A Sentiment Analysis Model Integrating Multiple Algorithms and Diverse Features

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By
Zhe Xu, B.S.
Graduate Program in Computer Science and Engineering

The Ohio State University
2010

Thesis Committee:
Rajiv Ramnath, Advisor
Mikhail Belkin
Hui Fang
Abstract

In this thesis, we propose a model for integrating multiple sentiment analysis algorithms that each cover separate features, and show that it can do better than single algorithms that deal with multiple features. The key idea behind this integration model is the selective use of the right algorithm for the right case. We propose two measures to estimate the effectiveness of an algorithm, and, based on these measures, a two-step process to construct the model based on the understanding of contextual properties of algorithms. Our experiments show that our model outperforms existing baselines.
Dedication

Dedicated to my parents
Acknowledgments

I would like to thank Rajiv Ramnath and Hui Fang for their support during the development of this thesis and my graduate studies. I also thank Mikhail Belkin for his suggestions on this thesis.
Vita

June 2007 .......................... B.S. Software College,

Xiamen University

September 2007 to present .......... Graduate Student,

Department of Computer Science and

Engineering, The Ohio State University

Publications


Fields of Study

Major Field: Computer Science
Table of Contents

Abstract.................................................................................................................. ii
Dedication............................................................................................................. iii
Acknowledgments............................................................................................... vi
Vita......................................................................................................................... v
List of Tables......................................................................................................... vii
List of Figures....................................................................................................... viii
Chapter 1: Introduction......................................................................................... 1
Chapter 2: Related Work...................................................................................... 11
Chapter 3: Methodology...................................................................................... 27
Chapter 4: Experiments...................................................................................... 38
Chapter 5: Conclusions and Future Work......................................................... 44
References............................................................................................................. 45
List of Tables

Table 1: Accuracy Comparison between Baselines and SV.........................42
Table 2: Performance of SV with Different Coverages...............................42
Table 3: Performance of SV with Different Coverages...............................43
List of Figures

Figure 1: Feature and Algorithm .............................................................. 2
Figure 2: Visualization of SentiWordNet ............................................... 16
Figure 3: Example of SVM .................................................................... 20
Figure 4: 28 Features Used for Neutral-Polar Classification ................. 24
Figure 5: Example of Dependency Parse Tree ...................................... 25
Figure 6: Features for Polarity Classification ........................................ 26
Figure 7: Threshold and Confidence ..................................................... 30
Figure 8: Threshold and Coverage ......................................................... 31
Figure 9: Performance of SV_1 ............................................................. 42
Figure 10: Performance of SV_2 ............................................................. 43
Chapter 1: Introduction

"What other people think" has always been an important factor affecting people’s decision-making process. The advent of the Internet and the exponential growth in opinion rich resources such as review sites, web blogs and online forums, provide users with an opportunity to leverage the opinions of many, at a scale hitherto unachievable. Opinion mining and sentiment analysis are computational techniques that seek to understand opinion and sentiment by analyzing large amount of opinion data efficiently to assist in human decision making. In business, sentiment analysis can help companies analyze customer opinions to improve their products, provide better customer service and even identify new business opportunities. In politics, sentiment analysis can predict shifts in public opinion regarding election candidates. In daily life, people may become better informed with respect to selecting electronic products, movies to watch or books to read, and hence make better buying decisions.

1.1 Sentiment Analysis Techniques

Various sentiment analysis methods have been described in the literature [2][3][4][5][9][12]. We study the sentiment analysis problem from two aspects:
the sentiment analysis feature and the algorithm. We define feature and algorithm as follows.

Definition 1. Sentiment Analysis Feature is a measurable property of documents ready for sentiment analysis.

Definition 2. Sentiment Analysis Algorithm is a method based on sentiment analysis features for deciding the polarity of documents.

Examples of sentiment analysis feature are polarity, frequency or part-of-speech information of terms in documents. Algorithms are built on features. Figure 1 describes the relationship between features and algorithms. The nodes within the left column indicates various features while the right-column nodes represent different algorithms. A line connecting a feature and an algorithm means the algorithm uses that feature in order to do analysis. Note that one feature may be used by multiple algorithms and one algorithm may use multiple features.

![Feature and Algorithm Diagram]

**Figure 1. Feature and Algorithm.**

---

1 When we use the term "documents", we refer to texts such as reviews or comments. We will use "document", "review" and "comment" interchangeably.
We divide features into two types: sentiment-lexicon features and non-sentiment-lexicon features. In algorithms, we study two important types of algorithms: sentiment-lexicon-based algorithms [2][3][4][5][12] and machine-learning-based approaches [9]. Most sentiment analysis algorithms fall into these two types. [15] makes a comparison between them.

