method is the ability for further expansion by considering more out-of-domain data sets as well as adding the additional sentiment lexicons. Obviously having more cases and polar lexicons can help to increase the performance of the approach [4].

3.7 Cases retrieval and reused for SA

Once the case base is generated, our CBR sentiment classifier can use it to predict the polarity of unseen documents in the test phase.

As already stated, in CBR systems, prior cases are reused to solve a new problem. In this work, to make a prediction for a new review or document, those cases that are similar to the new review are retrieved in order to reuse their solutions for calculating the polar values of the new review. We first retrieve the *k* closest cases to the unseen review based on their case descriptions, and then reuse their case solutions (lexicons) to predict the polarity of the new review.

In order to retrieve appropriate cases, one algorithm that performs well is using the k-Nearest Neighbor for some given numbers *k*. K-Nearest Neighbor (KNN) is one of the fundamental classification methods that is principally used when the classifier has little or no antecedent knowledge about the distribution of the data. KNN is a non-parametric approach. Non-parametric methods rely on the assumptions that the data are not inferred from a given probability distribution [63]. This method performs well in discriminate analysis specially when estimating the reliable parametric of probability densities are unknown or difficult to determine. Given a new document, KNN algorithm retrieves the k cases which have the closest case descriptions to the new document's case description. To measure the closeness of two cases, a distance metric is required. KNN method commonly uses Euclidean distance to compute the distance between the specific training samples and a test sample. Let $X(x_1, x_2, \dots, x_n)$ be a test example, and $Y(y_1, y_2, \dots, y_n)$ be a training sample. According to the Euclidean distance formula, the distance between these two samples in R^n is given by:

$$d = |X - Y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \textbf{(3.2)}$$

As presented in section 3.5.1, each case description is a 17-dimensional vector of numerical features that represent the document statistics and writing styles parameters. To scale these features in the range of $[0,1]$ we use the max-min normalization method.

Given an unseen document, in the test phase, first its case description is extracted and normalized. In the second stage, the distance between its case description and all cases in the case base is computed, and then case base is ordered in descending order based on these distances. The first k cases are retrieved as follows:

- Where *k=1*: the classifier uses all the lexicons obtained from the first case to predict the orientation of the test sample [4].
- Where *k>1*: a ranking method based on the frequency of the occurrence of lexicons out of the k retrieved cases is established. The lexicon(s) with most frequency are selected and reused for the prediction [4].

Since a case solution may record multiple lexicons, the outcome of retrieval and ranking it may yield more than a single lexicon. To prevent the tied results, we individually compute the sentiment scores using each lexicon, and then aggregate all obtained positive and negative scores together in a pair. The higher score determines the final polarity of the document [4].

Table 4: An example of ranking 3 case solutions during case retrieval process for *k=3* [4]

Retrieved cases	Ranking results	Selected lexicons
Case solution of case A: $\{l_1, l_2, l_5\}$ Case solution of case B: $\{l_1, l_5\}$ Case solution of case C: $\{l_1, l_4, l_5\}$	l_1 (3) l_5 (3) l_2 (1) l_4 (1)	$\{l_1, l_5\}$

Table 4 shows the process of ranking lexicons of three selected case solutions. In this example, three cases (let called *A, B, C*) are retrieved as the closest cases to an unseen hypothetical document. In order to find the right solution, we rank the lexicons based on their frequency. Since the maximum frequency is 3 that belong to the lexicons l_1 and l_5, they are chosen as solutions for the document. The aggregation of the scores achieved by them determines the document polarity [4].

3.8 Evaluation results

We evaluate our approach on six different datasets of film and user-generated product reviews in plain text as following:

- The film reviews which are extracted from www.IMDB.com and used in the work of Pang and Lee in [40]. They have been written by 312 authors before 2002.

- The hotel reviews dataset that are selected from TripAdvisor web site[26] used by Baccianella et al in [64]. These reviews are about hotels in the towns of Pisa and Rome[27].
- The product reviews about apparel, book, electronic devices, and music that are chosen from www.Amazon.com used in [65] and [66].

Table 5: Customer reviews datasets

Dataset	**Number of reviews**	**#Average token size**	
		#Negative reviews	**#Positive reviews**
Apparel	2000	68.41	65.16
Book	2000	192.86	167.38
Electronics	2000	119.25	112.03
Hotel	2000	210.12	177.59
Movie	2000	721.34	803.18
Music	2000	140.26	153.26

Table 5 shows our datasets along with their sizes (average token counts). Each dataset includes 2000 reviews that included equal number of positive and negative documents. We decide to have equal size for all datasets in order to eliminate any possible impacts of one domain on the others. In contrast, Ohana et al [4] consider datasets with different numbers of document.

Our evaluation phase has two steps: a training step, and a test step. In training phase, our program creates six distinct case bases. Each case base contains five domains while one domain is held out. In the test step, these case bases are used to classify reviews of the hold out domains.

[26] http://www.tripadvisor.com/
[27] The reviews are available in http://patty.isti.cnr.it/~baccianella/reviewdata/

Our goal is to evaluate if the case-based approach on the cross-domain can work well in sentiment analysis by measuring the accuracy of the classifier.

3.8.1 Training phase

In the training phase, our classifier takes a lexicon and a document words-list as input and computes the polarity of the document by querying for sentimental values of its all adjectives and verbs. We use the Porter and the Snowball stemmers to stem the verbs. Based on our preliminary experiments, the Snowball stemmer leads to better results.

Table 6 shows the size of each case base with the percentage of its discarded documents ratio. Note that each case base name reflects the hold out domain. Each training set consists of 10,000 documents. However, each case base has a different size that is related to the number of those documents that are predicted correctly by one or more lexicons (the cases). Also the mentioned positive and negative percentages are related to each case base, not the training sets themselves.

Table 6: Case base size and ratio of the discarded cases

Holdout Domain	**Case base Size #**	**Positive%**	**Negative%**	**Discarded ratio %**
Apparel	9076	53.11%	46.89%	9.24%
Book	9183	52.28%	47.72%	8.17%
Electronics	9112	53.04%	46.96%	8.88%
Hotel	9147	52.21%	47.79%	8.53%
Movie	9099	53.19	46.81	9.01%
Music	9158	52.47%	47.53%	8.42%

As it is shown in Table 6, the ratio of predicted positive documents is more than negative documents in all domains. This could be ascribed to some difficulties in predicting the negative emotions such as using the negated phrases, more sarcasm and irony, emoticons, and punctuations (e.g. *?* and *!* can be used as unfavorable emotions) in negative reviews [4]. Moreover, Book domain has less discarded documents than other domains. It can be attributed to its authors. For example, these authors use more formal and structured writing style that can decrease the error ratio of extracting the emotions.

3.8.2 Test phase

To evaluate our classifier, we apply it to an unseen domain to predict the polarity of its documents. For a given test document, first, its case description is extracted and computed. By using Euclidean distance as the similarity measure, the *k* nearest case(s) to this document is retrieved. In the second step, based on the value of *k,* we use the lexicons which are obtained from the case solutions of the retrieved cases to predict the document polarity [4].

As earlier mentioned, when *k=1* we directly apply all the lexicons of the most nearest case to predict the polarity of an unseen document. The final score of the document, for all values of *k*, is calculated by aggregating all positive and negative scores which are derived from high ranked lexicons. The highest accuracy at *k=1* belongs to Hotel domain (see Table 7).

Table 7: Accuracy results at *k=1*

Domain	**#TP**	**#FN**	**#TN**	**#FP**	**Accuracy%**
Apparel	803	197	556	444	67.95%
Book	898	102	353	647	62.55%
Electronics	787	213	530	470	65.85%
Hotel	979	21	468	532	**72.35%**
Movie	746	254	683	317	71.45%
Music	863	137	419	581	64.10%

At *k= 3*, we extract three nearest cases to the unseen document based on its case description. Then we rank the lexicons that are derived from these cases' solutions. The lexicon(s) with higher frequency are selected to predict the polarity of unseen document. In this case, our program achieves better results for Movies domain (see Table 8).

Table 8: Accuracy results at *k=3*

Domain	#TP	#FN	#TN	#FP	Accuracy%
Apparel	852	148	505	495	67.85%
Book	881	119	364	636	62.25%
Electronics	849	151	422	578	63.55%
Hotel	993	7	391	609	69.2%
Movie	691	309	729	271	**71%**
Music	891	109	382	618	63.65%

And the results at *k = 5* are depicted in Table 9. Like at *k=1*, The Hotel has the higher accuracy than other domains.

Table 9: Accuracy results at *k=5*

Domain	#TP	#FN	#TN	#FP	Accuracy%
Apparel	781	219	488	512	63.45%
Book	879	121	361	639	62%
Electronics	805	195	429	571	61.7%
Hotel	981	19	402	598	**69.15%**
Movie	585	415	726	274	65.55%
Music	877	123	383	617	63%

3.9 Discussion

Our evaluation results show the accuracy is between 61% and 73%, which is acceptable comparing to similar approaches that have already mentioned in this thesis. For different values of *k* , our performance is almost better than the performance that Ohana et al have achieved in [4], although our datasets contain of 12,000 documents compare to 15,445 documents in their study, and with this in mind that we still have not implemented the negation detection. Our datasets consist of the reviews of different products that are not similar together. The average token size of each domain shows that Apparel domain includes shorter reviews while in Movie reviews authors precise their opinion more completely. Also, in Hotel domain reviewers separate pros and cons with individual words and phrases. These reasons can explain why Hotel and Movie domains achieve better accuracy than others. In contrast, in Music domain, writers use more informal writing style and some slang such as *"In fact, I guarantee you'll be knocked out"*.

In addition, we examine the influence of different values of *k* on our classifier accuracy. For this purpose, we apply odd values, from 1 to 11, for *k* and find out a slight decrement in accuracy when *k* increases. These trends are depicted in Figure 3.3. For all domains except for Book (a slight difference) the highest performance is at *k=1*. This means using more similar cases reduces the accuracy of the system. It can be attributed to missing some relevant lexicons during the ranking process. For instance, in Table 4, lexicon l_2 is not chosen as solution because of its low frequency despite it is an appropriate solution for the closest case to the hypothetical document (case *A*). Moreover, the ranking process, in most cases, can decrease the numbers of selected lexicons.

