Recognizing contextual valence shifters in document-level sentiment classification

Sara Ahmed Morsy

THE AMERICAN UNIVERSITY IN CAIRO

SCHOOL OF SCIENCES AND ENGINEERING

RECOGNIZING CONTEXTUAL VALENCE SHIFTERS IN DOCUMENT-LEVEL SENTIMENT CLASSIFICATION

A THESIS SUBMITTED TO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR

THE DEGREE OF THE MASTER OF SCIENCE

BY:

SARA AHMED MOHAMED MORSY

SUPERVISOR

DR. AHMED RAFEA

SUMMER 2011

ACKNOWLEDGEMENT

In the name of Allah the most gracious and most merciful

I thank my great supervisor, Dr. Ahmed Rafea, who always helped me, welcomed my questions and gave me a lot of recommendations and suggestions. I would not have reached this phase, if it were not for his permanent support, advice, and guidance.

I would also like to express my thanks to my thesis committee members, Dr. Awad Khalil, Dr. Reem Bahgat and Dr. Sherif G. Aly, for their support, guidance and helpful feedback.

Finally, I thank my beloved parents, siblings and friends for their permanent support, appreciation and patience. I am also grateful for my husband, who has given me a lot of help and support during doing my research and while running the experiments. I would like to dedicate this thesis to them all.

Abstract

Sentiment classification is an emerging research field. Due to the rich opinionated web-content, people and organizations are interested in knowing others' opinions, so they need an automated tool for analyzing and summarizing these opinions. One of the major tasks of sentiment classification is to classify a document (i.e. a *blog*, *news article* or *review*) as holding an overall positive or negative sentiment. Machine learning approaches have succeeded in achieving better results than semantic orientation approaches in document-level sentiment classification; however, they still need to take linguistic context into account, by making use of the so-called *contextual valence shifters*. Early research has tried to add sentiment features and contextual valence shifters to the machine learning approach to tackle this problem, but the classifier's performance was low.

In this study, we would like to improve the performance of document-level sentiment classification using the machine learning approach by proposing new feature sets that refine the traditional sentiment feature extraction method and take contextual valence shifters into consideration from a different perspective than the earlier research. These feature sets include: 1) a feature set consisting of 16 features for counting different categories of contextual valence shifters (intensifiers, negators and polarity shifters) as well as the frequency of words grouped according to their final (modified) polarity; and 2) another feature set consisting of the frequency of each sentiment word after modifying its prior polarity. We performed several experiments to: 1) compare our proposed feature sets with the traditional sentiment features that count the frequency of each sentiment word while disregarding its prior polarity; 2) compare our proposed feature sets after combining them with stylistic features and n-grams with traditional sentiment features combined with stylistic features and n-grams; and 3) evaluate the effectiveness of our proposed feature sets against stylistic features and n-grams by performing feature selection. The results of all the experiments show a significant improvement over the baselines, in terms of the accuracy, precision and recall, which indicate that our proposed feature sets are effective in document-level sentiment classification.

CONTENTS

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

FCA	First Common Ancestor
FN	False Negative
FP	False Positive
IG	Information Gain
IMDb	Internet Movie Database
ML	Machine Learning
NB	Naive Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
SO	Semantic Orientation
SVM	Support Vector Machines
TN	True Negative
TP	True Positive

CHAPTER 1

INTRODUCTION

People are now interested in knowing what other people think. Due to the rich opinionated content on the web, such as blogs, reviews, web forums, social networks and others, people can now know others' opinions about different topics without even knowing them. Such available opinions are significant at the personal, corporate and governmental levels. At the personal level, most people now do online product research before buying any products to know what is said in the reviews about this product and their buying decisions get affected by these reviews. People also tend to gather political opinions and sharing them on the web. At the corporate level, companies always seek their consumers' feedback about their products and/or services by conducting surveys and focus groups, since it will be a laborious task to gather and analyze hundreds or thousands of reviews published online. Finally, at the governmental level, governmental organizations are interested in knowing people's reactions towards the new rules and regulations that they set [26].

Therefore, an automated tool is needed to crawl opinions about different topics on the web, according to one's interest, and display them in a summarized, meaningful, and less confusing or overwhelming way such that he/she does not have to manually read or analyze them. As a result, sentiment classification has been emerging as a new field in computational linguistics, natural language processing (NLP) and machine learning (ML). It is the area of research that attempts to identify the sentiment/opinion of a piece of text (whether a document, sentence or phrase) that is held from the author of the text [20].

In this study, we are interested in document-level sentiment classification, where the goal is to classify a whole document (i.e. a review, a blog, a news article ...etc) as holding an overall positive or negative sentiment. We have studied the early and current research done in this area, and we would like to improve the performance of the document-level sentiment classification by proposing new features to use.

This chapter is organized as follows: section 1.1 describes the problem definition for undertaking this specific area of research in our study; section 1.2 explains our motivation and goal for working in document-level sentiment classification as well as our main contributions in this study; and finally section 1.3 will list the chapters and sections in this study and what each one will talk about.

1.1 Problem Definition

Many researchers have focused their work on document-level sentiment classification, where the task is to identify the overall sentiment of a given document, assuming it holds the opinion of one author and talks about one object. There are mainly two approaches for this task: the ML and semantic orientation (SO) approaches, as will be discussed in section 2.2. Some researchers have developed document-level sentiment classification tools using the current ML algorithms, such as Support Vector Machines (SVM) and/or Naive Bayes (NB), using different feature sets to represent the documents; whereas others have centered their attentions on building sentiment dictionaries (lexicons) with all possible polar words with a corresponding number or score showing their polarity and/or semantic intensity. Both approaches have their advantages and disadvantages. For instance, the ML approach is typically supervised and needs a huge annotated training data, but it yields high performance when testing on data from the same domain. On the other hand, the SO approach is unsupervised, yet it does not generate such good results as the ML. Hence, there have been recently some attempts to combine both approaches using different techniques, as we will show in section 2.4. One example of such hybrid approaches is to add the frequency of SO-carrying words as features in the ML approach. However, counting the frequency of sentiment words does not yield accurate results due to the contextual polarity of some of these words. Computing the contextual polarity of sentiment words has shown a significant improvement in the SO approach [7, 34], but it has only been handled in sentence-level and phrase-level sentiment classification [39, 22, 8].

1.2 Motivation and Goal

Sentiment classification is different from classic text classification, since the latter depends on a set of keywords and phrases that can be used as features, for topical text classification for instance. However, sentiment classification cannot depend on a set of keywords and/or phrases only, since opinions can be expressed in more subtle ways that can be hard for the computer to learn. Hence, new features were designed and explored specifically for sentiment classification, such as: a sentiment lexicon. A sentiment lexicon should contain all words and/or phrases that carry SO. Many researchers have developed different sentiment lexicons, such as SentiWordNet [13, 5] and the Subjectivity dictionary [39]. Still, using individual words only to extract sentiment was not efficient, since the word's polarity can be affected by its context; e.g. the polarity and semantic intensity of the word "*good*" is different from that of the words "*not good*" and "*very good*", respectively. Also, not all sentiment words should be regarded as carrying SO in special conditions, such as conditional and contrast words as well as modal verbs; e.g. in the sentence "*Although he was **brilliant**, he **failed** the exam*", the adjective *brilliant* should be disregarded as a sentiment word, such that the overall sentiment of the sentence is negative, from the verb *failed*. Hence, the authors in [28] defined different categories for contextual valence shifters that can change the prior polarity of the SO-carrying words and hence should be considered when extracting sentiment, as we will explain in section 2.1. The authors in [39] developed a sentence-level classifier that considers two types of the contextual valence shifters; intensifiers and modifiers. In addition, the authors in [7, 34] developed a sentiment lexicon (SO-CALculator) and additional features that take contextual valence shifters into consideration and then used them to extract all sentiment words and sum up their polarity scores.

The ML approach has outperformed the SO approach in the literature for the sentiment classification task. Therefore, in this research, we wanted to improve the accuracy of the document-level sentiment classifier even further by incorporating contextual valence shifters in an effective way and refining the sentiment feature extraction method used in [9], where they extracted all SO-carrying words while disregarding their contextual polarity.

Our main contributions in this study are: 1) To refine the traditional sentiment feature extraction method that counts the frequency of each sentiment word while disregarding its contextual polarity; 2) To propose a new technique to extract contextual valence shifters in the ML approach in a way that the classifier can learn the effect of these shifters; 3) To show that our proposed feature set to handle the effect of contextual valence shifters is effective and relevant for the classifier; and 4) To improve the sentiment classification performance at the document-level by incorporating our proposed feature sets.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows: chapter 2 contains 3 sections. Section 2.1 defines and explains the categories of contextual valence shifters, section 2.2 reviews the different approaches used for document-level sentiment classification, section 2.3 explains the theoretical foundations of SVM and the dependency parser, and then section 2.4 compares these approaches and surveys the attempts to combine them together. In chapter 3, we talk about our proposed model and feature sets for document-level sentiment classification. Section 3.1 discusses the classification model and its different phases. Then, section 3.2 will list the tools that we will use for developing the model and our proposed feature sets and describe each one of them. Then, we explain how we will extract the proposed features in section 3.3, section 3.4 analyzes the running time for the algorithms used in terms of big O notation and section 3.5 explains the evaluation methodology that we will follow to evaluate our proposed feature sets performance. In chapter 4, we will discuss and analyze the 3 differents sets of experiments that we have performed to evaluate the performance of our proposed feature sets against the baseline and discuss the results. Finally, in chapter 5, we will conclude the thesis and list some directions for future work. There are 4 appendices at the end of the document. Appendix A shows the feature sets used in our experiments; appendix B includes some of the lists used to extract our proposed features; appendix C explains different combinations of the variables used in the proposed algorithm and the result of each one; and appendix D shows some sample documents from the datasets we have used in our experiments, their extracted features from both the baseline and our proposed feature sets and their classification in both cases.

CHAPTER 2

LITERATURE SURVEY

Sentiment classification has been recently the interest of many researchers, due to the enormous opinionated web content, including blogs, reviews and social network applications. Rather than searching for facts, where most search engines have focused, people are now also interested in searching for other people's opinions on the web. For example, when buying a new product, a person is interested in knowing the opinions of those who have already bought it, so he is interested in having an automated tool that crawls the reviews on this product on the web and gives him a summary of the reviews. Thus, given a certain piece of text, the tasks of sentiment classification are: 1) classify it as holding subjective or objective information (known as *subjectivity analysis*); 2) if it is subjective, determine its polarity as positive or negative; and 3) determine its semantic intensity as strong or weak. Researchers may focus on one or more of these tasks, and they can also focus on one or more type of text; whether it is a phrase, sentence or a document. For a more comprehensive review on sentiment classification, see [26, 20].

In this chapter, I review some of the research that has been done in sentiment classification, and my focus is at the document-level, where the problem is to classify a whole document (whether a review, news article, blog . . . etc) as holding an overall positive or negative sentiment. For this task, an assumption is being made that the document is subjective, talking about one object only and holding the opinion of one author [20]. There are mainly two approaches for document-level sentiment classification: the ML and the SO approaches. There has been recently some work trying to build a hybrid model combining both of them.

This chapter is organized as follows: section 2.1 defines contextual valence shifters and its importance in sentiment classification. In section 2.2, I talk more in-depth about the ML approach and the different feature sets used in the literature and then give a survey about the SO approach and the recent work done in this direction. Then, in section 2.3, I will explain in brief the theoretical foundations of SVM and the dependency parser, since they were used in the literature and will be used in our experiments. Finally, in section 2.4, I will compare both the ML and SO approaches and review the different techniques for combining them.

2.1 Contextual Valence Shifters

Traditional sentiment classification research tried to extract individual terms that indicate prior positive or negative polarity and build a dictionary of polar words. However, such individual terms are not sufficient to determine the true or contextual polarity of the document. The valence of a polar term may be modified by one or more words, the so-called "*contextual valence shifters*" [28]. These shifters can be categorized into several types, some of them are:

1. **Negators:**

   The polarity of a term may be reversed by a negation term like: "*not*", "*never*", "*nobody*" . . . etc. For example:

   *He **never** succeeds his exams.*

   The word "**never**" changes the polarity of the adjective "**successful**" from positive to negative [28].

2. **Intensifiers:**

   A polar term may be modified by one or more intensifiers. An intensifier is an adjective or an adverb that increases (called amplifier) or decreases (called down-toner) the semantic intensity of the word next to it. Adverbs often modify verbs, adverbs and nouns, whereas adjectives modify nouns only [34]. For example:

   *I was **very** excited to attend the party.*
   *There is **little** truth about that.*

In the first sentence, the adverb “**very**” increases (or amplifies) the polarity of the positive adjective “**excited**”, so if its polarity was +2, it would be +3. In the second sentence, the adjective “**little**” decreases (or down-tones) the semantic intensity of the noun “**truth**”, from, for example, +2 into +1.

3. **Modals and conditional words:**

   Language can express two types of events: those that truly happened (called *realis* events) and those that cannot have happened (called *irrealis* events). An *irrealis* event can be conveyed using a modal operator (such as: *might*, *could* or *should*) or a conditional word (such as: *if* or *unless*). Polar words that occur with modals or conditional words should not be taken into consideration as carrying any SO for they do not convey any. For example:

   *This **should** be a good movie.*
   ***If** John was nice, he would have more friends.*

   In the first sentence, “**should**” neutralizes the positive polarity of the adjective “**good**” since it expresses the author’s expectation about the movie. In the second sentence, the conditional word “**if**” and the modal “**would**” also neutralize the words “**nice**” and “**more**”. As we can tell from this sentence that John is not nice at all [28], so it would be better to reverse the polarity of the lexical items appearing after conditional words instead of neutralizing them.

4. **Presuppositional items:**

   These are words that can shift the base valence of the SO-carrying words towards the other polarity, such as *barely*, as in “*barely sufficient*”. There are also some nouns and verbs that shift the polarity of the lexical items, such as *failure*, *neglect*, *lack*, *fail*, *omit* . . . etc. For example:

   *He **failed** to succeed the exam.*
   *He **lacks** wisdom.*

   In these two sentences, the verbs “**failed**” and “**lacks**” shift the polarity of the verb “**succeed**” and the noun “**wisdom**”, respectively, so that the whole expressions “failed to succeed” and “lacks wisdom” are negative instead of positive [28].

5. **Connectors:**

   Connectors that show contrast, such as *although*, *but*, *on the contrary* and *in spite of*, affect the polarity of the SO-carrying words in the sentence. For example:

   ***Although** he was brilliant, he failed the exam.*

   In this sentence, the connector *"***although***"* neutralizes the polarity of the positive adjective "**brilliant**" leaving the negative verb "**failed**" to determine the final polarity of the sentence. Without doing this adjustment, the overall polarity would be neutral [28].

In addition to the above contextual valence shifters, there are others, such as: *irony*, *multi-entity evaluation* and *genre constraints*, but these are beyond our discussion here and can be found in [28].

