Opinion Analysis through Constraint Optimisation

Suraj Jung Pandey

Department of Computer Science
University of York

A dissertation submitted for the degree
MSc by Research

January 2011

Abstract

Opinion lexicon plays a vital role in sentiment classification. A previous study shows that a compositional model can be effective in sentiment classification. But such a model has been only applied using hand-crafted composition rules. The need for hand-crafted rules arise when dealing with conflicting polarity values within the same phrase. In this thesis, we show that an alternative is to employ a weighted polarity lexicon. There are several key advantages of a weighted polarity lexicon. Firstly, compositionality rules simply become linear sums without requiring conflict resolution rules. Secondly, a weighted polarity lexicon can be automatically learnt from review data using constraint optimisation. Thirdly, instead of providing just a binary *positive* or *negative* output, our model can be used to provide a graded overall sentiment. Our experiments show that our model provides state-of-the-art opinion classification.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly I would like to thank my supervisor Suresh Manandhar. He has been a great source of encouragement. His advices and assistance always got me back on the track whenever I was confused and astray. He has been a great source of inspiration for me.

A special thanks goes to Shailesh Pandey who initially helped me a lot to adjust in England and also recommended many valuable papers to read. He showed great enthusiasm and interest to answer any of my questions.

I owe a great deal to my friends at York. Particularly Matt Naylor who along with making my stay in York very pleasant also helped me a lot with Latex codes. Suresh Katwal, my friend from Nepal always made me smile with his phone calls.

Finally supporting me always although far was my family, Dad, Mum, Nisha and Jyoti.

Declaration

I hereby declare that I composed this thesis entirely myself and it describes my own research.

Suraj Jung Pandey
University of York

Chapter 1

Introduction

Sentiment Analysis (SA) is primarily the extraction and identification of *attitude* towards *something* in a text. *Attitude* may be anger, love, happiness, resentment, hate etc and *something* can be anything from a presidential candidate to a product like a TV, item of clothing, a nail, a movie, a book, an article etc. Sentiment Analysis is also synonymously known as *affect extraction*, *opinion mining* and *subjectivity analysis*. In its most basic form SA task is to classify a given text into a positive or a negative sentiment. For example, *"This is an excellent camera"* provides positive sentiment and *"His actions were appalling"* provides negative sentiment.

1.1 Scope

Due to the availability of huge volumes of online information, the opinion of the general public has become a major concern [23][1]. For example, people could check the different opinions of the voters before voting or see the reviews and opinions of a product before buying it. Opinions are even more important to the product manufacturer. A politician may want to know what people think about him, so that he knows what he can do to increase his popularity. A manufacturer could want to know consumers' opinions about their product so they can improve it accordingly. Online discussion

[1]The author shows a survey demonstrating how the availability of online reviews has bolstered people's decisions; Page 1.

forums could use SA techniques to track the sentiment of its users over a certain time period or even to track inflammatory messages.

1.2 Motivation

Sentiment analysis involves steps that include 1) identifying the sentiment-bearing sentences or phrases, 2) classifying them according to the sentiment they possess and finally 3) combining these to generate an overall sentiment of the text. Each individual step is a challenge in itself [1, 10, 15, 24, 33, 35, 38]. Many phrases will have a different sentiment depending upon the context, e.g. *unpredictable* may be a bad review for car steering but a good review for a movie [32]. Extracting sentiment-bearing sentences will mainly depend on the presence of sentiment-bearing words, but this is not always the case. For example, *"President of National Environment Trust"* has no sentiment even with the presence of the word *trust* [35]. Even after successful completion of these steps, computing the final overall sentiment of a text is also not straightforward. As shown in [24], generating the overall sentiment is not as simple as summing and averaging the constituent sentiments of the text.

Example 1.1

> This film should be *brilliant*. It sounds like a *great plot*, the actors are *first grade*, and the supporting cast is *good* as well, and Stallone is attempting to deliver a *good performance*. However, it *can't hold up*.

What overall sentiment should we assign to Example 1.1? One could argue that with all the positive words present, the review is a positive one. Alternatively, one could argue the final sentence is negative, thus it is a negative review. Therefore, SA is both complex, requiring multiple steps, and also challenging, since each step is hard.

1.3 Thesis Aims

The principal aim of this thesis is to provide a solution towards weighing each individual sentence and individual words with a polarity value. Associating polarity values with individual words results in a weighted opinion lexicon. This can in turn be used to provide a weighted polarity value to each review. With a weighted opinion lexicon we could classify the review in Example 1.1 as 3 star, where 5 star is the most positive and 1 star is the least positive. We aim to produce a complete sentiment analysis system with opinion lexicon, target lexicon, and an effective classifier that can exploit such a lexicon to classify the sentiment of the *review* text.

1.3.1 Thesis Contribution

The specific contribution of this thesis can be summarised as follows:

1. The thesis develops a new additive compositional model for opinion classification. The model consists of a weighted opinion lexicon that can be learnt from data. The additive model classifies the opinionated text through linear sum of the opinion value of words in the text. Accuracy obtained by the additive model is in par with current state-of-the-art supervised methods. This is a significant contribution because previously opinion classification through linear sum of constituent opinion words was considered incorrect. This is due to the fact that the opinion lexicon used was not weighted, but each word in the lexicon will have three values namely positive, negative and neutral. A linear sum with such lexicons can lead to incorrect decisions [19].

   This thesis shows that by using a weighted lexicon a linear additive model can be used for opinion classification.

2. The thesis explores the use of the compositional nature of the opinion text for both opinion classification and opinion lexicon extraction. Although the compositional nature of opinion text has been exploited quite successfully by [19], the model depends heavily on a pre-acquired opinion lexicon and hand-crafted rules for classification.

Firstly, we provide a fully automatic method for acquiring a weighted lexicon from data through constraint optimisation with compositional additive constraints. We then use this lexicon for opinion classification with an additive model. Secondly, we provide an effective way of incorporating the prior knowledge of the domain during the learning phase.

3. The thesis explains an easy-to-implement association-based approach to acquire targets (topics on which an opinion is expressed e.g. *camera*) of opinions from review topics. The method is fully unsupervised and does not require any prior knowledge on reviews and targets. Our method is based on the observation that words have either strong or weak association to different topics. The approach is very simplistic, it is efficient compared to manual labeling of the targets in the review and easier compared to the hand-crafted rules as our method does not require any domain knowledge. The targets acquired from the reviews can be used for fine-grained sentiment analysis.

4. We investigated different features used by supervised opinion classification system. This led us to acquire effective features for opinion classification. The thesis utilises a feature selection technique to form a commendable set of features for opinion classification without using any prior domain knowledge. This is an important contribution as the accuracy obtained using our set of features is highest among all the other supervised classification algorithms on the same dataset.

5. The thesis provides a systematic algorithm to acquire unit clauses which expresses opinion towards a single feature. This is a significant contribution as our algorithm can provide detailed opinion on multiple features of any product.

1.4 Related Work and Background

This section consists a brief discussion of the previous literature relating to the areas covered in this thesis. All individual chapters also include a more in-depth related work discussion.

1.4.1 Opinion Lexicon

A common approach to sentiment analysis is to use a lexicon with information about the polarity of the words. The compilation of such lexicons involves the task of classifying sentiment-bearing words or phrases into either a *positive* sentiment or *negative* sentiment. This task is difficult in itself, as pointed out by Wilson et al. [35], where the author showed the effect that context/topic will have on specifying the sentiment of given phrases.

Different approaches are taken to annotate phrases with polarity. The approaches vary from manual annotation [36] to various forms of automatic annotation. Most of the earlier work on automatic annotation for sentiment polarity have has been based on word association [20, 32]. In Turney's [32] work, similarity, *Pointwise Mutual Information (PMI)* between phrase and words *"excellent"* and *"poor"* was calculated by issuing queries to a web search engine and then counting the number of hits the phrase + seed-words get. Based on this concept the *Semantic Orientation* (SO) of the phrase was calculated using the formula:

$$SO(phrase) = log_2 \left[\frac{\text{hits(phrase Near "excellent") hits("poor")}}{\text{hits(phrase Near "poor") hits("excellent")}} \right]$$

A positive SO value means that the phrase is semantically closer to the word *"excellent"*, thus is a positive sentiment. Similarly, negative SO means the phrase has a negative sentiment. The results obtained were impressive, e.g. *"Low fees, SO=0.333,* and *"unethical practices", SO=-8.484* but they would fail in cases where the context senses different semantics to the phrases, e.g. *"Lesser evil", SO=-2.288.*

To overcome this problem Wilson et al. [35] manually annotated a Multi Purpose Question Answering (MPQA) corpus with contextual polarity and trained classifiers like SVM, Ripper etc on it using different features. For example in the sentence, *"They have not succeeded, and will never succeed (**positive**), in breaking the will of this valiant people"*, the feature will be the negative phrase *breaking the will* along with the polarity shifter phrase *will never succeed*. The polarity shifter will contribute towards making the sentiment of the sentence positive. The result was a classifier which could determine the polarity of the phrase with 71.6% accuracy (SVM). But then this approach would require huge amount of manually tagged data to work

reliably.

WordNet *glosses* have been used successfully to derive sentiment lexicon [1, 9, 10]. Esuli et al. [10] used WordNet synset and gloss to form *SentiWordNet*. Their semi-supervised approach to building a lexicon involved seed positive and negative polarity WordNet synset. Then iteratively, synsets connected with other synsets by WordNet's *also-see* relation were given the same polarity and opposite polarity were given to the synsets connected by *direct antonymy* relation. Vectors for input to classifier were formed by indexing the synset with its *gloss*, which represents the semantics in textual form. The classifier was learned with these vectors, then the trained classifier was applied to all of the vector representations of WordNet synset, thus producing "Sentiment classification of the whole WordNet". But the evaluation of *SentiWordNet* remains incomplete since author did not have any baseline approach for comparing the results.

1.4.2 Target Extraction

Target extraction or opinion feature mining from an opinionated text is an integral part of an opinion analysis system. Opinion targets are primarily used for detecting subjective sentences and for fine-grained feature-specific opinion analysis. Target extraction for opinion analysis has been done mainly through the following approaches:

Manual target extraction

In most of the opinion analysis systems, targets are assumed to be limited, i.e. the features that are reviewed are in few numbers and can be provided by the manufacturers for any kind of analysis. This can be true to some extent but [13] shows some of the irregularities that might occur when using the target lexicon provided by the manufacturer. For example, the customer may not use the same word for a certain feature as used by the manufacturer.

Pre-defined feature list can be expanded using Wikipedia's category system[2][11]. The category tree specifies the named entity such as product

[2]http://en.wikipedia.org/wiki/Portal:Contents/Categorical_index

names, proper names and brand names. Most of the time these entities are the targets of the review text.

Target extraction based on bootstrapping

Most of the time targets in an opinion review are considered to be nouns or noun phrases. Lui et al. [39] use an information extraction method to mine product targets and an opinion lexicon together. The Double Propagation approach is based on the fact that the opinion targets are modified by *mod* relation as given by the dependency parser. They use an initial opinion lexicon and target lexicon to search through the dataset to identify such a relation. The bootstrapping process continues until no further opinion words or targets are found. The targets extracted in this case are all nouns or noun phrases.

Yi and Niblack [37] used three different relations with respect to the topic of the review and feature, these relations are:

- a part of a relation with the given topic
- an attribute relationship with the given topic
- an attribute relation with a known feature of given the topic

After extracting such phrases only those are selected which have the grammatical constructs as: { "NN"; "NN NN"; "NN NN NN"; "JJ NN"; "JJ NN NN"; "JJ JJ NN"}. For all the candidate phrases a log-likelihood score $-2log\lambda$ is calculated. From the sorted list thus acquired, the only phrases considered as features are those which lie above a threshold margin.

Target extraction based on statistical measures

Hu and Liu [13] identify the targets from a review text, first by extracting all the nouns and noun phrases from the text and then by using the association mining rule [2]. The association mining rule can be considered as a frequent phrases mining algorithm, where the phrases which occur at least equal to a certain threshold are termed as frequent.

Another such approach is one used by Liu et al. [17] to develop a system called OPINE. OPINE first extracts noun phrases from the reviews and retains only those with frequency greater than a certain threshold. Then, it evaluates each noun phrases by computing the Pointwise Mutual Information (PMI) score between the phrases and meronymy discriminators associated with the product. Only those noun phrases were extracted as features whose PMI score was greater than a certain threshold.

1.4.3 Opinion Classification

Features used

The presence and the frequency of sentiment-bearing phrases are the key features used for generating the overall sentiment[24]. Subjectivity is always a prominent feature to extract sentiment [20, 21, 36]. Objective sentences can be misguiding and should be avoided. For example the objective sentence, *"The protagonist tries to protect her good name"* has the word "good" but does not portray any sentiment towards the topic(in this case a movie), so such sentences should be excluded.
Context can change the overall sentiment of a phrase. For example in the sentence, *"They have not succeeded, and will never succeed, in breaking the will of this valiant people",* even with the presence of negative sentiment phrases the sentence shows a positive sentiment. Detecting and classifying contexts has been done by machine learning [35] or by just using bigrams [32].
Negation reverses the sentiment polarity. For example, in the phrase *He is not good*, the word "not" reverses the polarity of "good". Adverbs are a good source for increasing or decreasing the intensity of sentiment. For example in the sentence, *The concert was thoroughly enjoyable*, the adverb "thoroughly" increases the effect of "enjoyable" and should not be disregarded during sentiment summarisation. Benamara et al. [4] proved the effect of adverbs by stating *"*Adverbs are better than Adjectives alone*"*.

The presence of lexicon which mentions the polarity of subjective expression has resulted in increasing the accuracy of sentiment summarisation [29, 6].

1.4.4 Linear programming/ Constraint optimisation/ Constraint Solver

Linear Programming (LP) is defined as the method of solving linear equalities or inequalities for its optimal value, where optimality is defined under certain criteria. Formerly, LP is defined as the problem of *maximising* or *minimising* a linear *objective function* subject to linear *constraints*, where constraints are linear equalities or inequalities. Mathematically, LP is written as:

$$\textbf{Minimise or Maximise}: \; c_1x_1 + c_2x_2 + ...c_nx_n$$

$$\begin{aligned} \textbf{Subject to}: \quad & a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n \sim b_1 \\ & a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n \sim b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n \sim b_m \end{aligned}$$

$$\textbf{with bounds}: \quad l_1 \leq x_1 \leq u_1 .....l_n \leq x_n \leq u_n$$

where the symbol $\sim$ may be any of the symbols from the set $[\leq, \geq, <, >, =]$ and the lower bound variable l and upper bound variable u can be any value from positive infinity to negative infinity or any real number. Also in the above equations:

$c_1...c_n$ is optimisation coefficient
$x_1...x_n$ is unknown variables
$a_{11}...a_{mn}$ is constraint coefficient
$b_1...b_m$ is the right hand side of the constraint equation

To see how a linear equation works consider a simple example:

Example 1.2

$$\textbf{Minimise or Maximise}: \; x_1 + x_2$$

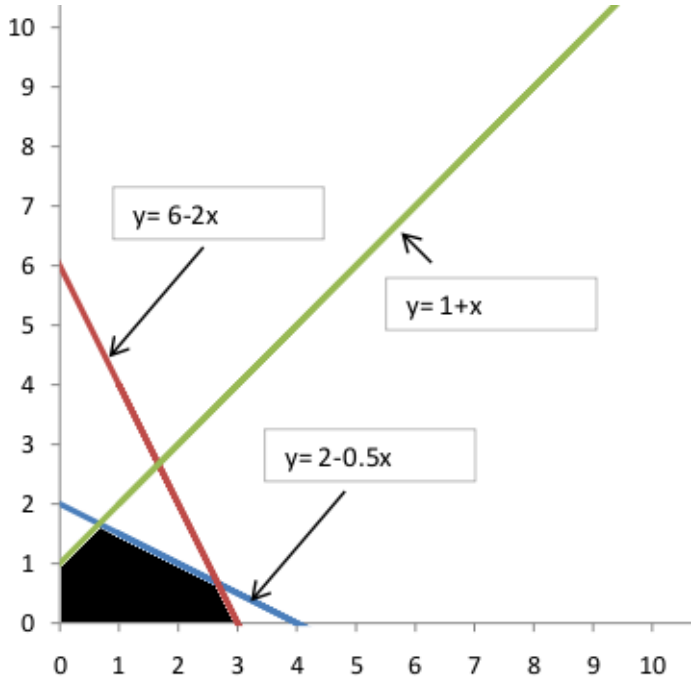


Figure 1.1: Graph plot of equations in Example 1.2

Subject to : $x_1 + 2x_2 \leq 4$
$4x_1 + 2x_2 \leq 12$
$-x_1 + x_2 \leq 1$

with bounds : $x_1 \geq 0, x_2 \geq 0$

The equations in Example 1.2 contains two unknown variables, thus we can solve them by plotting a graph for all the constraints as shown in Figure 1.1. Once the graph is plotted we can search the co-ordinate that maximises the objective function in the solution plane. Each constraint forms a plane on either side of the line obtained from its equation. The bound equation limits the plane to positive axes. The solution plane is the region where all planes of constraints intersect. It is shown in Figure 1.1 by the shaded region. The shaded region has five corners, the objective function being linear and the solution set being bounded, it is always the case that the minimum and maximum value will occur at one of these corners. We can see that the objective function is constant at slope -1, as we can write the objective function as equality $x_1 = -x2$. So, if we draw a line with slope

-1 and start moving it from the origin to the rightmost part of the solution region (as we are maximising the objective function), we will see that the maximum value for the objective function is attained at the intersection of lines $x_1 + 2x_2 \leq 4$ and $4x_1 + 2x_2 \leq 12$. The value for x_1 is 8/3 and the value for x_2 is 2/3 thus the value for objective function $x_1 + x_2$ is 8/3+2/3=10/3.

For this study we use CPLEX solver [3]. The software is free to use for academic purposes. We fed the above example to the solver and it took 0.06 seconds to solve the problem and the following result was obtained:

Objective value : 3.33e+00

Variable Name	**Solution Value**
x1	2.66
x2	0.66

CPLEX not only solves linear programs but also also mixed linear programs. A mixed linear program is a problem when either constraints or the optimisation equation are not linear. This property is essential to our study because we have non-linearity in our optimisation equation. All the equations described later in this report follow the general convention of writing a constraint programming problem as described above.

1.5 Conclusions

In the above sections we discussed various approaches for solving different aspects of sentiment analysis. The above sections shows that opinion lexicons are important for the opinion classification task. In this thesis we aim to provide a single model which can solve both the opinion lexicon extraction and opinion classification task. In addition to this we also aim to investigate other aspects of opinion analysis. We aim to provide a statistical approach to solve the opinion target extraction problem and see how it compares to the rule-based target extraction algorithm. We discussed various features that are used for opinion classification task but few are learned from the data itself. Most of the features are derived from prior knowledge. In this thesis we aim to investigate the usefulness of the features learnt from the opinion data itself.

[3] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Our aim is to investigate each above discussed aspect of opinion analysis and provide simple but effective directions on solving each problem.