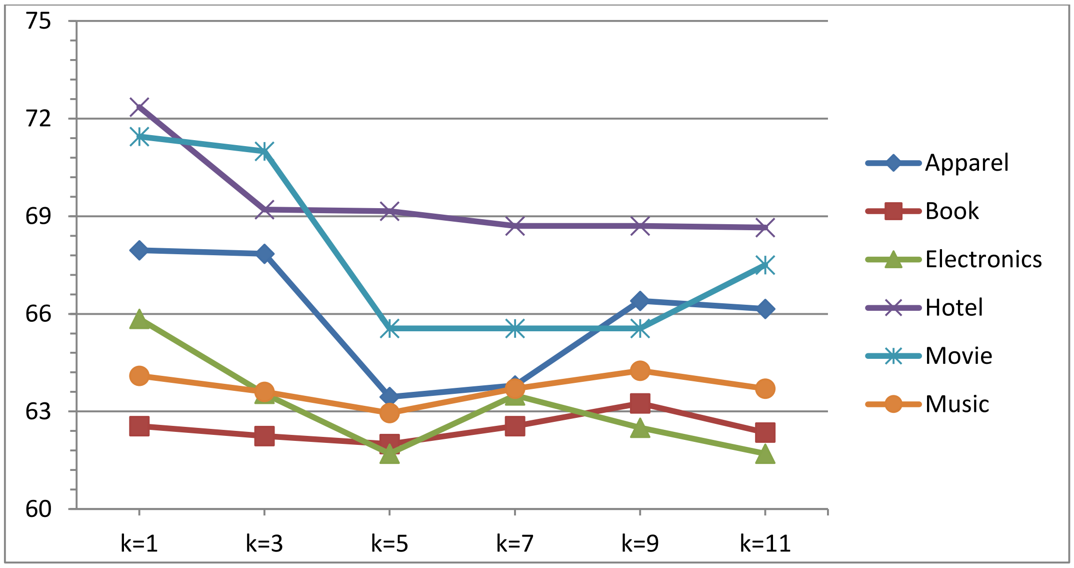


Figure 3.3: Accuracy for different values of *k*

Another issue that is worth to mention is the rate of false positive (FP) which is high in all domains. It can be addressed of using sarcasm, irony and more specifically negation structures. Therefore, we apply an algorithm to detect the negation in order to increase the performance of our classifier.

Chapter 4

Negation

In this chapter we propose an approach to find out the negation and its scope in a statement, and study its effect on our classifier accuracy. In following sections, we first explain the definition of negation and its scope in linguistic and natural language processing applications. In second section, we define the different negation keywords. The third section, introduces general rules for identifying the negation scopes. The following section presents some works that have been done on this concept and followed by some general remarks. In Section 6, we completely explain our approach. This chapter finished by providing results and discussion.

4.1 Definitions and main concepts

There are varied manners and several grammatical rules to express the same opinions or emotions that a review writer can use. One of the frequent ways to state an opinion is negation that completely changes the polarity of words and statements. Negation is one of the most common and yet complex issue in linguistic [67] and used in all languages [7]. There are different purposes to use negation including reversing the sentimental orientation of a statement (e.g. "*She does not like him"*), vouching on a negativity (e.g. "*She has not eaten nothing*"), and making weaker assertions (e.g. *"She is not amateur"*) [7]. Negation detection can help classifier to categorize more accurately a text, a sentence or a phrase as positive, negative or neutral by taking into account all relevant negation factors that influence on the polarity. Moreover, the presence of negation markers in a text does not mean that all the terms that convey opinion or emotion will be inverted. Thus identifying which elements in a sentence or a statement are affected by negation clues lead linguists to define the *scope* for negation keywords based on their part of speech or context (see section 4.3) [7]. Therefore, identifying the negation itself and more important, its scope, is one of the important task to find out the orientation of a text in SA. Since negation does not limit itself to "*not*", negation identification is not a simple and easy task. The linguistic patterns such as prefix (e.g. dis-, in-, un-, etc.) or suffix (e.g. –less), also other terms like; *no, never, nowhere, neither* and etc. increase the complexity of negation recognition. Another side of negation terms is contextual valence shifters that contain intensifier and diminisher words or phrases such as "*very*" or "*little*". These terms may not flip the words orientation but can increase or decrease the level of positivity or negativity of them [68].

Although negation detection can improve the performance of the sentiment classification, there are fewer efforts in this domain with respect to the general sentiment analysis task [68]. There are some reasons that has not paid enough attentions in this concepts such as low precision of negation words that involving in the reviews [69]. Therefore, the number of negated sentences encountered in texts is insignificant as compared to the amount of effort required to unravel the issues related to negation evaluations [68].

Since there are different negation terms, defining negation keywords and their type is a good starting point before embarking on the next sections.

4.2 Negation keywords

In general, there are six common typical categories for negation keywords as below [70]:

1. Auxiliary verbs such as *cannot*, *shouldn't*
2. Adjectives and adverbs like *impossible*, *nowhere* etc.
3. Nouns, e.g. *nothing*.
4. Conjunctions like *neither … nor* etc.
5. Negation keywords as "*no*", "*not*".
6. Prepositions for example "*without*".

Since we work with reviews, it is common that reviewers use ungrammatical structures and words in their writing. Thus considering these cases is important to compute the correct scope of a negation term in a sentence or a phrase. For instance, using "*ain't*" instead of "aren't" or "isn't" [70].

4.2.1 Complex keyword vs. sequence of keyword

The complex keyword in negation is a phrase (rather than a word) including the words that cannot express negation separately, e.g. "*not sure*" [70]. On the other hand, the semantic of the sub-components of the phrase are significantly different from the meaning of the whole phrase. Therefore, prepositions, determiners, adverbs and so on cannot be part of the complex keywords if the keywords can carry negation content on its own [50]. In contrast, the sequence of keywords is a group of words that can declare negation on their own such as "*neither…nor*". Thus, distinguishing the complex keywords from a sequence of clues should be accomplished carefully. Otherwise, finding the correct scope of negation would not be accurate [70].

4.2.2 Contextual valence shifters

In previous sections, we explain how negators can flip the positive valences to negative valences [71]. Generally, terms that can alter or modify the semantic orientation of polarized words are called "*contextual valence shifters*" [72]. Polanyi and Zeanen first postulate the importance of contextual valence shifters as contextual phenomena that change the prior polarity of a term in [71]. Beside of negators, there are three different categories of valence shifters as following [73]:

Intensifiers and diminishers: intensifiers and diminishers (also called amplifiers and downtoners [74]) are terms that increase and decrease, respectively, the degree of the semantic intensity of the expressed sentiment [73][72]. In terms of grammar, adjectives modify nouns whereas adverbs shift the polarity of verbs or adverbs [73]. For instance, in the sentence "*I am very excited to play guitar*", the phrase "*very excited*" is more positive than "*excited*" alone. In contrast, in the phrase "*little efficient*", the term "*little*" decreases the intensity of the adjective "*efficient*" [71].

Modals and conditional words: modal operators such as "*might*" or "*could*", and conditional words like "*if*" or "*unless*" indicate the possibility or necessity in a text [71]. Therefore, they should not be taken into account as altered factors to change the orientation of the following words, but the degree of the intensity. For example in the sentence "*This should be a good movie*" the term "*should*" expresses the expectation of the author and not necessarily about the quality of the movie itself [73].

Presuppositional items: words like "*even*" or "*barely*" that modify the base valence of evaluative words via their presuppositions are called *presuppositional items* [71], [73]. For instance, in the phrase "*barely sufficient*", the term "*barely*" changes the positive level of the word "*sufficient*" [71].

Kennedy et Inkpen use the contextual valence shifters in their term-counting method for sentiment classification of movie reviews and report the improvement in the accuracy of the classifier [72]. Both [71] and [65] have implemented the intensifiers and diminishers by adding and subtracting, respectively, one unit of the original value of the neighboring polarized words. For example, if a positive adverb has a polarity value of 2, an intensifier increases its value to 3 while a diminisher decreases it to 1 [75]. This method does not take into account any different between terms belonged to same subcategory. For example, "*extraordinarily*" and "*really*" as amplifiers equally augment the polarity value of neighboring terms while "*extraordinarily"* is a much stronger adverb than "*really*" [75]. Taboada et Voll [75] propose a method that associates different percentages to the valence shifters instead of adding or subtracting a fixed value as in [71] and [65]. For instance, they consider additional 50% positive value for "*extraordinarily*", and 15% positive value for "*really*".

4.3 The scope of negation

Scope in negation identification is a window between a negation term and following words or a punctuation [67]. By convention, this window can only include the words that are modified by a negation clue. As already stated, there are six different categories for negation keywords. Thus based on the structure of a sentences or a phrase, Konstantinova et al. in [70] propose the following methods to find the scope of a negation term:

- The scope of negation adjectives and adverbs generally begins right after the keywords and includes the rest of the sentence, phrase or clause. For example, in: "*It is impossible [to see the difference between the two models]*" the scope of adjective "*impossible*" starts from "*to*" and ends at the end of the sentence.
- In the case of verbal phrases including auxiliaries, the scope of negation starts immediately after the verbs and consists of all the rest of elements of the clause, containing relative or coordinate clauses, or the sentence, e.g. "*I didn't [realize so much of the book was focused on historical facts].*"
- For the subjects with the negative determinate "*no*" like "*no one*", usually the scope of negation extends to the entire sentence, either a passive or an active sentence, e.g.: "*No [one was herded during our journey].*"
- Complex keywords and negative conjunctions have one scope that is linked to entire keywords. For instance, the scope of "*neither… nor*" includes all words that are effected by both keywords.
- In elliptical sentences, the scope of negative keywords is neglected. In linguistics, an elliptical construction refers to a shorter form of a clause which some of its words have already been omitted, nevertheless they are recoverable or understandable from the context. For example in "*It has so many features that other laptops do **not.***" the negation term *"not*" does not have any scope.

4.3.1 Min-max strategy

For annotating negation keywords and their scopes in a corpus, a min-max strategy [76] can be applied. Based on this strategy, when identifying the keywords, a minimalist strategy considers the minimum unit that expresses negation as a keyword. However, special attention must be paid to complex keywords.

To find the scope of negation, a maximalist strategy extends the scope to the largest syntactic possible units. Therefore, the scopes possess the maximal length in contrast to the keywords. This strategy is supported by two facts. First scopes include every word between the keyword and the target to avoid empty scope. Second, the status of modifiers sometimes is ambiguous and not clear whether the modifier of the target word is included in its scope or not [76]. For example in "*There is [no] primary impairment of glucocorticoid metabolism in the asthmatics*", depending on the scope of "*no*" the sentence can describe two different situations. In first situation, the scope of "*no*" consists of only *primary*. Thus, the impairment of glucocorticoid metabolism is in the asthmatics but not as primary effect. In second situation, the scope of "*no*" extends to impairment, its complements and modifiers as well. Therefore there is not any impairment of the glucocorticoid metabolism at all [76].