2.2 Approaches for Document-level Sentiment Classification

There are two assumptions when performing document-level sentiment classification: 1) The document expresses the opinion of one author; and 2) The author talks about one object only. Document-level sentiment classification techniques can be mainly divided into two approaches: ML and SO. ML algorithms have succeeded in classic text categorization, and so they were implemented for sentiment classification, but with having the target classes as "positive" and "negative". These algorithms are discussed in detail in [31, 40]. Supervised ML algorithms, such as SVM, have been used extensively in the sentiment analysis research [20, 26]. In supervised ML, a piece of text is converted into a feature vector so that the classifier would learn from a set of data labeled with its class (called *training* data) that a combination of specific features yields a specific class, and then it can test its accuracy of learning on another set of data (called *testing* data). On the other hand, in the SO approach, a sentiment lexicon is built either manually, semi-automatically or automatically with each word having its semantic intensity as a number indicating whether it is positive or negative as well as its intensity. Then, this lexicon is used to extract all sentiment words from the document and sum up their polarities to determine if the document is holding an overall positive or negative sentiment besides its intensity.

2.2.1 The Machine Learning Approach

In the ML approach, each document is represented as a feature vector with representative features for the target class as well as its correct class. Then, the feature vectors are inserted into a classifier that uses a ML algorithm for training it. A set of documents are chosen for training (called *training data*) to build a model for predicting the class of unseen or new data (called *testing data*). Different ML algorithms have been used in the sentiment classification literature, but SVM have dominated most of the research done. The theoretical foundation of SVM is explained in brief in section 2.3.1. Various feature sets have been tried specifically for sentiment classification, as discussed in [26]. Some of the significant features that are related to our work are:

1. **N-grams:**

   N-grams are common features to use in text classification. They are words that are frequently repeated in the corpus. They vary from uni-grams (one word only, such as *director*), bi-grams (two neighboring words, such as *two stars*), to tri-grams (such as *science fiction film*). N-grams can be useful when capturing the sentiment of the document, for example, the *director* of a movie could have done outstanding work, so its occurrence in a review most probably indicates a positive sentiment. Many researchers have used n-grams, especially uni-grams, since they result in a high accuracy and they added other features as improvements to their systems [27, 38, 23, 1, 2, 9]. Some researchers use uni-grams and bi-grams [9], some use bi-grams only [41], while others use uni-, bi- and tri-grams [2, 9]. Several ML algorithms have been compared, and SVM outperformed other algorithms [27], and hence, it has been used as a common algorithm for sentiment classification, except for a few papers that have used NB or ME or a combination of two or more of them [41, 42].

2. **Part-Of-Speech tag n-grams:**

   Part-Of-Speech (POS) tag n-grams have been proven to be good indicators of sentiment [16, 14, 2]. For instance, the authors in [14] experimented the effect of a combination of positive and negative nouns, verbs, adverbs and nouns and have shown that the appearance of a positive adjective followed by a noun is more frequent in positive documents than in negative ones, and that the appearance of a negative adjective followed by a noun is more frequent in negative documents.

3. **Stylistic features:**

   These include lexical and structural attributes as well as punctuation marks and function words, as explained in [1, 43, 2]. The *lexical features* include character- or word-based statistical measures of word variation. Examples of the character-based lexical features are: 1) Total number of characters; 2) Characters per sentence; 3) Characters per word; and 4) The usage frequency of individual letters. Some examples of the word-based lexical features are: 1) Total number of words; 2) Words per sentence; 3) Word length distribution; and 4) Vocabulary richness measures; such as: the number of words that occur once (*hapax legemona*) and twice (*hapax dislegemona*) as well as several statistical measures like: Yule's K, Simposon's D and Brunet's W measures [36]. The *structural features* include text organization and layout, for example, signatures, number of paragraphs, average paragraph length, total number of sentences per paragraph and others.

   These features were used along with other features in sentiment classification research [1, 2, 9, 3] and they improved the system's accuracy when added [2].

4. **Negation and modification features:**

   These are two of the important categories of the contextual valence shifters discussed in the previous section. Early work in sentiment classification did not investigate the effect of negation or modifiers [27], as the authors only used Bag-Of-Words (BOW) and higher-order n-grams. As a result, two sentences like these: "*I like this movie*" and "*I don't like this movie*" would be similar to each other since both contain the verb "*like*", although the first one holds a positive sentiment while the second holds a negative sentiment [26]. Modifiers also play a crucial role when classifying sentiment since they can increase or decrease the semantic intensity of polar terms or even shift them towards the positive or negative sentiment, as discussed in the previous section.

Therefore, later research focused on how to detect and extract negation and modifiers and add them as features [39, 32, 18, 33, 8]. For instance, the authors focusing on sentence-level sentiment classification in [39] added a binary feature for each word token to determine whether it was negated or not in addition to other features to indicate whether it was modified by an adjective or adverb intensifier or one or more contextual valence shifters that they extracted: general, negative and positive polarity shifters. The general polarity shifter reverses the polarity of the word modified by it, e.g. *little* truth. The negative polarity shifter yields an overall negative sentiment for the overall expression, e.g. *lack* of wisdom. The positive polarity shifter changes the overall sentiment of the expression to positive; for example, *abate* the damage [39]. In addition, the authors in [18] used a dependency parser to extract typed dependencies and developed special rules to determine the scope of each negation term. Furthermore, the authors in [8] used typed dependencies and a negation and quantifiers lists to extract negated, amplified and down-toned sentiment words and build a sentence-level polarity and intensity sentiment classifier.

5. **Dependency relations:**

   Some researchers have explored the effect of dependency relations in sentiment classification, since they can hold more information than neighboring words (such as: higher-order n-grams). Dependency relations are typically extracted using a dependency parser. For example, the authors in [39] used a dependency parser to extract modifiers and other dependency relations (e.g. words connected by a conjunction, like *and*). Also, the authors in [41] extracted bi-grams and all dependency relations as features besides uni-grams to improve the system's performance. However, it was shown in [23] that adding dependency relations with focus on adjectives preceded by nouns and nouns that have a dependency relation with polar terms to n-grams did not improve the performance.

Researchers have tried using different ML algorithms to compare their performance in the sentiment classification task. It was shown in [27] that SVM outperforms both NB and ME when using unigrams, bigrams as well as other features; and hence, most research has used SVM and it became the default algorithm in sentiment classification. However, the authors in [41] showed that uni-grams perform better on SVM, whereas higher-order n-grams and dependency relations perform better on NB, due to the nature of the algorithms themselves. They explained the difference between the performance of SVM and NB classifiers as follows: unigrams tend to have more relevant features and less independency than higher order n-grams. The discriminative model, SVM, can capture the complexity of relevant features, whereas a more generative model, NB, performs well on bigrams and dependency relations since the feature independence assumption holds well [41].

Another problem in ML besides choosing the right features is feature selection. Since there can be redundant or irrelevant features in the feature vector that makes it unnecessarily a huge vector, feature selection is performed to reduce the dimensionality of the feature vector so that only representative features for the target class are remaining. By selecting the most relevant features for the target concept, the classifier learns better which class label to give to each vector. In addition, it reduces the running time required for the classifier on the training data, which is crucial for real-world applications [10]. Several researchers have implemented different feature selection methods that were used for classic text classification and applied them to sentiment classification [2, 9]. Other researchers have developed new algorithms for feature selection specified for sentiment classification [37, 24, 3]. But, most research uses the Information Gain (IG) heuristic as the feature selection method due to its reported effectiveness [2, 9, 17].

2.2.2 The Semantic Orientation Approach

In the SO approach, a document's polarity is calculated as the sum of its polar terms and/or expressions. Early work in this direction calculated the phrase's polarity as the difference between the Point Mutual Information (PMI) between the phrase and the word "excellent" as representative for the positive class and the PMI between the phrase and the word "poor" as representative for the negative class, with hit counts from AltaVista search engine using the NEAR operator. The PMI between two words is calculated as follows [35]:

$$PMI(word_1, word_2) = log_2(\frac{p(word_1 \& word_2)}{p(word_1)p(word_2)}) \tag{2.1}$$

where $p(word_1 \& word_2)$ denotes the probability that both words occur together. Later, researchers tried to build sentiment lexicons for words and expressions. Some researchers built a sentiment lexicon manually, such as:

1. The General Inquirer:
   The General Inquirer[1] (GI) is a content-analysis tool that is used for many purposes including sentiment classification. It contains several categories, such as *Positiv* and *Negativ* that are mainly used for sentiment classification, as well as other categories for words of pleasure and pain, overstatement and understatement, and others. Each category has a list of words and word senses, so it works with Word Sense Disambiguation to differentiate between different word senses. Also, it combines different dictionaries, such as "Harvard IV-4" and "Lasswell" dictionaries. It has been used in many sentiment classification research to build other lexicons and in the ML approach as well when using sentiment words as features such as [39, 22].

2. SO-CALculator (SO-CAL):
   SO-CAL is another manually-built sentiment lexicon. The authors in [7, 34] developed a sentiment lexicon manually, incorporating SO of individual words and contextual valence shifters, but did not take into account word sense disambiguation. Their dictionary includes a separate list for sentiment bearing nouns, adjectives, adverbs and verbs with a numerical score from -5 to +5 that indicates the word's polarity and strength. In addition, they developed a list of amplifiers (words that increase the semantic polarity of words after them, such as *very* good) and down-toners (words that decrease the semantic polarity of words after them, such as *less* successful). These modifiers include adjectives, adverbs and multi-word expressions. They also implemented a new method for negating words, called *shift negation*, instead of the traditional *switch negation* method that only reverses the sign (+ or -) of the word preceded by a negated term. This method shifts the polarity of the negated words towards the opposite polarity by a fixed number (4), so for example the word "excellent" whose score is +5 when negated, its polarity changes to +1 (5 - 1). They achieved comparable performance to other sentiment lexicons [34].

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

Other researchers built sentiment lexicons using automatic sources only or a combination of both manual and automatic sources, such as:

1. The Subjectivity dictionary:
   The authors in [39] built the subjectivity dictionary from both manual and automatic resources, but they used it in a ML approach by developing specific features for sentence-level sentiment classification. They started with a list of subjective clues and then expanded them using a dictionary and thesaurus as well as positive and negative words from the General Inquirer. They obtained a list of about 8,000 words labeled with their corresponding prior polarity (positive, negative, neutral or both), type (strong or weak subjective) and part of speech [39].

2. SentiWordNet:
   SentiWordNet was built on the English lexical database, WordNet [21], where each group of words (nouns, adverbs, adjectives and verbs) are grouped together according to their synsets and the concepts they hold. The annotation of words in SentiWordNet was done automatically through a 2-step process: a weak-supervision semi-supervised learning step, and a random-walk step. In the first step, the authors in [5] started with a small seed list of positive and negative words and searched for words with similar polarity using binary relations of WordNet. Then, in the second step, an iteration is run on the words until they converge to decide on their final polarity. Each word in SentiWordNet is associated with its Part-Of-Speech and 3 polarity types (objective, positive and negative) and each type having a score describing its intensity (from 0 to 1) [13, 5]. This sentiment lexicon has also been used in the sentiment classification research, such as [9] by incorporating them as features, as we will see in section 2.4.

The ML approach has dominated over the SO approach in the literature, since there are many features that can determine the polarity of documents. But, the SO approach has better generality across domains, as it is not domain-specific. Similar to the concern of the ML approach, which is choosing the right features, the concern in the SO approach is to build a comprehensive lexicon with correct prior polarity to the words and methods to handle the contextual polarity of them. This is what the authors in [34] have concentrated on when building their sentiment lexicon and they argued that building a sentiment lexicon manually is more efficient than building it automatically to produce more accurate lexicons.

2.3 Theoretical Foundations

In this section, I briefly describe the theoretical foundations of both SVM and the dependency parser, since they were used in the literature for sentiment classification and we will use them in our experiments. Section 2.3.1 explains what the theory behind SVM is and section 2.3.2 describes what a dependency parser is and how The Stanford parser works.

2.3.1 Theoretical Foundation of SVM:

There are different books and articles that explain the SVM method. Here, we describe its theory in brief since it was used extensively in the sentiment classification literature. In a classification problem, you are given a set of labeled and unlabeled data with their target class and you want to build a model to learn to classify the unlabeled data as well as any further unseen data. If you have two target classes only, this is called a binary classification problem, otherwise it is a multi-class problem. So, in order to build this model, you need first to decide on the features that you believe are relevant when determining the target class. Then, you have to extract these features from each document, so that each document will be represented as a feature vector with each feature having its corresponding value. A classifier can be represented as a function: $f(x) : R^d \to R$ where a binary classifier assigns a point to the positive class if its function value is greater than or equal to zero, and assigns it to the negative class if its function value is less than zero. SVM have been widely used in classification problems, such as text classification problems, and recently they have been also used in sentiment classification. They construct a special rule, which is called a linear classifier. A classifier is said to be linear if its function can be written as: $f(x;w,b) =< w,x > +b$ where w and b are the function parameters and $<$ and $>$ signs are the inner or dot product of the two vectors. The data points used for training the classifier can be represented in the form:

$$D = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^l, y_i \in \{-1,1\}\}_{i=1}^{l} \tag{2.2}$$

where **x** is the feature vector of the data point with length *l* and *y* is the class of the point and is either 1 (for a positive class) or -1 (for a negative class). These data points can be described in a *l-1*-dimensional hyperplane, as illustrated in figure 2.1. The goal of the classifier then is to find a hyperplane that separates the data points with the positive class from those with the negative class with the maximum margin possible from each set of points to the hyperplane, as illustrated in figure 2.1, where the data points on the margins are called “support vectors”.

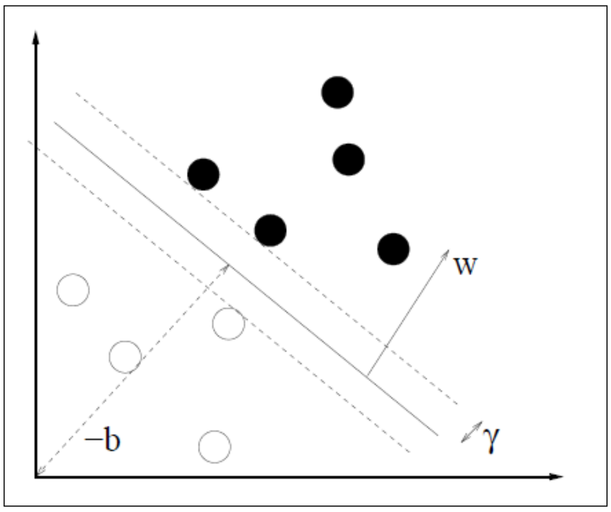

Figure 2.1: How SVM works [15]

An important property of the training data in SVM is to be *linearly separable*, where:

$$y_i f(x_i) > 0, \forall i = 1, ..., l \tag{2.3}$$

which means that the two hyperplanes of both margins can be selected in a way that there are no data points between them [15].