Chapter 2

Acquiring Weighted Opinion Lexicon through Constraint Optimisation

2.1 Introduction

A common approach in sentiment analysis is to use an opinion lexicon containing the polarity of the words. For example, a simple opinion lexicon can be represented as:

```
good  : positive
bad   : negative
great : positive
```

One simple method to infer the polarity of opinionated text is by considering the presence of words from the opinion lexicon. For example, the clause "a very *good* movie" will be positive but the clause "a *bad* movie" will be negative and a clause like "a movie" will be neutral provided both "a" and "movie" are not present in the opinion lexicon. Many effective sentiment analysis systems are based on using an opinion lexicon [7, 19, 26].

The task of opinion lexicon building can be divided into two coarse subtasks: 1) *to identify the opinionated words*, and 2) *to identify the polarity of the opinionated words*. Both of these tasks can be challenging. One obvious approach for the solution of the first task is to select all the adjectives from

the text. However, this is not always true. For example, words like "boring" (verb), "lack" (noun), "enjoy" (verb) and "love" (noun/verb). are highly opinionated and not adjectives. Similarly, not all the nouns and verbs are opinionated.

The task of identifying the polarity of the opinionated words is also challenging since the polarity of many words is highly domain dependent. For example, "unpredictable" in "the movie was very *unpredictable*" implies a positive review. On the other hand, in "the steering is very *unpredictable*", "unpredictable" implies a negative review. This property of the opinionated words rules out a general lexicon for all domains.

Also an opinionated word cannot simply be positive or negative, some are less positive/negative than others.

Example 2.1

1. "The product has *value*"
2. "*disappointed* in its *value*"

From sentence 1 in Example 2.1 it is easy to make out that "*value*" is an opinionated word and in this case is positive. Sentence 2 has two words which define the opinion of the whole sentence. Thus, both of these words, "*disappointed*" and "*value*", are opinionated. Sentence 2 implies negativity towards the product. However, one or both of these opinionated words in sentence 2 must be negative for the sentence to be negative. In sentence 1, "*value*" is already regarded as a positive word, thus for sentence 2 to be negative, "*disappointed*" should have the property to negate the positive opinion of the word "*value*".

Thus, an opinion lexicon should be weighted such that a word like "*disappointed*" is more negative than the absolute weight of a word like "*value*". The weighted lexicon may be the following:

```
good   : +10
bad    : -10
great  : +10
disapointed : -4
```

The use of compositionality for sentiment analysis is well explained in [19]. Apart from this paper, there is hardly any other published work on exploit-

ing the compositional nature of opinion-bearing sentences. For example, suppose we have a lexicon as {*"value=+1";"disappointed=-4"*}, then for sentence 1 in Example 2.1 polarity is equal to +1 and for sentence 2 in Example 2.1 it is $-4 + (+1) = -3$. This shows the additive nature of the sentiment-bearing phrases. It also implies that not all the words in the clause contribute to its overall polarity; we conveniently left out *the, product, has, in and its* from the calculation. At this point we have to point out that our work does not directly handle the negative and the positive polarity shifters [35]. For example, words like *"little"* can in some cases completely reverse the polarity of the sentence and in other cases can reduce/increase its polarity strength. For example, "$value''^{[+]}$; "$little\ value''^{[-]}$, "$criticism''^{[-]}$; "$little\ criticism''^{[less\ negative]}$. Handling of such polarity shifters for now is left as future work; intuitively a bigram lexicon might be effective for such cases.

In this chapter we propose an effective way of generating a domain-dependent weighted opinion lexicon through exploitation of the compositional nature of the opinion clauses by modeling the opinion text as a constraint optimisation problem.

Our work further explores the compositional nature of the opinionated text previously explained by [19] and owes a lot to their work.

We begin by describing how the compositional nature of the opinionated text can be used to accurately classify the opinion expressed by such texts. We then propose an additive compositional model which extends the previous compositional model capable of identifying polarity in binary form (positive and negative) by also providing a score for each opinionated text along with the polarity value.

We then show how such scores for the opinionated text on the training data can in turn be used to generate a weighted opinion lexicon and claim that such a lexicon in conjunction with an additive model can identify the polarity in a fully automatic setting as compared to the compositional model proposed in [19].

We then convert the opinionated text of the form {star rating, text} into a set of linear equations. Inspired by the effective use of the constraint programming in the natural language task by [28], we solve the linear equa-

tions through the constraint optimisation. We describe a baseline model, primarily based on the error minimisation for each equation.

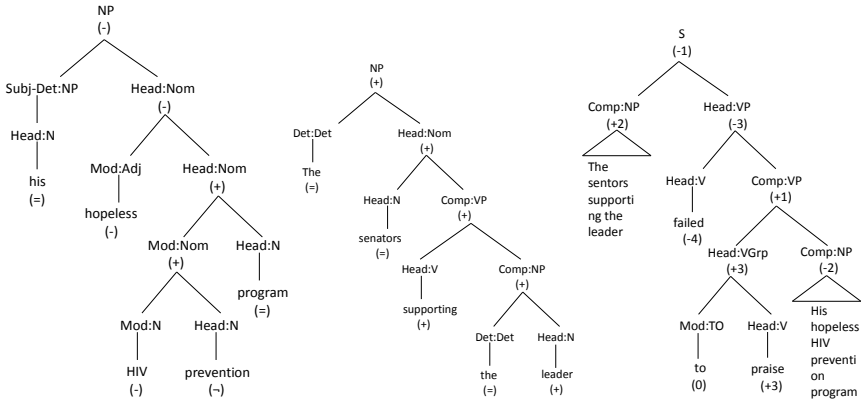
We then show how prior knowledge on opinionated text can be seamlessly integrated in our baseline model to further generate newer models more consistent with the nature shown by the opinionated text. We go on to prove that our additive model works accurately to identify the polarity of the opinionated text through evaluations and comparison with SVM, a popular classifier for the opinion classification task. The results obtained are promising and open gateways for other opinion analysis tasks such as multi-class opinion classification.

2.2 Simple Compositional Model [19]

Example 2.2

> "The senators $supporting^{[+]}$ the $leader^{[+]}$ $failed^{[-]}$ to $praise^{[+]}$ his $hopeless^{[-]}$ $HIV^{[-]}$ prevention program."

Example 2.2has 3 positive and 3 negative words, thus counting just the positive and the negative words would fail to recognise the negative opinion of the whole sentence. Figure 2.1 shows the compositional solution to



(a) Object Noun Phrase 1(b) Object Noun Phrase 2 (c) Sentence

Figure 2.1: Partial and complete parse tree of Example 2.2[19]

sentence 1. The compositionality ($\oplus$) is defined as $\{\texttt{[+]} \oplus \texttt{[=]} \rightarrow \texttt{[+]}\}$; $\{\texttt{[-]} \oplus \texttt{[=]} \rightarrow \texttt{[-]}\}$; $\{\texttt{[+]} \oplus [\neg] \rightarrow \texttt{[-]}\}$; $\{\texttt{[-]} \oplus [\neg] \rightarrow \texttt{[+]}\}$ for the non-

conflicting polarities. If we have a conflicting composition like $\{\texttt{[+]} \oplus \texttt{[-]}\}$ the decision is taken based on the conflict resolution rules.

Figures 2.1a and 2.1b show the object Noun Phrases (NP) of Example 2.2. From Figure 2.1a we can see that the negative sentiment of *HIV* is reversed by ¬*prevention*. The resulting positive sentiment propagates upwards, unaffected by the neutral sentiment of the word *program*. A conflict occurs when compared with the negative sentiment of *hopeless* but the dominance of pre-modifiers in this syntactic situation resolves the conflict and propagates the negative sentiment henceforth. Finally, the neutral sentiment of *his* means that the global sentiment of the whole NP will be negative. In a similar manner we can deduce the global polarity of the NP shown in Figure 2.1b to be positive.

Now, after we combine the two noun phrases and the remaining part of the sentence to form a complete sentence as shown in the Figure 2.1c, the NP [his hopeless HIV prevention program]$^{(-)}$ is reversed when it is combined with a verb group outputting positivity ([to praise]$^{(+)}$). The resultant (+) VP undergoes a polarity reversal through ¬failed, yielding a (-) VP ([failed to praise his hopeless HIV prevention program]$^{(-)}$). Lastly, the (+) subject NP combines with the (-) predicate, while the polarity conflict is resolved by choosing the polarity of dominant the constituent. The global polarity of the sentence will then be negative.

2.3 Weighted Additive Compositional Model

If we maintain a weighted opinion lexicon then the compositional solution proposed by [19] with a non-weighted lexicon can be solved through an additive compositional model. For example, assume the following weighted lexicon:

hopeless	-3	supporting	+1	praise	+3
HIV	-1	leader	+1		
prevention	+2	failed	-4		

The overall sentiment of Example 2.2can therefore be derived as follows:

$$his(0) + hopeless(-3) + HIV(-1) + prevention(+2) + program(0) = -2$$

and

$$The(0) + senator(0) + supporting(+1) + the(0) + leader(+1) = +2$$

thus,

$$(+2) + failed(-4) + to(0) + praise(+3) + (-2) = -1$$

We can see that through a weighted lexicon we can correctly predict the polarity of each of the noun phrases and finally the polarity of the sentence. The manual identification of the dominant constituent for the conflict resolution described in [19] becomes a default case with such a weighted lexicon. For example, the compositional solution of $(hopeless^{-} \oplus (program + HIV + Prevention)^{+})^{-}$ was based on the pre-assumed fact that the syntactic construct of *hopeless* dominates the syntactic construct of $[program + HIV + Prevention]$. Once we have a weighted lexicon we do not have to make such an assumption; we can see that the negative weight of *hopeless* cancelled the positive effect of its adjacent construct to give an overall negative opinion.

We can also observe this problem from another point of view. We have established that a weighted lexicon in conjunction with an additive model can provide an overall polarity score for a text. Following this statement we can say that the reverse of this must also be true, i.e. if a text has a polarity scored assigned to it, then the score is the additive solution of the score of the constituent words in the text. For example, if the review texts are represented as:

$$his + hopeless + HIV + prevention + program = -2$$

and

$$The + senator + supporting + the + leader = +2$$

thus,

$$NP2 + failed + to + praise + NP1 = -1$$

The scores on the right-hand side must come from the score of each word on the left hand-side. Thus, by representing texts by linear sum and solving those equations, we can learn the weight of the constituent words in the text.

POS tag	Examples	POS tag	Examples
AUX	do, done, have, is	NNPS	Americans
CC	and, both, either	PDT	all, both, half
CD	0.5, 1	POS	”s
DT	all, an, the	PRP	hers, herself, him
EX	there	PRP$	her, his, mine
FW	jeux	RP	along, across
IN	astride, among, whether	SYM	&
LS	DS-400, second	TO	to
NNP	Ranzer	WDT	that, what, which

Table 2.1: Parts of Speech tags removed from the review

There are two key advantages of this simple additive model:

1. The sentiment lexicon can be learnt from reviews
2. The overall score assigned to a sentiment text shows the degree to which the polarity is positive or negative.

The focus of this chapter is primarily 1. and although it is clear that our model provides 2. we leave the full evaluation for this as future work.

2.4 Equation Construction

We assume that for training our model, reviews for the given domain are available. Each review is of the form {star rating, review text}, where the star rating represents the polarity ranking of the review, star rating *1* being negative and star rating *5* being positive.

Example 2.3

1. *5* : I would highly recommend this book.
2. *1* : I was disappointed with this book.

Review 1 has star rating *5*, thus it is a positive review. Review 2 has star rating *1*, thus review 2 is a negative review. Before converting these reviews into equations, the review text is passed through various filters. We

used TextCat[1] to remove all non-English text. All the punctuation symbols were removed from the data. We used a morphological analyser and each word was replaced to its base form. For example, "disappointed" will be converted into "disappoint". Words with a certain part of speech that bears no sentiment, for example personal pronouns, were also removed from the text. Table 2.1 shows the full list of parts of speech tags which bear no sentiment. Finally, words were converted to lowercase.

Example 2.4 Sample review dataset after pre-processing

1. *5* : would highly recommend book
2. *1* : be disappoint book.

If each word type in a text is represented by x_i and star rating *1* is represented as -5 and star rating *5* as +5, then, following the discussion in Section 2.3, the additive model of Example 2.4 will be:

$$x_{would} + x_{highly} + x_{recommend} + x_{book} = 5$$
$$x_{be} + x_{disappoint} + x_{book} = -5$$

If we solve these equations simultaneously, one possible solution is:

$$0 + 2 + 3 + 0 = 5$$
$$0 - 5 + 0 = -5$$

yielding a lexicon $\{x_{would} = 0;\ x_{highly} = 2;\ x_{recommend} = 3;\ x_{disappoint} = -5\}$. Words that occur frequently in positive reviews are positive in nature and vice-versa. Also words that occur frequently in both positive and negative reviews tend to be neutral, for example "*book*" in the above case. Thus representing reviews as additive equations provides a mechanism to generate a weighted lexicon. We can observe from this example that not all the words in the opinionated text have certain values. Some words are neutral and their value should be set to zero.

In a bag-of-words representation, each dictionary word is represented by a variable x_i and the {star rating, review text} pair is represented by the equation:

$$\sum_{i=1}^{n} x_i = s$$

[1] http://www.let.rug.nl/~vannoord/TextCat/

where s represents the star rating.

As the input is noisy, when the number of variables and the number of equations both increase, a solution to the set of equations cannot be found. To overcome this problem, we introduce an error variable E_t for each equation. Following this formulation our example equations will be of the form:

$$x_{would} + x_{highly} + x_{recommend} + x_{book} - 5 = E_1$$

$$x_{be} + x_{disappoint} + x_{book} + 5 = E_2$$

Thus our equation can be re-written as:

$$\sum_{i=1}^{n} x_i - s = E_k$$

When a perfect solution is achieved, values of all the error variables will be zero. With this basic setup we build three different models to solve the opinion lexicon acquisition problem.

2.4.1 Baseline (MinErr)

We have established our basic form of the equation and also found out that the optimum solution desired is the one where all the error values are set to zero. Thus, the problem can be viewed as a minimisation problem, where the aim will be to minimise the sum of all the error variables.

Let $\{(X^1, S^1)....(X^R, S^R)\}$ denote the set of {review, star rating} pairs. Each X^i is a bag-of-words $\{x_1^i, ...x_{|X^i|}^i\}$. Multiple instances of the same word share the same variable in the equations. Thus our baseline model which minimises the error in each equation (MinErr) will be:

`Minimise`:

$$\sum_{k=1}^{R} |E_k|$$

`subject to` :

$$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R\}$$

`with bounds :`

$$-10 \leq x_i^k \leq +10 \quad : \forall k \; \forall i \; 1 \leq k \leq R \; , -1 \leq i \leq |X^k|$$

$$-100 \leq E_k \leq +100 \quad : \forall k \; 1 \leq k \leq R$$

where, -10 for x_i^k represents the highest weighted negative polarity and $+10$ represents the highest weighted positive polarity, and the range for E_k represents the allowed error range for each equation.

To balance the number of words in each review we set the value of "s" for the review with star rating *1* as -100 and for the review with star rating *5* we set it +100. Minimising the absolute value of E causes X to be as close as possible to its respective S value. The baseline model is purely based on the classical solution for the collection of linear equations where the aim is to minimise the error. This equation holds true for any graded text. Further models proposed in this chapter are more native to the characteristics of the opinionated text.

2.4.2 Force zeroes (MinErrFZ)

Following the discussion in Section 2.1 and Section 2.3, it is desirable that most of the words in the text have a value zero. Only a small number of words show sentiment and only they decide on the global polarity of the sentence. Thus we need to incorporate this prior knowledge into our model. This can be achieved by introducing a squared loss function.

`Minimise`:

$$\sum_{k=1}^{R} |E_k| + \sum_{i=1}^{|X^k|} (x_i)^2$$

`subject to :`

$$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R \}$$

`with bounds :`

$$-10 \leq x_i^k \leq +10 \quad : \forall k \; \forall i \; 1 \leq k \leq R \; , -1 \leq i \leq |X^k|$$

Dependency triples	Rules	Example
$amod(arg_1, arg_2)$	$\|arg_2\| > \|arg_1\|$, if and only if arg_1 is NN	*"trivial problem"*
$advmod(arg_1, arg_2)$	$\|arg_2\| > \|arg_1\|$, if and only if arg_1 is NN or RB	*"decreasingly happy"*
$pobj(arg_1, arg_2)$	$\|arg_1\| > \|arg_2\|$, if and only if arg_2 is NN	*"against racism"*

Table 2.2: Conflict resolution rules.

$$-100 \leq E_k \leq +100 \quad : \forall k \; 1 \leq k \leq R$$

The squared loss $\sum_{i=1}^{|X^k|}(x_i)^2$ forces the values of most variables to be close or equal to zero.

2.4.3 Conflict Resolution (MinErrFZCR)

In [19] approach, conflict resolution is a method dealing with the case when two adjacent words have different polarities. The decision is taken on the basis of the parts of speech of competing words. For example, in Example 2.2($[Mod : Adj]hopeless^- \oplus [Head : Nom]((program + HIV + Prevention))^+)^-$, overall negative polarity is chosen since adjectives are considered more opinionated than the nominal phrase [4].

Such linguistic knowledge can be easily incorporated into our model to further improve the quality of the learnt lexicon. Table 2.2 shows some of the conflict resolution rules in terms of dependency[2] triples. Following [19] we only consider dependency triples of the form $modifier(arg_1, arg_2)$. Conflict resolution imposes a stronger constraint that states that when conflicts arise in certain $modifier(arg_1, agr_2)$ constructions it is a priori the case that $|arg_1| > |arg_2|$ when the modifier is *pobj* and $|arg_2| > |arg_1|$ when the modifier is either *amod* or *advmod*. Rules in Table 2.2 are too general and are only intended to be applicable when two competing words have *opposite polarities*. Thus, when we have a neutral construction such as "*financial hub*" with the dependency relation *amod* $(hub, financial)$, generating a constraint such as $x_{financial} > x_{hub}$ does not make any sense.

To overcome this problem we implement conflict resolution as a two-step process. First we parse the full review text to generate a dependency parse of the review. Next, for all $amod(x_a, x_b)$, $advmod(x_a, x_b)$ and $pobj(x_b, x_a)$

[2]The dependency parser used in this case is Stanford's dependency parser[3], and all the abbreviations have their usual meaning.

relations, we employ the MinErrFZ model to find out the polarities of x_a and x_b. Then, whenever $sign(x_a) \neq sign(x_b)$ we add a new constraint $|x_b| > |x_a|$.

Algorithm 1

```
1. L = learn an opinion lexicon through MinErrFZ
2. R = dependency relations for confict resolution like
       advmod,pobj etc.
3. D = dependency parse the dataset
4. C = constraint of form arg2 > arg1, initially null
5. for each dependency triples in D
6.   Sel_D=select dependency triple which are in R
7.   for each dependency triple in Sel_D
8.      apply opinion to argument from L
9.      if sign of arg2 is not equal to sign of arg1
10.       add constraint arg2 > agr1 to C
```

Algorithm 1 describes the selection method for a conflicting pair.