Sentiment-lexicon features: These features are obtained from a sentiment lexicon. A sentiment lexicon labels each term in the dictionary with polarity information - usually a scaled numeric number to score how positive or negative a given word is, with the positive score indicating positive sentiment and the negative score suggesting its negative sentiment. We name the positive (negative) score of a term as term positive (negative) score.

Non-sentiment-lexicon features: These features are any feature except sentiment-lexicon features, such as term presence, term frequency and syntax information.

Sentiment-lexicon-based algorithms: Several unsupervised sentiment analysis approaches belong to this track. These algorithms construct functions based on features provided by sentiment lexicon such as term positive (negative) scores to calculate the polarity of the tested review. Usually this function generates another two scores: review positive score - indicating the overall positive degree of the review, and review negative score - indicating the overall negative degree of the review. One simple but commonly used method to decide the polarity of a given review is to compare the average of term positive scores of each word in the review with the
average of term negative scores, with the review being considered positive if
the average term positive score is larger, and vice versa. If the two scores are
close, the review may be considered as neutral in its sentiment. In our work,
we do not consider neutral reviews and take reviews falling into the third case
as negative. This approach to decide a review's polarity is used as one of the
baseline methods in this paper. We call this algorithm Average Sentimental
Value (ASV). Assume a review R has n terms. Let the review positive score
be POS_R and the review negative score be NEG_R. For a term T_i, it has a
positive score POS_{T_i} and a negative score NEG_{T_i}. POS_R and NEG_R are
computed in the following way.

$$POS_R = \frac{1}{n} \sum_{1 \leq i \leq n} POS_{T_i}$$  \hspace{1cm} (1)

$$NEG_R = \frac{1}{n} \sum_{1 \leq i \leq n} NEG_{T_i}$$  \hspace{1cm} (2)

If POS_R is larger than NEG_R, R is positive; otherwise, R is negative.

Machine-learning-based algorithms: Common techniques of this type include
Support Vector Machines [6]. In these methods, training data are documents
manually labelled with sentiment values. Within this thesis, these labels are
either positive or negative. Before [16], machine-learning-based algorithms all
use non-sentiment-lexicon features. [16] takes into account sentiment-lexicon
features along with other features to conduct SVM with promising results.
Existing work presumes that the best sentiment analysis performance is achieved by a single algorithm considering as many as possible features. However, no research is conducted to support this point of view. Given that each algorithm has its own strengths and weaknesses, we believe that a solution integrating all algorithms where each algorithm does not have to cover all features might be better than using a single algorithm covering all features. Assume we have a given review dataset and a given algorithm which covers some given features. Within the dataset there must be some reviews for which the algorithm can successfully identify their sentiment polarities. Meanwhile, there must also be some reviews for which the algorithm fails to identify their sentiment polarities. Our position is that we may heuristically decide which algorithms may do the best analysis and which algorithms may fail, hence choose the right algorithms for the given review. The intuition is to find a set of algorithms that complement each other in performance, such that for any given review, if one algorithm fails to identify the polarity, there is another algorithm which will succeed. Essentially, we speculate that not all features are appropriate for each algorithm. In other words, an algorithm may get satisfying results using some features and get bad results using others.

Specifically, we try to prove the following statement: a model for sentiment analysis that uses appropriate algorithms with appropriate features in appropriate situations is better than using any single algorithm with all features in all situations. How to choose complementing algorithms is one
issue. How to select features for each algorithm is also another issue. In the next subsection we show how to address these issues based on understanding the contextual characteristics of algorithms.

1.2 Context in Sentiment Analysis

One aspect through which sentiment-lexicon-based methods (using sentiment-lexicon features) and machine-learning-based methods (using non-sentiment-lexicon features) may complement each other is their contextual properties.

When we refer to context, we interpret it at two different levels: first at the domain level, and second, at expression level.

Domain context: This is context provided by the subject of the review (such as movies, electronics, and so on). For a given word, the context provided by different domains may assign it different polarities with respect to its sentiment. For example, "unpredictable" is negative when used to describe the stability of an mp3 player. But "unpredictable" may be a positive sentiment for movie plots.