2.3.2 Theoretical Foundation of the Dependency Parser:

A parser, or syntactic analyzer, is a program that extracts the grammatical relationships between different phrases of the sentence. A typed dependency parser is a parser that outputs the grammatical relations between single words of the sentence, for example, what is the subject and object of a verb. A typed dependency parser usually uses a phrase structure parser to output typed dependencies [11]. Parsers were developed probabilistically by training them on manually-parsed sentences (the Penn Treebank) to try to produce the most likely grammatical relations between the words of new (unseen) sentences. Different dependency parsers use different methods to extract typed dependencies. Here, we explain in brief how Stanford typed dependencies are extracted. To extract Stanford typed dependencies from phrase structure trees, there are mainly 2 phases:

1. Dependency Extraction:
   In this phase, each sentence is first parsed with a phrase structure parser, usually a Penn Treebank parser. Then, the semantic head of each constituent in each sentence is extracted using some head rules, and then the words that depend on each head are identified.

2. Dependency Typing:
   This phase is concerned with identifying the dependencies between each head word and its different dependent words by defining some patterns and then matching each one against every tree node and taking the most likely dependency.

Each dependency is represented as follows: dependency_name(governor, dependent), where the governor is the parent of the relation and dependent is its child. The typed dependencies are represented as a directed acyclic graph with one root only. These are called "basic dependencies", as illustrated for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas." in figure 2.2. However, there are some situations where prepositions and conjunctions could be treated as dependencies. So, Stanford has developed other types of dependencies, such as "collapsed dependencies", as illustrated on the same sentence in figure 2.3 [11].

2.4 Comparing the Approaches and Combining them into a Hybrid Approach

There are some significant differences between the ML and SO approaches. First of all, the ML approach performs better than the SO approach on a single domain; the highest accuracy achieved on the Polarity dataset version 2.0 [25] using a classifier was 91.7%

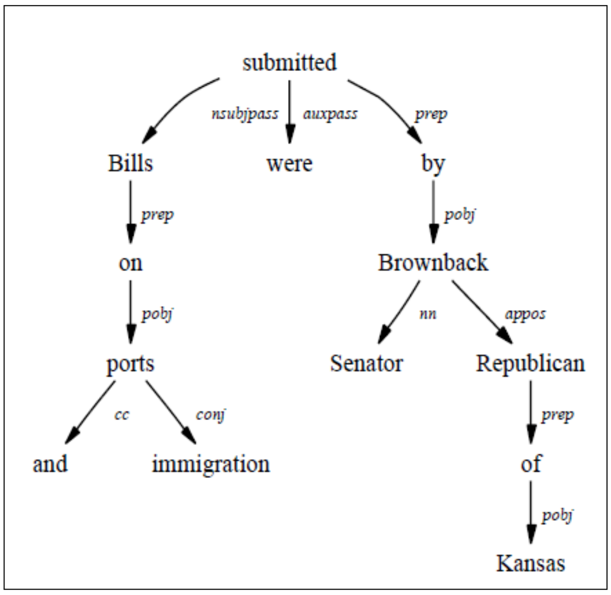

Figure 2.2: An example of basic typed dependencies [11]

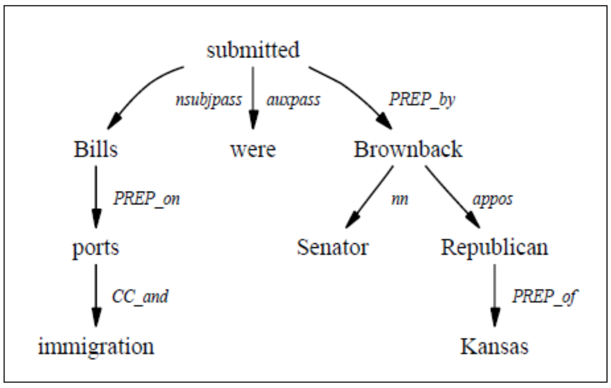

Figure 2.3: An example of collapsed typed dependencies [11]

[2] versus an accuracy of 76.37% using SO-CAL [34]. Secondly, the classifier can learn domain-specific polarity; e.g. "***long** battery life*" (+ve) vs "***long** time to focus*" (-ve), unlike the lexicon where a prior polarity is known for each word. However, to build a high accuracy classifier, we need to have a huge corpus labeled with its class (positive or negative) whereas a dictionary does not need one. This can be a labor-intensive task; to collect data from different genres (reviews, news articles, blogs . . . etc) and domains (movies, products, politics . . . etc) and label them manually. In addition, the authors in [7, 34] argued that a ML approach does not take linguistic context into account, such as negation and intensification, since there can be three features "*good*", "*very_good*" and "*not_good*" but the classifier does not know they are related to each other. In this thesis, we are trying to solve this major drawback in the ML approach, as will be explained in chapter 3.

Since both the ML and SO approaches have some advantages and disadvantages, some researchers have attempted to combine them together to benefit from the advantages of each approach [4, 29, 30, 17, 9]. For example, the authors in [9] built a classifier with unigrams and bigrams with a threshold of 5 and stylistic features and they developed a feature-calculation strategy to extract sentiment features (verbs, adjectives and adverbs) from SentiWordNet 3.0. Similarly, the authors in [4] tried to develop a high-accuracy domain-independent system at the sentence-level by building an ensemble model of two classifiers; one with unigrams, bigrams and trigrams as features and trained on in-domain data, and the other one with sentiment words from both WordNet and the General Inquirer as features. The authors in [29] then improved this system to be self-supervised so that it does not need labeled data. They developed a two-phase model; the first one is a sentiment lexicon and a negation list to label the data and they took the high-confidence labeled documents as training data to a classifier with sentiment words being the feature set. Another technique to solve the problem of manually labeling the data was shown in [17], where the authors trained an initial classifier with sentiment features from the Subjectivity lexicon. Then, from the high-confidence labeled data, they extracted the most indicative features for each class (using the IG heuristic) as self-learned features to train another classifier instead of training it with self-labeled data. However, all these models did not take into account contextual valence shifters described in section 2.1. Therefore, in this thesis, we are trying to fill this gap in the current literature.

CHAPTER 3

PROPOSED APPROACH: RECOGNIZING CONTEXTUAL VALENCE SHIFTERS IN DOCUMENT-LEVEL SENTIMENT CLASSIFICATION

In this research, we are interested in using a hybrid approach by incorporating contextual valence shifters as features into an SVM classifier in addition to other important features whose performance has been tested and proven to be significant in sentiment classification. By adding contextual valence shifters, such as negation, intensification and connectors, from a new perspective other than the ones described in chapter 2 into a ML approach, the classifier will learn how to take linguistic context into consideration. In this way, by developing a self-supervised model, as described in the literature, it would be a practical technique to use instead of using a sentiment lexicon for being an unsupervised technique.

This chapter is organized as follows: in section 3.1, I explain the classification model, its components and the different feature sets that will be used; section 3.2 will describe the different tools used for building the model; section 3.3 explains the algorithms for extracting the proposed feature sets; section 3.4 analyzes the running time for these algorithms and finally section 3.5 will provide the evaluation methodology we will follow in order to evaluate our proposed feature sets.

3.1 Document-level Sentiment Classification Phases

After conducting research on the current and past work done in document- and sentence-level sentiment classification, we decided to develop a document-level sentiment classification model that recognizes contextual polarity. The document-level sentiment classification model consists of several components as illustrated in figure 3.1.

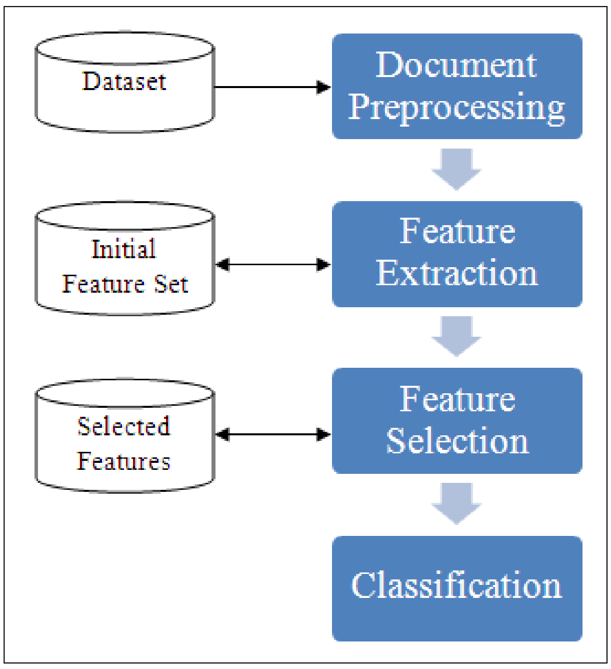

Figure 3.1: The document-level sentiment classification model

The model goes through the following phases:

1. **Document preprocessing:**
   The datasets need to be preprocessed to be ready for feature extraction. Since the multi-domain dataset is in an XML format, we had to extract only the review body and its overall sentiment using an XML parser. In addition, the two datasets have to be tokenized, tagged into their corresponding POS and parsed with their typed dependencies.

2. **Feature extraction:**
   After preprocessing the documents, the feature extractor is used to generate the following feature sets that are listed in appendix A:

(a) ***Content-free features (F1):***

These include stylistic features: 87 lexical features, 8 punctuation marks and 150 function words. These features are described in detail in [43]. We will denote to these features as F1.

(b) ***Content-specific features (F2):***

These include n-grams (unigrams, bigrams and/or trigrams) with a pre-defined frequency threshold. Their number varies from one dataset to another. We will denote to these features as F2.

(c) ***Sentiment features (F3):***

We will use the Subjectivity dictionary provided by [39] to extract sentiment features. This feature set includes the frequencies of each sentiment word (found in the lexicon) in each document. Some preprocessing was performed on the dictionary to remove all words whose prior polarity was "neutral" or "both". The total number of sentiment features varies from one dataset to another. We will denote to these features as F3. We will also refine F3 so that sentiment words modified by contextual valence shifters as well as sentiment words after conditional or contrast words or modal verbs are not counted. Moreover, we will lemmatize the words before looking them up in the lexicon to get more accurate results. We will denote to these features as 'refined F3'.

(d) ***Contextual Valence Shifters (F4):***

All the previous features were used in [9], so in this study, we are adding contextual valence shifters as a new feature set to the ML approach. We believe that this feature set will improve the baseline's performance. In the baseline [9], the authors add all sentiment words disregarding any modification to sentiment words, so for example, if a sentence has the words *not good*, the frequency of the sentiment word *good* will be incremented by 1, though this is not a valid increment. Therefore, we will only count the sentiment features that were not modified by contextual valence shifters. In addition, we will use the list of intensifiers and contextual valence shifters provided by [39] as well as a list of conditional and contrast words, listed in Appendix B. We will denote to these features as F4.

3. **Feature selection:**

   Having generated the features, we should select the most significant features that are good predictors of the class label. Since the IG heuristic was used extensively in the sentiment classification literature, as explained in section 2.2, we will use it as the feature selection method for our feature sets.

4. **Classification:**

   In this step, the feature vectors are generated and inserted into a sparse ARFF file that is the input file for the classifier. We choose the Weka suite software to run the classifier.

3.2 Tools

We will use a set of tools and English lists in order to implement our proposed feature sets and test it against the baseline, as follows:

1. **The Subjectivity dictionary, intensifiers and contextual valence shifters:**

   These lists are provided by [39]. The Subjectivity dictionary contains 8,221 entries (sentiment words). Each sentiment word is described using some tags, as follows:

   type=<type> len=1 word1=<word> pos1=<POS> stemmed1=<stemmed> priorpolarity=<polarity>

   where *type* is either *strongsubj* or *weaksubj* representing the semantic intensity of the word in most context as either strong or weak, *len1* represents the length of the sentiment expression (all entries have a length of 1), *word1* is the word token, *pos1* indicates the word's part of speech (maybe *anypos* – any part of speech), *stemmed1* indicates whether the sentiment word should match all its unstemmed variants with the corresponding POS, and *priorpolarity* indicates the prior polarity of the sentiment word that is out of any context.

   The intensifiers list consists of 121 adjectives and 123 adverbs that are used as intensifiers; e.g. *absolute*, *most*, and *highly*. Adjectives may modify nouns only, e.g. *total* *failure*, whereas adverbs may modify verbs, adverbs and adjectives, e.g. *it greatly affected his performance*.

The contextual valence shifters list contains: 15 *negation* terms (single and multi-word expressions), 5 *positive polarity shifters*, and 72 *negative polarity shifters*. The negation terms typically reverse the polarity of the words after them, e.g. he *never* succeeds. The positive polarity shifter changes the overall sentiment of the expression to positive; for example, *abate* the damage. The negative polarity shifter yields an overall negative sentiment for the overall expression, e.g. *lack* of wisdom. It also contains 5 multi-word expressions labeled as *nonshifter*. These expressions are used along with the negation terms to check if the negation term should be counted as such. For instance, the expression “not only” is a nonshifter so that “not” is not counted as a negation term [39].

2. **The Stanford tagger, tokenizer and parser:**

   To perform the POS tagging, tokenizing and parsing, we will use the Stanford tagger version 3.0.1[1] and parser version 1.6.6[2]. The POS tagger is a piece of software written in Java that takes raw text as an input and outputs each work token with its corresponding part of speech. It uses the Penn Treebank tag set. We use the tagger when extracting sentiment features (F3) since each sentiment word has its corresponding POS. The dependency parser is a program that extracts the grammatical relations (known as *typed dependencies*) between different words of each sentence. It also uses the Penn Treebank tag set. We use the dependency parser when extracting contextual valence shifters and intensifiers (F4).

3. **Weka Suite Software:**

   We will use the Weka Suite Software version 3.6.4[3] as the environment for classification. Weka is a piece of software written in Java that provides a set of ML algorithms, such as SVM, NB and others as well as feature selection methods such as IG. It also provides a number of test options, such as cross validation and percentage split. It can

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

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

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

be run directly by inserting the dataset into the program or from the command line (when the dataset size is large).

4. **MontyLingua Lemmatizer:**

   Instead of stemming word tokens to search for them in the Subjectivity dictionary, we prefer to lemmatize them, since the lemmatizer takes the POS into consideration unlike the stemmer. We use the MontyLingua lemmatizer version 2.1[4] written in Python.

   A lemmatizer is a natural language program that takes a word in its inflected form and POS and then outputs its base form. For instance, consider the word "meeting". If it is a noun, as in "our meeting yesterday was successful", then its lemmatized form will still be "meeting". But, if it is a verb, as in "we are meeting tomorrow", then the lemmatizer will output the word "meet".

3.3 Extracting Refined Sentiment Features and Contextual Valence Shifters

In this section, we describe in detail the algorithms we have developed to extract the refined sentiment features (refined F3) and contextual valence shifters (F4). F4 contains a set of 16 features that counts the number of intensifiers, negation, and polarity shifters grouped together according to the polarity and type of sentiment words modified by them, in addition to the frequency of sentiment words grouped together with their final (modified) polarity and type. These features are described as follows:

1. *Intensifier_pos_strongsubj:* frequency of words with positive prior polarity and strong subjectivity type preceded by an intensifier.