The new minimisation model obtained will thus be:

Minimise:

$$\sum_{k=1}^{R} |E_k| + \sum_{i=1}^{|X^k|} (x_i)^2$$

subject to :

$$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R \}$$

and

$$|x_b| > |x_a| \text{ for each conflicting pair } (x_a, x_b)$$

with bounds :

$$-10 \leq x_i^k \leq +10 \quad : \forall k \; \forall i \; 1 \leq k \leq R \; , -1 \leq i \leq |X^k|$$

$$-100 \leq E_k \leq +100 \quad : \forall k \; 1 \leq k \leq R$$

where a and b are the indices of selected words.

Setting	Positive	Negative	Neutral
MinErr	1195	881	0
MinErrFZ	328	136	1612
MinErrFZCR	540	360	1176

Table 2.3: Number of positive, negative and neutral words in each setting.

The added constraint $|x_b| > |x_a|$ adds prior knowledge about the opinionated text into our model.

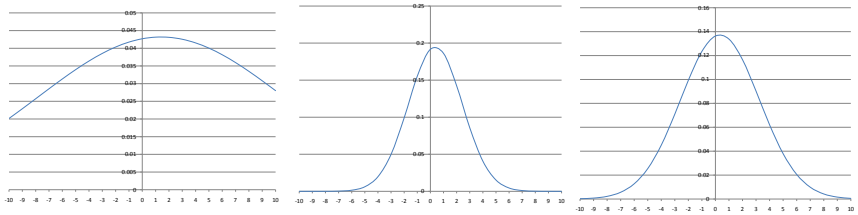
2.5 Experiments

For all the experiments shown here we used the Multi-Domain Sentiment Dataset[4] that contains product reviews taken from Amazon.com. We chose book reviews for our experiment. All the reviews in the dataset have a helpfulness ranking. This score shows whether the review written is liked by the reader or not. This counts the number of "likes" and "dislikes" posted by the reader for reviews. A review with its number of "likes" set at zero is generally spam. These are the reviews which provide no significant information and sometimes are even misleading. For example, a review which was read by 11 people and none of them liked it has a sentence ' *"I did not like the book because it's a country book"*. The sentence is very misleading since books have genres, and to point out that one does not like the book because it is a "country book" is an improper review. Thus such reviews are filtered in the initial phase. From the remaining reviews we extracted 1000 reviews with star rating *1* as negative reviews and 1000 reviews with star rating *5* as positive reviews. All the experiments were conducted using the ILOG cplex solver[5].

Table 2.3 shows the distribution within the generated lexicons from each model. We can see that MinErr generates no neutral words, i.e. all the words are labelled as either positive or negative. The distribution of the lexicon value generated by MinErr as shown in Figure 2.2a shows a high percentage region around zero (but no actual zeroes) and a fairly flat distribution.

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

[5] http://www-01.ibm.com/software/integration/optimisation/cplex-optimizer/



(a) MinErr (b) MinErrFZ (c) MinErrFZCR

Figure 2.2: Distribution of opinion words in the generated lexicon. The x-axis corresponds to the polarity and the y-axis corresponds to the total number of words.

MinErrRZ generates the highest proportion of neutral words and adding conflict resolution decreases the number of neutral words. The distribution diagram reflects the nature of our models. MinErr was about minimising the equation error, thus the optimal solution was more concerned with error value than the actual value of the words. This is the reason we did not get any zeroes as expected (since opinionated text has many neutral words) from MinErr. But the distribution shows a clear high percentage data around zero, thus even though there were no actual zeroes, the high percentage of data around zero supports our claim about the nature of opinionated text and in turn shows that linear additive equations can model opinionated text. The squared loss function forced most of the words to attain a zero value and finally the MinErrFZCR with the conflict rules reduced the number of zeroes or neutral variables. The effectiveness of each model can be further seen in the evaluation section (2.5.1).

We can observe that the number of positive opinion words is higher than negative opinion words. This is due to the fact that people tend to express negative opinion in most cases by actually negating the positive word rather than using the negative words itself. For example, using "*not impressive*" instead of "*unimpressive*".

Some of the constraints added by MinErrFZCR are

- *frustrating* > *reading*
- *useless* > *joke*
- *terrible* > *book*

MinErr			MinErrFZ			MinErrFZCR		
Positive	**Negative**	**Neutral**	**Positive**	**Negative**	**Neutral**	**Positive**	**Negative**	**Neutral**
skill	website		love	disappoint	french	clear	disappoint	website
dedicate	gross		highly	nothing	dozen	love	poor	sticker
2nd	disappoint		wonderful	bad	please	fun	bad	future
carefully	dry		life	instead	conclusion	highly	poorly	sister
stream	worthless		good	poorly	publish	favourite	completely	teaching
thank	error		recommend	however	book	wonderful	useless	relation
concisely	file		read	money	review	life	disappointing	turner
nicely	sorry		must	poor	someone	excellent	error	collect
warm	cent		excellent	useless	pics	great	boring	desk
essential	difficult		easy	little	people	easy	little	version

Table 2.4: Top 10 positive, negative and neutral opinion words learnt in each setting

Polarity	Count	Examples
Positive	134	reliable; commercial; nice; apprehensive
Negative	91	implausible; long; unglorified; incorrect

Table 2.5: Polarity statistics of the adjectives from the dataset

- *outdated > information*

Table 2.4 shows the top 10 ranked positive and negative words along with the neutral words in each setting. The neutral column for MinErr is empty because no neutral words, or words with a value exactly zero, were generated in MinErr. The ordering of some of the words are changed when comparing MinErr with MinErrFZ. For example, in a positive column of MinErrFZCR, words like *clear, great, favourite* are introduced in the top 10 which are clearly missing in MinErr. For a book review, all of these words are significant for a positive review. Similarly in the negative column of MinErrFZCR, words like *error and boring* have made their way to the top 10. These words are absent from the top 10 in MinErrFZ. Also, the word *useless* has gone up the ranks in MinErrFZCR as compared to MinErrFZ.

2.5.1 Evaluation

Precision-Recall

The first part of the experiment is the evaluation of the lexicon extracted in terms of precision and recall. To carry out this experiment we extracted the sentiment-bearing word from the dataset. Adjectives are considered to be highly opinionated and thus we only extracted adjectives from the data

Negative	Positive
stupid	nice
confusing	good
horrible	ultimate
same	open
useless	spiritual
difficult	comprehensive
disappointing	terrific
online	worth
real	serious
obvious	wonderful
free	cool
basic	glad
outdated	essential

Table 2.6: Positive and negative adjectives extracted from the algorithm

Polarity	Precision	Recall
Positive	0.95	0.52
Negative	0.85	0.33

Table 2.7: Precision and recall for positive and negative adjectives extracted

set. The adjectives are manually annotated [6] with the polarity based on the context that they are used. Table 2.5 shows the count for both positive and negative adjectives in the dataset.

The lexicon generated from each setting namely MinErr, MinErrFZ and MinErrFZCR, contains words which are adjectives along with words which have other parts of speech. For this experiment we chose to use the lexicon generated from MinErrFZCR. From all the positive and negative words generated by MinErrFZCR we only extracted those words which are adjectives. Table 2.6 lists some of the adjectives that are selected from the whole lexicon.

The precision and recall for the extracted adjectives is shown in Table 2.7. The precision for both the positive and negative adjectives is very high. This shows that our method works well to extract the correct opinion for the words. However, on the other hand the recall is low. The low recall is

[6]The annotation was done by a single person; a better approach would be to use two annotators and compute the agreement between them.

Positive	Value	Negative	Value
clear	10	disappoint	-10
love	10	poor	-10
fun	10	bad	-10
highly	10	poorly	-10
informative	7.49	completely	-10
unique	6.98	outdated	-7.67
depth	6.91	return	-6.67
nicely	5.14	similar	-4.93
intermediate	3.45	repeat	-3.93
colorful	3	sleep	-3

Table 2.8: Sample lexicon generated using MinErrFZCR along with their polarity values.

due to the less frequent occurrence of the annotated words in the data set. Experiments showed that words like *"satanic", "blatant", "unabridged"* and many others were used just once in the dataset. Our algorithm performs poorly when the words are used infrequently in the dataset. Most of the time such words are coined as neutral by our algorithm. Also we have to keep in mind that these are just the adjective part of our whole lexicon. The lexicon contains other words which are not adjectives but are still highly opinionated. Even though the recall for the adjective is very low, the high precision obtained for all the extracted adjectives and the significant percentage of non-adjectives in the lexicon proves that our algorithm works significantly well to extract opinionated words from the text.

Accuracy

The next part of the evaluation is based on polarity detection using our generated lexicon.*Accuracy* measures the total number of reviews correctly identified by using the generated lexicon in an additive compositional model. We used 10-fold cross validation with the dataset for evaluation.

We follow the same pre-processing steps as during training to generate the bag-of-words corresponding to each review. We then take a test review, replace the words with their respective value (zero value assigned for unknown words) from the learnt opinion lexicon and add them. If the sum results in a negative value then the review is deemed negative, but if the

Setting	**Positive**	**Negative**	**Overall**
MinErr	83.7%	80.5%	82.1%
MinErrFZ	90.6%	71%	80.8%
MinErrFZCR	88.2%	87.6%	87.9%

Table 2.9: Accuracy scores on opinion classification.

	Features	**Positive**	**Negative**	**Overall**
a)	All words	88.45%	84.67%	86.56%
b)	Top ranked features by SVM	90.76%	86.14%	88.45%
c)	Positive and negative terms generated by MinErrFZCR	91.86%	89.54%	90.70%

Table 2.10: Accuracy for opinion classification using SVM.

sum results in a positive value then the review is deemed positive. For example, the sentence "*This book was totally disjointed*" will have values, $this(0)$, $book(0)$, $was(0)$, $totally(-3.00)$,
$disjointed(-3.39)$, resulting in the equation:

$$0 + 0 + 0 - 3.00 - 3.39 = -6.39$$

Thus we will classify the above sentence as *negative*.

Table 2.9 shows the accuracy obtained by each setting on the test set. The high accuracy on MinErr supports our initial claim that opinionated text is compositional, and an additive model works well for classifying opinionated text.

From Table 2.9, we also observe that for MinErrFZ there is a major dip in accuracy in negative text. This is due to the fact that MinErrFZ is more biased towards positive text. We can see from Table 2.3 that among all the three settings MinErrFZ has the highest positive-to-negative ratio, 2.4 compared to 1.3 and 1.5 of MinErr and MinErrFZ, respectively. Among the three settings MinErrFZCR has the highest average accuracy of 87.9%. This is a clear improvement over our baseline accuracy of 82.1% and is highly statistically significant ($p < 0.01$, paired t-test). This shows that the automatic method of resolving the conflict in the opinion text worked very well to increase the accuracy of the system.

Next we ran an SVM[7] on the same dataset (prepared following the same pre-processing steps described in 2.4) with unigram as a feature. On the test set, using the SVM resulted in 86.56% accuracy. The accuracy obtained by our best-performing system is slightly higher than the accuracy obtained by SVM (however, this is not statistically significant).

Next, to test the effectiveness of the lexicon obtained by our method, we compared it with the lexicon obtained by SVM. To extract the lexicon learnt by the SVM we used a feature selection method to extract top-weighted unigrams [14]. We extracted the top-ranked unigrams from the SVM classifier trained on the training dataset. Then the SVM was trained again using just the extracted features on the same training dataset.

Also a separate SVM was trained by using only a lexicon generated by MinErrFZCR. The idea behind this setting is that this would reveal the effectiveness of our lexicon when compared to features selected by SVM.

Following this setting, we can see from Table 2.10 that the accuracy obtained by SVM trained on a lexicon generated by MinErrFZCR is higher and is statistically significant ($p < 0.05$, paired t-test) than that obtained by SVM trained on a lexicon generated by SVM itself. Thus we can say that our method of opinion lexicon generation works significantly well.

To show that the constraint optimisation model adds value when compared with the less expensive methods, we performed a baseline experiment using association-based method. The idea behind this method is to generate words which are highly associated with either positive reviews or negative reviews. Using the association method we can also generate weight of the association. Thus, two separate lexicons will result, one containing words with positive polarity and the other containing words with negative polarity. We used the Log-Likelihood (LL) method on our training dataset (prepared following the same pre-processing steps described in 2.4) with a unigram as a feature. The data preparation, the motivation and the formulas to implement LL method are described in detail in Chapter 5.

Table 2.11 shows the positive and negative words generated from the LL method. For each word in the lexicon appearing in both the positive and the negative lists, a choice is made to remove it from one of the lists. The

[7] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Positive	**Value**	**Negative**	**Value**
great	+79387.09	not	-79336.56
easy	+79390.89	money	-79360.95
do	+79401.71	waste	-79373.34
no	+79403.9	disappoint	-79405.2
content	+79407.5	poorly	-79415.32
must	+79409	disappointment	-79417.44
excellent	+79409.42	instead	-79417.68
nothing	+79411.07	error	-79418.85
love	+79412.19	useless	-79418.85
read	+79416.15	enjoy	-79420.49

Table 2.11: Sample lexicon generated using LL along with their polarity values.

choice is based on the index (a highly associated word gets 1 and so forth) and the weight assigned to the word. Priority is given to the index rather than the weight. If both the index and the weight for the word appearing in both the lexicon are the same, then the word is termed neutral and assigned weight 0. The sign(+/-) for all the words in the positive lexicon is assigned "+" and "-" for the words in the negative lexicon. Applying the LL lexicon to the additive model classified the test set into positive and negative reviews with 63.5% accuracy. The accuracy obtained is significantly less than the accuracy achieved by the baseline method (MinErr). Therefore, we can claim that the constraint optimisation method is more suitable for generating opinion lexicons than the association method. Thus, the additional resources requirement for implementing a constraint optimisation method for opinion lexicon generation is rational.

For further comparison we applied an additive model to the dataset used later in Chapter 4. The dataset used is the one used in Pang et al. [24] and it consists of 1,000 positive and 1,000 negative movie reviews. The dataset was filtered using all the pre-processing tasks mentioned above. Accuracy is measured in a similar setting as mentioned above.

Table 2.12 shows the accuracy obtained in the movie review dataset. The nature of the accuracy for all the 3 different settings matches the discussion we have provided above. Table 2.13 shows the accuracy obtained by the SVM after applying various feature selection techniques (explained in Chapter 4). The first 4 rows show the accuracy taken from methods used in

Setting	Positive	Negative	Overall
MinErr	84.3%	81.5%	82.9%
MinErrFZ	88.9%	82.1%	85.5%
MinErrFZCR	93.3%	89.6%	91.45%

Table 2.12: Accuracy scores on opinion classification for the movie review dataset.

Chapter 4. Among them, the feature selection technique which uses selected unigrams and frequent sub-sequences outperforms the rest with an accuracy score of 97.69%. Next we used, as features, positive and negative words generated by MinErrFZCR. The accuracy obtained in this case is 94.0%. This is an acceptable score and thus further strengthens our claim that the lexicon generated by our system is of significance to polarity detection and performs considerably well when applied for the same.

Features	Accuracy %
Unigram	85.0
Unigram selected	86.3
Unigram + frequent sub-sequences	85.8434
Unigram selected + frequent sub-sequences	97.69
Positive and negative terms generated by MinErrFZCR	94.0

Table 2.13: Accuracy before and after feature selection

2.6 Related Work

The compositional nature of opinion expression has already been shown in [19], but linear programming has not been used to date to exploit the compositional nature for learning weighted opinion lexicons. In this paper we show how linear equations can be formulated for opinion expression which exploits its compositional nature (Section 2.2). A weighted opinion lexicon can be derived by solving such equations.

Our work can be related to the sentiment compositional model introduced in [19] in the sense that both uses the compositional property of opinionated text. In [19] an opinion lexicon is given and opinion classification is done by using handcrafted rules. Thus their approach can be considered more manual than automatic. In contrast, we exploit the compositional nature

of the opinionated text to learn an additive model. The model results in a sentiment lexicon that can be effectively used for classifying sentiment text. Thus, unlike [19], we do not pre-assume an opinion lexicon but instead generate it automatically.

Since not all the words occurring in the opinion clauses bear sentiment, the weight of most of the clause in the lexicon should be zero or, in other words be absent from the lexicon. Instead of searching opinion words in text as done in [16, 15, 12, 10], we show an efficient approach which forces most of the words towards zero, thus generating non-zero words as polarity words. Within our framework the task of identifying opinionated words and their polarity is solved simultaneously through constraints optimisation.

Certain parts of speech always have higher sentiment weight than others [4]. For example, adjectives are considered to be more opinionated than nouns in most cases. Previous literature shows that such rules can be used with high effect whenever the compositional nature of the opinion text is exploited [19]. We show how such linguistic knowledge can be incorporated into our model.

Minimum-cuts also a type of linear programming, have been used to classify opinion text into positive and negative [21]. The statistically significant result obtained by incorporating context information like sentence proximity in the minimum-cut framework shows that linear programming can be used successfully for opinion classification. Instead of classifying text, we used linear programming for classifying words and also incorporated different prior knowledge like conflict resolution.

Another work closely related to our work is the Integer Linear Programming (ILP) approach to adapt a polarity lexicon to a specific domain by [7]. They employed the relation between the word and the sentiment expression to calculate the degree of polarity of the word. They defined the degree of polarity of the word by the number of different sentiment expressions containing the word. The primary disadvantage of their approach is the fact that an initial sentiment lexicon is needed in their approach. In contrast, our method can be employed to directly learn a domain-specific weighted sentiment lexicon.

2.7 Conclusions

In this chapter we developed a weighted additive model for sentiment classification. Our model allows learning a weighted opinion lexicon through constraint optimisation. We also showed how prior knowledge of the domain can be seamlessly integrated into our model to obtain improved results. Finally, we showed that using the acquired lexicons as features within an SVM allows us to obtain state-of-the-art results on opinion classification.

Chapter 3

Multi-class and Features-Related Opinion Classification

3.1 Introduction

Many papers define Opinion Classification as classification of opinion text into one of the two classes, namely, positive and negative [24, 20, 32, 19, 31]. This is true in many cases, and the most fundamental thinking of anyone reviewing a product, whether the product is *a camera, a movie, a food item* or even *an election candidate*, is that either you like that product or you don't. However this is a very generalised point of view; this is what we say when we talk about a product with friends or when we are in real a hurry. Written reviews are more elaborate. This is driven by the need for information. You would not want to go out and buy a camera without knowing the pros and cons of its every aspect (*lens, viewfinder, body* etc.), and you would still go to a restaurant which serves *bad desserts* but has amazing fish. Consumers' taste are different; some would not mind a mind-boggling action flick with mediocre acting and some would prefer a mediocre plot with awesome acting. So there is certainly a need for elaborate information, but why we have all this information? The answer is "*features*".

Every product has *features*. A camera has *lens, viewfinder, body* etc; a movie

has *actors, acting, plot, direction*; an election candidate has his or her own set of *policies*. In a best case scenario or in worst case scenario all the features will get a positive review or negative review respectively. But there are also cases when some features get positive reviews while others get negative reviews. In a review data set taken from Amazon, where a star rating of 1 is termed negative and 5 is termed positive, of 3791 reviews 1180 reviews were rated with stars 2 and 4. This statistic states that approximately 31% of the reviews were neither positive nor negative. Thus a document opinion classification which separates a positive review from negative ones starts off with a 31% error rate without even seeing any training data. This is a major problem, a problem which arises from sentences like, *The camera is fine but the bland brown colour is not eye catching*. This sentence is taken from a review about a camera from Amazon which has a star rating of 4.