4.4 General remarks

In most negation annotation approaches the following rules are considered as general principles [70]:

- Only consider the sentences or phrases with the instances of negation clues, not all sentences.
- Question statements are not annotated for negation.
- The negation clues are not included in the scope.
- Transition words or phrases (e.g. *moreover*) are removed from negation scope.

4.5 Previous works

The SFU Corpus[28] of the work of Konstantinova et al. [77] contains of 400 reviews of 8 different domains as follows: movie, books, cars, computers, hotels, cookware, music, and phones. Each category has a different number of documents. The authors perform a manual annotation by the help of two annotators. The aim of their work is identifying the scope of negation and speculation in the corpora at the token and sentence levels. According to Vincze in [78], “speculation is understood as the possible existence of a thing is claimed neither its existence nor its non-existence is known for sure”.

Dadvar et al. in [67] study the effects of negation words such as “*not”* and “*hardly”* in the sentiment classification of movie reviews. They examine how five different window sizes, or scopes, of negated words effect on the classifier accuracy. Their classifier works based on the term frequency. The authors consider two word lists: one positive word list that consists of 136 positive adjectives and adverbs such as “*good*” and its synonyms like “*favorable*” and etc., and the second list includes 109 negative words like “*boring*” and its synonyms such as “*uninteresting*”, and so on. These lists are extracted from online dictionaries, such as synonym.com, and the work of Pang et al. in [32]. They also consider “?” and “!” as negative words following [32] because many opinions are stated in the form of question or surprise. They conclude although their negation detector recognizes more emotional sentences and phrases in the task of the sentiment analysis, however, applying different window sizes do not have significant effect in the classification accuracy. Their study on falsely classified reviews discloses that not identifying the indirect or implicit negation or sarcasms, which is not considered in their work, increases the chance of misclassification especially in negative reviews. Another possible reason is negative reviews including fewer adjectives and adverbs in comparison with positive documents. Moreover, since the window size is started after the location of negation words, if the

[28] https://www.sfu.ca/~mtaboada/research/SFU_Review_Corpus.html

negation words appear after the adjectives or adverbs, the classifier cannot find the scope correctly.

Das and Chen [79] apply negation identification in their experiment to extract the sentiments from stock market message boards. According to them, negation words like "*not*", "*never*", "*no*", etc. in a sentence can reverse the meaning of it. To do this, they use a list of negation words to find out if a sentence contains them and if so, mark the sentence as negated in the classification process.

In 2002, Pang et al. [80] adopt the same approach as [79] and consider scope of each negation keyword starting right after the keyword to the first following punctuation. This technique has two limitations. First the scope is not accurate and it may contain words or clues that are not negated especially for embedded clauses. Second, this manner is based on a bag of negation words, and making a completed list, in any language, is not achievable.

The linguistic structure of a sentence is another issue that can be considered in sentiment analysis. As proposed in [81], the polarity of a sentence is dependent on its parts such as noun phrases (NP), verb phrases (VP) and words' part of speech. In this case, negated words in a sentence can effect on entire sentence or only some parts of it [82] [68].

In [83] Choi and Cardio propose a semantic relationship analysis approach. Semantic relation or dependency refers to the relationship between meanings, for example synonym or antonym. Their approach consists of two processing phases. First, they extract the polarity of words that are classified based on their strength in term of the scope in a sentence. Second, they identify the polarity modification features based on the inference rules. For instance, in the sentence "*They could not eliminate my doubt*", the negated word "*not*" reverses the polarity of "*eliminate*" while the verb "*eliminate"* is inversing the orientation of "*doubt*". This approach performs well for simple sentences. For compound sentences, especially when there is word-base or sentence-base dependency, it has failed.

In addition to semantic dependency, grammatical or syntactic relationships between the words within a sentence help to extract textual relations and dependencies. Reschke and Anand [84] propose a contextual aware approach for opinion mining where the sentiments are extracted toward target events, entities and etc. In this method the scope of negation words is computed based on the verb and noun clauses or phrases in a sentence. The heuristic rules that join the clauses are used to understand the sentiment in a sentence [60].

The research of Asmi and Ishaya [68] proposes a way to represent the role of negation in sentiment classification. Their work provides a framework for automatic negation identification in textual data. This framework uses Stanford Dependency Parser[29] to identify the negation and the part of speech tags that are associated with in a sentence. All identified sentences are marked as NEGATION. In the phase of computing the polarity of sentences, the algorithm uses the SentiWordNet lexicon in order to evaluate the scores of adjectives and nouns for noun phrases, and verbs and adverbs for verb phrases. If a noun or a verb phrase of a sentence is market as NEGATION by the syntactic parser, its scores are reversed. Their results show negation detection improves their sentiment analyzer performance.

Jai et al. [69] investigate the problem of negation identification in opinionated text and provide a methodology to compute the scope of each negation term. They consider both individual words such as "*not*" or "*hardly*", and phrases like "*no longer*" as negation keywords. They use Stanford Parser to extract the minimal logic unit as candidate scope. Minimal logic unit in a sentence is a subset of the sentence words that are following a negation term and consists of the leaf nodes of non-terminal nodes in the parse tree of the sentence. The candidate scope does not contain independent clauses of the sentence and will be pruned to extract the actual scope. In order to do it, they propose the three following rules to eliminate words from candidate scope. First, static delimiter are unambiguous words that begin with another clues such as "*because*" or "*unless*". Second, dynamic delimiters are ambiguous words such as "*like*" or "*for*" that require being disambiguated by using contextual information like their POS tag. And third, heuristic rules that involve sentimental verb, noun and adjective such that the immediate word after these sentimental

[29] http://nlp.stanford.edu/software/lex-parser.shtml

terms acts as a delimiter. Jia et al. conduct two sets of experiments. The first one at sentence level is evaluated by accuracy which consists of 1000 random sentences from the review corpus of Retiatall.com. The second set involves the ranked opinionated documents that are retrieved from 3.2 million TREC documents[30]. Both their experiments outperform other techniques like [32] and [85].

4.5.1 NegEx algorithm

In addition of user-generated contents, another area of textual information that has studied for negation detection is narrative reports in medical records. These clinical documents contain wealth of data about patient medical conditions and are used to manage patient information and predict trends in diseases. However, since they are expressed in narrative form, not only they are unavailable for automated systems to improve patient care or further medical researches, but they also are difficult for humans to approach for clinical researches or teaching purposes. Therefore, researching on relevant clinical terms requires information retrieval (IR) methods to build effective techniques that index automatically these narrative clinical reports. However, discrimination between terms pointed to a present disease and negated terms is not generally applicable in IR methods. This downside of IR technique has led researchers to find out ways to identify negated terms [85]. One of the most commonly and yet easy algorithms is NegEx that is proposed in [85] by Chapman et al.

NegEx algorithm determines whether findings and diseases, which have already been indexed by IR methods in a sentence from patients' summaries, are negated or not. This algorithm accepts a sentence contained UMLS indexed terms as input and identifies whether these terms are affirmed or denied [85]. The Unified Medical Language System (UMLS) is a project that is developed to improve the computer ability to understand

[30] http://trec.nist.gov/data/docs_eng.html

biomedical vocabularies [86]. UMLS is a summary of terminological biomedical vocabularies, standards and classifications[31].

To identify the negation scope, NegEx algorithm uses two regular expressions (RE) as below [85]. The first RE finds out the negated phrases preceded the UMLS term:

$$< negation\ phrase > * < UMLS\ term >$$

In the second RE, the negation phrase comes after the UMLS term:

$$< UMLS\ term > * < negation\ phrase >$$

It is worth noting that, to capture the longest possible subset of the sentence, the asterisk indicates 0 to 5 intervening tokens, i.e. words or UMLS terms. Chapman et al. propose a set of 35 negation phrases (see Appendix 1) that divided into two groups. The first group called "pseudo-negation" contains 10 phrases that, although they include the negated words such as "*not*" or "*without*", but they identify double negation like "*not ruled out*" or false triggers e.g. "*not necessarily*". The second group consists of negated phrases used to deny findings and diseases like "*absence of*" in the first form of regular expression, or "*declined*" in the second one.

4.6 Our approach

Our approach is inspired by Blanco and Moidovan [7] that propose some heuristics to identify the negation detection role in statements. They study both scope and focus of negation keywords. According to them, scope is all items in a statement that are negated by individual negation marker, and focus is a part of scope that carries the most prominent negated burden in a statement. Usually identifying the focus is more difficult than scope,

[31] http://www.nlm.nih.gov/research/umls/quickstart.html

while some linguistic aspects such as stress and intonation have influences on it [7]. We mention here that addressing the focus is beyond of the scope of this thesis.

In order to identify the scope of negation, Blanco and Moidovan [7] use the WSJ section of the Penn Treebank. This section consists of full syntactic rules. Also, they study how often and which negation bearings occur more often in negated texts. To find out this information, in the first step, they enumerate the occurrences of negation markers in their corpus. The total number of incidences of negation terms is 7,169 where 5,707 times of them (means 79.61%) corresponds to "*not*" and its affixed form "*n't*". The average number of the rest of negation words' frequency is between 64 to 1,000 times which shows significant difference. So, they choose "*not*" and "*n't*" as negation keywords in their work [7].

In the second phase, they establish four syntactic patterns to extract negated statements from a sentence or a phrase. These syntactic structures which are depicted in Table 10 are stated based on the Penn Treebank rules. They consider two auxiliary verb: "*have*" and "*do*" [7].

Table 10: The syntactic rules for detecting negation scope

VP-be-not-PRG	**VP-be-not-VP**	**VP-aux-not-VP**	**VP-MD-not-VP**
(VP (* <be>) (…) (RB n|o'|t) (…) (*-PRD *))	(VP (* <be>) (…) (RB n|o'|t) (…) (VP *))	(VP (* <aux>) (…) (RB n|o'|t) (…) (VP *))	(VP (MD *) (…) (RB n|o'|t) (…) (VP *))

However, they do not consider every item within the constituent as scoped elements. They eliminate prepositional phrases (PP), and sub-clauses (S) and (SBAR). The overall accuracy they achieve is 66% [7].

In our work, we expand the above approach by adding more negation bearings and patterns. In addition to *"not"* and its affixed form *"n't"*, modal and auxiliary verbs, we consider 19 more negation makers including *"no", "none", "nobody", "nowhere", "never", "barely", "scarcely", "hardly", "nothing", "neither…nor", "no one", "no person", "anyone", "anybody", "without", "lack", "lacked", "lacking", "lacks"* that are involved in 90 pattern structures (see Appendix 3).