2. *Intensifier_pos_weaksubj:* frequency of words with positive prior polarity and weak subjectivity type preceded by an intensifier.

3. *Intensifier_neg_strongsubj*: frequency of words with negative prior polarity and strong subjectivity type preceded by an intensifier.

[4]http://web.media.mit.edu/~hugo/montylingua/

4. *Intensifier_neg_weaksubj*: frequency of words with negative prior polarity and weak subjectivity type preceded by an intensifier.
5. *Negation_pos_strongsubj*: frequency of words with positive polarity and strong subjectivity type preceded by a negation term.
6. *Negation_pos_weaksubj*: frequency of words with positive polarity and weak subjectivity type preceded by a negation term.
7. *Negation_neg_strongsubj*: frequency of words with negative polarity and strong subjectivity type preceded by a negation term.
8. *Negation_neg_weaksubj*: frequency of words with negative polarity and weak subjectivity type preceded by a negation term.
9. *Shiftneg_strongsubj*: frequency of words with positive polarity and strong subjectivity type preceded by a negative polarity shifter.
10. *Shiftneg_weaksubj*: frequency of words with positive polarity and weak subjectivity type preceded by a negative polarity shifter.
11. *Shiftpos_strongsubj*: frequency of words with negative polarity and strong subjectivity type preceded by a positive polarity shifter.
12. *Shiftpos_weaksubj*: frequency of words with negative polarity and weak subjectivity type preceded by a positive polarity shifter.
13. *Pos_strongsubj*: frequency of words with positive polarity and strong subjectivity type after modifying their polarity and type if they were preceded by any of the intensifiers or contextual valence shifters.
14. *Pos_weaksubj*: frequency of words with positive polarity and weak subjectivity type after modifying their polarity and type if they were preceded by any of the intensifiers or contextual valence shifters.
15. *Neg_strongsubj*: frequency of words with negative polarity and strong subjectivity type after modifying their polarity and type if they were preceded by any of the intensifiers or contextual valence shifters.

16. *Neg_weaksubj*: frequency of words with negative polarity and weak subjectivity type after modifying their polarity and type if they were preceded by any of the intensifiers or contextual valence shifters.

In order to extract these features, we have used the Stanford parser to get typed dependencies that show the grammatical relationships between words in each sentence and developed some algorithms. Algorithms 1 and 2 show the general steps taken to extract refined F3 and F4 in each document. Appendix C shows how F4 is computed when there are different combinations of the contextual valence shifters and intensifiers that modify the sentiment word using these algorithms. To illustrate how these algorithms work correctly with some combinations, consider the following sentences:

1. *"This is a problem":*

   According to the Subjectivity lexicon, the prior polarity of the noun *problem* is negative and its type is weak (weaksubj). Since it is not preceded by a modal or conditional or contrast word or by an intensifier or a polarity shifter, its final polarity will remain the same (it will not go through any modifications). Therefore, we only want to increment both the sentiment feature *problem_noun* and the frequency of its final polarity *neg_weaksubj* by 1. Thus, in algorithm 1 the sentiment word *problem* will only go through lines 4 and 13 - 15. In algorithm 2, it will only go through line 15, where *word_count* will be 1, since the variables *negated*, *intensified* and *shifted* are all false. After that, the sentiment feature corresponding to the noun *problem_noun* in refined F3 will be incremented by 1 as well as the feature *neg_weaksubj* in F4 (lines 14 and 15 in algorithm 1). Hence, the values of the 16 features of F4 for this sentence before and after modification are shown in table 3.1.

2. *"This is not a problem":*

   In this sentence, the noun *problem* is negated, so we want to increment the feature *negation_neg_weaksubj* by 1 and decrement the sentiment feature *problem_noun* by 1. The algorithm will find the "neg" dependency between the negation term *not* and the word *problem* and no other modifiers. Therefore, the variable *negated* will be set to *TRUE* (lines 7 - 19). Then, it will go to algorithm 2, where it will go through lines

Feature	Before Modification	After Modification
intensifier_pos_strongsubj	0	0
intensifier_pos_weaksubj	0	0
intensifier_neg_strongsubj	0	0
intensifier_neg_weaksubj	0	0
negation_pos_strongsubj	0	0
negation_pos_weaksubj	0	0
negation_neg_strongsubj	0	0
negation_neg_weaksubj	0	0
shiftneg_strongsubj	0	0
shiftneg_weaksubj	0	0
shiftpos_strongsubj	0	0
shiftpos_weaksubj	0	0
pos_strongsubj	0	0
pos_weaksubj	0	0
neg_strongsubj	0	0
neg_weaksubj	0	1

Table 3.1: F4 values for sentence #: 1

9, 11 and 12, where *word_count* will be -1 (since the word is negated), the negation feature with the word's polarity, *negation_neg_weaksubj*, will be incremented by 1 and the final polarity of *problem* will be negated (from *neg_weaksubj* to *pos_weaksubj*) according to algorithm 4. Then, it will go back to algorithm 1 and in line 14 the sentiment feature of the word, *problem_noun* will be decremented by 1 and the feature *pos_weaksubj* will be incremented by 1 in line 15. Hence, the values of the 16 features of F4 for this sentence before and after modification are shown in table 3.2.

Feature	Before Modification	After Modification
intensifier_pos_strongsubj	0	0
intensifier_pos_weaksubj	0	0
intensifier_neg_strongsubj	0	0
intensifier_neg_weaksubj	0	0
negation_pos_strongsubj	0	0
negation_pos_weaksubj	0	0
negation_neg_strongsubj	0	0
negation_neg_weaksubj	0	1
shiftneg_strongsubj	0	0
shiftneg_weaksubj	0	0
shiftpos_strongsubj	0	0
shiftpos_weaksubj	0	0
pos_strongsubj	0	0
pos_weaksubj	0	1
neg_strongsubj	0	0
neg_weaksubj	0	0

Table 3.2: F4 values for sentence #: 2

3. *"This is a huge problem":*

   In this sentence, the noun *problem* is intensified by the adjective *huge*. Thus, we want to increment the feature *intensifier_neg_weaksubj* as well as the sentiment feature *problem_noun* by 1. In algorithm 1, it will find an "amod" dependency between *problem*

Algorithm 1 Extract Refined F3 and F4

```
1:  $F3, F4 \Leftarrow 0$
2:  for each sentence in the document do
3:    for each sentiment word w found in the sentence do
4:      $final\_polarity \Leftarrow prior\_polarity$
5:      if w is not preceded by a modal, conditional or contrast word then
6:        $final\_polarity \Leftarrow check\_polarity\_shifters$(w,final_polarity)
7:        if w is preceded by a negation term then
8:          $negated \Leftarrow TRUE$
9:        end if
10:       if w is preceded by an intensifier then
11:         $intensified \Leftarrow TRUE$
12:       end if
13:       $final\_polarity \Leftarrow modify\_polarity$(negated,intensified,shifted)
14:       $word\_POS \Leftarrow word\_POS + word\_count$
15:       $polarity\_type \Leftarrow polarity\_type + 1$
16:     end if{end of search for negation and intensification}
17:   end for{end for each sentiment word}
18: end for{end for each sentence}
```

and *huge* in line 10 which calls algorithm 6. So, it will go through line 11 where the variable *intensified* will be set to *TRUE*. Then, it will go to algorithm 2, where the variable *word_count* will be set to 1 (line 15) and the feature *intensifier_neg_weaksubj* will be incremented by 1 (since *intensifier_count* will be 1 as there is only 1 intensifier, according to algorithm 6) (line 18). And then, the final polarity will be intensified, according to algorithm 5 to *neg_strongsubj* (line 20). Then, back to algorithm 1, both features *problem_noun* and *neg_strongsubj* will be incremented by 1 (lines 14 and 15). Hence, the values of the 16 features of F4 for this sentence are shown in table 3.3.

4. *"This is not a huge problem":*

   Here, the sentiment word *problem* is intensified with the adjective *huge* and *huge* is negated by the negation term *not*. Hence, we want to first intensify the polarity of *problem* and then negate the expression *huge problem* by incrementing the feature *negation_neg_strongsubj* and the sentiment feature *problem_noun* by 1, since there is a problem that is not huge. So, in algorithm 1, the variable *negated* will be set to *TRUE* (lines 7 - 9). Then, *intensified* will be also set to *TRUE* since there is an "amod" dependency between *huge* and *problem* (lines 10 - 12). It will then go to algorithm 2. Since both *negated* and *intensified* are *TRUE*, it will go through lines 6, 7, 11

Algorithm 2 Modify polarity($negated$, $intensified$, $shifted$)

```
1:  if $negated = TRUE$ then
2:      if $shifted = TRUE$ then
3:          $word\_count \Leftarrow 1$
4:      else {$shifted$ = false}
5:          if $intensified = TRUE$ then
6:              $final\_polarity \Leftarrow intensify\_polarity(final\_polarity)$
7:              $word\_count \Leftarrow 1$
8:          else
9:              $word\_count \Leftarrow -1$
10:         end if
11:         $negation\_final\_polarity \Leftarrow negation\_final\_polarity + 1$
12:         $final\_polarity \Leftarrow negate\_polarity(final\_polarity)$
13:     end if{end if $shifted$ = TRUE}
14: else {the word is not negated}
15:     $word\_count \Leftarrow 1$
16:     if $intensified = TRUE$ then
17:         if $shifted! = TRUE$ then
18:             $intensifier\_final\_polarity \Leftarrow intensifier\_final\_polarity + intensifier\_count$
19:         end if
20:         $final\_polarity \Leftarrow intensify\_polarity(final\_polarity)$
21:     end if
22:     if $shifted = TRUE$ then
23:         $word\_count \Leftarrow -1$
24:         $shift\_final\_polarity \Leftarrow shift\_final\_polarity + 1$
25:     end if
26: end if{end if the word is negated} return $final\_polarity$
```

Algorithm 3 Check polarity shifters(w, $final\_polarity$)

```
if $w$ is positive, not negated and preceded by a shiftneg then
    $shifted \Leftarrow TRUE$
    $final\_polarity \Leftarrow neg\_weaksubj$
else if $w$ is negative, not negated and preceded by a shiftpos then
    $shifted \Leftarrow TRUE$
    $final\_polarity \Leftarrow pos\_weaksubj$
end if
return $final\_polarity$
```

Feature	Before Modification	After Modification
intensifier_pos_strongsubj	0	0
intensifier_pos_weaksubj	0	0
intensifier_neg_strongsubj	0	0
intensifier_neg_weaksubj	0	1
negation_pos_strongsubj	0	0
negation_pos_weaksubj	0	0
negation_neg_strongsubj	0	0
negation_neg_weaksubj	0	1
shiftneg_strongsubj	0	0
shiftneg_weaksubj	0	0
shiftpos_strongsubj	0	0
shiftpos_weaksubj	0	0
pos_strongsubj	0	0
pos_weaksubj	0	0
neg_strongsubj	0	1
neg_weaksubj	0	0

Table 3.3: F4 values for sentence #: 3

and 12. Thus, the final polarity will be intensified, according to algorithm 5 to be *neg_strongsubj* and *word_count* will be set to 1. Then, the negation feature of the final polarity will be incremented by 1 and then the final polarity will be negated according to algorithm 4 to *neg_weaksubj*. It will then go back to algorithm 2, where the sentiment feature of the word *problem_noun* will be incremented by 1 (line 14) and the frequency of the final polarity (*neg_weaksubj*) will be also incremented by 1 (line 15). Hence, the values of the 16 features of F4 for this sentence are shown in table 3.4.

Feature	Before Modification	After Modification
intensifier_pos_strongsubj	0	0
intensifier_pos_weaksubj	0	0
intensifier_neg_strongsubj	0	0
intensifier_neg_weaksubj	0	0
negation_pos_strongsubj	0	0
negation_pos_weaksubj	0	0
negation_neg_strongsubj	0	1
negation_neg_weaksubj	0	0
shiftneg_strongsubj	0	0
shiftneg_weaksubj	0	0
shiftpos_strongsubj	0	0
shiftpos_weaksubj	0	0
pos_strongsubj	0	0
pos_weaksubj	0	0
neg_strongsubj	0	0
neg_weaksubj	0	1

Table 3.4: F4 values for sentence #: 4

5. *"He lacks wisdom":*

In this sentence, the noun *wisdom*, whose prior polarity is *positive* and type is *strong-subj* is preceded by a negative polarity shifter *lacks*. Thus, the polarity of *wisdom* (*pos_strongsubj*) should be shifted toward the negative polarity (*neg_weaksubj*). Algorithm 1 will go through line 6 where it will go to algorithm 3 and change the final polarity to *neg_weaksubj* since the noun *wisdom* is positive, not preceded by a negation term and preceded by a negative polarity shifter *lacks* and the variable *shifted* will be set to *TRUE*. Then, it will go to algorithm 2, where *word_count* will be first set to 1 (line 15) and then changed to -1 (line 23) and the feature *shiftneg_strongsubj* will be incremented by 1 (line 24). Then, back to algorithm 1, the sentiment feature *wisdom_noun* will be decremented by 1 (since *word_count* is -1) and the feature *neg_weaksubj* will be incremented by 1 (lines 14 and 15). Hence, the values of the initial 16 features of F4 for this sentence are as follows: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. After modification, their values are as follows: 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1.