There is a need to divert from binary classification of opinion to more elaborate graded classification. There are possibly two directions that can be taken. One is to model opinion analysis as multi-class classification where each class resembles the magnitude of the sentiment in the document. For example, there could be 5 classes, each class representing a star rating in reviews, where 1 is the most negative and 5 is the most positive. The other direction is to model opinion analysis as opinion on features instead of opinion on the whole product. The system would output a result after reading a review of a camera under three categories {**Feature** : **Opinion** : **Sentence**} where "**Feature**" is the specific feature of the product, "**Opinion**" is the opinion expressed in the product either positive or negative and "**Sentence**" is the sentence in the review showing the opinion. An example output for a camera review could be:

Feature	**Opinion**	**Sentence**
`lens`	`Positive`	`the lens in the camera is spectacular`
`viewfinder`	`Negative`	`you can barely look through the viewfinder`
`body`	`Positive`	`the body is light and sturdy`

In this report we show the implementation on both the directions. The major contributions of this study are:

- We propose a diversion from the classical model of opinion classifica-

tion as a two-class model to a multi-class model. We show a method of classifying opinionated text into multiple classes, namely 4 classes termed as 1 star, 2 star, 4 star and 5 star. We left out star value 3 because these reviews are more neutral in nature and lack distinction from 2 star and 4 star. We propose a method based on mixed linear programming where classification is done based on the weights attained by each feature sentence. This is a more fine-grained method of classifying opinionated text than blindly counting the number of positive sentences and negative sentences. For example, in the review text "*The lens is very good. The viewfinder is bad.*", there is 1 positive sentence and 1 negative sentence, thus a count of positive or negative sentence would lead us nowhere. But we propose a system which weighs the opinion and classifies the text accordingly. In the above sentence, since the positive review is more stressed (i.e. the sentiment intensifier *very* is used) than the negative, the product has a higher chance of getting 4 stars, which is plausible.

- We also propose a fine-grained opinion analysis model where an opinion text or review is split into sentences reviewing different features and an opinion (positive or negative) is assigned to such sentences. The concept of fine grain analysis of opinion text is well established in the opinion analysis domain. We propose a method of identifying such sentences from the review by using targets (opinion feature), opinionated words and grammatical constructs. The target is extracted from an annotated review. The opinionated words are extracted by using the lexicon extraction method described in Chapter 2. The method successfully extracts opinionated sentences on a particular feature from the review text. The opinion assignment on the extracted sentence is done according to the weighted lexicon and compositionality shown by the opinionated text.

3.2 Motivation for using Mixed Linear Programming

The major motivation for using Mixed Linear Programming is because of the way the compositional nature of the opinionated text lends itself to linear programming formulations. The compositionality of the opinionated text is explained in Chapter 2 Section 2.2. In very simple terms, the additive model for opinionated text is: *selected* words which *adds* up to give a certain rating (star value) to the text. For example,

- **Text 1 :** good lens + attractive design + excellent speed = 5
- **Text 2 :** good lens + average shutter speed = 4
- **Text 3 :** worst camera + horrible = 1

The example shows the probable words in any review which may have a star rating of 5 or 4 or 1. The words which are not opinionated are removed. We can see that each text can be represented as a linear equation. Solving the collection of these equations gives us an estimate for each variable, resulting in an opinion lexicon. The compositional nature of opinionated text which lends itself as a linear equation fits perfectly to the linear programming paradigm which makes it obvious to use linear programming to solve the problem.

One of the prominent requirements of such an additive model is that not all the words in the text can be added, i.e not all the words in the text have an opinion. For example, a full sentence for Text 1 could be *The brown camera that came out today has a good lens.* The additive model just extracts the phrase *good terms* from the whole sentence. What we are effectively saying is that all the words in the sentence other than *good* and *lens* should have their values set to zero. Thus if we form our linear equations with all the words in the sentence, we need to have a global function which forces the value of each variable to be zero unless otherwise absolutely necessary. Such condition can be implemented easily with the use of an objective function of the linear equation. The ease in the usability and implementation is another motivating factor for using linear programming. A simple solution to achieve above condition would be to write an objective function which

minimises the value of each variable. This will cause the linear program to select the solution in which the words have as minimum value as possible. In our case, for better results we set the objective function to minimise the square of each variable. Since our objective function is not linear we used mixed linear programming.

In an opinion analysis system and especially in our model where we want to separate opinions into multiple classes, variables (words) should have different weights. A linear programming approach will effectively result in weighted variables following an optimal setting. In addition to this, a linear programming setting can also use prior knowledge of data. The prior knowledge can be easily added in as constraints. From the work of Moilanen et al. [19] we know that certain grammatical constructs outweigh other in terms of opinion expression. For example, an adverb modifier is weighted higher than the noun it modifies, e.g. $trivial >> problem$. For a more detailed explanation of such rules see Chapter 2 Section 2.4.3. To incorporate such knowledge into our system we just have to add some constraints which state that certain variables are greater than others. Thus, because of the way the whole idea of opinion analysis fits easily in the mixed linear programming and the desired output also fits the output of any linear programming system, the use of mixed linear programming becomes obvious.

3.3 Multi-class opinion classification

We classify opinionated text in terms of stars, 1-4, increasing in positivity. We have already shown in Chapter 2 an opinion on a sentence cannot be always inferred correctly by counting the number of positive and negative words in the sentence. On the other hand, opinionated sentences show compositionality [19], which can be used to correctly classify a positive opinion from a negative opinion. We showed in Chapter 2 how the compositional property of opinionated text can be used to form a linear additive model from which we could learn a weighted opinion lexicon. Such weighted opinion lexicons not only could classify the positive opinion from a negative opinion but also can give a degree to the polarity assigned. For example,

$$The(0) + senator(0) + supporting(+1) + the(0) + leader(+1) = +2$$

We can see that the additive model not only classifies the sentence as positive but also gives a score to the sentence, in this case 2. We use such scores to classify opinionated text into multiple classes. Thus, our multi-class opinion classification completes in two steps: first we learn a weighted opinion lexicon from the data, then, use this opinion lexicon under a linear additive model for classification.

3.3.1 Experimental Settings

Data set

For all the experiments shown here we used the Multi-Domain Sentiment Dataset[1] that contains product reviews taken from Amazon.com. The training set consists of reviews with star ratings 1, 2, 4 and 5. Star rating 3 was left out because there is no clear separation between the text in star rating 3 with text in either star rating 2 or star rating 4.

Equation Construction

Each review is of the form {star rating, review text} where the star rating represents the graded polarity of the text and ranges from 1 to 5 in increasing order of polarity. Following the approach taken in chapter 2, in a bag-of-words representation each dictionary word is represented by a variable x_i and the {star rating, review text} pair is represented by the equation:

$$\sum_{i=1}^{n} x_i - s = E$$

where, s represents the star rating. The star values are assigned as two graded values, where each star rating has an opposite pair. Star rating 1 has the value -100 and its opposite pair, i.e. star rating 5, has the value +100. Similarly star rating 2 has the value -50 and its opposite pair, i.e. star rating 4, has the value +50. Figure 3.1 shows a graphical representation of the star rating value across the y-axis. E represents the error value allowed

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

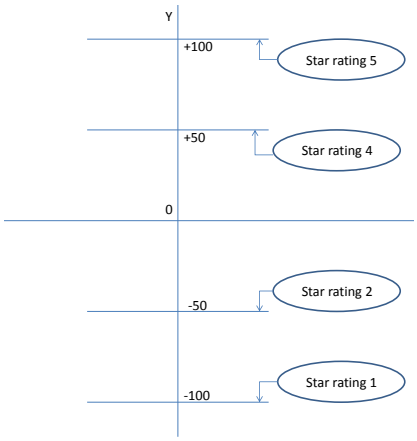


Figure 3.1: Star scale along y-axis

so that even noisy data can have a solution. In case of a perfect solution the value of E will be zero.

For completion we will revisit our three minimisation models already described in detail in chapter 2:

1. **Baseline (MinErr)**
   Let $\{ (X^1, S^1)....(X^R, S^R)\}$ denote the set of (review,star rating) pairs. Each X^i is a bag-of-words $\{x_1^i, ...x_{|X^i|}^i\}$. Multiple instances of the same word share the same variable in the equations.

   `Minimise`:

   $$\sum_{k=1}^{R} \mid E_k \mid$$

   `subject to` :

   $$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R\}$$

   `with bounds` :

   $$-10 \leq x_i^k \leq +10 \quad : \forall\ k\ \forall i\ 1 \leq k \leq R\ , -1 \leq i \leq \mid X^k \mid$$

   $$-100 \leq E_k \leq +100 \quad : \forall\ k\ 1 \leq k \leq R$$

   where, -10 for x_i^k represents the highest weighted negative polarity and $+10$ represents the highest weighted positive

polarity,
while the range for E_k represents the allowed error range for each equation.

2. **Force zeroes (MinErrFZ)**

`Minimise`:

$$\sum_{k=1}^{R} \mid E_k \mid + \sum_{i=1}^{|X^k|} (x_i)^2$$

`subject to` :

$$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R\}$$

`with bounds` :

$$-10 \leq x_i^k \leq +10 \quad : \forall\ k\ \forall i\ 1 \leq k \leq R\ , -1 \leq i \leq \mid X^k \mid$$

$$-100 \leq E_k \leq +100 \quad : \forall\ k\ 1 \leq k \leq R$$

3. **Conflict Resolution (MinErrFZCR)**

`Minimise`:

$$\sum_{k=1}^{R} \mid E_k \mid + \sum_{i=1}^{|X^k|} (x_i)^2$$

`subject to` :

$$\{ \sum_{i=1}^{|X^1|} x_i^1 - s^1 = E_1, ..., \sum_{i=1}^{|X^R|} x_i^R - s^R = E_R\}$$

and

$$\mid x_b \mid > \mid x_a \mid \ \text{for each conflicting pair } (x_a, x_b)$$

`with bounds` :

$$-10 \leq x_i^k \leq +10 \quad : \forall\ k\ \forall i\ 1 \leq k \leq R\ , -1 \leq i \leq \mid X^k \mid$$

$$-100 \leq E_k \leq +100 \quad : \forall\ k\ 1 \leq k \leq R$$

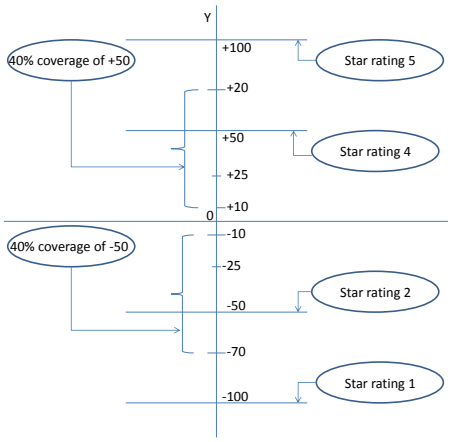


Figure 3.2: 40% coverage for +50 and -50

Experiments

For all the experiments shown here we used the Multi-Domain Sentiment Dataset[2] that contains product reviews taken from Amazon.com. We chose book reviews for our experiment. For training we extracted 600 reviews with star rating 1, 300 reviews with star rating 2, 300 reviews with star rating 4 and 600 reviews with star rating 5. Since the observations showed that there can be similarities between star rating 2 and star rating 4, to scale the training data we only extracted 300 reviews from each.

Before converting these reviews into equations, the review text was passed through various filters. We used TextCat[3] to remove all non-English text. All punctuation symbols were removed from the data. We used a morphological analyser and each word was replaced by its base form. For example, "disappointed" would be converted into "disappoint". Words with certain parts of speech that bear no sentiment, for example personal pronouns, were also removed from the text. Finally, words were converted to lower case.

All the experiments were conducted using the ILOG CPLEX solver[4].

[2]http://www.cs.jhu.edu/~ mdredze/datasets/sentiment/
[3]http://www.let.rug.nl/~ vannoord/TextCat/
[4]http://www-01.ibm.com/software/integration/optimisation/cplex-optimizer/

star 1	star 2	star 4	star 5
61.91	43.33	47.33	64.94

Table 3.1: Accuracy for additive model

3.3.2 Evaluation

Each of our three models were run to get three separate opinion lexicons. The values obtained from these lexicons are used in an additive model to get the star rating for each review. With the scaling we are expecting a review with the star rating 1 to have the value -100, star rating 2 to have the value -50, star rating 4 to have +50 and star rating 5 to have +100. It is almost impossible to get the exact value for each class, i.e. using the additive model it is impossible to get the value -100 for star rating 1 on the test set or +100 for star rating 5. Thus, for each star rating we allowed a certain range defining the coverage of each star rating. The coverage is defined by the gap between each star rating and the next. Figure 3.2 shows the 40% coverage for star rating 2 (-50) and 4 (+50). The gap between -50 and -100 is less than that of -50 and +50, thus the coverage of each on either side is of a different value.

The evaluation is based on correctly classifying the review text into any of 4 classes using our generated lexicon. The test set consists of 70 unseen reviews with star rating 1, 30 reviews with star rating 2, 30 reviews with star rating 4 and 70 reviews with star rating 5 taken from the same data set. *Accuracy* measures the total number of reviews in the test set correctly graded to its respected star rating by using the generated lexicon. We used 10-fold cross validation with the data set for evaluation. We can see from Table 3.1 that the results obtained are not encouraging. Classifying the documents into four different classes highly degraded the performance. The result obtained for coverage less than 40% and with models other than MinErrFZCR are very poor. We only considered the result with 40% coverage using the model MinErrFZCR.

We can see that the accuracy on star ratings 1 and 5 are relatively high than those on 2 and 4. This shows that our classifier still works relatively well to separate positive from negative ones. The higher results on either end are encouraging to further pursue the additive model for the opinion

star 1	star 2	star 4	star 5
78.0366	64.2551	65.0873	79.5511

Table 3.2: Accuracy for SVM

classification task.

For comparison we used SVM to classify the same training data set into multiple classes by using the one vs the rest strategy. Four different training sets are sampled from the original training data set. The training set to classify star rating 1 from the rest consisted of 600 reviews of star rating 1 and 1200 of the other remaining reviews. The training set for the rest of the star ratings were created in a similar way. The pre-processing that was done was similar to that done before the equation construction. LIBSVM[5] with optimised learning parameter C with a unigram as a feature is used for training. The test set is the same as was used in our experiments with the additive model.

Table 3.2 shows the accuracy score obtained by SVM. We can see that SVM also performs relatively well on either end. The accuracy results for the star rating 2 and star rating 4 are lower than those for star rating 5 and star rating 1. The accuracy obtained by SVM is higher than that obtained by our classifier. The thing that has to be noted here is that our classifier does the whole classification with one trained classifier, i.e a single classifier is classifying the data into 4 different classes. The SVM used here uses different training sets to classify 2 classes at a time. We did not set up our classifier with the same 1 vs the rest strategy because we felt that it negates the whole purpose of using an additive model to get a score for each review. Our assumption is that star rating 4 is different from star rating 5 in some ways, and if we did a 1 vs the rest training then on multiple instances we have to consider these two different star ratings as one. Thus we avoided the 1 vs the rest scenario for our classifier.

Our additive model has already shown an impressive performance in classifying the data into two classes (see Chapter 2). Even though the result obtained for the multi-class classification is poor, the former shows an encouraging paths to explore the additive model to improve on the multi-class

[5]http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

classification.

3.3.3 Related Work

The multi-class opinion classification has not been explored much in the opinion analysis domain. Of the few researches done on this task the most prominent one is the work on rating-inference problem carried out by [22]. They first evaluated the classification problem on the basis of human performance. The results showed that the performance decreased as the number of ratings i.e stars increased. The reason for this problem is the same as we discussed before; as the rating scale increases the adjacent scales are more similar than different. The authors also applied a regression algorithm and found that the results obtained were not impressive. Then they applied a metric algorithm to alter the result of n-ary's classifier so that the similar items received similar scores. For this they counted the total number of positive words in each review. This improved the results obtained by just using SVR (for regression). This is an encouraging result, as similar to metric labeling, our weighted score can be used in conjunction with any regression algorithm combined to proper feature selection to produce better results. We leave this task for future work. In this study we wanted to test how an additive model can tackle a multi-class opinion classification problem.

3.3.4 Conclusions

In this section we classified the opinionated text into multiple classes using our additive compositional method. Although the results obtained were not exactly what we had hoped for, the result obtained is still encouraging and shows the direction for future work. We compared our approach with SVM. SVM also showed comparable poor results in classifying the opinionated text into four different classes. Considering the success of our additive model and also SVM in classifying the opinionated text into two polar classes, we can say that classifying opinionated text into multi-classes is a hard problem.

The problem is due to the grey line between the reviews with star rating 5 and star rating 4 and between the reviews with star rating 1 and star rating 2. Although human reviewers can decide which review they want to tag as

star 1 and which they want to tag as star 2, there is no clear algorithm or rules governing such. A review with star rating 2 has a more positive sense than a review with star rating 1, and similarly a review with star rating 4 has a more negative sense than a review with star rating 5. Our additive model considers such cases but the whole constraint system cannot quantify a consistent value throughout.

The future work to solve the problem could be a multi-iteration process where the first step will be to detect a negative anomaly on star rating 4 and a positive anomaly on star rating 2 and then introduce a parameter, say α, to balance out the equation. For example:

> This film should be *brilliant*. It sounds like a *great plot*, the actors are *first grade*, and the supporting cast is *good* as well, and Stallone is attempting to deliver a *good performance*. However, it *can't hold up*.

The above review could be represented as:

> This film should be *brilliant*. It sounds like a *great plot*, the actors are *first grade*, and the supporting cast is *good* as well, and Stallone is attempting to deliver a *good performance*. α(However, it *can't hold up*) =**4**

Training such parameters could be a better way of achieving a consistent optimisation.

We showed in this section that multi-class opinion analysis is a difficult problem, and even though we could not achieve better results, we believe that the additive method can be used as the base platform for future research on multi-class opinion analysis.

3.4 Features-related opinion classification

Any expression which expresses *opinion* towards some *targets* is an opinionated expression or *subjective expression*. Subjective expressions contain at least a *target* and an *opinion expression* reviewing the *target*. For example, *good camera* is a subjective expression because it has both target (*camera*) and opinion (*good*). On the contrary, *this camera has a lens* is not

a subjective expression because even though it has two targets (*camera* and *lens*), it lacks any opinion towards these targets. Features-related opinion classification is completed in 4 steps:

1) **Extracting targets of opinion from the review text:** For this study, targets are extracted from the annotated data.

2) **Extracting opinion words from the review text:** Opinion words are extracted from the review text using constraints optimization over an additive model as described in Chapter 2.

3) **Extracting subjective expression unique to each feature**: Subjective expressions are extracted by mining the expression which has both opinion words and target words present in it. A different text mining technique is used to identify use of multiple features in one subjective expression.