Beside these 90 patterns, we exclude the pseudo-negation pattern *“not only…but also…”*. In this regard, we consider the following pattern to eliminate the effect of negated item “*not*” in this structure:

"RB=rb . __=label : (=rb < @NOT) : (=label < @Only)"

Moreover, we consider *but clauses*. But clauses use to indicate two opposed aspect of an object. For example, the sentence *“She is not talkative but smart!”* the term *“not”* is not applied to *“smart”* although they are in the same noun phrase.

To better illustrate our approach we show it in an example. Consider the following sentence: *“There is no good movie in this collection!”*.

In the first step, as mentioned earlier, our algorithm tokenizes the sentence:

Tokenized sentence: [*there, is, no, good, movie, in, this, collection, !*]

In the second step, it looks for negation keyword, and if it finds anyone, then applies a negation analyzer. In this example, the word *“no”* is a negation marker. The negation analyzer sends the tokenized sentence to Stanford Parser in order to create its parse tree as following (see Figure 4.1):

Main tree: *(ROOT (S (NP (EX there)) (VP (VBZ is) (NP (NP (DT no) (JJ good) (NN movie)) (PP (IN in) (NP (DT this) (NN collection))))) (. !)))*

In the third phase, the negation analyzer uses a negation processor to detect the corresponding pattern that matches with above parse tree. It finds the following pattern:

Pattern: *VP << NP=np : (=np < (DT < no))*

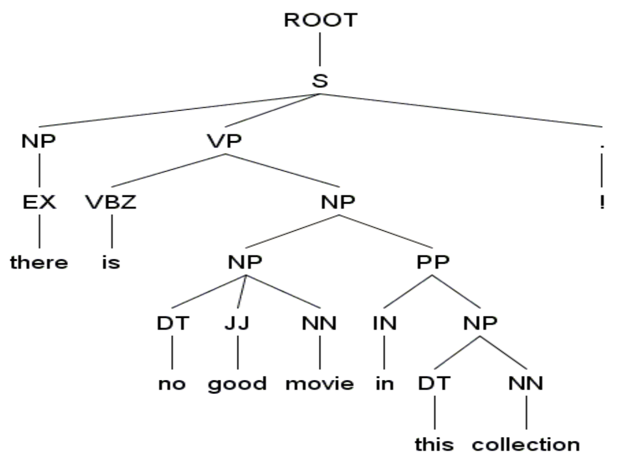


Figure 4.1: Parse tree

In the fourth step, based on the negation pattern, the negated subtree(s) are extracted from the main parse tree. Like Blanco and Moldovan [7], we prune the negated subtree by eliminating prepositional phrases (PP), sub-clauses (S) and (SBAR), and moreover parentheses. All the rest of leaves are marked as negated items and their polarity scores multiplied by -1.

Negated subtree: *(VP (VBZ is) (NP (NP (DT no) (JJ good) (NN movie)) (PP (IN in) (NP (DT this) (NN collection)))))*

Negated items: *[is/VBZ, good/JJ, movie/NN, in/IN, this/DT, collection/NN]*

We mention again that we just consider verbs and adjectives as polar terms. So the final negated terms are "*is*" and "*good*".

4.7 Evaluation results for CBR approach in SA

We use the same program and approach that is described in Chapter 3 with adding the negation detection to improve the results. Before expressing our evaluation, we study the accuracy of our scope detector by examining it on more than four hundreds negated and affirmative sentences.

4.7.1 Scope detection of negation

We use the SFU corpus from work of Konstantinova et al [77] that consists of 400 reviews of book, movie, and product reviews including cars, music, hotel, computers, and cookware. This corpus contains 200 positive and 200 negative reviews that are annotated at token level with speculative and negative clues, and with their linguistic scope at sentence level [77].

We randomly choose 479 sentences, with and without negation keywords, from book, hotels, movie, and music domains to have more compatibility with our domains. These sentences are selected from raw corpus (without annotations). We compare the extracted negation scopes of our program with the scopes of the annotated corpus. In total, there are 121 negated sentences out of 479 sentences. Our program can identify the negation scope of 107 sentences precisely. Moreover, our algorithm finds the scope of modal verbs that are not considered by Konstantinova et al [77]. The accuracy of our negation scope detector is 88.43%. Some of the unrecognized scopes can be due to eliminating prepositional phrases and the closed sentences with parentheses, or some phrase structures that we have missed in our patterns. For positive sentences, our algorithm work well and does not detect them as negated sentences.

4.7.2 Training phase with negation detection

The training process to construct the case base is the same as described in the section 0, except that we have added a negation finding phase at this stage.

The results in Table 11 show an increment in case base sizes (between 60 to 76 cases) in all domains as well as slightly improvement in the ratio of negative reviews that are correctly predicted (true negative).

Table 11: Case base size and ratio of discarded cases after applying negation detection

Domain	**Case base Size#**	**Positive%**	**Negative%**	**Discarded ratio %**
Apparel	9136	52.4%	47.37%	8.64%
Book	9249	51.864%	48.13%	7.51%
Electronics	9175	52.596%	47.41%	8.25%
Hotel	9223	51.67%	48.31%	7.77%
Movie	9168	52.75%	47.25%	8.32%
Music	9224	52%	48.01%	7.76%

Also, the average discarded ratio is reduced about 6.67% which are depicted in Figure 4.2. The minimum discarded ratio belongs to the Book domain and the maximum one is for the Apparel domain. However, the percentage of prediction for positive reviews shows a little decrement that can be because of existing negated statements in some positive reviews.

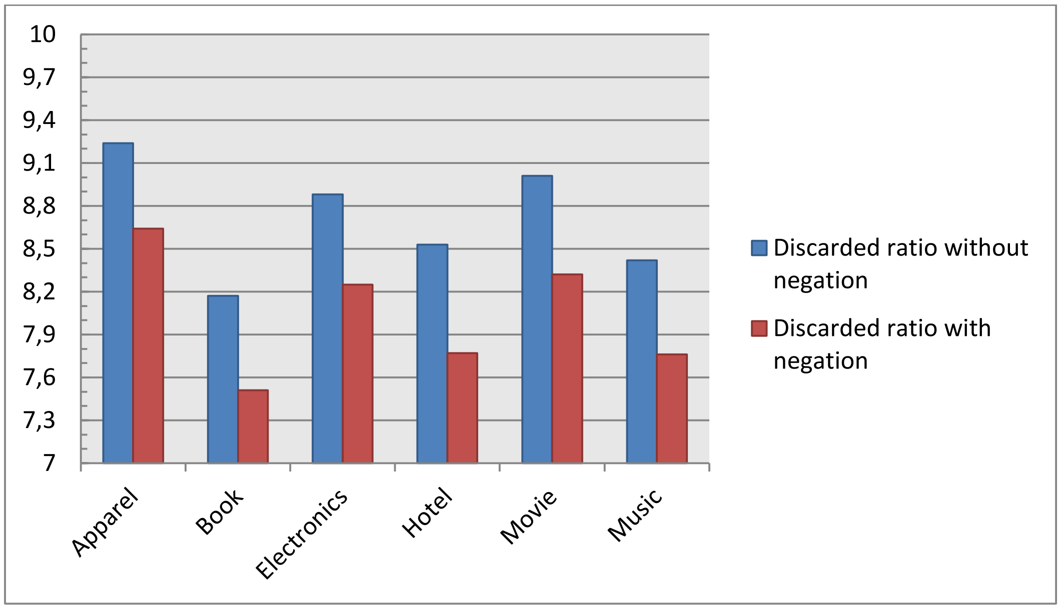


Figure 4.2: Comparing the discarded ratio with and without negation detection

4.7.3 Test phase with negation detection

Like section 0, we evaluate our classifier on unseen reviews. At this stage, for each domain, our approach uses a case base that consists of the other five domains to train our classifier in order to classify unseen documents. We present the results for *k=1* to *k=5* as following:

Table 12: The results at *k=1* after applying the negation detection

Domain	**#TP**	**#FN**	**#TN**	**#FP**	**Accuracy%**
Apparel	829	171	611	389	72%
Book	888	112	410	590	64.9%
Electronics	790	210	537	463	66.35%
Hotel	978	22	473	527	72.55%
Movie	723	277	720	280	72.15%
Music	872	128	457	543	66.45%

Table 13: The results at *k=3* after applying the negation detection

Domain	**#TP**	**#FN**	**#TN**	**#FP**	**Accuracy%**
Apparel	859	141	591	409	72.5%
Book	870	130	415	585	64.25%
Electronics	840	160	491	509	66.55%
Hotel	992	8	402	598	69.7%
Movie	701	299	749	251	72.5%
Music	885	115	422	578	65.35%

As the tables depict the performance of our classifier is improved after applying the negation detection up to a maximum 6.9% (for Apparel). In addition, there is a reduction in the ratio of false positive (FP) in all domains. This difference is illustrated in Figure 4.3 for *k=1*. For other values of *k,* there is an improvement in the ratio of false positive as well. Apparel and Book domains have the most important difference, with 55 and 52 instances respectively. In contrast, Hotel and Electronics datasets contain the lowest diversity, 5 and 6 instances respectively.

Table 14: The results at *k=5* after applying the negation detection

Domain	#TP	#FN	#TN	#FP	Accuracy%
Apparel	790	210	617	383	70.35%
Book	883	117	394	606	63.85%
Electronics	802	198	522	478	66.2%
Hotel	986	14	415	585	70.05%
Movie	580	420	771	229	67.55%
Music	866	134	437	563	65.15%

These results indicate that negation identification influence positively in the performance of the classifier and therefore, it is important to take it into account in sentiment analysis.