Feature	Before Modification	After Modification
intensifier_pos_strongsubj	0	0
intensifier_pos_weaksubj	0	0
intensifier_neg_strongsubj	0	0
intensifier_neg_weaksubj	0	0
negation_pos_strongsubj	0	0
negation_pos_weaksubj	0	0
negation_neg_strongsubj	0	0
negation_neg_weaksubj	0	0
shiftneg_strongsubj	0	1
shiftneg_weaksubj	0	0
shiftpos_strongsubj	0	0
shiftpos_weaksubj	0	0
pos_strongsubj	0	0
pos_weaksubj	0	0
neg_strongsubj	0	0
neg_weaksubj	0	1

Table 3.5: F4 values for sentence #: 5

Algorithm 4 Negate Polarity(*pol_type*)

```
if $pol\_type = pos\_strongsubj$ then
    $pol\_type \Leftarrow pos\_weaksubj$
else if $pol\_type = pos\_weaksubj$ then
    $pol\_type \Leftarrow neg\_weaksubj$
else if $pol\_type = neg\_strongsubj$ then
    $pol\_type \Leftarrow neg\_weaksubj$
else if $pol\_type = neg\_weaksubj$ then
    $pol\_type \Leftarrow pos\_weaksubj$
end if
return $pol\_type$
```

Algorithm 5 Intensify Polarity(*pol_type*)

```
if $pol\_type = pos\_weaksubj$ then
    $pol\_type \Leftarrow pos\_strongsubj$
else if $pol\_type = neg\_weaksubj$ then
    $pol\_type \Leftarrow neg\_strongsubj$
end if
return $pol\_type$
```

To extract adjectival and adverbial intensifiers that modify sentiment words, algorithm 6 was implemented. This algorithm first searches for dependencies "amod" and "advmod" in the dependency tree with the sentiment word being the governor (or parent). The first dependency, "amod", represents an adjectival modifier, where an adjective modifies a noun. The second dependency, "advmod", represents an adverbial modifier, where an adverb modifies a verb, adverb, or adjective [12]. If the sentiment word has any of these two dependencies and the dependent (or child) is in the intensifiers list, then we increment the intensifier count of the word by 1. Most researchers have implemented this method to extract intensifiers. However, we also found that intensifiers are not necessarily followed by the sentiment word. For example, in the sentences, *This problem is huge* and *This problem has become huge*, we can see that the adjective *huge* modifies the noun *problem* but it occurred after it. The dependency tree for these two sentences are as follows:

1. *Sentence 1* - "This problem is huge":

    - det(problem, this)
    - nsubj(huge, problem)
    - cop(huge, is)

2. *Sentence 2* - “This problem has become huge”:

   - det(problem, this)
   - nsubj(become, problem)
   - aux(become, has)
   - acomp(become, huge)

Algorithm 6 Extract Intensifiers

```
for each child/dependent $c$ of the sentiment word $s$ in the dependency tree do
    if the dependency is “amod” and $c$ is an adjectival intensifier then
        $intensifier\_count \Leftarrow intensifier\_count + 1 + check\_other\_intensifiers(\mathrm{c})$
    else if the dependency is “advmod” and $c$ is an adverbial intensifier then
        $intensifier\_count \Leftarrow intensifier\_count + 1$
    end if
end for
for each governor/parent $g$ of the sentiment word $s$ in the dependency tree do
    if the dependency is “nsubj” then
        if $g$ is an adjectival intensifier then
            $intensifier\_count \Leftarrow intensifier\_count + 1 + check\_other\_intensifiers(\mathrm{g})$
        else
            for each child $c$ of $g$ in the dependency tree do
                if the dependency is “acomp” and $c$ is an adjectival intensifier then
                    $intensifier\_count \Leftarrow intensifier\_count + 1 + check\_other\_intensifiers(\mathrm{c})$
                end if
            end for
        end if
    end if
end for
```

Therefore, in addition to the traditional search method for intensifiers, the algorithm also searches for intensifiers that do not necessarily appear before the word that they modify. To do so, we check the governors (or parents) of the sentiment word in the dependency tree. If a dependency is “nsubj” (which represents a nominal subject, where the dependent is the subject and the governor is the verb or the complement of a copular verb (adjective or noun)), we check if the governor is an adjectival intensifier. If so, like the first sentence, we increment the intensifier count for this word. Otherwise, we check the dependents of the governor of the “nsubj” relationship. If a dependency is “acomp” (which represents the adjectival complement of the verb; that is, the adjective which acts as the object of the verb [12]) and the dependent is an adjectival intensifier, like the second sentence, then we increment the intensifier count. In addition, if the intensifier found is an adjective in any of the above cases, we look for other intensifiers that could be modifying the same sentiment word, as in algorithm 7. For example, in the sentence, *This problem is very huge and puzzling*, we see that *very* intensifies *huge* and both *huge* and *puzzling* intensify *problem*. Thus, in algorithm 7, we check the dependents of the adjectival intensifier to see if there is an "advmod" or "conj_and" dependencies with an adverbial or adjectival intensifier, respectively (the dependency "conj_and" represents a conjunction between two words with *and* or its synonyms). If so, we increment the intensifier count as well. Also, we used algorithm 5 to intensify the polarity of sentiment words preceded by an intensifier. As illustrated in the algorithm, we only change the type of sentiment words whose type is weak. For example, *good* has a *positive* polarity and *weaksubj* type. When intensified, e.g. *very good*, its final polarity and type are *positive* and *strongsubj*, respectively. Often, sentiment words with *strongsubj* type are not intensified, so we do not often say *very excellent*. Hence, we only intensify the sentiment words with *weaksubj* type.

To extract negation terms and whether or not they negate the sentiment words, we followed the method proposed by [8], in which the authors search for the negation term that appear in a "neg" dependency or in a negation term list, then they get the First Common Ancestor (FCA) between the negation term and the word immediately after it. They then assume that all descendant leaf nodes are to be negated. In our implementation though, we keep the descendant leaf nodes that appear after the negation term and within a window of size 6 to be affected by the negation term, since a negation term does not often affect the words before it or the words that follow it with a large distance. Also, we omit the negation terms that appear in expressions that do not represent negation, e.g. *not only* and *not just*. To negate the polarity of sentiment words, we used algorithm 4. As proposed by [34], where the authors shift the polarity of the negated sentiment words towards the opposite polarity by a fixed number (in their implementation, they use a numerical score to show the polarity and strength of sentiment words - from -5 to +5). For example, *not excellent* changes from 5 to 1 (5 - 4), *not terrible* changes from -5 to -1 (-5 + 4) and *not good* changes from 3 to -1 (3 - 4). Thus, this *shift negation* method is equivalent to algorithm 4, where *pos_strongsubj* is converted to *pos_weaksubj*, *pos_weaksubj* is converted to *neg_weaksubj*, *neg_strongsubj* is converted to *neg_weaksubj*, and *neg_weaksubj* is converted to *pos_weaksubj*.

Algorithm 7 Check Other Intensifiers($adj\_intensifier$)

```
$intensifier\_count \Leftarrow 0$
for each child $c$ of the adjectival intensifier in the dependency tree do
    if the dependency is “advmod” and $c$ is an adverbial intensifier then
        $intensifier\_count \Leftarrow intensifier\_count + 1$
    else if the dependency is “conj_and” and $c$ is an adjectival intensifier then
        $intensifier\_count \Leftarrow intensifier\_count + 1$
    end if
end for
return $intensifier\_count$
```

3.4 Analysis of Algorithms Running Time

In section 3.3, we have explained the different algorithms we have developed in order to extract the refined sentiment features and features for contextual valence shifters. In this section, we will compute the running time of each of these algorithms in terms of big O notation. Since algorithm 1 uses all the other algorithms we will start with the basic ones and then compute the running time of the whole algorithm used. Starting with algorithm 2, its running time is equal to $O(1)$. Similarly, algorithms 3, 4 and 5 have all $O(1)$. For algorithm 7, if we say n is the number of children for the sentiment word, then the algorithm's running time will be $O(n)$. For algorithm 6, if we say m is the number of parents for the sentiment word, then the first for loop will be $O(n^2)$ whereas the second for loop, since it contains another nested loop, will be $O(mn^2)$. Finally, for algorithm 1 which uses all the other algorithms, if we say that l is the number of sentences for each document and q is the number of sentiment words in the sentence, then the algorithm will take $O(lqmn^2)$, which is the total running time for each document. To approximate this time, we will assume that l = m = n = q, so it will be $O(q^5)$. It is important to note that in our experiments, we ran the algorithm on all the documents of each dataset in parallel on a supercomputer.

3.5 Evaluation Methodology

The first three sets of features (F1, F2 and F3) construct the baseline for our proposed features, whereas refined F3 and F4 are the new feature sets that we added. In order to evaluate the performance of our proposed feature sets against the baseline, we used two of the most used datasets in the sentiment classification literature:

- **The Polarity dataset v 2.0:**

  This is the most used dataset used in sentiment classification. It contains 1,000 positive and 1,000 negative movie reviews collected from the IMDb archive [25].

- **The multi-domain dataset:**

  This dataset contains 1,000 positive and 1,000 negative reviews from four different review genres (books, DVDs, electronics and kitchen appliances), for a total of 8,000 reviews [6].

We will perform several experiments to evaluate the performance of the new features that we have proposed. First, we will evaluate the effectiveness of refining sentiment feature extraction against the traditional sentiment feature extraction. The experiments to perform this comparison between the baseline and the proposed method are adding: 1) negation features; 2) intensifier features; 3) negation and intensifier features; 4) negation, intensifier and polarity shifter features without ignoring sentiment words appearing after modals, or conditional or contrast words; and 5) all features with ignored sentiment words against adding the frequency of all sentiment words without their contextual polarity (baseline). Also, we will evaluate the effectiveness of adding refined F3 and F4 to F1 and F2 as well as refined F3 and F4 (with ignored sentiment words) to F1 and F2 against the baseline (F1, F2 and F3). Then, we will perform feature selection using the IG with a threshold of 0.0025 (as proposed by [9]) on the baseline (F1 to F3), all features (F1 to F4) and all features with ignored sentiment words (F1 to F4 with ignored words) to see whether there are any improvements in the performance.

Like any other text classification problem, sentiment classification research has mainly used three measures of classification effectiveness; namely, the accuracy, precision and recall, which are explained in detail in [31]. In addition, the two-tailed t-test is used to show the statistical significance of the new feature sets proposed. We used the same measures in our work to compare our proposed feature sets with the baseline in addition to feature selection:

1. *Accuracy:*

The accuracy of a classifier is defined as the percent of correctly classified objects. It is calculated as:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

TP denotes the number of positively-labeled test documents that were correctly classified as positive;

TN denotes the number of negatively-labeled test documents that were correctly classified as negative;

FP denotes the number of negatively-labeled test documents that were incorrectly classified as positive; and

FN denotes the number of positively-labeled test documents that were incorrectly classified as negative.

2. *Precision:*

   The precision for a class is defined as the probability that if a random document should be classified with this class, then this is the *correct* decision. Precision for the positive class for instance is calculated as follows:

$$P = \frac{TP}{TP + FP}$$

3. *Recall:*

   The recall for a class is defined as the probability that if a random document should be classified with this class, then this is the *taken* decision. Recall for the positive class for instance is calculated as follows:

$$R = \frac{TP}{TP + FN}$$

4. *The two-tailed t-test:*

   To test the statistical significance of our proposed feature sets, we performed the two-tailed *t*-test on the baseline against the proposed feature sets. This was done by randomizing each dataset 40 times to get different training and testing sets in each time, running the classifier on the 40 randomized orders on each dataset, and computing the *t*-ratio, the *p* value and the mean accuracy difference.

The *t*-ratio is calculated with the following equation:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{{SD_1}^2}{N_1} + \frac{{SD_2}^2}{N_2}}}$$

where $\bar{X_1}$ and $\bar{X_2}$ are the mean accuracies of groups 1 (baseline) and 2 (proposed feature sets), SD_1 and SD_2 are the standard deviations of the accuracies of the 2 groups, and N_1 and N_2 are the number of samples in each group, which is 40 in our case.

The larger the *t*-ratio, the smaller the probability that the mean accuracies of the 2 groups are the same and hence the higher the statistical significance of one group over the other. This probability is denoted to as the *p* value.

5. *Feature Selection:*

To test whether the proposed features are relevant in sentiment classification, we have performed feature selection when adding all the feature sets to check whether the proposed features of F4 are selected or not.

CHAPTER 4

EXPERIMENTAL RESULTS AND EVALUATION

We have performed several experiments to evaluate the performance of our proposed feature sets against the baseline. The first set of experiments (E1) shows the performance of adding different kinds of features in F4 (intensifiers, negation, and polarity shifters) while counting the sentiment words in the documents and changing their final polarity against counting the sentiment words while disregarding its contextual polarity that we will explain in section 4.1. The second set of experiments (E2) shows the performance of adding refined F3 and F4 (with and without ignored sentiment words) to F1 and F2 (all and all+ign) against adding F3 to F1 and F2 (F1 + F2 + F3), as we will discuss in section 4.2. Then, the last set of experiments (E3) shows the performance of the same features in E2 after performing feature selection on each one, which we will explain in section 4.3. Finally, we discuss the results of all experiments in section 4.4.

Instead of using SentiWordNet 3.0 as a sentiment lexicon, that was used in [9], we used the Subjectivity dictionary, since its effectiveness was reported when using it with contextual valence shifters in the SO approach [34]. To extract content-specific features (F2), different n-grams were extracted from the two used datasets:

1. The Polarity dataset: all unigrams, bigrams and trigrams with a frequency threshold of 10 were extracted, as in [2].

2. The multi-domain dataset: all unigrams and bigrams with a frequency threshold of 5 were extracted, as in [9].

We tried different combinations of n-grams for each dataset, and the ones listed above yielded the highest accuracy.

We used the term frequency instead of the term presence when extracting features, since we did a pilot experiment to compare their performance and the former outperformed the latter. For classification, we used SVM as the classification algorithm with default kernel settings due to being widely used in the literature, as discussed in chapter 2, and ran the classifier on the two datasets with 10-fold cross validation in each experiment. For feature selection, we used the IG heuristic to extract features whose IG is greater than 0.0025 using the whole dataset as the training set.

4.1 Comparing Sentiment Words with their Prior Polarity Against Contextual Polarity

In this section, we describe the first set of experiments that were performed to evaluate the performance of refining the traditional sentiment feature extraction method to take contextual polarity into consideration. We refer to this set of experiments as E1. This set includes the following experiments:

1. **Constructing the baseline for sentiment feature extraction:**
   This experiment is performed to construct the baseline for our proposed feature sets. It consists of sentiment features, which are the frequency of each sentiment word found in the document without taking its contextual polarity into consideration. We refer to this feature set as F3. The number of sentiment features for both datasets is shown in table 4.1, and the results of running the classifier on these datasets using F3 only are shown in tables 4.2, 4.3 and 4.4.

2. **Evaluating the effectiveness of refining the sentiment feature extraction method and adding other features for contextual valence shifters:**
   In this experiment, we wanted to evaluate the impact of refining the sentiment feature extraction method against the features used in experiment 1. So, instead of adding the frequency of each sentiment word with its prior polarity, its contextual polarity was

calculated first using the algorithms explained in section 3.3 and the new proposed features (intensification, negation, polarity shifters and frequency of sentiment words grouped with their polarity and type) were added. Also, lemmatization was performed on the words before searching for them in the sentiment lexicon. We refer to the refined sentiment features and features for these valence shifters as F3 + F4. The number of these features is listed for each dataset in table 4.1. As we can see from the table, the number of refined sentiment features is greater than sentiment features (baseline) because of lemmatization. The accuracy of the classifier using F3 + F4 on both datasets is shown in table 4.2 while the precision and recall are shown in table 4.3 and the overall precision and recall for the positive and negative classes on all datasets are illustrated in table 4.4. The results showed a clear improvement in the accuracy of the classifier when refining the sentiment feature extraction method and adding new features for contextual valence shifters (F3 + F4) against the traditional extraction method (F3), as there is an increase in the accuracy from 0.8-5% with an overall increase by more than 2% among all the datasets. In addition, tables 4.3 and 4.4 show an overall increase in both the precision and recall for the two classes (positive and negative) with a greater improvement in the average precision for the negative class and the average recall for the positive class. In addition, the results of the two-tailed t-test of F3 + F4 against F3 are shown in table 4.5. As we can see from the results, the t-ratio is large in each of the books, electronics and kitchen appliances datasets with a very small p value, which shows the statistical significance of our proposed feature sets against the baseline in these 3 datasets. Also, the mean accuracies in both the Polarity and multi-domain datasets using F3 + F4 are higher than the mean accuracies using F3 only.