4) **Classifying the opinion and summarising the results**: Once we have the subjective expression we identify the polarity of that expression. Once the polarity has been decided we summarise the result as discussed in Section 5.1.

3.4.1 Target Extraction

The data set used for this study already came with annotated targets. To get the best result out of our algorithm to extract fine-grained subjective expressions we opted to use the annotated target. We also developed our own target extraction algorithm which is described in detail in Chapter 5.

3.4.2 Opinion words extraction

Opinion words are extracted from the review text using constraints optimisation over an additive model as described in Chapter 2. We used our best performing model (MinErrFZCR) for the opinion word extraction.

3.4.3 Extracting subjective expressions unique to each feature

An expression which shows an opinion towards a certain target is termed as a subjective expression. For example, *the camera is beautiful* is a subjective expression, whereas *the camera comes in black,* is non-subjective because it describes a feature of the camera rather than pointing out any opinion towards camera.

Filters based on features and opinion words (Feat&Op)

Once we have the potential *features* and *opinion lexicon* for a domain, we can extract subjective sentences from the review text by only selecting those sentences which contain *both* features and opinion words.

Example 3.1 Sentences containing features/targets and opinion words:

1. "The *picture* turned out quite *nicely*."
2. "In a word, *awesome* is how I would describe this *camera*."

Sentences are extracted using feature lexicon {"picture"; "camera"} and opinion lexicon {"nice"; "awesome"}.

By extracting only those sentences which have both feature and opinion words we filter out sentences which are non-subjective. For example, sentences which describe a product rather than provide an opinion about it, like *The camera comes in black.* Even though this sentence contains a feature word, the whole sentence mentions nothing about the quality of the feature.

Each sentence in Example 3.1 is considered as the review for the target it mentions. This method can separate subjective sentences from the non-subjective sentences but will fail to identify sentences unique to the feature in certain cases. These methods will work in cases where there is only one target mentioned in the sentence, but as the number of targets in the sentence increases, we cannot identify a sentence as a review of a particular target. For example, the sentence, "It has a beautiful *design*, would use it but the *battery* is horrible" has two target words, thus we cannot regard the whole sentence as a review of either *design* or *battery*.

Phrase level splitting (SBARsplit)

Example 3.2

1. "It has a beautiful *design*, would use it but the *battery* is horrible"
2. "the *menus* are easy to navigate and the *buttons* are easy to use"

Sentences are extracted using feature lexicon {"design"; "battery; "menu"; "buttons"} and opinion lexicon {"beautiful"; "horrible"; "easy"}.

We can see in Example 3.2 that each sentence is reviewing multiple features, for example, sentence 1 of Example 3.2 reviews *design* and *battery*. An opinion classification algorithm would classify the above sentences as either positive or negative. For sentence 2 of Example 3.2 , since both of the features have the same polarity, it would be valid if they share the overall polarity of the sentence. But in sentence 1 of Example 3.2 we can see that each feature has the opposite opinion towards them. Thus any opinion we get for sentence 2 won't be true for one of the features. Thus it is necessary to split the sentences into different parts, where each part is a review of a single feature.

In sentences which contain opinions for more than one feature, it is the case that clausal boundaries would help extract unit reviews. Thus, separating sentences into clauses might solve the problem shown in Example 3.2 . We split a sentence into clauses with occurrences of nodes labeled "*SBAR*", which indicates the root of a sub-ordinate clause in a phrase structure parse tree of a sentence. We used Stanford's parser[6] to obtain the phrase structure parse tree of a sentence. The clauses of sentences in Example 3.2 are:

Example 3.3

1. Clauses of Sentence 1 of Example 3.2
   1.1. it has a beautiful design,
   1.2. would use it but
   1.3. the battery is horrible.

[6] http://nlp.stanford.edu:8080/parser/

2. Clauses of Sentence 2 of Example 3.2

    2.1. the menus are easy to navigate and the buttons are easy to use

We can see from the first example of Example 3.3 that by splitting a sentence into clauses, we can successfully extract parts which have an opinion about a single feature. For example, Sentence 1.1 of Example 3.3 is a review about the *design* and Sentence 1.3 of Example 3.3 is a review about the *battery*.

Not only this, but we can again apply Feat&Op on the extracted clauses and filter out non-subjective clauses. For example, clause 1.2 of Example 3.3 will be filtered out since it does not contain feature terms and opinion words.

Using SBARsplit we can get clauses unique to a feature but this does not work in all cases. We can see in sentence 2.1 of Example 3.3 that the clause extracted still contains multiple features. This points the necessity of further splitting the clauses into smaller parts.

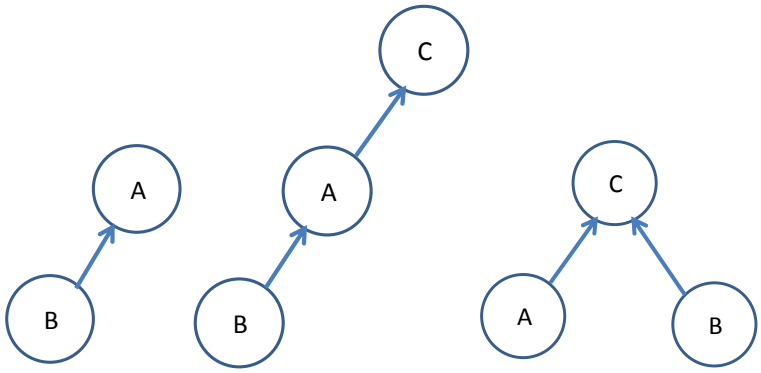
Direct Dependency Extraction (DDE)

Example 3.4

1. the menus are easy to navigate and the buttons are easy to use
2. it takes great pictures, operates quickly, and feels solid
3. who is looking for excellent quality pictures and a combination of ease of use and the flexibility to get advanced with many options to adjust
4. has a great lens, but a horrible viewfinder.

Sentences in Example 3.4 are the clauses containing more than one feature, extracted after applying SBARsplit. Our Direct Dependency Extraction (DDE) method to extract clauses unique to a particular feature is based on the fact that there is a natural relationship between opinion words and features as the former *modifies* the latter. Furthermore as shown in [26], there are various grammatical relations between polarity influencers (opinion words in our case) and their target feature. We define such relations via the dependency parser based on dependency grammar.

Dependency grammar describes the relation between words in a sentence. After parsing a sentence by the dependency parser, we will get relations between words in the sentence. For example, in a sentence, "It is a good lens.", *good* is an opinionated word and *lens* is its target. After parsing the sentence with the dependency parser we will find that *good* depends on *lens* by *mod* relation. A direct dependency (DD) relation is defined as the direct dependency of two words with modifiers. Any multilevel dependency relation is considered indirect dependency (ID) relation. Figure 3.3a shows the type 1 direct dependency where a word A is dependent on word B directly by a modifier. Figure 3.3b is the type 2 direct dependency where a word A is related to word C with type 1 DD and word B is also related to word C with type 1 DD, thus word A shows a direct dependency relation to word B.



(a) DD type 1 (b) DD type 2 (c) ID

Figure 3.3: Propagation of polarity in a sentence

An opinion word is related to the target word through a dependency relation. Dependency relations have been successfully applied to extract {target, opinion} a pair by [13] and as a feature for sentiment classification by [35]. We use dependency relations to extract clauses unique to features. We propose that, in any sentence which has an opinion about multiple features, each feature is modified by its own opinion expression. For example, the dependency parse of sentence 4 of Example 3.4 yields:

```
amod(lens-3, great-2)
dobj(has-1, lens-3)
cc(lens-3, but-5)
```

```
amod(viewfinder-7, horrible-6)
conj(lens-3, viewfinder-7)
```

The dependency relation shows that *lens* and *great* are dependent on each other by the modifier *amod* and *viewfinder* and *horrible* are dependent on each other also through the modifier *amod*. Thus, *lens* is only related to *great* and *viewfinder* is only related to *horrible*. The possible clause extracted for each feature will be:

1. lens great
2. viewfinder horrible

Thus a dependency triples $modifier(arg_1, arg_2)$, where arg_1 and arg_2 are opinion words and target words pair, can identify the opinion towards a target. Thus, each direct dependency of type 1 with constraints on syntactic relation is a part of the unique review for the feature word present in the relation. These dependency triples are the *seed* to generate independent clauses for each feature.

The syntactic constraints, M, is the set of{*mod, subj, comp, conj*}. An example of each modifier is shown in Table 3.3. Only the syntactic modifier in set M is considered to be relevant to opinion analysis; the rest of the modifier relations are ignored. From the syntactic modifier in set M, only those dependency triples are selected as seed dependency triples which have one argument as the target word and the other as the opinion word. All the direct dependency of type 2 and indirect dependency between the target and opinion words are ignored because they add to the noise and the clauses that expanded from such constructs are little different from the original sentence. Therefore, we formulate a seed relation **Mod-OF** as a quadruple of $<< DDT1,\ S,\ w_o,\ w_t >>$ where DDT1 is the direct dependency relation of type 1 between w_o and w_t, S is the syntactic modifier in set M and w_o and w_t are the arguments of the dependency triples where w_o is the opinion word in the opinion lexicon and w_t is the target/feature word in the target lexicon.

There are many instances when the opinions on two features are expressed with the conjunction relation to both. For example, in the sentence.,"*The camera and the lens are both awesome*", both the features *camera* and *lens*

Modifier	Dependency triple	Sentence
mod	amod(a,b)	this is an example sentence
	vmod(a,b)	this is also an example sentence
subj	dsubj(a,b)	this is an example sentence
	isubjmod(a,b)	this is also an example sentence
comp	xcomp(a,b)	this is an example sentence
	acomp(a,b)	this is also an example sentence
conj	conj(a,b),conj(a,b)	The camera is great and so is the viewfinder.

Table 3.3: Syntactic relation with {feature, opinion} pair

have one opinionated word, *awesome*, related to them. The parse tree of such a construct yields:

Example 3.5

```
det(viewfinder-2, The-1)
nsubj(awesome-8, viewfinder-2)
cc(viewfinder-2, and-3)
det(lens-5, the-4)
conj(viewfinder-2, lens-5)
cop(awesome-8, are-6)
dep(awesome-8, both-7)
```

In such cases where two features are related to each other by dependency modifier *conj*, it is the case that the opinion on one of the features is also the opinion on the other. Thus, a dependency triple with modifier *conj* and both the arguments as feature words are also considered a seed dependency triple if one of the arguments of such triples is of Mod-OF. In Example 3.5 the dependency triple *conj(viewfinder-2, lens-5)* is considered a seed dependency triple because one of its arguments *viewfinder* satisfies Mod-OF in dependency triple *nsubj(awesome-8, viewfinder-2)*. Therefore, we formulate a second seed relation Mod-FF as $<< DDT1,\ conj,\ w_{t1},\ w_{t2},\ NotIN,\ IN >>$ where DDT1 is the direct dependency relation of type 1 between w_{t1} and w_{t2}, *conj* is the syntactic modifier, w_{t1} and w_{t2} are the arguments of the dependency triples where w_{t1} is the target/feature word and w_{t2} is the another target/feature word present in the target lexicon. $NotIN$ states that either w_{t1} or w_{t2} has not already been extracted and IN states that either w_{t1} or w_{t2} satisfies the relation Mod-OF.

Once we have our seed relations we extract all the direct dependency of type 1 involving the arguments of the seed dependency triples as the clauses for each feature. We term this step as *Expand*. The procedure can be explained by the following algorithm.

Algorithm 3.1

```
1. fill opinion lexicon
2. fill target lexicon
3. D=Dependency parse the sentence si
4. F=features in sentence si
5. for each dependency triples in D
6.   Sel_D=select dependency triples which satisfies relation
     Mod-OF
7.   if NOT all features in F IN Sel_D
8.   Sel_D=Sel_D+ dependency triples which satisfies relation
     Mod-FF
9. for each dependency triples in Sel_D
10.  select all the dependency triples which share DD type 1
     with one of the arguments.
11 Expand for each feature to get feature clauses.
```

If we apply Algorithm 3.1on sentence 4 of Example 3.4 , in each step we will get the following output, in which the opinion lexicon and target lexicon are assumed to be given:

Step 3. dependency parse the sentence

```
amod(lens-3, great-2)
dobj(has-1, lens-3)
cc(lens-3, but-5)
amod(viewfinder-7, horrible-6)
conj(lens-3, viewfinder-7)
```

Step 4. select features in the sentence

```
{lens, viewfinder}
```

Step 6 . select dependency triples which satisfy relation Mod-OF

```
amod(lens-3, great-2)
amod(viewfinder-7, horrible-6)
```

Step 7 . since the dependency triples for all the features in F have been found, further searches are not necessary

Step 9 . for each selected dependency triples, expand the DD type 1 relation

```
Feature ‘‘lens’’
amod(lens-3, great-2)
DD type 1 relations:
 dobj(has-1, lens-3)
 cc(lens-3, but-5)
 conj(lens-3, viewfinder-7)

Feature ‘‘viewfinder’’
amod(viewfinder-7, horrible-6)
DD type 1 relations:
  conj(lens-3, viewfinder-7)
```

Step 11. Expand to get clauses

```
 Feature ‘‘lens’’
lens great has but viewfinder

Feature ‘‘viewfinder’’
viewfinder horrible lens
```

Thus we can see that we can extract clauses unique to each feature. Let us assume the following lexicon for opinion words and features:

Opinion Lexicon	Feature Lexicon
easy	menu
great	buttons
operates	picture
excellent	feel
ease	options
advanced	pictures
horrible	used
	lens
	viewfinder

Following Algorithm 3.1, the clauses extracted from the sentences in Example 3.4 are:

Example 3.6

1. the menus are easy navigate and
2. easy the buttons are, use
3. great pictures operates
4. operates feels solid
5. for excellent quality pictures and combination
6. has great lens, but
7. but horrible viewfinder

We can see that we managed to extract most of the clauses unique to each feature. Since we only used the direct dependency relation of type 1 we missed out on some of the clauses in sentence 3 of Example 3.4 . We can also see that the clause which we extracted from sentence 3 of Example 3.4 is correct to the feature. This points to the fact that even though the recall of each clause unique to the feature could be low with our algorithm, the precision can be quite high.

Thus our method of extracting subjective expressions unique to each feature is an iterative approach which can be described as:

Feature	**Sentences/Clauses**
picture	The picture turned out quite nicely
	great pictures operates
	for excellent quality pictures and combination
camera	In a word, awesome is how I would describe this camera.
design	it has a beautiful design
battery	the battery is horrible
menu	the menus are easy navigate and
button	easy the buttons are, use
feel	operates feels solid
lens	has great lens, but
viewfinder	but horrible viewfinder

Table 3.4: Feature table with unique subjective sentence for each feature

Algorithm 3.2

```
1: Extract subjective sentences using Feat&Op
2: store each sentence with single feature mention
   to feature table.
3: For each sentence with multiple features do
4:   SBARsplit
5:   For each clauses extracted by SBARsplit do
6:     filter using Feat&Op
7:     store each clause with single feature mention
       to feature table
8:     For each sentence with multiple features do
         DDE
9:       store each clause to its respective
         feature table
```

□

If we take all the example sentences from Example 3.1 to Example 3.4 and apply Algorithm 3.2 , the feature table would look like that shown in Table 3.4.

Classifying the opinion in the sentences/clauses

The opinion class of each sentences/clause is either positive or negative. For classification we use our linear additive classifier from Chapter 2. We use our best performing classifier, namely, **MinErrFZCR**. We train the classifier on the review data set containing complete sentences, i.e. a data set not divided into individual clauses. Since our model is additive and it gives weight to each part of the sentences, we can safely use a classifier trained for complete sentences to classify the polarity of clauses obtained from the sentences. For example, consider the following sentence:

Example 3.7

> "The senators supporting the leader failed to praise his hopeless HIV prevention programme."

Then the overall sentiment of Example 3.7 can be derived as follows:

1. $his(0) + hopeless(-3) + HIV(-1) + prevention(+2) + programme(0) = -2$
2. $The(0) + senator(0) + supporting(+1) + the(0) + leader(+1) = +2$
3. $failed(-4) + to(0) + praise(+3) = -1$
4. $-2 - 1 + 2 = -1$

We can see that a lexicon learnt from a complete sentence through an additive model can classify correctly the opinion expressed in each part of the sentence.

3.4.4 Experiments

For all our experiments we used the Customers' Review Dataset[7]. The data set contains review of different products. We chose reviews on cameras("Canon" for the training and "Nikon" for the testing) for our experiments. The data set already has all the targets labelled. Each target is also labelled with a polarity score. The polarity score for each target ranges from [-1, +3]. To train our MinErrFZCR we converted the whole data set

[7] http://www.cs.uic.edu/∼liub/FBS/sentiment-analysis.html

number of positive reviews	314
number of negative reviews	85

Table 3.5: Number of positive and negative reviews

number of positive words	102
number of negative words	51
total words	328

Table 3.6: Number of positive and negative words extracted by MinErrFZCR

to positive and negative reviews. All the sentences with a positive score towards the feature is considered a positive review and all the sentences with a negative score towards them is considered a negative review. In case of multiple targets in a single sentence, we added the opinion score provided for each feature. If the total sum is negative then the review is considered negative and if the total sum is positive, the review is considered positive.

Table 3.5 shows the total number of positive and negative reviews in the data set. Table 3.7 shows the number of positive and negative words extracted by applying MinErrFZCR on the data set. The total number of words is the number of unigrams used to train the MinErrFZCR. The total number of words is quite low because most of the reviews are single sentence reviews. Also we applied pre-processing as discussed in Chapter 2 to remove irrelevant words from the dataset.

Negative	Positive
obstruct	great
problem	quality
only	easy
blurry	love
heavy	good
plastic	happy
average	awesome
lack	excellent
minor	price
flaw	please

Table 3.7: Sample positive and negative words extracted by MinErrFZCR

Positive	Negative
80.98%	74.11%

Table 3.8: Accuracy for correctly identifying positive and negative features

3.4.5 Evaluation

The test was performed on the different Camera review (Nikon). For the evaluation of all the fine-grained clauses/sentences extracted by our method, we first collected the features/targets from the data set along with their polarity values. In the whole data set there are 107 different kinds of features, with each feature having multiple polarity assigned to it. The features are both single words and multiple words, for example , "picture" is a feature and "picture quality" is also a feature. For the experiment we only used single word features. We re-labelled the data with a single word feature where we could and avoided it where we could not. For example, some instances of "picture quality" were changed to "picture". Since we have a labelled data set, experiments exploiting the labelled data seemed more reasonable. Also the major target of this work is more to extract fine-grained opinion on each feature than to extract the feature itself, thus we opted for using the labelled feature.

With these labelled features and the opinion words extracted through the opinion lexicon generated by MinErrFZCR (we only considered words with positive and negative values, words with value 0 are neutral and thus not opinionated), we calculated how accurately we could extract the correct number of opinions from each feature. Accuracy is defined as the percentage average number of correct answers for each feature, for example if the feature "*lens*" has 3 positive opinions and 1 negative opinion on the whole data set and if our system can extract 3 clauses for "lens" and says all of them are positive, then the accuracy for positive for "*lens*" is 100% and for negative is 0%. Accordingly accuracy for each individual clauses for each feature was calculated. The final accuracy is the average of the accuracy obtained on each individual clauses. The overall accuracy is the number of correct positive reviews identified for each feature divided by the actual number of positive reviews for that feature, and the same for the negative.