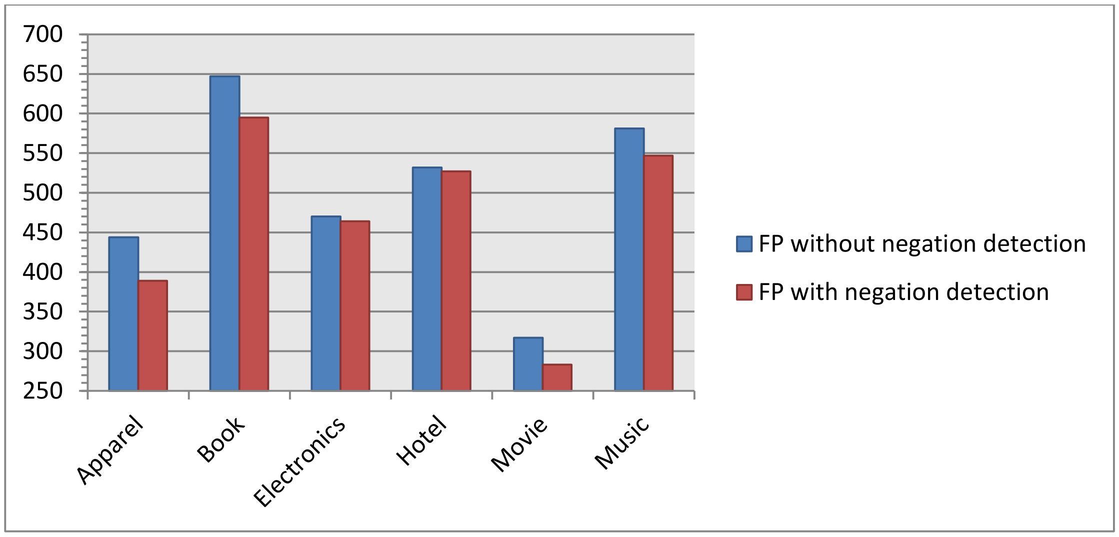


Figure 4.3: Difference of false positive at *k=1*

4.8 Discussion

The above evaluation shows better results than the test phase in section 0 (our CBR approach without negation identification). The accuracy for all values of *k* is improved comparing with the outcomes of Ohana et al. [4]. Among the listed domains, the Hotel performs better than the others except at *k=3* where Apparel has better accuracy.

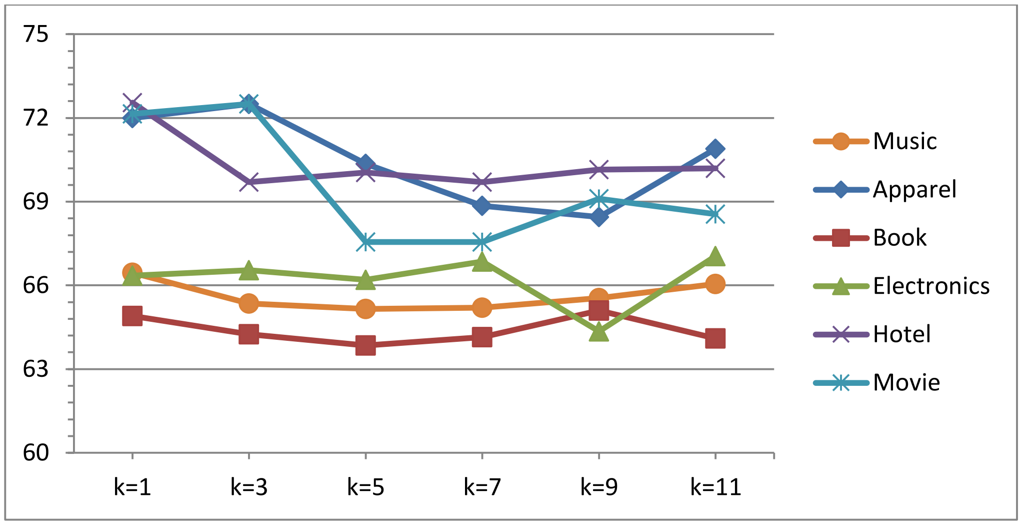


Figure 4.4: Accuracy for varying values of *k* after applying negation detection

To study the influence of *k*, we examined five discrete values, odd values, from 1 to 11 as depicted in Figure 4.4. Comparing the results with Figure 3.3 shows that the shape of the each individual curve over *k* seems generally similar. The only exception is Electronics domain that behaves quite differently. In addition, the Apparel is out of order compared to the other five domains. This would indicate that negation detection gives a small boost in accuracy about 5% without changing the general behavior of the algorithm.

Chapter 5

Conclusion and Future works

This final chapter is started by reviewing the most prominent problems existing in sentiment analysis and continued by the conclusion of our project. We round off the thesis with the future works.

5.1 Difficulties of sentiment analysis

As mentioned earlier, sentiment analysis is not an easy task. In the following, we mention some of the problems in this field:

- As the first step to determine opinionated texts, building a subjectivity classifier, as a subtask of sentiment analysis, is a complicated task in the classification. Usually there is a thin-line difference between subjective and objective contexts [31].
- Another matter that makes SA a difficult task is determining the opinion holder and distinguishing he/she from the commenter [31].
- Dependency on domains, as mentioned before, is another problem in extracting the true opinions from a text or a corpus [31].
- Negation identification and finding out its scope is another difficult task in the literature and in particular in SA (see Chapter 4).
- Since an object can have several features, a person can be interested on some aspects while having criticisms on other features. In this case, sentiment analysis

applications need to distinguish the merits and shortcomings of the object in proper way [2].

- Automated extraction of sentiment data from different types of documents such as reviews, tweets, Facebook posts, and etc. is a difficult task especially because of the dependency on the informal language structures and domains [2].
- Recognizing named entities is another challenge in SA applications. In another words, sometimes identifying the exact target purpose of an author involves ambiguity. For example, if a writer mentions "*300 Spartans*", it is not enough clear from the computer program side if he is speaking about a group people of ancient Greeks or the name of a movie [2].
- Detecting what an opinionated phrase (i.e. a noun phrase or a verb phrase or etc.) refers to is another problem that is called *anaphora resolution*. For example an audience after watching a movie posts. *"We watched X movie and went to dinner, it was awful!"* what does *awful* refer to? To the movie or to the dinner? [2]
- Identifying the sarcasm and the irony are another challenge in sentiment analysis tasks. For instance, a woman tweets her opinion about the movie X that she has recently watched as "*So much fun! X is my new favorite movie!*" [17]. Here there are two positive sentences. At the same subject, her teenage son tweets *"So much fun seeing X with mom ... NOT."* [17]. Many classifiers will classify the boy's tweet as positive because the word "NOT" comes at the end of the sentence and some negation detection algorithms consider no scope for it. Even if an algorithm marks the tweets as negative, there is still another issue as well; the negative sense of the tweet does not refer to the movie itself [17]. General speaking, if we do not have any background knowledge about the author, understanding the sarcasm or the irony will be difficult task even for human [2].
- Non-literary writing style is another issue in sentiment classification. For example using of abbreviations, poor punctuations, and existing of poor grammar, informal or irregular words or poor spelling, short length of sentences, and lack of capitals have made a new evolution in languages [2][24]. Determining such changes that are happening and growing every day is complex for SA systems.

- Part-of-speech tagging and parsing of unstructured texts are not accurate enough. Most reviews and tweets or other social media opinionated posts are written in informal style. Thus handling POS tagging or parsing them are challenging tasks for developers of SA systems.

5.2 Conclusion

The main contribution of this thesis was to study a supervised algorithm including a case-based approach on cross-domains for sentiment analysis. Sentiment analysis (SA) is an application of artificial intelligence and natural language processing that automatically extracts the sentiment and emotion from reviews. SA is a very useful application for any organizations that are looking for people's opinions about their products and services. Since sentiment analysis naturally is a domain dependent approach that makes it problematic and expensive task, we studied a cross-domain algorithm which trained our classifier on different domains instead of make it limited to one specific domain.

In this thesis we used a case-based approach on cross-domains for sentiment analysis at the document level. Our approach consists of two parts: in first part we generates a binary supervised classifier based on the case-based theory and train it on 6 discrete domains including Apparel, Book, Electronics, Hotel, Movie and Music that are extracted from IMDB and Amazon databases. Each domain contains 2000 reviews with equal numbers of positive and negative documents in plain text. In training phase, we hold out one domain and made our case base on the remaining of five domains. In this phase, the classifier used five different sentiment lexicons to extract the reviews' polarity along with the corresponding scores. Thus at the end of this process, we have six case bases that our classifier has already trained on them.

In the test phase, the classifier predicted the polarity of the hold out domains by using a set of the processed cases. We used accuracy as evaluation metric to measure how much our classifier acts precisely. In this stage, the Hotel and Movie embody better performance than others. We repeated our experiment for five different values of *k* (*1,3,5,7,9,11*). The outcomes revealed our program achieved better results at *k=1* for all domains.

In the second part of our project, we added negation detection to improve the accuracy of the classifier. We have written 90 patterns to extract the scope of negation from the negated sentences. We have observed better result in both training and test stages.

It is worth mentioning that our approach allows for further developments by adding more sentiment lexicons and datasets in the future.

5.3 Future works

This study can extend by improving some aspects and subtasks of our approach. In the following, we propose some suggestions for promising avenues of further research:

- One issue that can be added to this approach is using N-grams, e.g. bigram, instead of unigram which was used in this project. Many researchers suggest that considering the order of words can keep more information of data.
- The rapid growth of abbreviated and irregular words in recent years has created the need for attaching these new vocabularies to the lexicons. Recognizing the abbreviations (for example "*omg*"), emoticons (e.g. ☺ or☹), and interjections (like "*lol*") obviously has positive impacts on the performance of SA classifiers [24].
- The raw opinionated texts can be very noisy. Thus applying some pre-processing steps such as substituting the web address with the word "*URL*", replacing some irregular form of words whit their correct forms (e.g. "*hate'n*" to "*hating*"), eliminating repeated letters (e.g. converting "*goooood*" to "*good*") [31], turning some contiguous repeated punctuations like question mark (?) and exclamation mark (!) to ones punctuation from the same type can improve the performance of SA applications [87].
- Recognizing the irony and sarcasm by defining new algorithms and using more data sets.
- Finally it is worth mentioning that our classifier can be combined with some stronger machine learning methods like the SVMs classifier to have more accurate and practical sentiment analyzer.