3. **Evaluating the effectiveness of ignoring some sentiment words while refining the sentiment feature extraction method:**
   In this experiment, we wanted to measure the effectiveness of ignoring sentiment words appearing after modal verbs or conditional or contrast words while refining the sentiment feature extraction method and adding the features for contextual valence shifters against the features used in experiment 2. We refer to these features as F3 + F4 + ign. The number of features for this experiment is illustrated in table 4.1 for each

dataset. As we can see from the second and third rows of the table that show F3 + F4 and F3 + F4 + ign, respectively, the number of ignored sentiment words ranged between 60 and 80 for each dataset. The results of this experiment are shown in tables 4.2, 4.3 and 4.4. As illustrated in table 4.2, ignoring sentiment words appearing after modals or conditional or contrast words were only effective in the DVDs and electronics datasets with a little improvement in the accuracy. In table 4.3, some of the precision and recall results increased in the DVDs, electronics and kitchen appliances datasets from F3 + F4 to F3 + F4 + ign. But, on the average, we noticed that ignoring sentiment words degraded the performance of the classifier in terms of the accuracy, precision and recall.

Feature Set	Number of Features				
	Polarity Dataset	Multi-domain Dataset			
		Books	DVDs	Electronics	Kitchen Appliances
F3(baseline)	4420	3030	2920	1345	1247
F3+F4	4868	3262	3158	1448	1362
F3+F4(ign)	4804	3176	3088	1376	1293
F3+int	4860	3254	3150	1440	1354
F3+neg	4860	3254	3150	1440	1354
F3+int+neg	4864	3258	3154	1444	1358

Table 4.1: Number of features in each feature set in E1

4. **Evaluating the effectiveness of adding intensification only while refining the sentiment feature extraction method:**

   In this experiment, we wanted to measure the performance of the classifier when refining the sentiment feature extraction method with intensification only against the features used in experiment 1. Thus, only the first 4 features of F4 that count the intensified words according to their polarity and type and the last 4 features of F4 that count the frequency of sentiment words grouped with their final polarity and type (shown in section 3.3) are used along with the refined sentiment features. We refer to these features as F3 + int. The number of these features for each dataset is shown in table

Measure	Feature Set	Polarity	Multi-domain Dataset				Average
		Dataset	Books	DVDs	Electronics	Kitchen Appliances	
Accuracy	F3(baseline)	79%	72.95%	77.10%	76.05%	75.55%	76.13%
	F3+F4	**79.85%**	**77.90%**	77.35%	78.70%	**79.55%**	**78.67%**
	F3+F4(ign)	78.80%	75.45%	**77.70%**	**78.90%**	79.15%	78%
	F3+int	78.9%	75.15%	77.25%	77.95%	77.85%	77.42%
	F3+neg	79.4%	77.15%	76.85%	78.3%	79%	78.14%
	F3+int+neg	79.25%	77.45%	77.55%	78.4%	79.5%	78.43%

Table 4.2: Accuracy of different feature sets in E1

4.1 and the classifier's performance is shown in tables 4.2, 4.3 and 4.4. As we can see from table 4.2, the overall accuracy of F3 + int is 77.42% compared with that of F3, which is 76.13%. Also, both the overall precision and recall have improved from F3 to F3 + int. Thus, we concluded that intensification has a significant effect when refining the traditional sentiment feature extraction method.

5. **Evaluating the effectiveness of adding negation only while refining the sentiment feature extraction method:**
This experiment was performed to measure the impact of adding negation features only while refining the sentiment feature extraction method against the features used in experiments 1 and 4. This is done by negating the polarity and type of sentiment words preceded by negation terms and adding the 4 negation features from F4 described in section 3.3 as well as the last 4 features of F4 that count the frequency of sentiment words grouped with their final polarity and type. We refer to these features as F3 + neg. The number of these features is illustrated in table 4.1 and the results of performing this experiment are illustrated in tables 4.2, 4.3 and 4.4. As we expected before performing this experiment, the accuracy of the classifier improved much better than its accuracy when running it with F3 + int, which shows that adding negation is more effective than adding intensification. Moreover, there is a significant increase in the overall precision and recall from F3 + int to F3 + neg, which also indicates that negation features are

Dataset	Measure	Polarity	F3	F3+F4	F3+F4+ign	F3+int	F3+neg	F3+int+neg
Movies	Precision	pos	0.795	**0.798**	0.791	0.79	0.792	0.793
		neg	0.785	**0.799**	0.785	0.788	0.796	0.792
	Recall	pos	0.782	**0.800**	0.783	0.787	0.798	0.792
		neg	**0.798**	0.797	0.793	0.791	0.79	0.793
Books	Precision	pos	0.713	**0.762**	0.736	0.734	0.754	0.756
		neg	0.748	**0.798**	0.776	0.771	0.792	0.795
	Recall	pos	0.767	**0.811**	0.793	0.788	0.807	0.81
		neg	0.692	**0.747**	0.716	0.715	0.736	0.739
DVDs	Precision	pos	0.752	0.760	0.763	0.755	0.76	**0.767**
		neg	0.793	0.788	**0.793**	**0.793**	0.778	0.785
	Recall	pos	**0.808**	0.799	0.804	0.807	0.785	0.792
		neg	0.734	**0.768**	0.75	0.738	0.752	0.759
Electronics	Precision	pos	0.768	0.786	**0.793**	0.781	0.787	0.785
		neg	0.753	**0.788**	0.785	0.778	0.779	0.783
	Recall	pos	0.746	**0.788**	0.782	0.777	0.776	0.782
		neg	0.775	0.786	**0.796**	0.782	0.79	0.786
Kitchen	Precision	pos	0.786	0.798	**0.802**	0.795	0.793	0.797
		neg	0.731	**0.793**	0.782	0.763	0.787	**0.793**
	Recall	pos	0.702	**0.792**	0.774	0.75	0.785	0.791
		neg	**0.809**	0.799	**0.809**	0.807	0.795	0.799

Table 4.3: Precision and recall of different feature sets in E1

Measure	F3	F3+F4	F3+F4+ign	F3+int	F3+neg	F3+int+neg
Pos. Precision	0.763	**0.781**	0.777	0.771	0.777	0.780
Neg. Precision	0.762	**0.793**	0.784	0.779	0.786	0.790
Pos. Recall	0.761	**0.798**	0.787	0.782	0.790	0.793
Neg. Recall	0.762	**0.779**	0.773	0.767	0.773	0.775

Table 4.4: Average positive and negative precision and recall of E1 experiments

Measure	Polarity Dataset	Multi-domain Dataset			
		Books	DVDs	Electronics	Kitchen Appliances
t-ratio	0.78	7.01	0.54	2.22	1.94
P value	0.44	2.1E-8	0.59	0.03	0.06
Mean accuracy Diff.	0.8	5.45	0.6	2.3	1.5

Table 4.5: The two-tailed *t*-test results for F3+F4 against F3

more beneficial than intensification features.

6. **Evaluating the effectiveness of adding both intensification and negation while refining the sentiment feature extraction method:**
Finally, we performed the last experiment in E1 to assess the performance of the classifier when adding both intensification and negation features while refining the sentiment feature extraction method and compare it with adding all features of the contextual valence shifters (that include: intensification, negation and polarity shifters - the features used in experiment 2). We refer to these features as F3 + int + neg. This feature set includes all sentiment words after modifying their polarity if they are preceded by intensifiers and/or negation terms as well as the counts of these intensifiers and negation terms and the frequency of the words according to their final polarity and type. The number of these features is shown in table 4.1 for each dataset and the results of the classifier are shown in tables 4.2, 4.3 and 4.4. As we can see from the tables, there is only a slight increase in the accuracy, precision and recall from F3 + int + neg to F3 + F4. This show that adding the polarity shifters is not very effective, which was anticipated since the list of polarity shifters is not comprehensive and not long enough and so they were not found as much as negation terms and intensifiers. In appendix D, we show some sample documents from the datasets we used in our experiments that were misclassified when using the baseline and correctly classified when using our proposed feature sets and we show their sentiment features (F3) and their refined sentiment features and features for contextual valence shifters (F3 + F4).

4.2 Comparing All Features with Sentiment Words with their Prior Polarity Against Contextual Polarity

In this section, we show the results of the second set of experiments (E2), that compares the performance of adding F1, F2 and F3 (F1 + F2 + F3) against the performance of all 4 feature sets without ignored sentiment words (all) and with ignored sentiment words (all+ign). These experiments are explained as follows:

1. **Constructing the baseline for adding all features:**
   In this experiment, we added stylistic features (F1), ngrams (F2) and sentiment features (F3) together to construct our second baseline. We refer to these features as F1 + F2 + F3. The number of these features for each dataset is shown in table 4.6 and the results of this experiment are shown in tables 4.8, 4.9 and 4.10.

2. **Evaluating the effectiveness of refining the sentiment feature extraction method when adding all features:**
   We performed this experiment to measure the impact of refining the sentiment feature extraction method when adding them to F1 and F2 against the baseline. We refer to these features as: all. The number of features for each dataset is shown in table 4.7 and the results of the classifier are shown in tables 4.8, 4.9 and 4.10. As we can see from table 4.8, the accuracy of the classifier improved in all datasets with an overall increase of about 1.3%, except the polarity dataset where the performance degraded with 0.2%. In addition, we observed a significant improvement in the overall precision and recall from experiment 1, which also indicates that refining the sentiment feature extraction method and adding new features for contextual valence shifters are effective when adding them to F1 and F2 compared with the traditional sentiment feature extraction method.

3. **Evaluating the effectiveness of ignoring some sentiment words while refining the sentiment feature extraction method when adding all features:**
   In this experiment, we wanted to assess the effect of ignoring sentiment words appearing after modals or conditional or contrast words when adding all features (F1, F2, F3

and F4) against experiment 2. We refer to these features as: all-ign. The number of these features for each dataset is illustrated in table 4.7 whereas the accuracy, precision and recall are illustrated in tables 4.8, 4.9 and 4.10. From the results, we observed that ignoring sentiment words degraded the accuracy in all datasets, with an overall decrease of 0.3% while some of the precision and recall results increased slightly in the books, DVDs and electronics datasets. So, it may not be practical to ignore such words in the review domain, as was proposed by [28].

Feature Set	Number of Features				
	Polarity Dataset	Multi-domain Dataset			
		Books	DVDs	Electronics	Kitchen Appliances
F1	245	245	245	245	245
F2	10008	4180	4592	2482	2160
F3	4420	3030	2920	1345	1247
F1+F2+F3	**14673**	**7455**	**7757**	**4072**	**3652**

Table 4.6: Number of features in each feature set in E2 (baseline)

Feature Set	Number of Features				
	Polarity Dataset	Multi-domain Dataset			
		Books	DVDs	Electronics	Kitchen Appliances
F4	16	16	16	16	16
ref-F3	4852	3246	3142	1432	1346
all	**15121**	**7687**	**7995**	**4175**	**3767**
ref-F3+ign	4788	3160	3072	1360	1277
all+ign	**15057**	**7601**	**7925**	**4103**	**3698**

Table 4.7: Number of features in each feature set in E2 (proposed feature sets)

Measure	Feature Set	Polarity	Multi-domain Dataset				Average
		Dataset	Books	DVDs	Electronics	Kitchen Appliances	
Accuracy	F1+F2+F3	**84.85%**	77.20%	79.50%	79.20%	81.25%	80.4%
	all	84.65%	**78.65%**	**80.60%**	**81.45%**	**83.00%**	**81.67%**
	all+ign	84.45%	78.35%	80.50%	81.10%	82.45%	81.37%

Table 4.8: Accuracy of different feature sets in E2

Dataset	Measure	Polarity	F1+F2+F3	all	all+ign
Movies	Precision	pos	**0.857**	0.850	0.848
		neg	0.840	**0.843**	0.841
	Recall	pos	0.836	**0.841**	0.84
		neg	**0.861**	0.852	0.849
Books	Precision	pos	0.762	**0.781**	0.774
		neg	0.783	**0.793**	**0.793**
	Recall	pos	0.791	0.797	**0.800**
		neg	0.753	**0.776**	0.767
DVDs	Precision	pos	0.787	0.799	**0.8**
		neg	0.803	**0.814**	0.81
	Recall	pos	0.809	**0.818**	0.813
		neg	0.781	0.794	**0.797**
Electronics	Precision	pos	0.789	0.807	**0.819**
		neg	0.796	**0.823**	0.822
	Recall	pos	0.798	**0.827**	0.824
		neg	0.786	**0.802**	0.798
Kitchen	Precision	pos	0.809	**0.827**	0.822
		neg	0.816	**0.833**	0.827
	Recall	pos	0.818	**0.835**	0.829
		neg	0.807	**0.825**	0.820

Table 4.9: Precision and recall of different feature sets in E2

Measure	F1+F2+F3	all	all+ign
Pos. Precision	0.801	**0.813**	**0.813**
Neg. Precision	0.808	**0.821**	0.819
Pos. Recall	0.810	**0.824**	0.821
Neg. Recall	0.798	**0.810**	0.806

Table 4.10: Average positive and negative precision and recall of E2 experiments

4.3 Comparing Selected Features with Sentiment Words with their Prior Polarity Against Contextual Polarity

In this section, we perform feature selection on the feature sets used in the previous section to evaluate the effectiveness of our proposed feature sets and test if they were significant and relevant among all other features or not. We selected features whose IG are greater than 0.0025 using the whole dataset as the training set. We refer to this set of experiments as E3. Like E2, this set of experiments includes 3 experiments, as follows:

1. **Constructing the baseline for feature selection when adding sentiment features to stylistic features and ngrams:**
   This experiment was performed to construct the baseline for E3. The features used are the same as those in the previous section in experiment 1; that is, F1 + F2 + F3, after performing feature selection on them. The number of reduced features for this experiment is illustrated in table 4.11 and the results of the classifier are shown in tables 4.12, 4.13 and 4.14.

2. **Evaluating the effectiveness of feature selection when refining the sentiment feature extraction method and adding all features:**
   In this experiment, we wanted to evaluate the effectiveness and relevance of our proposed features against the other features (F1 + F2 + F3). So, we added all the features (F1 + F2 + F3 + F4) together and then performed feature selection on them. The number of selected features for each dataset is shown in table 4.11. It is important to note that most of the 16 features in our proposed feature set F4 were selected among the

whole features, with some of them having the highest IG. This indicates the importance and relevance of our proposed feature set among the other features in classifying the documents. The results of this experiment are shown in tables 4.12, 4.13 and 4.14. The results showed a clear improvement in the accuracy, except for the DVDs dataset which remained the same, with an overall increase of about 1%. Also, the precision and recall increased in most cases in the datasets.