Table 3.8 shows the accuracy of classifying positive and negative reviews

based on each feature. The above results are quite impressive owing to the fact that the opinion lexicon is learnt from the data. The major cause of not performing significantly well is that in the sentences not all the features are used in their original form. For example, a sentence with the feature "camera" with a positive opinion was:

```
the more i work with it, the more i love it
```

Here, "it" refers to "camera". We have not used a pronominal anaphora resolution, so such sentences will not be extracted by our system. Thus the accuracy dips with such sentences.

Another reason for the dip in accuracy is due to the lemmatised use of features for labelling. All the features are labelled in its root grammatical form, such as , in the sentence, "This camera offers functional conveniences." The feature listed in this sentence is 'function", but our tool to extract lemma of words does not return a lemma "function" for "functional". Thus the sentence won't be detected.

Another reason for the dip in accuracy is that the features are not mentioned in the review explicitly. For example, a sentence tagged with the feature "weight" is "rather heavy for point and shoot". Our system does not incorporate any knowledge to find a relation between "weight" and "heavy".

On the other hand, our system also detected some features which were not tagged as feature in the sentence. For example, in the following sentence:

> i 'd highly recommend this camera for anyone who is looking for excellent quality pictures and a combination of ease of use and the flexibility to get advanced with many options to adjust if you like .

Features tagged were "picture", "use" and "option". Our system also extracted a clause "i 'd highly recommend this camera for anyone" as a review for the camera and the classifier classified this as positive.

3.4.6 Related Work

We would like our work to differ from subjective expression extraction as our work is more about extracting topic span. Topic span is the collection

of words which provide the overall opinion on that topic. Subjectivity in our case is defined in a very simple manner. If an expression contains both a topic word and an opinion word then such an expression is subjective. We believe that even if such expressions are not subjective, then our algorithm further filters out such expressions. Lots of work has been done in classifying subjective expression from non-subjective expression. One of the early works is of Pang and Lee [21], who used graph-based min-cuts method to classify subjective text from non-subjective text. Their work showed that subjectivity extracts can compress a review to a much smaller size but still retain the polarity of the whole text.

Target extraction and opinion words/expression extraction are done in conjunction with each other. With a seed opinion and target word, a bootstrapping process based on the fact that target and opinion modify each other through some syntactic constructs is used to generate further opinion and target words. Hu and Liu [13] and Qui et al. [12] are some of the works which follow the bootstrapping method. Hu and Liu [13] identify the opinion on the target based on the polarity of the opinion word that modifies the target. Our work does not only depend on the polarity of the opinion words but also considers the other words that target words and opinion words modify. Such information is necessary as they contribute a lot towards the overall opinion of the expression. For example, the polarity of "not good" is opposite to that of "good". Wiebe et al. [34] introduced the concept of "Direct Subjective Expressions" (DSEs) and "Expressive Subjective Expressions" (ISEs). For example, in the sentence,

> Tsvangirai **said** the election result was *illegitimate* and a clear case of *highway robbery*.

The bold face span of text is the DSE and the span of words in italics are the ISEs. This matches our concept of extracting only the related text spans from the whole sentence. Breck et al. [5] used conditional random field to tag such expressions. Their approach requires data tagged with DSEs and ISEs, whereas we concentrate on extracting fine-grained subjective expression based on the feature term itself. If an opinion and feature lexicon is provided then our algorithm does not require any further annotation to extract subjective text spans.

Fahrni and Klenner [11] used Wikipedia's category system to extract targets and used syntactic relations to further identify target-specific opinion lexicons. Such lexicons are important because phrases like "cold coke" and "cold pizza" don't share the same opinion. We concentrate more on extracting topic span than generating a topic-specific lexicon. We claim that once a topic span is created a classifier can be trained to disambiguate such anomalies.

Yi and Niblack's [37] work on fine-grained opinion analysis is very closely related to our work. They define a syntactic pattern that forms an opinion expression. For example,

```
This camera takes excellent pictures.
   - predicate: take
   - pattern: <"take" OP SP>
   - subject phrase (SP): this camera
   - object phrase (OP) : excellent pictures
   - sentiment of the OP: positive
```

Although there is no comparison done to their work, we believe that our work takes into account the context more than them. We let the expression build from the sentence itself and mine every direct relationship to the target and opinion words in contrast to matching a database pattern. In doing so we extract almost every piece of contextual information relating to target and opinion words.

3.4.7 Conclusions

In this work we presented a step-wise and descriptive algorithm to extract clauses/sentences unique to a single feature of the product. The uniqueness refers to the fact that the extracted clauses shows opinions about a single feature of any product. Classifying such clauses into positive and negative opinions leads us to summarise the opinions on each individual feature of the product concerned. Through this we can get the fine-grained opinions on each feature of the product, as:

```
Product 1:
 Feature 1:
```

```
    Positive : 21   {Positive clauses/sentences}
    Negative : 3    {Negative clauses/sentences}
-------------------------------------------------
  Feature 2:
    Positive : 10   {Positive clauses/sentences}
    Negative : 4    {Negative clauses/sentences}
-------------------------------------------------
  Feature 3:
    Positive : 2   {Positive clauses/sentences}
    Negative : 14  {Negative clauses/sentences}
-------------------------------------------------
.
.
.
```

Even though our algorithm showed some promising results, it also has some significant shortcomings. Our algorithm does not take into account the pronominal resolution, thus it becomes ineffective when pronouns are used to identify features. This can be resolved by using a pronominal resolution on the data, and we leave this to future work. Also other possible future directions could be to use word sense disambiguation to extract words with similar sense as the feature word, e.g. "weight" used in the same sense as "heavy". Using such filters can also improve the performance of our algorithm.

Chapter 4

Feature Exploration for Sentiment Classification

4.1 Introduction

Document Sentiment classification is a task to label documents as either positive or negative, depending upon the content of the document.

Example 1

- Document 1

  ```
  The movie Avatar has a great story.  The actors in
  the movie are brilliant.
  ```

- Document 2

  ```
  The movie lacks substance.  The acting is really
  appalling.
  ```

The task of the document level sentiment classification system is to label document 1 as positive and document 2 as negative. A person who knows about movies would quickly identify keywords such as *movie-great-story* and *actors-brilliant* from document 1 and keywords such as *lacks-substance* and *acting-appalling* from document 2 and with the knowledge to infer meaning from such words, they can easily tag document 1 as positive and document

2 as negative. A machine can also be trained to do the task in a similar manner. First devise an algorithm so that the machine can pick up such keywords, and then make the machine learn the meaning of such words. Thus, the task of document sentiment classification can be divided into two sub-tasks: 1) to identify relevant keywords from the document, and 2) to learn the extracted pattern for classification. A supervised setting can make task 2 quite easy. Instead of teaching the machine the meaning of each word, we could feed the machine with sufficient examples of positive keywords and also of negative keywords, then write an algorithm so that the machine can separate positive patterns from negative patterns. Such supervised pattern recognition and classification algorithms have been researched a lot and have been perfected rapidly. Support Vector Machine (SVM) is one such supervised classification algorithm which we will be using in this research.

The classification problem of our task is sorted out, but the major problem still remains, which is to devise an algorithm so that the machine can pick up keywords which are significant for the sentiment classification of the document. Many works have been done in this area too. The works are based mainly on two aspects. One is to mine the keywords from the text using popular techniques like unigram-based model in Pang et al. [21] and others are mining keywords by applying well formulated rules, like rules which extract dependency relations from the text as done in Wilson et al. [35]. Both of the techniques work reasonably well, but they have certain shortcomings. The unigram or even n-gram model fails to capture significant links between two non-consecutive words in the text. For example, in document 1 of Example 1, a unigram model will fail to capture the link between *movie-great*, thus it misses out on the context. A carefully crafted rule may be able to capture such a relation but, as the data grows, rules might not work properly. Formulating rules requires proper and exact data knowledge and also such rules might not migrate easily when the dataset is changed.

This study reports on a method which concentrates on overcoming the non-context capturing nature of unigram models, removing unwanted features when applying the n-gram model and capturing related keywords without applying hand-crafted rules. The sentiment classification result obtained using such features beats the results obtained thus far in the same dataset. Following are the contributions of this report:

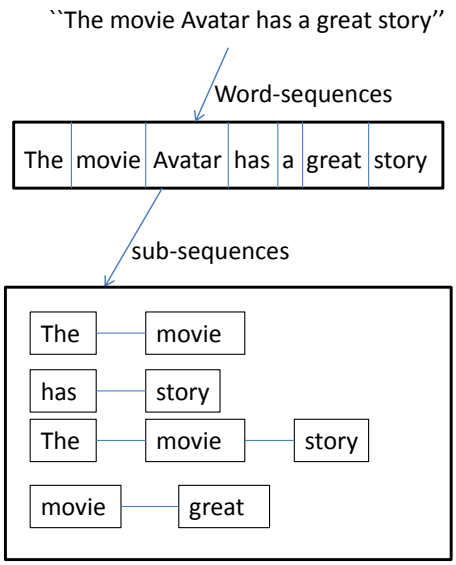


Figure 4.1: Word sub-sequences of sentence, *"The movie Avatar has a great story"*

- To overcome the non-context capturing nature of the unigram model, we explore the sub-sequence mining from the text. Sub-sequences are the sets of sequences of words obtained after removing a non-zero number of words from a sentence. An example of sub-sequence mining is shown in Figure 4.1. We can see from the figure that there is a direct link between *movie* and *great*, which is very desirable as a feature for opinion classification.

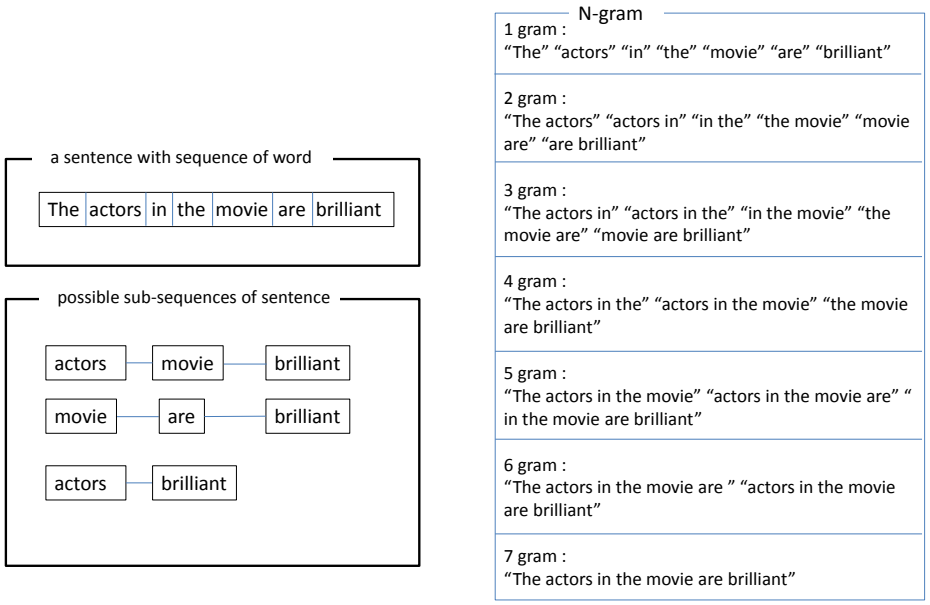
- When using n-grams, say, bi-grams or trigrams, or even sub-sequences, we capture much non-related text. For example, one of the trigrams in the sentence *The movie Avatar has a great story* will be *"the movie Avatar"*, which is irrelevant to the sentiment analysis model. There are only certain context which are relevant, and these contexts define the subjectivity of the text. In a given domain, such contexts are frequently occurring. For example, in a movie domain, patterns like, *"movie-great"* or *"movie-bad"* occur more often than patterns like *"movie Avatar"*. This report explains how such frequent patterns can be mined from the text. Mining frequent patterns from the text is an automatic way of generating relevant context information without using hand-written rules.

- Most of the words that occur in the opinion text are irrelevant for sentiment analysis; for example, in the movie domain, the name of the actor, movies, etc. To filter out these non-related terms from the text we propose a two-step feature selection process. In the first step we will learn from all the features and in the next we will apply a feature selection process to select only the relevant features. This report also shows how the accuracy of the system can be improved by applying feature selection techniques.

4.2 Background

4.2.1 Word sub-sequences

A word sub-sequence is defined as a set of sequences of words obtained after removing a non-zero number of words from a sentence. The order of the words in the sentence is maintained in the sub-sequence. We can see from



(a) Sub-sequences (b) N-grams

Figure 4.2: Word sub-sequences and n-gram patterns of the sentence *“The actors in the movie are brilliant”*

Figure 4.2 that n-gram can only extract N continuous occurring words from

a text, but if we take a sub-sequence of a word it extracts not only the co-occurring words but also non-co-occurring words. Also the sub-sequence is not restricted to any specific N number of words; it can be any number of words within the sentence. Thus using sub-sequences as the keywords for classification becomes more effective.

For the sentence *"The actors in the movie are brilliant"* it is already established that the most potential candidate as the feature for sentiment analysis is *"actor-brilliant"*. We can see from Figure 4.2b that it is not possible to extract a direct relation from the word "actors" to the word "brilliant" using any of the n-gram technique. Only when we use 6-gram and 7-gram do both of these words occur in the same frame. Taking such a long frame can hurt the classification, since it will be hard to find a matching pattern for such lengthy frames. For example, taking a 7-gram will result in the phrase "*The actors in the movie are brilliant*"; the probability of finding another phrase from the text which is an exact match is very low. A classifier learns from similar patterns, so using long frames will make features very sparse, thus resulting in poor performance. In contrast to this, we can see in Figure 4.2a that one of the sub-sequences is *"actors-brilliant"* which is very desirable to be used as a feature for the classification task. Also the probability of finding a sub-sequence phrase "*actors-brilliant*" in an opinionated document about movies is quite high, thus we get desirable features from sentences in abundance. A simple pseudo-code to extract all possible sub-sequences from a sentence is given below:

```
Define List<String> subseq
Define token[]=sentence.toTokens()
for (int i = 0; i < token.length; i++)
     if (NOT(token[i] In subseq))
         Add token[i] to subseq

     String seq = token[i];
     for (int j = i + 1, l = i + 1; j < token.length; j++)
          seq = seq + " " + token[j];
          if (NOT(seq In subseq))
              Add seq to subseq

          if (j == token.length - 1)
              j = l;
              l++;
              seq = token[i];
```

Sentence	The actors in the movie are brilliant
Sub-sequences	The, The actors, The actors in, The actors in the, The actors in the movie, The actors in the movie are, The actors in the movie are brilliant, The in, The in the, The in the movie, The in the movie are, The in the movie are brilliant, The the, The the movie, The the movie are, The the movie are brilliant, The movie, The movie are, The movie are brilliant, The are, The are brilliant, The brilliant, actors, actors in, actors in the, actors in the movie, actors in the movie are, actors in the movie are brilliant, actors the, actors the movie, actors the movie are, actors the movie are brilliant, actors movie, actors movie are, actors movie are brilliant, actors are, actors are brilliant, actors brilliant, in, in the, in the movie, in the movie are, in the movie are brilliant, in movie, in movie are, in movie are brilliant, in are, in are brilliant, in brilliant, the, the movie, the movie are, the movie are brilliant, the are, the are brilliant, the brilliant, movie, movie are, movie are brilliant, movie brilliant, are, are brilliant, brilliant

Table 4.1: All the possible sub-sequences of the sentence *The actors in the movie are brilliant*

Table 4.1 shows all the possible 63 sub-sequences obtained by applying the above pseudo code to the sentence "*The actors in the movie are brilliant*".

We can see that sub-sequence mining captures all the necessary syntactic relations from a sentence. On the contrary, sub-sequence mining can also lead to overwhelming features. From a single sentence, 63 unique sub-sequences were extracted, not all of which are necessary. We are only interested in those sequences which occur frequently in our dataset (barring some common grammatical sequence like *"in are"*). This leads to the necessity of mining frequent clauses from the sentences.

4.2.2 Mining Frequent Sub-sequence Patterns

Frequent sub-sequences are mined from a sequential dataset by using sequential pattern mining algorithms. The sequential pattern mining problem was first introduced by Agrawal et al. [3]:

> *"Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user specified minimum support threshold, sequential pattern mining is to find all of the frequent sub-sequences, i.e., the sub-sequences whose occurrence frequency in the set of sequences is no less than minimum support."*

To mine frequent sub-sequences we used PrefixSpan [25]. PrefixSpan (Projected Sequential Pattern Mining) mines the complete set of frequent patterns but greatly reduces the effort of candidate sub-sequence generation by exploring prefix projection in sequential pattern mining.

PrefixSpan builds in a simple logic that for any sequence to be frequent, it has to be that the prefix of that sequence is also frequent. Thus instead of mining all the sequences in the dataset, the algorithm only expands those sub-sequences which have a frequent prefix. If a minimum possible subsequence is a single word, and a sentence is a single sequence, PrefixSpan first mines all the frequent words, i.e. all the words whose occurrence in a multiple sequence is greater than a certain threshold. Then the algorithm expands each already-obtained frequent sub-sequence of size k by attaching a new item to obtain a frequent sequence of size k + 1. By repeating the latter step recursively, the algorithm obtains all frequent sub-sequences.

However, expanding a sub-sequence by attaching a new item to an arbitrary position leads to duplicated enumeration of the same candidate subsequence. To avoid such enumeration, the algorithm restricts the position to attach a new item to the end of the newly-obtained sub-sequence in left-to-right order.

4.2.3 Feature Selection

Mining frequent sub-sequences has to be done with a sequence size greater than or equal to two, or else we run the risk of including many irrelevant words as features. But we cannot neglect the significance of a *single word token* for the determination of opinions in a document. Most of the time a document with a significant number of positive words is positive. Words like "good", "bad", "brilliant" and "appalling" are used quite a lot in opinion text, especially in movie reviews. We do want to use the effectiveness of sub-sequences as a feature, but also we do not want to miss out on significant single words which are highly opinionated. Thus, we propose a feature selection technique to acquire only those unigrams which have significance in opinion classification.