Appendix 1

I. Pseudo-negation phrases including of false trigger, double negation or ambiguous negation:

gram negative
no further
not able to be
not certain if
not certain whether
not necessarily
not ruled out
without any further
without difficulty
without further

II. Negation phrases used in the regular expression first form(<negation phrase> * <UMLS term>)

absence of	doubt
declined	negative for
denied	no
denies	no cause of
denying	no complaints of
did not exhibit	no evidence of
no sign of	versus
no signs of	without
not / n’t	without indication of
not demonstrate	without sign of
patient was not	ruled out
rules out	

III. Negation phrases used in the regular expression second form (<UMLS term> * <negation phrase>)

declined
unlikely

Appendix 2

SMART system (System for the Mechanical Analysis and Retrieval of Text) contains 524 common words or stop-words that we consider in our work as following:

"a", "a's", "able", "about", "above", "according", "accordingly", "across", "actually", "after", "afterwards", "again", "against", "ain't", "all", "allow", "allows", "almost", "alone", "along", "already", "also", "although", "always", "am", "among", "amongst", "an", "and", "another", "any", "anybody", "anyhow", "anyone", "anything", "anyway", "anyways", "anywhere", "apart", "appear", "appreciate", "appropriate", "are", "aren't", "around", "as", "aside", "ask", "asking", "associated", "at", "available", "away", "awfully", "b", "be", "became", "because", "become", "becomes", "becoming", "been", "before", "beforehand", "behind", "being", "believe", "below", "beside", "besides", "best", "better", "between", "beyond", "both", "brief", "but", "by", "c", "c'mon", "c's", "came", "can", "can't", "cannot", "cant", "cause", "causes", "certain", "certainly", "changes", "clearly", "co", "com", "come", "comes", "concerning", "consequently", "consider", "considering", "contain", "containing", "contains", "corresponding", "could", "couldn't", "course", "currently", "d", "definitely", "described", "despite", "did", "didn't", "different", "do", "does", "doesn't", "doing", "don't", "done", "down", "downwards", "during", "e", "each", "edu", "eg", "eight", "either", "else", "elsewhere", "enough", "entirely", "especially", "et", "etc", "even", "ever", "every", "everybody", "everyone", "everything", "everywhere", "ex", "exactly", "example", "except", "f", "far", "few", "fifth", "first", "five", "followed", "following", "follows", "for", "former", "formerly", "forth", "four", "from", "further", "furthermore", "g", "get", "gets", "getting", "given", "gives", "go", "goes", "going", "gone", "got", "gotten", "greetings", "h", "had", "hadn't", "happens", "hardly", "has", "hasn't", "have", "haven't", "having", "he", "he's", "hello", "help", "hence", "her", "here", "here's", "hereafter", "hereby", "herein", "hereupon", "hers", "herself", "hi", "him", "himself", "his", "hither", "hopefully", "how", "howbeit", "however", "i", "i'd", "i'll", "i'm", "i've", "ie", "if", "ignored", "immediate", "in", "inasmuch", "inc", "indeed", "indicate", "indicated", "indicates", "inner", "insofar", "instead", "into", "inward", "is", "isn't", "it", "it'd", "it'll", "it's", "its", "itself", "j", "just", "k", "keep", "keeps", "kept", "know", "knows", "known", "l", "last", "lately", "later", "latter", "latterly", "least", "less", "lest", "let", "let's", "like", "liked", "likely", "little", "look", "looking", "looks", "ltd", "m", "mainly", "many", "may", "maybe", "me", "mean", "meanwhile", "merely", "might", "more", "moreover", "most", "mostly", "much", "must", "my", "myself", "n", "name", "namely", "nd", "near", "nearly", "necessary", "need", "needs", "neither", "never", "nevertheless", "new", "next", "nine", "no", "nobody", "non", "none", "noone", "nor", "normally", "not", "nothing", "novel", "now", "nowhere", "o", "obviously", "of", "off", "often", "oh", "ok", "okay", "old", "on", "once", "one", "ones", "only", "onto", "or", "other", "others", "otherwise", "ought", "our", "ours", "ourselves", "out", "outside", "over", "overall", "own", "p", "particular", "particularly", "per", "perhaps", "placed", "please", "plus", "possible", "presumably", "probably", "provides", "q", "que", "quite", "qv", "r", "rather", "rd", "re", "really", "reasonably", "regarding", "regardless", "regards", "relatively", "respectively", "right", "s", "said", "same", "saw", "say", "saying", "says", "second", "secondly", "see", "seeing", "seem", "seemed", "seeming", "seems", "seen", "self", "selves", "sensible", "sent", "serious", "seriously", "seven", "several", "shall", "she", "should", "shouldn't", "since", "six", "so", "some", "somebody", "somehow", "someone", "something", "sometime", "sometimes", "somewhat", "somewhere", "soon", "sorry", "specified", "specify", "specifying", "still", "sub", "such", "sup", "sure", "t", "t's", "take", "taken", "tell", "tends", "th", "than", "thank", "thanks", "thanx", "that", "that's", "thats", "the", "their", "theirs", "them", "themselves", "then", "thence", "there", "there's", "thereafter", "thereby", "therefore", "therein", "theres", "thereupon", "these", "they", "they'd", "they'll", "they're", "they've", "think", "third", "this", "thorough", "thoroughly", "those", "though", "three", "through", "throughout", "thru", "thus", "to", "together", "too", "took", "toward", "towards", "tried", "tries", "truly", "try", "trying", "twice", "two", "u", "un", "under", "unfortunately", "unless", "unlikely", "until", "unto", "up", "upon", "us", "use", "used", "useful", "uses", "using", "usually", "uucp", "v", "value", "various", "very", "via", "viz", "vs", "w", "want", "wants", "was", "wasn't", "way", "we", "we'd", "we'll", "we're", "we've", "welcome", "well", "went", "were", "weren't", "what", "what's", "whatever", "when", "whence", "whenever", "where", "where's", "whereafter", "whereas", "whereby", "wherein", "whereupon", "wherever", "whether", "which", "while", "whither", "who", "who's", "whoever", "whole", "whom", "whose", "why", "will", "willing", "wish", "with", "within", "without", "won't", "wonder", "would", "would", "wouldn't", "x", "y", "yes", "yet", "you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves", "z", "zero".

Appendix 3

Negation identification macros that we used in our project:

- "@VB"-> "/^VB/"
- "@AD"-> "ADVP|ADJP"
- "@PRN"-> "NP|ADJP|ADVP|S|PP"
- "@SPRN"-> "NP|ADJP|ADVP|S|PP"
- "@RB-NOT"-> "(RB=rb < @NOT)"
- "@RB-T"-> "(RB < @T)"
- "@Neither"-> "(__ < /^(?i:neither)/)"
- "@Nor"-> "(__ < /^(?i:nor)/)"
- "@Aux"-> "HAVE|Have|have|HAS|Has|has|HAD|Had|had|'VE|'ve|’VE|’ve|'S|'s|’S|’s|S|s|DO|Do|do|DOES|Does|does|DID|Did|did"
- "@Be"-> "AM|Am|am|'M|'m|’M|m|AI|Ai|ai|IS|Is|is|ARE|Are|are|WAS|Was|was|WERE|Were|were|'S|'s|’S|’s|S|s|'RE|'re|’RE|’re"
- "@WILL"-> "/^(?i:will)/|/^(?i:wo)/|/^(?i:would)/"
- "@CAN"-> "/^(?i:can)/"
- "@NADV"-> "never|Never|NEVER|BARELY|Barely|barely|SCARCELY|Scarcely|scarcely|HARDLY|Hardly|hardly"
- "@Only"-> "Only|ONLY|only|SOLELY|Solely|solely"
- "@Nothing"-> "/^(?i:nothing)/"
- "@Nowhere"-> "/^(?i:nowhere)/"
- "@NoOnes"-> "One|one|One|Person|person|PERSON"
- "@Nobody"-> "/^(?i:nobody)/"
- "@Anybody"-> "/^(?i:anybody)/"
- "@Anyone"-> "/^(?i:anyone)/"
- "@Lack"-> "LACK|Lack|lack|LACKS|Lacks|lacks|LACKED|Lacked|lacked|LACKING|Lacking|lacking"
- "@None"-> "None|NONE|none"
- "@NOT"-> "NOT|Not|not|n’t|n't|N'T|N’T|n't"
- "@NOOT"-> "NOOT|Noot|noot"
- "@NO"-> "NO|No|no"
- "@A"-> "(DT < /[Aa]/)"

Negation detection patterns that we use in our approach:

- **/** Without, 1 pattern **/**
  - ("PP=head < IN=in : (=in < /^(?i:without)$/)")

- **/** Nothing, 7 patterns **/**
  - ("S=head < NP=np < VP=vp : (=np <+(__) @Nothing $. =vp) : (=vp !< VP)")
  - ("VP=head , NP=np : (=np <+(NP) NN=nn) : (=nn < @Nothing)")
  - ("VP=head < @VB=vb < NP=np : (=np <+(NP) (NN < @Nothing))")
  - ("VP=head < @VB=vb < @AD=ad : (=ad ,, =vb < (NN < @Nothing))")
  - ("VP=head < @VB=vb < @PRN=prn : (=prn << NN=nn) : (=nn < @Nothing) : (=nn !>> PP)")
  - ("NP=head < __=label : (=label < @Nothing)")
  - ("VP=head ,+(NP) @AD=ad : (=ad < __=label) : (=label < @Nothing)")
- **/** Nobody, 2 pattern **/**
  - ("VP=head , NP=np : (=np <+(NP) NN=nn) : (=nn < @Nobody)")
  - ("VP=head < @PRN-SBAR=prn : (=prn << @Nobody)")
- **/** None of, 4 patterns **/**
  - ("VP=head $,, NP=np : (=np <+(NP) __=label) : (=label < @None=none) : (=none ?. of|Of|OF)")
  - ("VP=head < NP=np : (=np <+(NP) __=label) : (=label < (@None ?. of|Of|OF))")
  - ("NP=head << NP=np << PP=pp : (=np <+(NP) __=label) : (=label < @None) : (=pp << of|Of|OF) : (=np $. =pp)")
  - ("@AD=head < NP=np : (=np <+(NN) @None)")
- **/** Nowhere 2 patterns **/**
  - ("VP=head < @VB < @AD=ad : (=ad << RB=rb) : (=rb < @Nowhere)")
  - ("VP=head < @VB < NP=np : (=np << JJ=adj) : (=adj < @Nowhere)")
- **/** Aux (have/do) not VP, 2 patterns **/**
  - ("VP=head < @VB=vb < VP : (=vb < @Aux $. @RB-NOT)")
  - ("VP=head < @VB=vb < VP=vp : (=vb < @Aux .. RB=rb) : (=rb < @NOT=not !> PP) : (=not >> =head) : (=vp < @VB)")
- **/** TO Be not PRD/VP, 6 patterns **/**
  - ("VP=head < @VB=vb < VP : (=vb < @Be $. @RB-NOT)")
  - ("VP=head < @VB=vb < RB=rb : (=vb $.. =rb) : (=rb < @NOT) : (=rb !$,, CC)")
  - ("VP=head < @VB=vb < @PRN : (=vb < @Be $. @RB-NOT)")
  - ("VP=head < @VB=vb < @PRN : (=vb < @Be=be .. RB=rb) : (=rb < @NOT ,, =be) : (=rb > =head)")
  - ("VP=head < @VB=vb < RB=rb < VP : (=vb < @Be .. (=rb < @NOT)) ")
  - ("VP=head < @VB=vb < @AD=ad : (=vb < @Be $. =ad) : (=ad < @RB-NOT)")
- **/** Modal, 4 patterns **/**
  - ("VP=head < MD=md < VP : (=md $. @RB-NOT)")
  - ("VP=head < MD=md < VP : (=md $.. @AD=ad) : (=ad < @RB-NOT)")
  - ("VP=head < MD=md < VP : (=md $. @AD=ad) : (=ad $. @RB-NOT)")
  - ("VP=head < MD=md < VP : (=md < @CAN $. @AD=ad) : (=ad < @RB-T)")
- **/** Short answer and elliptic sentences, 2 patterns **/**