3. **Evaluating the effectiveness of ignoring some sentiment words while performing feature selection on all features:**
   We performed this experiment to evaluate the impact of ignoring sentiment words that appear after modals or conditional or contrast words after adding all features and performing feature selection on them. The number of these features for each dataset is shown in table 4.11 and the results are shown in tables 4.12, 4.13 and 4.14. As we can see from the tables, the accuracy has only increased slighty from using "all" features in the DVDs dataset by 0.2%, it remained the same in the kitchen appliances dataset and decreased in the remaining datasets, making an overall decrease of 0.24%. In addition, some of the precision and recall results have increased marginally in the polarity, DVDs and kitchen appliances datasets, but on the average, all the precision and recall results have decreased.

Feature Set	Number of Features				
	Polarity Dataset	Multi-domain Dataset			
		Books	DVDs	Electronics	Kitchen Appliances
F1+F2+F3	1471	482	503	401	408
all	1490	493	549	423	430
all+ign	1491	495	543	420	428

Table 4.11: Number of selected features in each feature set in E3

Measure	Feature Set	Polarity	Multi-domain Dataset				Average
		Dataset	Books	DVDs	Electronics	Kitchen Appliances	
Accuracy	F1+F2+F3	87.05%	84.20%	84.60%	83.55%	84.85%	84.85%
	all	**87.15%**	**85.50%**	84.60%	**84.85%**	**86.55%**	**85.73%**
	all+ign	86.9%	84.6%	**84.8%**	84.6%	**86.55%**	85.49%

Table 4.12: Accuracy of different selected feature sets in E3

Dataset	Measure	Polarity	F1+F2+F3	all	all+ign
Movies	Precision	pos	0.874	0.874	0.868
		neg	0.867	0.869	**0.87**
	Recall	pos	0.866	0.868	**0.87**
		neg	**0.875**	**0.875**	0.868
Books	Precision	pos	0.823	**0.837**	0.829
		neg	0.864	**0.874**	0.865
	Recall	pos	0.872	**0.881**	0.872
		neg	0.812	**0.829**	0.82
DVDs	Precision	pos	0.820	**0.828**	0.83
		neg	**0.876**	0.867	0.869
	Recall	pos	**0.886**	0.874	0.876
		neg	0.806	0.818	**0.82**
Electronics	Precision	pos	0.819	**0.840**	0.837
		neg	0.854	**0.857**	0.855
	Recall	pos	**0.862**	0.861	0.859
		neg	0.809	**0.836**	0.833
Kitchen	Precision	pos	0.859	0.870	**0.875**
		neg	0.839	**0.861**	0.857
	Recall	pos	0.834	**0.859**	0.853
		neg	0.863	0.872	**0.878**

Table 4.13: Precision and recall of different selected feature sets in E3

Measure	F1+F2+F3	all	all+ign
Pos. Precision	0.839	**0.850**	0.848
Neg. Precision	0.860	**0.866**	0.863
Pos. Recall	0.864	**0.869**	0.866
Neg. Recall	0.833	**0.846**	0.844

Table 4.14: Average positive and negative precision and recall of E3 experiments

4.4 Discussion of the Results

In this chapter, we have carried out some experiments to evaluate the performance of our proposed feature sets. First, in section 4.1, we evaluated the performance of the refined sentiment features (F3) after calculating their contextual polarity and adding features that count the number of intensifiers, negation terms, and polarity shifters grouped according to the polarity and type of the words modified by them as well as the frequency of sentiment words grouped with their final polarity and type (F3+F4) against counting all sentiment words while disregarding their contextual polarity (F3). The results we got showed that refining the sentiment feature extraction method by taking their contextual polarity into consideration and adding features to count the contextual valence shifters have clearly improved the classifier's performance in terms of the accuracy, precision and recall. Furthermore, the two-tailed *t*-test showed statistical significance of our proposed feature sets against the baseline in 3 out of the 5 datasets used. In addition, we showed that ignoring sentiment words appearing after conditional or contrast words or modal verbs improved the classifier's accuracy in some cases (3 datasets) and decreased its performance in other cases (2 datasets), but on the average, it degraded the accuracy of the classifier by 1%. Furthermore, we have shown that the effect of adding negation features is better than adding intensifier features, and that adding both negation and intensifier features has a significant improvement on the classifier.

The second set of experiments, described in section 4.2, was performed to evaluate the performance of adding the refined sentiment features and features for contextual valence shifters (F3 and F4) to content-free and content-specific features (F1 and F2). We also got promising results that were better than the baseline, that adds all sentiment features with their prior polarity (F3) to F1 and F2, except for the Polarity dataset, where the accuracy decreased by 0.2% and the precision of the positive class as well as the recall of the negative class decreased slightly. Furthermore, we proved that ignoring sentiment words that appear after conditional or contrast words or modal verbs with these feature sets reduced the classifier's accuracy in all datasets (with an average of 1%), but slightly improved its precision and recall in some datasets.

Finally, we performed the last set of experiments that we explained in section 4.3 to evaluate the performance of feature selection on the feature sets described in section 4.2. Feature selection has shown that our proposed feature set that counts the different contextual valence shifters (F4) is effective among all the feature sets, since most of them were selected and some of them were among the features with the highest IG. Also, the results we got after feature selection on the classifier showed an improved accuracy in all datasets in addition to an overall increase in the precision and recall among all of them.

As a result, refining the sentiment feature extraction method and adding our proposed feature set to handle contextual valence shifters were more effective in product reviews than in movie reviews, due to the nature of movie reviews, which can contain a lot of objective sentences about the plot or characters or the movie that contain sentiment words. These sentiment words should not be counted when determining the overall sentiment of the author [19]. But overall, we have shown that refining the sentiment feature extraction method and adding our proposed feature set are effective for document-level sentiment classification.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Document-level sentiment classification is an emerging research field in NLP and ML. Researchers' interests have been mainly divided into two main approaches: ML and SO. In the ML approach, researchers have been attempting different feature sets to improve the classifier's performance. They have tried the same features used in classic text classification, such as n-grams and they have yielded good results. Some have also tried extracting words with SO, for example, the authors in [9] have implemented a sentiment feature extraction method from SentiWordNet 3.0. However, the method proposed in that paper is not efficient to grasp the true polarity of the sentiment words, as the authors in [28] have noted that there are different categories of the so-called *contextual valence shifters* that can change or neutralize the polarity and/or intensity of sentiment words.

Therefore, in this study, we have proposed a document-level sentiment classifier with a new feature set to include contextual valence shifters in a non-trivial way as well as a refinement for the sentiment feature extraction method that is proposed in [9]. The results we have obtained from our system showed a much better performance than other systems who have tried incorporating sentiment features and contextual valence shifters in the ML approach.

There are different directions for extending this system. One direction could be merging features of contextual valence shifters (F4) that have equivalent polarity and/or type together into one feature; for example, we can merge *pos_strongsubj* with *pos_weaksubj* into one *pos* feature representing the total number of positive words, and *neg_strongsubj* with *neg_weaksubj* into a *neg* feature representing the total number of negative words. Also, we can try merging *shiftpos_strongsubj* and *shiftpos_weaksubj* with *pos* and the same for *shiftneg_strongsubj* and *shiftneg_weaksubj*. Another direction for future work would be to modify the quantifiers list provided by [7], which contains single- and multi-word expressions that intensify or diminish the polarity of the words after them, e.g. *it is less expensive*, since each quantifier has a numerical score (from -5 to +5) showing its polarity and intensity. So, we could modify these scores to indicate their polarity and type (*pos_strongsubj*, *pos_weaksubj*, *neg_strongsubj*, and *neg_weaksubj*). And then we can add another 4 features to F4 to represent diminished words grouped together according to the polarity and type of the sentiment words modified by them, e.g. *diminisher_pos_strongsubj*. Furthermore, we can try to build an Arabic sentiment lexicon and a list of intensifiers and other contextual valence shifters to build an Arabic document-level classifier with our proposed feature sets. Finally, as we stated earlier in the thesis, two assumptions are being made in document-level sentiment classification: 1) The document expresses the opinion of one author only; and 2) The document talks about one object. Therefore, two important directions for future work are to: 1) Extract the sentences that express the opinion of the author; and 2) Perform Named Entity Recognition (NER) and topic extraction to extract topic sentences and sentences talking about different objects and separate them so that we can classify the sentiment of each object being talked about in the document.

BIBLIOGRAPHY

[1] A. Abbasi and H. Chen. Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20:67–75, September 2005.

[2] A. Abbasi, H. Chen, and A. Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Trans. Inf. Syst.*, 26:12:1–12:34, June 2008.

[3] A. Abbasi, S. France, Z. Zhang, and H. Chen. Selecting attributes for sentiment classification using feature relation networks. *IEEE Trans. on Knowl. and Data Eng.*, 23:447–462, March 2011.

[4] A. Andreevskaia and S. Bergler. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. In *Proceedings of ACL-08 HLT*, number June, pages 290–298, Columbus, Ohio, USA, 2008. Association for Computational Linguistics.

[5] S. Baccianella, A. Esuli, and F. Sebastiani. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th conference on International Language Resources and Evaluation*, volume 25, page 2010, 2010.

[6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Annual Meeting-Association For Computational Linguistics*, volume 45, pages 440–447, 2007.

[7] J. Brooke. A semantic approach to automated text sentiment analysis. Masters thesis, Simon Fraser University, 2009.

[8] J. Carrillo de Albornoz, L. Plaza, and P. Gervás. A hybrid approach to emotional sentence polarity and intensity classification. In *Proceedings of the Fourteenth Conference*

on Computational Natural Language Learning, CoNLL '10, pages 153–161, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] Y. Dang, Y. Zhang, and H. Chen. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems*, 25:46–53, July 2010.

[10] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(1-4):131–156, 1997.

[11] M. De Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Citeseer, 2006.

[12] M. De Marneffe and C. Manning. Stanford typed dependencies manual, 2010.

[13] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on International Language Resources and Evaluation*, volume 6, pages 417–422. Citeseer, 2006.

[14] Z. Fei, J. Liu, and G. Wu. Sentiment classification using phrase patterns. In *Proceedings of the The Fourth International Conference on Computer and Information Technology*, CIT '04, pages 1147–1152, Washington, DC, USA, 2004. IEEE Computer Society.

[15] D. Fradkin and I. Muchnik. Support vector machines for classification. *Discrete methods in epidemiology*, 70:13–20, 2006.

[16] M. Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[17] Y. He. Learning sentiment classification model from labeled features. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1685–1688, New York, NY, USA, 2010. ACM.

[18] L. Jia, C. Yu, and W. Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1827–1830, New York, NY, USA, 2009. ACM.

[19] A. Kennedy and D. Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, 2006.

[20] B. Liu. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing,*, pages 978–1420085921, 2010.

[21] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November 1995.

[22] T. Nakagawa, K. Inui, and S. Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics, 2010.

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

[24] C. Nicholls and F. Song. Comparison of Feature Selection Methods for Sentiment Analysis. *Advances in Artificial Intelligence*, pages 286–289, 2010.

[25] B. Pang and L. Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[26] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135, January 2008.

[27] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empiri-*

cal methods in natural language processing-Volume 10, pages 79–86. Association for Computational Linguistics, 2002.

[28] L. Polanyi and A. Zaenen. Contextual valence shifters. *Computing attitude and affect in text: Theory and applications*, pages 1–10, 2006.

[29] L. Qiu, W. Zhang, C. Hu, and K. Zhao. Selc: a self-supervised model for sentiment classification. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 929–936, New York, NY, USA, 2009. ACM.

[30] J. Read and J. Carroll. Weakly supervised techniques for domain-independent sentiment classification. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, TSA '09, pages 45–52, New York, NY, USA, 2009. ACM.

[31] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[32] M. Sokolova and G. Lapalme. Verbs speak loud: Verb categories in learning polarity and strength of opinions. *Advances in Artificial Intelligence*, pages 320–331, 2008.

[33] M. Sokolova and G. Lapalme. Classification of opinions with non-affective adverbs and adjectives. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing*, pages 416–420, 2009.

[34] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, (Early Access):1–41, 2010.

[35] P. D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[36] F. Tweedie and R. Baayen. How variable may a constant be? Measures of lexical richness in perspective. *Computers and the Humanities*, 32(5):323–352, 1998.

[37] S. Wang, D. Li, Y. Wei, and H. Li. A Feature Selection Method Based on Fishers Discriminant Ratio for Text Sentiment Classification. *Web Information Systems and Mining*, pages 88–97, 2009.

[38] C. Whitelaw, N. Garg, and S. Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

[39] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[40] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.

[41] R. Xia and C. Zong. Exploring the use of word relation features for sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1336–1344, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[42] R. Xia, C. Zong, and S. Li. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 2010.

[43] R. Zheng, J. Li, H. Chen, and Z. Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, 57:378–393, February 2006.

Appendices

Appendix A: **Feature Sets**

This appendix shows the list of feature sets that were used in our experiments.

1. Stylistic Features:

   These features were provided by [43]:

   1. Total number of characters(C)

   2. Total number of alphabetic characters/C

   3. Total number of upper-case characters/C

   4. Total number of digit characters/C

   5. Total number of white-space characters/C

   6. Total number of tab spaces/C

   7-32. Frequency of letters (26 features)

   33-53. Frequency of special characters (21 features)

   54. Total number of words (M)

   55. Total number of short words (less than four characters)/M

   56. Total number of characters in words/C

   57. Average word length

   58. Average sentence length in terms of character

   59. Average sentence length in terms of word

   60. Total different words/M

   61. Hapax legomena*

   62. Hapax dislegomena*

   63. Yules K measure*

   64. Simpsons D measure*

   65. Sichels S measure*

   66. Brunets Wmeasure*

   67. Honores R measure*

   68-87. Word length frequency distribution /M (20 features)

   88-95 Frequency of punctuations (8 features)

   96-245 Frequency of function words (150 features)

Note: The definitions of measures with "*" can be found in [36].

2. Content-specific Features:

   These features vary from one dataset to another. For the Polarity dataset, we extracted all unigrams, bigrams, and trigrams whose frequency is greater than 9, and for the multi-domain dataset, we extracted all unigrams and bigrams with a frequency threshold of 5. The number of features and some examples of them for each dataset are as follows:

   - The Polarity dataset:

     This dataset included 10,008 n-grams. Some examples are: director, hangs, archer, correctly, quinlan, real estate, police force, good script, rocky horror picture, star wars trilogy, and hong kong action.

   - The books dataset:

     This dataset included 4,180 n-grams. Some examples are: steinbeck, capitalism, practically, designed, initial, character development, global warming, book reads, white people, and worst book.

   - The DVDs dataset:

     This dataset included 4,592 n-grams. Some examples are: adopted, purchase, awards, menacing, reviews, blues, high quality, viewing experience, original film, dvd collection, commander geordi, and world war.

   - The electronics dataset:

     This dataset included 2,482 n-grams. Some examples are: glasses, charge, programmed, procedure, packaged, decent, flash drives, firmware update, worked perfectly, wireless network, and radio reception.

   - The kitchen appliances dataset:

     This dataset included 2,160 n-grams. Some examples are: lids, shelves, returned, toasting, amount, good coffee, product arrived, fitted sheet, long time, and stopped working.