To select only significant unigrams, a classifier is trained with a unigram as a feature on the dataset. Then we select only those unigrams which are coined by the classifier as the most relevant for the classification purpose. Previous study shows that not all the words in the text are relevant for polarity detection. Only limited numbers of words are significant for polarity detection. Thus, from the sorted list of the unigrams obtained from the classifier, we only select top half words as *selected unigrams*. We use Support Vector Machine (SVM)[1] for the classification. For this study we use the linear kernel of the SVM. The process is very straightforward, first train the linear SVM with the unigram on the dataset. This results in a classifier which can separate a positive opinion document from the negative opinion document. In SVM, the classifier actually is a hyperplane which separates positive examples and negative examples as shown in Figure 4.3. These examples can be represented by a normal, a vector perpendicular to the hyperplane, as shown by w in the Figure 4.3. Only those features are selected

[1] http://www.csie.ntu.edu.tw/∼ cjlin/libsvm/

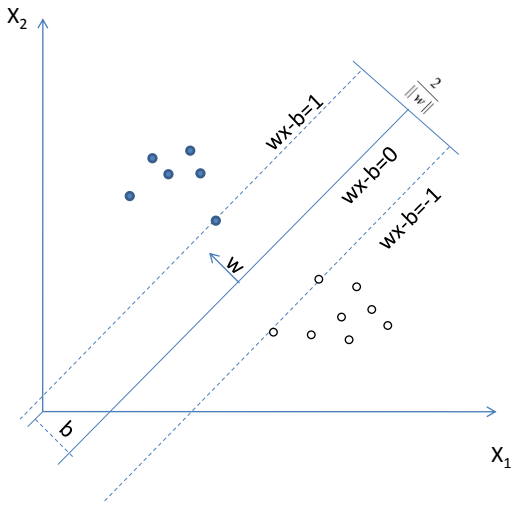


Figure 4.3: SVM hyperplane and normal

from the normal which have higher weights than a given threshold.

Mathematically, if data features are described with vectors, $\mathbf{x}_i = (x_{i1}...x_{in})$, where n is the feature dimension. Then, the class predictor trained by SVM using linear kernel ($K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$) has the form:

$$sgn[b + \mathbf{w}^T\mathbf{x}]$$

for $\mathbf{w} = \sum_i a_i\mathbf{x}_i$

where $\mathbf{w} = (w_1...w_n)$ can be computed and accessed directly. As shown in Figure 4.3 the class predictor uses the hyperplane to separate the positive examples to the negative examples and $\mathbf{w}$ is the normal to the hyperplane.

The linear classifier categorises new data instances by testing whether the linear combination $w_1x_1 + .. + w_nx_n$ of the components of the vector $x = (x_1, ..., x_n)$ is above or below some threshold b; mostly 0. In our feature selection approach we use the absolute value $\mid w_i \mid$ as the weight of feature i. We retain only those features for which the value of $\mid w_i \mid$ exceeds a certain threshold.

POS tag	Examples	POS tag	Examples
AUX	do, done, have, is	NNPS	Americans
CC	and, both, either	PDT	all, both, half
CD	0.5, 1	POS	”s
DT	all, an, the	PRP	hers, herself,him
EX	there	PRP$	her,his,mine
FW	jeux	RP	along,across
IN	astride, among, whether	SYM	&
LS	DS-400, second	TO	to
NNP	Ranzer	WDT	that, what, which

Table 4.2: Part of the speech tags removed from the review

4.3 Experiments

4.3.1 Data Set

The dataset used is the one used in Pang et al. [24] and it consists of 1000 positive and 1000 negative movie reviews. All the experiments are carried out using the same setting as in Pang et al. [24]. All the experiments are carried out using 10-fold cross validation.

4.3.2 Feature Extraction

The report shows the experiment done with the word unigram, word bigram and word sub-sequences. There are certain grammatical constructs which show no opinion and also provide minimum knowledge about the opinion of the sentence. For example, in the sentence, “*The movie was good*”, the word “*The*” does not bear any opinion and can be easily discarded for the opinion analysis task. Matsumoto et al. [31] have listed out such parts of the speech tags which do not show any opinion. Table 4.2 shows such parts of the speech tag with examples. In all the experiments, words with such parts of the speech tag were removed from the dataset. The dataset was tagged with parts of the speech tags using Stanford’s part of speech tagger [2]. Also all the punctuation symbols were removed from the dataset. The following section explains how each features is extracted from the dataset.

[2]http://nlp.stanford.edu/software/tagger.shtml

- **Unigram**:
  After applying all the filters mentioned above, all the distinct single token words were extracted from the dataset. Only those words whose count in the dataset was more than two were kept.

- **Bigram**:
  After applying all the filters mentioned above, all the distinct bigrams were extracted from the dataset. Only those bigrams whose count in the dataset was more than two were kept.

- **Frequent Sub-sequence**:
  Frequent sub-sequence was extracted from the dataset by applying the algorithm PrefixSpan as explained in Section 4.2.2. PrefixSpan has a simple data input format:

  ```
  Sequence ID (SID) : Sequence (Seq)
  ```

  The number of patterns extracted by the algorithm grows exponentially as the length of the *sequence* grows. Thus, it is a good idea to shorten the sequence as much as possible. Thus instead of using a single sentence from a document as a sequence, the sentences were subdivided into clauses and each clause was considered as a sequence. To divide the sentence into clauses, Stanford's parser was used [3]. From the parsed sentence, occurrence of the Penn Treebank tag *SBAR* is considered as the pivot point to separate the sentence into clauses.

  Figure 4.4 shows the parse tree obtained by parsing the sentence *"Although grand new technology exists that makes the technical sequences, including several mechanical sharks , obsolete , none of it could improve the film because it only would lead to overkill ."* by Stanford's parser. Figure 4.4 also highlights the *SBAR* tag. Table 4.3 shows the clause extracted from the sentence by splitting the sentence using SBAR tag brackets.

  Once the clauses are extracted, all the filters described in Section 4.3.2 are applied. Following the input pattern to the PrefixSpan algorithm, the extracted clauses are fed in to the algorithm as:

  01: although -1 grand -1 new -1 technology -1 exists

[3] http://nlp.stanford.edu:8080/parser/

```
(ROOT
  (S
    (SBAR (IN Although)
      (S
        (NP (JJ grand) (JJ new) (NN technology))
        (VP (VBZ exists)
          (SBAR (IN that)
            (S
              (VP (VBZ makes)
                (NP
                  (NP (DT the) (JJ technical) (NNS sequences))
                  (, ,)
                  (PP (VBG including)
                    (NP
                      (NP (JJ several) (JJ mechanical) (NNS sharks))
                      (, ,)
                      (NP (JJ obsolete)))))))))))
    (, ,)
    (NP
      (NP (NN none))
      (PP (IN of)
        (NP (PRP it))))
    (VP (MD could)
      (VP (VB improve)
        (NP (DT the) (NN film))
        (SBAR (IN because)
          (S
            (NP (PRP it))
            (ADVP (RB only))
            (VP (MD would)
              (VP (VB lead)
                (S
                  (VP (TO to)
                    (VP (VB overkill)))))))))))
    (. .)))
```

Figure 4.4: Parsed sentence highlighting the SBAR tag

Sentence	Although grand new technology exists that makes the technical sequences, including several mechanical sharks , obsolete , none of it could improve the film because it only would lead to overkill.
Clauses	1. Although grand new technology exists 2. that makes the technical sequences, including several mechanical sharks, obsolete 3. none of it could improve the film 4. because it only would lead to overkill.

Table 4.3: The clauses extracted by using SBAR as pivot

02: that -1 makes -1 the -1 technical -1 sequences -1 including -1 several -1 mechanical -1 sharks -1 obsolete
03: none -1 of -1 it -1 could -1 improve -1 the -1 film
04: because -1 it -1 only -1 would -1 lead -1 to -1 overkill

-1 is put between words to tell the PrefixSpan algorithm that the words are ordered, i.e. each item occurs before the next one. This is necessary because we want to preserve the word order in the sub-sequences generated. PrefixSpan takes a set of sequences as input. It finds all the sub-sequences that appear in at least minimum support (MINSUP) % of sequences. MINSUP is a parameter that has to be chosen when running PrefixSpan. For example, in the above four sequences, if we want to mine a single word sequence which occurs in at least half of the sequences, then we have to set MINSUP as 50%. After running the PrefixSpan we will get the result:

Pattern 1 : *the* SID : 02,03
Pattern 2 : *it* SID : 03,04

We get "*the and it*" as a pattern because only "*it*" and "*the*" occur in two different sequences. All the experiments are carried out using MINSUP at 10% and sequence length at 2 or greater than 2. Thus from all the clauses extracted from the sentences of 2000 review documents only those sub-sequences are extracted which occur in at least 10% of the total number of clauses. Table 4.4 shows the frequent sub-sequences extracted from the dataset with the above-mentioned

Frequent Sub-sequences
oddly enough
family entertainment
rather good
always great
yet performance
actor performance
dialogue bad
sense movie
fun movie
special bad
problem film
special effects good
film very entertaining
is very good film
is violence would be

Table 4.4: Sample 15 frequent sub-sequences extracted from dataset

Features	*Accuracy %*
Unigram	85.0
Bigram	83.2
Frequent sub-sequences	86.4458

Table 4.5: Accuracy obtained on each feature type

setting.

4.3.3 Results

After pre-processing and sub-sequence feature generation, the features are trained using the SVM. The features are fed in as presence rather than its count in the SVM feature file. LIBSVM[4] with linear a kernel is used in all the experiments. The learning parameter of the linear kernel, *soft margin parameter* (C), is adjusted using the grid search tool provided with the LIBSVM. The grid search tool takes in the feature vector and with a cross-validation technique returns the optimised value of parameter C.

Table 4.5 shows the result obtained in all the different feature settings.

[4]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Authors	*Accuracy %*
Pang et al. [24] (unigram)	87.0
Matsumoto et al. [31] (dependency subtree)	93.2
Mullen et al. [20] (lemmatized unigram+semantic orientation of words)	84.6

Table 4.6: Accuracy obtained on the same dataset by different authors

Table 4.6 shows the top accuracy obtained by different authors on the same dataset. Unigram-based accuracy is considered as the baseline accuracy. We can observe that a unigram alone can achieve decent accuracy. The respectable accuracy obtained by using a unigram alone backs our point of using feature selection to extract highly related unigrams.

Context capturing is essential in opinion analysis. We can take a very simple example to illustrate the importance of context in opinion analysis. The word "*good*" has a positive opinion but when used as "*not good*", the whole term becomes negative. Bigrams are used as a feature to help capture contexts that a unigram cannot capture. But the accuracy obtained by bigrams is lower than that of a unigram as shown in Table 4.5. This proves that bigrams as features are not effective in capturing the context.

On the contrary, the sub-sequences have quite high accuracy compared to bigrams and higher than unigram too. Although the increase in accuracy from *unigram* to *frequent sub-sequences* is not statistically significant but the difference in accuracy from *bigram* to *frequent sub-sequences* is statistically significant ($p < 0.05$, paired t-test). This proves that using sub-sequences as a context capturing mechanism is quite effective compared to using bigrams. This is because, as shown in Section 4.2, the context-related information is in a longer range rather than in a shorter range. Thus, instead of using continuous words as a context, a non-continuous scheme such as sub-sequences is more effective in capturing context.

Features	*Accuracy %*
Unigram + Frequent sub-sequences	85.8434
Unigram selected	86.3
Unigram selected + Frequent sub-sequences	97.69

Table 4.7: Accuracy before and after feature selection

Table 4.7 shows the accuracy obtained by combining unigram and subsequence features. The accuracy obtained in this setting is 97.69%, which

is the highest and highly statistically significant ($p < 0.01$, paired t-test) compared to all the other features and also the highest accuracy obtained in this dataset with similar settings. It outperforms all the other methods used thus far.

For combining two different features, two different experiments are carried out. First, all the extracted unigrams and all the extracted frequent features are combined and trained with SVM. The accuracy obtained is 85.84% which is less than using sub-sequences alone. This proves our previous statement that using all unigram adds to the noise in the features. With the addition of all the unrelated features, the classifier performs poorly.

Secondly, we used our feature selection technique as described in Section 4.2.3 and only extracted those unigrams which were significant. All the unigrams were sorted according to their weight of the normal from the classifier's hyperplane. From this sorted data only the top half of the unigrams was combined with the frequent sub-sequences. Table 4.8 shows the top 15 unigrams obtained after sorting. The classifier, thus trained, obtained an accuracy of 97.69%, which is quite high compared to all the other accuracies for different features and is also higher than the results obtained by other methods as shown in Table 4.6.

The relatively high accuracy obtained by using only a unigram as shown in Table 4.5 points to the fact that the unigram can be an important feature for opinion analysis. But when used in conjunction with proper context capturing techniques like sub-sequences, it hurts the performance by adding too much noise. But with proper feature selection techniques the advantage of a unigram can be used in combination with sub-sequences to achieve commendable accuracy.

4.4 Related Work

Our work is primarily an automatic way of extracting features from the data. Our work expands on the work done by Matsumoto et al. [31]. Their work is based on using frequent sub-sequences and frequent dependency sub-trees as features for sentiment classification. Our work is primarily based on frequent sub-sequences as features. Our thinking is that using a dependency

Top 15 selected unigrams
worst
awful
waste
bad
nothing
poor
memorable
unfortunately
plot
only
boring
as
mess
excellent
have

Table 4.8: Top 15 weighted unigram features

sub-tree makes the features redundant, as there is no relation left which sub-sequences do not capture. Also we use a feature selection technique to extract important word tokens from the dataset and combine them with frequent sub-sequences to get a very high accuracy for classification. For completion we had to re-define various aspects of sub-sequences, already defined in [31], but we have put on a slightly different approach to it.

Another closely related work is a features extraction technique used by Wilson et al. [35]. They used a manually annotated MPQA dataset and extracted different features from the data and learnt a polarity classifier for a clue instance present in the data. For example, in the sentence :

```
‘‘They have not succeeded, and will never succeed’’
(positive), in breaking the will of this valiant people.
```

The clue instance is the phrase *They have not succeeded, and will never succeed*. Even though the phrase itself is negative its contextual polarity in the whole sentence is positive. With data annotated with such clue instances they built a classifier which could predict the contextual polarity of such clue instances. To accomplish this they used different features, namely:

- Word tokens: includes the word used in the clue instance along with

words before and after the clue instance. Prior polarity of such a word token, if present in a lexicon, is included.

- Modification features : these are the binary features which are true if, clue instance is *preceded by an adjective*, or *preceded by an adverb* or *preceded by an intensifier [26]* and other polarity modifiers.
- Negation features : these are also binary features whose values are true if the clue instance is negated by words present within certain word frames of the clue instance.

An SVM classifier trained with the above features could disambiguate the contextual polarity of any clue instance with **81.6%** accuracy. We worked on a document level polarity classification rather than a clause level classification, but their work on feature extraction was a great inspiration for us. They showed how carefully crafted features can radically increase the accuracy of opinion classification of text. They extracted modification features from the dependency tree of the sentence. Most of the modification features did not occur right after the clue instance, thus using an n-gram model was not efficient. We took more automatic ways of extracting the features, rather than manually annotating all the features. Sub-sequences as the features were successful to capture relevant features for sentiment classification. Also we chose to learn the lexicon from the data to be used as a prior polarity rather than use a manually generated lexicon.

Mullen at al. [20] used Semantic Orientation (SO) as a feature. The SO for a *phrase* is the difference between its Point-wise Mutual Association (PMI) with the word "excellent" and its PMI with the word "poor". The PMI for a phrase was calculated by counting the number of *hits* obtained by querying the search engine with keywords {*phrase* + "poor"} or {*phrase* + "excellent"}. SVM trained with a lemmatised unigram combined with SO obtained an accuracy of 84.6%. Instead of using the SO of the words as a feature we chose to use selected words as features.

4.5 Conclusions

We proved that sub-sequences are better as features in sentiment analysis than bigrams. The results obtained by using only sub-sequences as features

easily beats the results obtained by using bigrams as features. The result also compares to the result obtained by using manually crafted rules as in [35] (see Section 4.4). This further proves that sub-sequences are more efficient and effective in capturing long-range relations (see Sec5.1) and better at capturing context than using an n-gram approach. Also we conclude that word tokens can become very effective for sentiment classification if used selectively. Using all the words in an opinionated text hurts the classification, but selecting a small fraction of the total words through a feature selection technique greatly improves the accuracy. The result obtained by combining the selected unigram and the frequent sub-sequences generated beats all the results obtained in the data set till to date.

Chapter 5

Target Extraction

5.1 Introduction

A review expresses an opinion on a certain topic. These topics are the targets of the opinions. The target could be a *movie*, or a product like a *camera, kitchen appliance, book etc* . For example,

(a) This *knife* is great for cutting.
(b) The *movie* was superb.

The above two sentences are reviews about a *knife* and a *movie* respectively. Without the review targets, sentiment analysis is of little value. It makes no sense in identifying the orientation of the review without identifying its target. For example, for a sentence *The book was great*, from a review about a *movie*, it would be positive if it was analyzed without the target. The analysis would be true for a book's review but for a review about a *movie* the sentence can easily be regarded as a negative sentence.

Within a single topic there can be multiple targets. For example, a review about a movie consists of reviews about actors, cinematography, story etc. A combination of all these reviews form a review for a movie. It is also not necessary that a negative review about a movie will have a negative review about all its constituents' targets. A reviewer can easily dislike the *story* of a movie but find the *acting* superb. A camera can easily have a magnificent *shutter speed* but an appalling *viewfinder*. The *likes* and *dislikes* of a product

is an emerging market trend. The pros and cons initiate/force improvements and support purchases. Thus it is important to identify different targets within a single review and extract the sentiment being expressed in each.

This chapter explains an easy-to-implement association-based approach to acquire targets on different review topics. The method assumes no prior knowledge on any particular topic and only requires data grouped into different topics. This unsupervised method is based on the observation that word has different associations relating to different topics. The association relation can be acquired from statistical hypothesis test.

The basic principle of this approach is to extract targets for a topic depending on its association with the given topic and also other topics. The method is simple yet effective. Target pruning is applied to improve the results. In evaluation, the results are tested against the manually annotated data and also a comparison between other target extraction methods are shown. The results are compared to other methods which use more prior knowledge than our approach.

5.2 Main Concept

Log-Likelihood (LL) target extraction system is based on a very simple concept of association between the review and its topic.

A review consists of primarily opinion words, targets of the opinions expressed by sentiment-bearing words and other grammatical necessities. For review of any products, this basic form is valid. Thus, a review about a camera and a movie will have the same set of opinion words and the same set of grammatical necessities. For example, consider the following sentences:

(a) It was a *fantastic* movie.
(b) This camera takes *amazing* pictures.

Sentence (a) is a review about a movie and sentence (b) is a review about a camera. Now, if we rewrite the sentences by interchanging the italicised words, we will get:

(a) It was an *amazing* movie.
(b) This camera takes *fantastic* pictures.

	Camera	Grocery
Sentences	g3 worth every single cent i spent on it; i really haven't taken a bad picture yet; the lens cover is surely loose	Coconut had bad flavour; Choc bar cost a little more than cent; The noodles loosen up after you stir it
Targets	g3, picture, lens, cover	Coconut, Choc bar, noodles

Table 5.1: Review sentences and potential non-common targets

We can see that interchanging the words makes no significant difference to the subjectivity of the sentence, even if it had changed the orientation of the polarity like in *unpredictable movie* and *unpredictable steering*. Words like *fantastic and amazing* could have been used in either review. The only major difference in the context of sentiment analysis in these two sentences is the targets. One review is about a camera while the other is about a movie. This simple fact about the nature of the reviews leads to a simple hypothesis that:

> *The content of two reviews about different products would primarily only differ by the product they are reviewing.*

Thus if we filter out all the common parts between the reviews about different topics, what remains is the target of the reviews. Such a filter is achieved by using the log-likelihood ratio principle. Log-likelihood ratio rewards distinct associations but punishes common associations. Table 5.1 shows different sentences taken from a review about a camera and a review about grocery products. The removal of common words (including similar grammatical constructs) between these reviews and also considering only nouns yields words, which are the targets of each individual review.