  - ("VP=head < @VB=vb [<- @RB-NOT | <- @Neither] : (=vb < @Be|@Aux)")
  - ("VP=head < MD [<- @RB-NOT | <- @Neither]")
- **/** Negative adverb of frequency, 7 patterns **/**
  - ("VP=head < MD < @AD=ad : (=ad < RB=rb) : (=rb < @NADV)")
  - ("VP=head < @VB=vb < @AD=ad : (=vb < @Be) : (=ad < RB=rb ,, =vb) : (=rb < @NADV)")
  - ("VP=head < @VB=vb < @AD=ad : (=vb $, =ad) : (=ad << @NADV)")
  - ("VP=head < @VB < @AD=ad : (=ad < RB=rb) : (=rb < @NADV)")
  - ("VP=head $,, @AD=ad : (=ad < RB=rb) : (=rb < @NADV)")
  - ("VP=head > S=s : (=s $, @AD=ad) : (=ad < RB=rb) : (=rb < @NADV)")
  - ("@AD=head < @AD=ad : (=ad <+(RB) @NADV)")
- **/** No, 16 patterns**/**
  - ("S=head < NP=np < VP=vp : (=np >, =head <+(__) @NO) : (=vp !< VP)")
  - ("S=head < INTJ=intj < VP=vp : (=intj >, =head <+(__) @NO) : (=vp !< VP)")
  - ("VP=head <@VB=vb < NP=np : (=vb $. (=np < DT=dt)) : (@NO >, =dt)")
  - ("VP=head $,, NP=np : (=np <1 (DT < @NO)) : (=np <2 (NN < @NoOnes))")
  - ("VP=head $, NP=np : (=np < (DT=dt < @NO)) : (=dt [!.. /,/ & !.. /:/]) : (=head !< CC)")
  - ("VP=head < NP=np : (=np <+(NP) (DT < @NO))")
  - ("VP=head < @VB=vb < @AD=ad : (=ad $,, =vb < (RB < @NO))")
  - ("VP=head < @VB < VP=vp : (=vp < @AD=ad) : (=ad <+(RB) @NO)")
  - ("VP=head < @VB=vb < @AD=ad : (=vb < @Be) : (=ad << @NO)")
  - ("VP=head < @AD=ad : (=ad < (__ < @NO))")
  - ("VP=head < @VB=vb < @AD=ad < @VP=vp : (=ad $, =vb $. =vp << @NO) : (=vb [< @Be| < @Aux])")
  - ("VP=head $, @AD=ad : (=ad < (__ < @NO))")
  - ("NP=head < DT=dt : (=dt < @NO >, =head)") //?
  - ("@AD=head < __=label : (=label < @NO)")
  - ("INTJ=head < UH=uh : (=uh < @NO)")
  - ("NP-TMP=head $, RB=rb : (=rb [< @NOT | < @NO])")
- **/** NOT and NOOT, 17 patterns**/**
  - ("S=head <<: VP=vp , RB=rb: (=vp <1 TO) : (=rb < @NOT)")
  - ("VP=head < RB=rb < VP=vp ?< CC=cc : (=rb < @NOT) : (=cc < /^(?i:and)$/ !$. =rb) : (=vp $,, =rb)")
  - ("VP=head $, RB=rb > VP=parent : (=rb < @NOT >1 =parent)") //#a
  - ("VP=head $,, __=label : (=label <1 @RB-NOT) : (=label <2 (NN < @Anyone))")
  - ("VP=head $,, RB=rb $,, NP=np : (=rb < @NOT $. =np) : (=np <+(NP) (NN < @Anyone))")
  - ("VP=head $, NP=np : (=np <+(NP) (@RB-NOT $. (NN < @Anyone)))")
  - ("VP=head $, NP=np : (=np <+(NP) (RB < @NOT=not)) : (=np !<< (@Only , =not))")
  - ("VP=head < @VB=vb < RB=rb : (=rb $.. =vb < @NOT)")

    - ("VP=head $,, __=label : (=label < @NOT=not) : (=not !. @Only)")
    - ("S=head $, WHADVP=wh : (=wh < (RB=rb < @NOT)) : (=rb >- =wh)")
    - ("NP=head < __=label : (=label < @NOOT)")
    - ("NP=head $,, __=label: (=label < @NOOT)")
    - ("@AD=head $,, __=label : (=label < @NOOT=not) : (=not !. @Only)")
    - ("@AD=head < RB=rb ?< CC=cc : (=rb < @NOT) : (=cc < /^(?i:and)$/ !$. =rb)")
    - ("S=head $, __=label: (=label < @NOOT)")
    - ("FRAG=head < @AD=ad < RB=rb : (=ad >, =head) : (=rb >2 =head)")
    - ("SBAR=head $, RB=rb : (=rb < @NOOT)")
- **/** Neither Nor structure, 9 patterns **/**
    - ("VP=head $, NP=np : (=np << @Neither << @Nor)")
    - ("VP=head $, @AD=ad : (=ad << @Neither !<< @Nor)")
    - ("VP=head < @AD=ad : (=ad << @Neither << @Nor)")
    - ("VP=head < @VB <+(S) NP=np : (=np << @Neither << @Nor)")
    - ("VP=head < @VB < @SPRN=prn : (=prn << @Neither ?<< @Nor)")
    - ("VP=head $, NP=np !< @Nor: (=np << @Neither !<<@Nor)")
    - ("VP=head , @Nor")
    - ("NP=head < @Neither ?<@Nor")
    - ("NP=head < @Nor !< @Neither")
- **/** FRAG, 5 patterns **/**
    - ("VP=head $, RB=rb : (=rb < @NOT) : (FRAG < =head < =rb !$, WHADVP)")
    - ("@AD=head $,, RB=rb : (=rb < @NOOT) : (=head >> FRAG=frag) : (=frag !$, WHADVP)")
    - ("CONJP=head < __=label : (=label < @Not)")
    - ("NP=head $,, @RB-NOT >> FRAG=frag : (=frag !$, WHADVP)")
    - ("FRAG=head < RB=rb : (=rb < @NOOT)")
- **/** Lack, 7 patterns**/**
    - ("NP=head < JJ=jj : (=jj < @Lack)")
    - ("PP=head $, NP=np > NP=parent : (=np <+(NN) @Lack > =parent) : (=head <+(IN) /^(?i:of)$/)")
    - ("PP=head $, NP=np : (=np <+(NN) @Lack) : (=head <+(IN) /^(?i:of)$/)")
    - ("NP=head < NP=np < PP=pp : (=np <+(NN) @Lack $. =pp) : (=pp <+(IN) /^(?i:of)$/)")
    - ("VP=head < @VB=vb < NP : (=vb < @Lack)")
    - ("NP=head < NN=nn : (=nn < @Lack=lack) : (=lack !>: =nn)")
    - ("NP=head <+(NP) NN=nn < NP : (=nn < @Lack=lack >>, =head)")

Bibliography

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Inf. Retr. Boston.*, vol. 2, no. 1, 2008.

[2] P. Routray, C. K. Swain, and S. P. Mishra, "A Survey on Sentiment Analysis," *Int. J. Comput. Appl.*, vol. 76, no. 10, p. 18, 2013.

[3] B. Liu, "Sentiment analysis and subjectivity," *Handb. Nat. Lang. Process.*, vol. 2, pp. 627–666, 2010.

[4] B. Ohana, S. J. Delany, and B. Tierney, "A Case-Based Approach to Cross Domain Sentiment Classification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7466 LNAI, pp. 284–296, 2012.

[5] P. Cunningham, N. Nowlan, S. J. Delany, and M. Haahr, "A Case-Based Approach to Spam Filtering that Can Track Concept Drift," in *Workshop on Long-Lived CBR Systems (ICCBR'03)*, 2003, vol. 3, no. Ml, pp. 3–2003.

[6] S. Gindl, A. Weichselbraun, and A. Scharl, "Cross-domain contextualization of sentiment lexicons," *Front. Artif. Intell. Appl.*, vol. 215, pp. 771–776, 2010.

[7] E. Blanco and D. Moldovan, "Some Issues on Detecting Negation from Text," *Twenty-Fourth Int. FLAIRS Conf.*, pp. 228–233, 2011.

[8] P. Russel, Stuart; Norvey, *Artificial intelligence: a modern approach*, vol. 74, no. 01. Prentice hall, 2010.

[9] P. McCorduck, "Machines who think," 2004.

[10] M. Alan, "Turing. Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[11] T. M. Mitchell, "Machine learning. WCB." McGraw-Hill Boston, MA:, Singapore, 1997.

[12] D. Jurafsky, *Speech and Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2009.

[13] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[14] J. Nichols and T. Warnow, "Tutorial on Computational Linguistic Phylogeny," *Lang. Linguist. Compass*, vol. 2, no. 5, pp. 760–820, 2008.

[15] G. Wilcock, "Introduction to Linguistic Annotation and Text Analytics," *Synth. Lect. Hum. Lang. Technol.*, vol. 2, no. 1, pp. 1–159, 2009.

[16] B. Pang, “Automatic analysis of document sentiment,” Citeseer, Cornell University, 2006.

[17] AARON MANDELBAUM, “A Guide to Sentiment Analysis Tools,” 2014. [Online]. Available: http://www.econtentmag.com/Articles/Editorial/Feat.

[18] N. Xue, “Annotation of Subjective Language What is subjectivity ?,” *Brandeis University*, 2011. .

[19] E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens, “Automatic sentiment analysis in on-line text,” in *ELPUB*, 2007, pp. 349–360.

[20] L. Ku, Y. Liang, H. Chen, K. Lun-Wei, L. Yu-Ting, and C. Hsin-Hsi, “Opinion Extraction, Summarization and Tracking in News and Blog Corpora,” *Artif. Intell.*, vol. pages, no. 2001, pp. 100–107, 2006.