3. Sentiment Features:

   These features vary from one dataset to another. Also, as we explained in chapter 3, there are sentiment features from the baseline (F3) and from our proposed feature sets (refined F3). The number of these features and some examples of them for each dataset are as follows:

- The Polarity dataset:
  In this dataset, F3 included 4,420 features, whereas refined F3 included 4,852 features. Some examples are: admission_noun, adeptly_adverb, apathy_noun, amicable_adj, affirm_verb, ambitiously_anypos, admirable_adj, abuse_noun, active_adj, and advantage_noun.
- The books dataset:
  In this dataset, F3 included 3,030 features, whereas refined F3 included 3,246 features. Some examples are: amazingly_anypos, abide_anypos, allowable_adj, afraid_adj, abdicate_verb, astonishing_adj, accidental_adj, applaud_verb, anger_noun, awful_adj, and awesome_adj.
- The DVDs dataset:
  In this dataset, F3 included 2,920 features, whereas refined F3 included 3,142 features. Some examples are: assure_verb, accomplish_verb, ambiguous_adj, asinine_adj, archaic_adj, astonishing_adj, acclaim_noun, ache_verb, agony_noun, and abhor_verb.
- The electronics dataset:
  In this dataset, F3 included 1,345 features, whereas refined F3 included 1,432 features. Some examples are: argumentative_anypos, advice_noun, attractively_anypos, accomplish_verb, admire_verb, admittedly_anypos, avoidance_noun, amusing_adj, attractive_adj, and annoyance_noun.
- The kitchen appliances dataset:
  In this dataset, F3 included 1,247 features, whereas refined F3 included 1,346 features. Some examples are: afford_verb, alas_anypos, ashamed_adj, avoid_verb, appealing_adj, agree_verb, admittedly_anypos, accurate_adj, absence_noun, and allow_verb.

4. Contextual Valence Shifters:

These features are the 16 features that handle contextual valence shifters. They are explained in detail in section 3.3 and are listed below:

(a) *Intensifier_pos_strongsubj*

(b) *Intensifier_pos_weaksubj*

(c) *Intensifier_neg_strongsubj*

(d) *Intensifier_neg_weaksubj*

(e) *Negation_pos_strongsubj*

(f) *Negation_pos_weaksubj*

(g) *Negation_neg_strongsubj*

(h) *Negation_neg_weaksubj*

(i) *Shiftneg_strongsubj*

(j) *Shiftneg_weaksubj*

(k) *Shiftpos_strongsubj*

(l) *Shiftpos_weaksubj*

(m) *Pos_strongsubj*

(n) *Pos_weaksubj*

(o) *Neg_strongsubj*

(p) *Neg_weaksubj*

Appendix B: **Modal Verbs and Conditional and Contrast Words**

Modal Verbs

can	could	had better	have to	has to	may
might	must	need to	need not	needn't	ought
ought to	shall	should	will	would	

Conditional Words

if unless whether provided that

Contrast Words

though	although	nonetheless	nevertheless	even so	despite
in spite of	notwithstanding	instead of	regardless of	irrespective of	disregardless of
even though	rather than				

Appendix C: **Combinations of Valence Shifters and Intensifiers**

In section 3.3 in algorithms 1 and 2, there are 3 boolean variables that show if the word is intensified, negated or preceded by a polarity shifter; namely, *intensified*, *negated*, and *shifted*, respectively. In this appendix, we show how the final polarity of the sentiment word as well as the contextual features (F4) are affected by the different combinations of the values of these 3 variables, as follows:

1. *negated* = FALSE, *intensified* = FALSE and *shifted* = FALSE:
   Since all 3 variables are set to FALSE, this means there are no changes that happened to the sentiment word. Thus, algorithm 1 will go to algorithm 2 in line 13 where only line 15 in the algorithm will be executed by setting *word_count* to 1. Then, back to algorithm 1, lines 14 and 15 will be executed. So, the final polarity will be the same as the prior polarity, and both the sentiment feature of the word and the frequency of the words with the same polarity and type will be incremented by 1.

2. *negated* = FALSE, *intensified* = FALSE and *shifted* = TRUE:
   Here, the sentiment word is preceded by a polarity shifter only. So, its polarity will be shifted towards the opposite polarity. So, algorithm 1 will go to algorithm 3 where either one of the two conditions will be true (based on the prior polarity of the word and whether it is preceded by a positive or negative polarity shifter) and the final polarity will be shifted. After that, it will go back to algorithm 1 which will go to algorithm 2. Line 15 will be executed by setting *word_count* to 1. Then, lines 23 and 24 will be executed by changing the *word_count* of the sentiment word to -1 and the shifter feature of the word will be incremented by 1 (if the word's polarity is *positive* the shifter feature will be *shiftneg* and if it is *negative*, then the feature will be *shiftpos*; and the feature's type will be the same as the word's type - *strongsubj* or *weaksubj*). Then, back to algorithm 1, the word's sentiment feature will be decremented by 1 and the frequency of the words with the same polarity and type as the final (shifted) polarity will be incremented by 1 (lines 14 and 15).

3. *negated* = FALSE, *intensified* = TRUE and *shifted* = FALSE:

Here, only *intensified* is set to TRUE, so the word is intensified but not negated or preceded by a polarity shifter. So, we only want here to increment the intensifier feature with the word's polarity and type by 1 and increment the sentiment feature of the word. Also, we need to intensify the polarity of the word and increment the frequency of the words with the intensified polarity and type by 1. Thus, lines 15, 18 and 20 in algorithm 2 and lines 14 and 15 in algorithm 1 will be executed and these changes will be saved to the corresponding features.

4. *negated* = FALSE, *intensified* = TRUE and *shifted* = TRUE:
   When the sentiment word is preceded by both an intensifier and a polarity shifter, the shifter will shift the intensified word (e.g. *he lacks numerous abilities*). So, the final polarity of the word will be shifted towards the opposite polarity (algorithm 3) and then in algorithm 2, the prior polarity of the word will be intensified (line 20), the word count of the word will be -1 and the shifter feature of the final (intensified) polarity will be incremented by 1 (lines 23 - 24). Finally, in algorithm 1, the sentiment feature of the word will be decremented by 1 and the frequency of the words with the shifted polarity will be incremented by 1 (lines 14 and 15).

5. *negated* = TRUE, *intensified* = FALSE and *shifted* = FALSE:
   When a sentiment word is negated, all the needed changes are to negate its polarity and decrement its count. Algorithm 2 in this state will go through lines 9, 11 and 12. So, *word_count* will be set to -1, the negation feature of the word's polarity will be incremented by 1, and the final polarity of the word will be negated. Then, in algorithm 1, the sentiment feature of the word will be decremented, and finally the the frequency of the words with the negated polarity and type will be incremented by 1 (lines 14 adn 15).

6. *negated* = TRUE, *intensified* = FALSE and *shifted* = TRUE:
   When both *negated* and *shifted* are TRUE, it is as if the word is negated twice (e.g. *he does not lack wisdom*). Thus, the contextual polarity is the same as the prior polarity of the sentiment word. In algorithm 2, *word_count* will be set to 1 (line 3), and then in algorithm 1, the word's sentiment feature will be incremented by 1 and the frequency of the words with the same polarity and type will be also incremented by 1 (lines 14

and 15).

7. *negated* = TRUE, *intensified* = TRUE and *shifted* = FALSE:
   When the sentiment word is negated and intensified, the negation affects the intensified word, not just the word (e.g. *he is not very good*). Here, the sentence negates the expression *very good* and not just *good*. So, we first want to intensify the word's polarity and then negate it. Algorithm 2 will go through lines 6 and 7 to first intensify the word's polarity and set the word's count to 1. Then, in line 11, the negation feature of the final (intensified) polarity will be incremented by 1 and in line 12, the final polarity will be negated. Then, back to algorithm 1, the sentiment feature of the word will be incremented by 1 in line 14, since the negation does not totally remove the effect of the word (in the sentence, it means that he is good but not very good (or not excellent)). And in line 15 the frequency of the words with the same polarity and type as the final negated polarity will be incremented by 1.

8. *negated* = TRUE, *intensified* = TRUE and *shifted* = TRUE:
   This state is somehow similar to the state where both *negated* and *shifted* are TRUE and *intensified* is FALSE. Consider, for example, the sentence *he does not lack great abilities*. In this sentence, the negation *not* and the polarity shifter *lack* will remove the effect of each other, but the presence of the intensifier does not mean that he has *great* abilities. So, we do not intensify the polarity of the word and we treat it like the case when *intensified* is set to FALSE (case number 6).

Appendix D: **Classification of Document Samples with Different Features**

In chapter 4, we have demonstrated the significance of our proposed feature sets compared with traditional sentiment features. In this appendix, we will show real examples of some sample documents that were used in our experiments, and how their predicted classification was changed from the wrong class (with using the baseline features) to the correct class (with using our proposed features). To do this, we selected 900 documents from each class in the books dataset to build the classifier model with 10-fold cross validation, and then used the remaining 100 documents from each class as the new testing set. We then ran the classifier with sentiment features alone with their prior polarity (F3) and then with refined sentiment features and features for contextual valence shifters (F3 + F4).

1. **Change of classification from negative to positive:**
   The actual classification of the following sample documents is positive. They were classified as negative when running the classifier with F3 (baseline), and as positive when running it with refined F3 and F4.

   (a) Sample document #1:
   "Some of Patterson's conclusions were a bit of a reach. Some of his segways weren't all that smooth. But this book is just plain entertaining. It is chopped full of stories on rockstars making deals with the Devil,lingering around after death, and backed up with just enough innuendo-ish research for it to almost be believable. Well, some of it actually believable. Patterson scribes on about various rockstar-occult alliances (in particular the Rolling Stones and Led Zeppelin) and uses quotes from the musicians themselves."

   The feature set F3 for this document included the following features:

   - believable_adj: 2 - a positive strongsubj word.
   - death_noun: 1 - a negative weaksubj word.
   - devil_noun: 1 - a negative strongsubj word.
   - entertaining_adj: 1 - a positive strongsubj word.
   - just_anypos: 2 - a positive weaksubj word.

- well_adverb: 1 - a positive weaksubj word.
- smooth_adj: 1 - a positive weaksubj word.

When classified with refined F3 and F4, refined F3 contained the same features as F3 with *smooth_adj* getting the value -1 (since it was negated), in addition to the feature: {back_verb: 1 - a pos weaksubj word} since lemmatization was performed in refined F3. F4 contained the following features (with values greater than 0):

- intensifier_pos_strongsubj: 1 - for the expression (plain entertaining)
- negation_pos_weaksubj: 1 - for the expression (n't .. smooth)
- pos_strongsubj: 3 - for the words (entertaining - believable (twice))
- pos_weaksubj: 3 - for the words (back - just - well)
- neg_strongsubj: 1 - for the word (devil)
- neg_weaksubj: 2 - for the words (n't .. smooth - death)

(b) Sample document #2:
"This is the first of a great series of books. Knights & squires, quests & fairies, action & discovery —- old stories re-told in an easy-to-read style with a dose of dry humor. Geared towards older children (my daughter LOVES them all), they don't insult the intelligence of the reader (I enjoyed them more than much of the "grown-up" fiction I read). The series is best when you read them in order, as you meet the characters again in future books. The books were such a hit, I actully purchased them in hardback so that the rest of my kids can read & re-read them as they get older. We're eagerly waiting for the 7th book (and the 8th...) to come out in print!"

The feature set F3 for this document included the following features:

- best_anypos: 1 - a positive strongsubj word.
- eagerly_anypos: 1 - a positive strongsubj word.
- fiction_noun: 1 - a negative weaksubj word.
- great_adj: 1 - a positive strongsubj word.
- insult_verb: 1 - a negative strongsubj word.

- intelligence_noun: 1 - a positive weaksubj word.

When classified with refined F3 and F4, refined F3 contained the same features as F3, but both *insult_verb* and *intelligence_noun* got the value -1 (since they were negated). In addition, the features: {enjoy_verb: 1 - a positive weaksubj word} and {love_verb: 1 - a positive strongsubj word} were added in refined F3 due to lemmatization in extracting refined F3. F4 contained the following features (with values greater than 0):

- negation_neg_strongsubj: 1 - for the expression (n't insult)
- negation_pos_weaksubj: 1 - for the expression (n't .. intelligence)
- pos_strongsubj: 4 - for the words (best - eagerly - great - love)
- pos_weaksubj: 1 - for the word (enjoy)
- neg_weaksubj: 3 - for the expressions (fiction - n't insult - n't .. intelligence)

2. **Change of classification from positive to negative:**
The actual classification of the following sample documents is negative. They were classified as positive when running the classifier with F3 (baseline), and as negative when running it with refined F3 and F4.

   (a) Sample document #1:
   "This book doesn't make any sense. It is so, so, so b-o-r-i-n-g. The book is not well written, and the sentences are way too short. I woudn't recommand this book to anyone"

   The feature set F3 for this document included the following features:

   - sense_noun: 1 - a positive weaksubj word.
   - too_anypos: 1 - a negative weaksubj word.
   - well_adverb: 1 - a positive weaksubj word.

   When classified with refined F3 and F4, refined F3 contained the same features as F3, but both *sense_noun* and *well_adverb* got the value -1 (since they were negated). F4 contained the following features (with values greater than 0):

   - negation_pos_weaksubj: 2 - for the expressions (n't .. sense - not well)
   - neg_weaksubj: 3 - for the expressions (n't .. sense - not well - too)

(b) Sample document #2:

"While the author does go into detail about numerous topics and informs the reader about what is necessary to survive. What I think the books lacks is the detail necessary on all of the major topics. An example would be that he describes how to build a leanto for shelter, but doesn't go into detail on how to tie the structural members together. He just tells you that you need to tie them together. What if you don't have a shoelace to use? What other alternates are there? This book would be great for a teenager who goes camping and may need to build a fire, but it's not enough for the serious camper"

The feature set F3 for this document included the following features:

- great_adj: 1 - a positive strongsubj word.
- just_anypos: 1 - a positive weaksubj word.
- necessary_adj: 2 - a positive weaksubj word.
- serious_anypos: 1 - a negative strongsubj word.
- shelter_noun: 1 - a positive weaksubj word.
- survive_verb: 1 - a positive weaksubj word.

When classified with refined F3 and F4, refined F3 contained the same features as F3, but without *necessary_adj* since it occurred twice (and the second one was negated by a polarity shifter *lacks*, so 1 - 1 = 0). Also, *serious_anypos* got the value -1 (since it was negated). F4 contained the following features (with values greater than 0):

- negation_neg_strongsubj: 1 - for the expression (n't .. serious)
- shiftneg_weaksubj: - 1 for the expression (lacks .. necessary)
- pos_strongsubj: 1 - for the word (great)
- pos_weaksubj: 4 - for the expressions (just - necessary - shelter - survive)
- neg_weaksubj: 2 - for the expressions (lacks .. necessary - n't .. serious)