5.2.1 Log-Likelihood

To find the association between the word and the topic we use the log-likelihood ratio (LL) [8]. The word with the highest topic-to-word LL is most strongly associated with the topic. The association of the word to the topic is calculated in comparison to the association of the same word with a different topic. From the above discussion we conclude that targets are very specific to the domain. Thus we want to reward words which are

highly associated with the domain of concern and at the same time punish the words which are highly associated with some other domain. In a simple form, this can be represented by the following equation:

$$\text{Association Ratio} = \frac{\text{Association with current domain}}{\text{Association with other domain}}$$

Thus a potential target is one which has a higher value for the numerator and lower value for the denomenator, i.e a potential target will have a high *association ratio*. We calculate the association ratio by calculating the log-likelihood ratio of topic T to word W. In order to compute the log-likelihood ratio of topic T to word W, we create a contingency table for each topic. The contingency table is shown in Table 5.2. The contingency table contains the observed value taken from the corpus.

C[i,j]	**Count in topic camera**	**count in topic ¬camera(eg Grocery)**		
lens	11	1	**12**	*C[lens]*
¬lens	1304	6535	**7839**	*C[¬lens]*
	1315	**6536**		
	C[Total camera]	*C[Total ¬camera]*		

Table 5.2: The contingency table to calculate LL ratio. Here, *C[i,j]* denotes the count of the number of times j occurs in i. Total corpus size is N=7851.

The LL value of topic T and word W is given by,

$$LL(T,W) = \sum_{i\epsilon\{T,\neg T\},j\epsilon\{W,\neg W\}} 2C(i,j)log\frac{C(i,j)N}{C(i)C(j)}$$

Table 5.2 shows the contingency table created for calculating the log-likelihood ratio for topic *camera* to the word *lens*. The data used is a review text on the *camera* domain and the other domain (¬camera) is the review about *groceries*. We can see that of the 1315 tokens in camera data the word *lens* occurs 11 times and of the 6536 word tokens in the grocery data the word *lens* occurs 1 time. Thus LL will be computed as:

$$LL(camera,lens) = \sum_{i\epsilon\{\text{camera},\neg\text{camera}\},j\epsilon\{\text{lens},\neg\text{lens}\}} 2C(i,j)log\frac{C(i,j)7851}{C(i)C(j)}$$

or

$$LL(camera,lens) = -6148.88$$

Similarly, if we take the LL value of *"good"* which occurred 75 times in the

Canon(camera)	Nokia(mobile)	Zen(mp3)	DVD Player	Nikon(camera)
camera	phone	player	player	camera
picture	nokia	software	dvd	picture
canon	product	ipod	apex	card
lens	radio	song	disc	battery
g3	service	music	unit	mode
image	battery	battery	picture	nikon
product	screen	zen	problem	pics
battery	speakerphone	amazon	button	image
photo	feature	computer	christmas	shot
flash	option	nomad	movie	lens
mode	camera	button	dvds	setting
film	menu	cd	output	quality
viewfinder	reception	device	display	model
card	game	unit	model	resolution
software	voice	product	feature	flash

Table 5.3: Top 15 words extracted for each category.

camera review and 80 times in the grocery review then

$$LL(camera, good) = -6126.28$$

the magnitude of LL(camera,good) is quite low compared to LL(camera,lens) considering there is a very minute change in LL value over large change in count value of the words. Thus through LL we can separate out target words from non-target words.

5.2.2 Filter

According to [13] target words are commonly nouns. Following this approach, only nouns were extracted as targets. This significantly increased precision. The tagged targets from the dataset are also all nouns.

Table 5.3 shows the top 15 ranked targets extracted for each dataset by the LL system.

Dataset	Number of tokens	Number of noun tokens	Number of targets
Canon G3(camera)	11547	2311	55
DVD player	12051	2264	52
Zen(mp3)	31705	6000	96
Nikon(camera)	6498	1315	31
Nokia(mobile)	9290	1851	67

Table 5.4: Corpus Statistics

5.3 Experiments

This section evaluates the proposed method. The dataset is described first, then the evaluation and comparison of results with the double propagation [12] method is shown.

5.3.1 Data Set

The data used is the customer review [1] from [13]. The dataset consists of reviews of 5 different products: 2 cameras, 1 DVD player, 1 MP3 player and 1 mobile. The detailed statistics of each dataset is given in Table 5.4. The data set is already tagged with targets for each applicable sentence. The tagged target is in its base form and is converted to lower case.

Data Preparation

All the punctuation marks from the data ware removed. The data were then converted into tokens and each category had one file which had a single word in each line. The tokens were then lemmatised using treeTagger[2].

The association test could be performed in pairs of topics or in a one vs the rest strategy. The experiments showed an insignificant difference between the results obtained from both of the methods. When considering the pair method for association mining, the best result is obtained when the pair are as dissimilar as possible. For example, if the pair considered for the test are the topics *camera* and *dvd player*, then the result might be poor since both topics have many features in common. For example, *battery, screen and picture* are potential targets for both camera and DVD player. So naturally

[1] http://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html
[2] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

Dataset	Precision
Canon G3(camera)	0.65
DVD player	0.71
Zen(mp3)	0.72
Nikon(camera)	0.55
Nokia(mobile)	0.76

Table 5.5: Precision obtained in each dataset by LL method

Dataset	Precision
Canon G3(camera)	0.87
DVD player	0.90
Zen(mp3)	0.81
Nikon(camera)	0.90
Nokia(mobile)	0.92

Table 5.6: Precision obtained in each dataset by Liu et al.

the contents of the topics would have a higher presence of these words. Since the log-likelihood test punishes common association, these words will have lower values as potential targets and may be disregarded. On the other hand, if the pair considered for the topics are, for example, *camera* and *grocery*, there is very little chance that the targets of these topics are the same. Thus, targets in this case will be extracted with higher precision.

5.3.2 Result

Table 5.5 shows the precision obtained by the LL system in each dataset. The precision is calculated by selecting the top N targets from the ranked list generated by LL system. N is set to the number of annotated targets extracted for each dataset. With this setting calculation of recall becomes insignificant as it equals the precision.

The precision obtained by the LL system cannot be directly compared to current state of the art [12] which also uses the same dataset. The reason for this is that they have included both a single word target and phrasal target like *battery life* in their evaluation. The highest precision obtained by their system for each dataset is shown in Table 5.6. The data was taken from their paper. The dataset used is a medium sized dataset with a significantly small number of target words. The Double Propagation [12] system works

well in medium-sized corpora but will introduce noise in larger corpora and leads to low precision. When applied to larger corpora their precision falls to as low as 0.62 [39]. This can also be seen from Table 5.6, where there is a distinct fall in precision from the dataset with fewer number of word tokens to the dataset with a comparatively higher number of tokens: *Nokia Pr: 0.90 #:6498 to Zen Pr: 0.81 #:31705.* Contrary to this, the LL system proposed here reacts better to the increase in the number of tokens or the potential targets, as seen by the increase in the precision in Zen MP3 player and Nokia mobiles. This is due to the fact that, as the size of the dataset increases, the ratio of the number of distinct words in each topic to the number of common words between topics decreases, thus the association between the distinct words will have a comparatively higher value. This will lead to identifying more targets with much higher ranks than the non-targets words.

To support this theory, a simple experiment was done. To increase the amount of data and number of potential targets, both of the camera reviews in the dataset (Canon + Nikon) were combined to form one whole review. This led to a significant increase in the number of tokens and a slight increase in the number of targets, as most of the targets were common to both the reviews. The new statistics are shown in Table 5.7.

Dataset	Number of tokens	Number of noun tokens	Number of targets
Camera(Canon+Nikon)	18054	3626	64

Table 5.7: Corpus statistics of the camera

The log-likelihood ranking thus generated with this new data showed an increase in precision by 6% to the previous high for the camera. This proves that the LL system works better with larger data and a larger number of targets. As the data size increases, the top ranked product becomes more distinct which results in the increase of precision. The top 15 products extracted from the new combined data are shown in Table 5.8.

Camera(Canon+Nikon)
camera
picture
product
battery
lens
canon
card
image
g3
mode
photo
flash
nikon
shot
quality

Table 5.8: Top 15 target extracted from combined Camera dataset

5.4 Related Work

- [18] uses the topic sentiment mixture model to extract topic and sentiment together. The LL approach is different from this, as it only extracts topics. Mei et al's [18] approach lacks clear evaluation for any comparison and such topic modeling methods can only extract coarse/general topics. The LL approach proposed here can even extract very infrequent targets.

- [13] classified targets in any review as frequent nouns and noun phrases. The LL system also uses only nouns. But instead of just mining frequent nouns it uses the association principle. It is not always the case that frequent nouns are the target in reviews. For example, in our dataset on the camera review, the word "software" occurred just five times. One of the review of the software is, "Software in it is quite bad". Thus we can see that "software" is the potential target in the review. In a corpus with other targets such as "camera" occurring 137 times, 5 is very infrequent and according to Hu's approach can be neglected. Our system ranked "software" quite high as the potential target.

- Popescu et al. [27] also followed the same approach as Hu's to mine noun targets. Their algorithm requires that the product feature is known. The algorithm determines whether nouns/noun phrases are targets by computing Pointwise Mutual Information (PMI) between phrases and class-specific determiners, e.g. "x has", "x is". The LL system is different from theirs as it does not need to search through the dataset to identify the part relation which can be quite time consuming for a large dataset.

- Lui [39] uses an information extraction method to mine product targets and opinion lexicons together. The Double Propagation approach is based on the fact that the opinion targets are modified by some *mod* relation as given by the dependency parser. They use an initial opinion lexicon and target lexicon to search through the dataset to identify such relations. The bootstrapping process continues until no further opinion words or targets are found. LL Approach differs from theirs as the LL approach does not need any seed lexicon thus it is purely unsupervised. Also the double propagation method uses only direct dependency relations but the target and opinion words are also connected through the indirect dependency relation [12]. Thus the double propagation method fails in such cases, and also when the size of data increases it tends to insert more noise thus decreasing precision. The result section shows that the LL method shows better results with an increase in size of the dataset.

- Stoyanov at al. [30] extract topics through topic co-reference resolution. Their work is based on the hypothesis that the two opinions are topic co-referent if they share the same opinion topic. To accomplish this they form a cluster of co-referent topics and label the cluster with the names of the topics. They train a classifier on the clusters. Even though the idea of target clusters according to topics shares similarities with the LL approach, the LL approach is quite different from theirs as it is unsupervised and does not need any labelled data.

5.5 Application

As already discussed in Section 5.1, target extraction plays an important role in sentiment analysis. The most important application is to extract clauses which are relevant towards providing the sentiment of the review. It is already established that opinion is expressed towards the target, thus any opinionated clause will have a target present in it. Thus, to extract any opinionated clause, we can extract clauses which have a target word present in it. Below are the clauses which were extracted from the camera review.

- the **software** was terrible
- It 's **batteries** died all the time and lost all of the pictures .
- The neck **strap** is not worth the difference
- And the **price** is right for the features and MP
- This **lens** is one of my favorites
- **Camera** comes with the software

The clauses were extracted with the highlighted targets. It can been seen that the target-based extraction of the opinionated clauses works reasonably well. Since the pronominal resolution of the corpus was not done, clauses like "**it** was great" were not extracted. The other limitation of this approach is that the target containing term may not be subjective/opinionated all the times. It may be the case that the reviewer is just talking about the product without expressing any opinion on it. For example, the last clause in the above list has a target word *camera* but does not show any opinion. But this can be solved by extracting the clauses which contain both targets and the opinionated word. So a subjective clause will be the one which has the presence of not only the target word but also the word which expresses an opinion towards that target.

Chapter 6

Conclusions and Directions for Future Work

This thesis investigated the various aspects of opinion analysis, namely, opinion lexicon extraction, opinion classification (including multi-class), opinion target extraction and summarising opinions of reviews.

The opinion lexicon extraction and opinion classification focused on using the compositional property of opinionated text. The thesis described an additive model with constraints optimisation which is used for both opinion lexicon extraction and opinion classification.

The thesis described an unsupervised algorithm to acquire opinion targets from the reviews. It also investigated the relations between opinion targets and opinionated words to devise an algorithm for extraction of fine-grained subjective clauses.

The thesis also showed the use of feature selection methods to improve on a previous work of opinion classification.

6.1 Summary of Results and Contribution

6.1.1 Weighted Opinion Lexicon

The major part of the thesis revolves around the idea that in a given domain two opinionated words can have different weights, even if they share the same polarity. If such is the case, the thesis showed with examples how a weighted lexicon can be used to successfully identify the polarity of the opinion on a text. The thesis explained an effective algorithm to generate such weighted lexicon from opinionated text. The generated lexicon when used for opinion classification showed promising results. The result bolstered our claim that the opinion lexicon should not only contain the polarity of the words but should also contains weight of the polarity.

6.1.2 Linear Sum for Opinion Classification

The thesis showed how an opinionated text can be represented by a linear additive equation. This part of the work was influenced by the successful use of compositionality for opinion classification in [19]. Contrary to prior belief that opinion classification of a text cannot be done by linear sum of constituent opinion words, the thesis successfully built a model which used the linear sum of weighted opinion words for opinion classification.

6.1.3 Need for Multi-class Opinion Classification

The thesis showed that a text cannot just be positive and negative; some are more positive/negative than others. With examples, the thesis showed the need for multi-class opinion classification. The experiments showed that our additive model which worked well for binary opinion classification failed to produce impressive results for multi-class opinion classification. We also used SVM for the same task; the results obtained by SVM for multi-class classification were also very low on accuracy as compared to SVM's result on binary classification. This led us to the conclusion that the irregularities in tagging a review as either 1 star or 2 star (5 star or 4 star) makes the multi-class opinion classification a very difficult problem.

6.1.4 Fine-grained subjectivity by exploiting opinion targets and opinion words relations

Many previous works used relations between opinion targets and opinion words to extract opinion targets and opinion words. The thesis showed that such relations can also be used to extract sentences/clauses which show an opinion towards a single feature/target of a topic. Using the output of syntactic parser, we developed hierarchical rules to successfully extract feature/target specific clauses/sentences from the whole review text. Since not all the words in a review text are opinionated and an opinion word modifies its target through certain syntactic modifiers, we can select a subset of words from the review text which will be subjective and the opinion will be targeted to a single feature.

6.2 Directions for Future Work

The important contribution of our work is the use of a linear additive model to classify the polarity of the opinionated text. We successfully showed that classification tasks can be achieved by the linear sum. We tried to use the same model for the multi-class classification but could not attain good results. Our model is based on the principle of weight distribution of opinion throughout the opinionated text. The lack of better accuracy on multi-class opinion classification is due to non-disambiguating opinion weight distribution among the classes in the middle, i.e 2 and 4 stars in our case. Possible future work would be to incorporate additional knowledge about the text with star rating 2 and star rating 4 and add this knowledge as an additional parameter in our model. The most basic knowledge could be the number of positive and negative sentences in the text. The use of such knowledge has shown an increase in performance in previous studies. We could formulate our constraints not only to incorporate an opinion lexicon but also parameters for multi-class opinion classification.

Further future work will be to extract not only single words as opinion words and target words but to extract phrases as potential opinion expression and target expression. The phrasal target extraction can in turn improve our feature selection algorithm.

Bibliography

[1] Alina Adreevskaia and Sabine Bergler. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 209–216, 2006.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. pages 3–14, 1995.

[4] Farah Benamara, Carmine Cesarano, Antonio Picariello, Diego Reforgiato, and V. S. Subrahmanian. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007. Short paper.

[5] Eric Breck, Yejin Choi, and Claire Cardie. Identifying expressions of opinion in context. In *IJCAI*, pages 2683–2688, 2007.

[6] Yejin Choi and Claire Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *EMNLP '08 : Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[7] Yejin Choi and Claire Cardie. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *EMNLP*, pages 590–598, 2009.

[8] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *COMPUTATIONAL LINGUISTICS*, 19(1):61–74, 1993.

[9] Andrea Esuli and Fabrizio Sebastiani. Determining the semantic orientation of terms through gloss analysis. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*, 2005.

[10] Andrea Esuli and Fabrizio Sebastiani. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of Language Resources and Evaluation (LREC)*, 2006.

[11] Angela Fahrni and Manfred Klenner. Old Wine or Warm Beer: Target-Specific Sentiment Analysis of Adjectives. In *Proc.of the Symposium on Affective Language in Human and Machine, AISB 2008 Convention, 1st-2nd April 2008. University of Aberdeen, Aberdeen, Scotland*, pages 60 – 63, 2008.

[12] Jiajun Bu Guang Qiu, Bing Liu and Chun Chen. Opinion word expansion and target extraction through double propagation. *COMPUTATIONAL LINGUISTICS*, 2010.

[13] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM.

[14] N. Milic-Frayling J. Brank, M. Grobelnik and D. Mladenic. Feature selection using linear support vector machines. In *Technical report, Microsoft Research*, 2002.

[15] Nobuhiro Kaji and Masaru Kitsuregawa. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1075–1083, 2007.

[16] Hiroshi Kanayama and Tetsuya Nasukawa. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP*, pages 355–363, 2006.

[17] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA, 2005. ACM.

[18] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 171–180, New York, NY, USA, 2007. ACM.

[19] Karo Moilanen and Pulman Stephen. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, pages 378–382, September 27-29 2007.

[20] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 412–418, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[21] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278, 2004.

[22] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.

[23] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, January 2008.

[24] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[25] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. pages 215–224, 2001.

[26] Livia Polanyi and Annie Zaenen. Contextual lexical valence shifters. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, 2004.

[27] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[28] Dan Roth and Wen tau Yih. A linear programming formulation for global inference in natural language tasks. In *In Proceedings of CoNLL-2004*, pages 1–8, 2004.

[29] Vikas Sindhwani and Prem Melville. Document-word co-regularization for semi-supervised sentiment analysis. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1025–1030, Washington, DC, USA, 2008. IEEE Computer Society.

[30] Veselin Stoyanov and Claire Cardie. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 817–824, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[31] Shotaro Okumura Takamura, Hiroya Matsumoto and Manabu. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceeding of PAKDD'05, the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 301310, page 3518, 2005.

[32] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424, 2002.

[33] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *CIKM '05: Proceedings of the 14th*

ACM international conference on Information and knowledge management, pages 625–631, New York, NY, USA, 2005. ACM.

[34] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210, 2005.

[35] Theresa Wilson and Janyce Wiebe andPaul Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433, 2009.

[36] Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. *Data Mining, IEEE International Conference on*, 0:427, 2003.

[37] Jeonghee Yi and Wayne Niblack. Sentiment mining in webfountain. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 1073–1083, Washington, DC, USA, 2005. IEEE Computer Society.

[38] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[39] Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. Extracting and ranking product features in opinion documents. In *Coling 2010: Posters*, pages 1462–1470, Beijing, China, August 2010. Coling 2010 Organizing Committee.