[21] M. Hu and B. Liu, “Opinion extraction and summarization on the web,” *Aaai*, pp. 1–4, 2006.

[22] F. W. Lancaster, “Precision and recall,” *Encyclopedia of Library and Information Science*, 2003. .

[23] S. R. Das and M. Y. Chen, “Yahoo! for Amazon: Sentiment extraction from small talk on the web,” *Manage. Sci.*, vol. 53, no. 9, pp. 1375–1388, 2007.

[24] H. Saif, Y. He, and H. Alani, “Semantic sentiment analysis of twitter,” in *The Semantic Web--ISWC 2012*, Springer, 2012, pp. 508–524.

[25] A. Andreevskaia and S. Bergler, “Mining WordNet for a Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses.,” in *EACL*, 2006, vol. 6, pp. 209–216.

[26] V. Hatzivassiloglou and K. R. McKeown, “Predicting the semantic orientation of adjectives,” in *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, 1997, pp. 174–181.

[27] N. Kaji and M. Kitsuregawa, “Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents.,” in *EMNLP-CoNLL*, 2007, pp. 1075–1083.

[28] C. Musto, G. Semeraro, and M. Polignano, “A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts,” *Inf. Filter. Retr.*, p. 59, 2014.

[29] F. Chaumartin, “UPAR7: A knowledge-based system for headline sentiment tagging,” *Proc. 4th Int. Work. Semant. Eval.*, no. Association for Computational Linguistics, pp. 422–425, 2007.

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

[31] G. Gebreselassie and G. Date, "Sentiment Analysis of Twitter Posts About News," *Monogr. Soc. Res. Child Dev.*, vol. 76, no. 2, p. 123, 2011.

[32] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 2002, pp. 79–86.

[33] 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*, 2003, pp. 519–528.

[34] K.-L. Liu, W.-J. Li, and M. Guo, "Emoticon Smoothed Language Models for Twitter Sentiment Analysis.," in *AAAI*, 2012.

[35] D. Bollegala, D. Weir, and J. Carroll, "Cross-domain sentiment classification using a sentiment sensitive thesaurus," *Knowl. Data Eng. IEEE Trans.*, vol. 25, no. 8, pp. 1719–1731, 2013.

[36] A. Aue and M. Gamon, "Customizing Sentiment Classifiers to New Domains : a Case Study," *Proc. Recent Adv. Nat. Lang. Process. RANLP*, vol. 49, no. 2, pp. 207–18, 2005.

[37] Z. Xu, "A sentiment analysis model integrating multiple algorithms and diverse," The Ohio State University, 2010.

[38] D. Bollegala, D. Weir, and J. Carroll, "Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 132–141.

[39] T. Mullen and N. Collier, "Sentiment Analysis using Support Vector Machines with Diverse Information Sources.," in *EMNLP*, 2004, vol. 4, pp. 412–418.

[40] 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*, 2004, p. 271.

[41] R. Prabowo and M. Thelwall, "Sentiment analysis: A combined approach," *J. Informetr.*, vol. 3, no. 2, pp. 143–157, 2009.

[42] M. Pantic, "Introduction to machine learning and case-based reasoning," vol. 1107. Computing Department, Imperial College London.

[43] R. Schank, "Memory-based expert systems," Northwestern University, Evanston,

1987.

[44] A. Aamodt and E. Plaza, “Case-based reasoning: Foundational issues, methodological variations, and system approaches,” *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[45] A. Aamodt and E. Plaza, “Case-based reasoning: Foundational issues, methodological variations, and system approaches,” *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[46] M. M. Richter, “On the notion of similarity in case-based reasoning,” in *Proceedings of the ISSEK94 Workshop on Mathematical and Statistical Methods in Artificial Intelligence*, 1995, pp. 171–183.

[47] A. Meduna, *Formal Languages and Computation: Models and Their Applications*. CRC Press, 2014.

[48] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[49] M. Porter, “Snowball: A language for stemming algorithms,” *English*, 2001. [Online]. Available: http://snowball.tartarus.org/texts/introduction.html.

[50] G. Salton and M. E. Lesk, “The SMART automatic document retrieval systems—an illustration,” *Commun. ACM*, vol. 8, no. 6, pp. 391–398, 1965.

[51] L. A. Shalabi, Z. Shaaban, and B. Kasasbeh, “Data mining: A preprocessing engine,” *J. Comput. Sci.*, vol. 2, no. 9, pp. 735–739, 2006.

[52] J. Z. Namenwirth and R. P. Weber, *Dynamics of culture*. Allen & Unwin, 1987.

[53] G. R. Semin and K. Fiedler, “The linguistic category model, its bases, applications and range,” *Eur. Rev. Soc. Psychol.*, vol. 2, no. 1, pp. 1–30, 1991.

[54] P. J. Stone, D. C. Dunphy, M. S. Smith, and D. M. Ogilvie, *The General Inquirer: A Computer Approach to Content Analysis*, vol. 08. MIT Press, 1966.

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

[56] S. Baccianella, A. Esuli, and F. Sebastiani, “SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.,” in *LREC*, 2010, vol. 10, pp. 2200–2204.

[57] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity in phrase-level sentiment analysis,” *Proc. Conf. Hum. Lang. Technol. Empir. Methods Nat. Lang. Process. HLT 05*, vol. 17, no. October, pp. 347–354, 2005.

[58] E. Riloff and J. Wiebe, “Learning extraction patterns for subjective expressions,” in

Proceedings of the 2003 conference on Empirical methods in natural language processing, 2003, pp. 105–112.

[59] J. R. L.-B. Bernard, *The Macquarie Thesaurus*. Macquarie Library, 1986.

[60] S. Mohammad, C. Dunne, and B. Dorr, “Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus,” *EMNLP ’09 Proc. 2009 Conf. Empir. Methods Nat. Lang. Process.*, vol. 2, no. August, pp. 599–608, 2009.

[61] S. M. Mohammad and P. D. Turney, “Crowdsourcing a word--emotion association lexicon,” *Comput. Intell.*, vol. 29, no. 3, pp. 436–465, 2013.

[62] B. Ohana, B. Tierney, and S.-J. Delany, “Domain Independent Sentiment Classification with Many Lexicons,” *2011 IEEE Work. Int. Conf. Adv. Inf. Netw. Appl.*, pp. 632–637, Mar. 2011.

[63] M. Nikulin, B. Vilijandas, and K. Julius, “Non-parametric Tests for Complete Data,” 2011.

[64] S. Baccianella, A. Esuli, and F. Sebastiani, “Multi-facet Rating of Product Reviews,” *Adv. Inf. Retr.*, vol. 5478, no. 9, pp. 461–472, 2009.

[65] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, “Learning bounds for domain adaptation,” in *Advances in neural information processing systems*, 2008, pp. 129–136.

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

[67] M. Dadvar, C. Hauff, and F. De Jong, “Scope of Negation Detection in Sentiment Analysis,” *Proc. Dutch-Belgian Inf. Retr. Work. (DIR 2011)*, pp. 16–20, 2011.

[68] A. Asmi and T. Ishaya, “Negation identification and calculation in sentiment analysis,” in *The Second International Conference on Advances in Information Mining and Management*, 2012, pp. 1–7.

[69] L. Jia, C. Yu, and W. Meng, “The effect of negation on sentiment analysis and retrieval effectiveness,” *Proceeding 18th ACM Conf.*, no. c, pp. 1827–1830, 2009.

[70] N. Konstantinova and S. C. M. De Sousa, “Annotating Negation and Speculation: the Case of the Review Domain.,” in *RANLP Student Research Workshop*, 2011, pp. 139–144.

[71] L. Polanyi and A. Zaenen, “Contextual valence shifters,” in *Computing attitude and affect in text: Theory and applications*, Springer, 2006, pp. 1–10.

[72] A. Kennedy and D. Inkpen, “Sentiment classification of movie reviews using contextual valence shifters,” in *Computational Intelligence*, 2006, vol. 22, no. 2, pp. 110–125.

[73] S. A. Morsy, “Recognizing contextual valence shifters in document-level sentiment classification,” American University in Cairo, 2011.

[74] R. Quirk, D. Crystal, and P. Education, *A comprehensive grammar of the English language*, vol. 397. Cambridge Univ Press, 1985.

[75] M. Taboada, K. Voll, and J. Brooke, “Extracting sentiment as a function of discourse structure and topicality,” *Group*, vol. 20, no. September 2010, pp. 1–22, 2008.

[76] V. Vincze, G. Szarvas, R. Farkas, G. Móra, and J. Csirik, “The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes,” *BMC Bioinformatics*, vol. 9, no. Suppl 11, p. S9, 2008.

[77] N. Konstantinova, S. C. M. de Sousa, N. P. C. Díaz, M. J. M. López, M. Taboada, and R. Mitkov, “A review corpus annotated for negation, speculation and their scope.,” in *LREC*, 2012, pp. 3190–3195.

[78] V. Vincze, “Speculation and negation annotation in natural language texts: what the case of BioScope might (not) reveal,” in *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, 2010, pp. 28–31.

[79] S. R. Das and M. Y. Chen, “Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web,” *Manage. Sci.*, vol. 53, no. 9, pp. 1375–1388, Sep. 2007.

[80] P. D. Turney, “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews,” *Proc. 40th Annu. Meet. Assoc. Comput. Linguist.*, no. July, pp. 417–424, 2002.

[81] K. Moilanen and S. Pulman, “Sentiment construction,” in *Proceedings of RANLP*, 2007.

[82] M. Wiegand, A. Balahur, B. Roth, D. Klakow, and A. Montoyo, “A Survey on the Role of Negation in Sentiment Analysis,” *Proc. Work. Negation Specul. Nat. Lang. Process.*, no. July, pp. 60–68, 2010.

[83] Y. Choi and C. Cardie, “Learning with compositional semantics as structural inference for subsentential sentiment analysis,” *Proc. Conf. Empir. Methods Nat. Lang. Process. - EMNLP ’08*, no. October, p. 793, 2008.

[84] K. Reschke and P. Anand, “Extracting contextual evaluativity,” in *Proceedings of the Ninth International Conference on Computational Semantics*, 2011, pp. 370–374.

[85] W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, and B. G. Buchanan, “A simple algorithm for identifying negated findings and diseases in discharge

summaries.," *J. Biomed. Inform.*, vol. 34, no. 5, pp. 301–310, 2001.

[86] D. A. Lindberg, B. L. Humphreys, and A. T. McCray, "The Unified Medical Language System.," *Methods Inf. Med.*, vol. 32, no. 4, pp. 281–291, 1993.

[87] 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)*, 2013.