Expression context: The composition of an expression in a given text provides a context for understanding its sentiment. For instance, a negation word such as "not" and "no" can change the polarity of the following word. For example, "not bad" expresses a different sentiment from that of "bad". Excellent work has been done in phrase level context ([8][13]). However, phrase level context investigates words within a short-span proximity, which means that
these words must be close in position. Sentence level is about the long-span proximity ([16]). We will use sentence structure context in this thesis. We now compare sentiment-lexicon-based algorithms using sentiment-lexicon features and machine-learning-based algorithms using non-sentiment-lexicon features with respect to context. We look at sentiment-lexicon-based algorithms first, which use sentiment lexicon features. Sentiment lexicons usually correctly estimate the generic polarity of terms, i.e., in a way that does not take domain information into account. Thus, if the review data contain multiple domains which are randomly distributed across the data, sentiment-lexicon-based methods perform well. However, this type of approach is weak in text containing single domain due to the lack of domain contextual information. Machine-learning-based algorithms with non-sentiment-lexicon features have quite opposite properties compared with sentiment-lexicon-based approaches. Machine learning algorithms "train" on a representative sample of data. Usually the training data is quite dependent on the domain of review data due to the fact that the essence of machine learning is to capture patterns which also cover domain contextual information. Once enough training data is learned by the machine learning algorithm, sentiment analysis on corresponding domain data will have very good results. However, the precondition for success by machine learning algorithms is enough training data. Several techniques are used to get large amount of training data. For instance, some review sites encourage people not only to write reviews but
also rate products (typically in a numerical scale). These ratings can be used as indicators of the real polarity of the corresponding reviews. The mechanism of ratings and reviews is considered intuitively reasonable as training data, since people should have consistent ratings with their own reviews. Further, the extraction of the ratings is not difficult as ratings embedded in html code or in plain-text documents. The whole extraction process is automatic and large amount of training data may be easily obtained using this technique. In our experiment, we also use this approach to generate training data. However, in most situations training data requires the tedious, manual work of labeling documents. In the real practice, only a small set of training data is typically available. However, decreasing the size of training data usually result in a drop in performance. For instance, if a word never appears in training data but appears in testing data, machine learning will not accurately estimate the polarity of that word. Another problem is that training data for certain domain can not be reused for other domains².

Our observation is that if we appropriately integrate both types of algorithms, disadvantages of both methods can be avoided in the unified model while the original advantages remain. The lack of contextual information of sentiment-lexicon-based approaches can be made up by the contextual strength of machine learning algorithms.

1.3 Contributions of this Thesis

² The "unpredictable" example mentioned earlier is a good illustration for this point.
In this thesis we introduce two measures to examine a sentiment analysis algorithm's effectiveness: Sentiment Confidence Indicator and Sentiment Confidence Triple. The Sentiment Confidence Indicator is a variable whose values can provide information on the trust that can be placed in a given algorithm's analysis of a given review. The Sentiment Confidence Triple, which is based on the Sentiment Confidence Indicator, aims to identify the suitable subset of a given review dataset for a given algorithm. That is, the algorithm will have high performance on the selected documents in the subset. Note that these two measures can be used on any sentiment analysis algorithm.

Based on the proposed metrics, we introduce a model which integrates multiple sentiment analysis algorithms. The model can takes in an arbitrary number of algorithms. In this thesis we use three components to construct the model: one from sentiment-lexicon-based methods, one from machine-learning-based methods and another approach capturing sentence structure by extracting frequently appearing substrings (or expressions). The purpose of the third algorithm is to enhance the overall strength of the model. The process is composed of two phases: selection and voting. For selection the appropriate algorithms for analysis are selected while voting step aims to resolve conflicts in the selection phase. Experiment results show our model has significant improvement over other baselines, including a machine-learning-based algorithm which considers both sentiment-lexicon and non-sentiment-lexicon features.
The rest of the thesis is organized as follows. Section 2 describes related work. Section 3 is a detailed illustration of our methodology. In section 4, we describe and discuss the experiments conducted and their results. Finally, we conclude and describe future work in section 5.
Chapter 2: Related Work

Sentiment-lexicon-based methods to sentiment analysis: Sentiment lexicons are usually built without supervision ([2][3][4][5][12]). Polarity mining may be turned into a clustering problem on two words linked by conjunctions such as "but" and "and" [5]. The use of mutual information and co-occurrence with a small set of seeds is described in [12]. WordNet-defined relations may also be used to build a sentiment lexicon [3][2][4]. SentiWordNet [3] associates three numerical scores to describe how positive, negative and objective a term in WordNet [11] is. Quantitative analysis of glosses associated to synsets and semi-supervised synset classification are then used to develop SentiWordNet.

Machine-learning methods for sentiment analysis: Machine learning methods for sentiment analysis are compared in [9]. In this work, the Support Vector Machine (SVM) approach has been highlighted as the one with the best performance. Before [27], machine-learning-based algorithms all use non-sentiment-lexicon features. [27] combined sentiment-lexicon features with others in one algorithm, with promising results. In contrast to the above work, we introduce a model that integrates multiple algorithms, that individually use sentiment-lexicon or non-sentiment-lexicon features, by understanding contextual information.
**Context in sentiment analysis:** Information about the context is an important factor in understanding opinion. The change in base attitudinal valence of a lexical item due to lexical and discourse context is studied in [8], which also proposes an implementation to deal with a set of contextual shifters. Contextual polarity at phrase level is studied in [13] which describes the use of a classifier that utilizes 28 features to do neutral-polar classification. [16] studies sentiment analysis by using sentence structure. We also use sentence structure in one of the components in our model.

We review several representing techniques in the following subsections based on three topics: sentiment-dictionary-based methods, machine-learning-based methods and context-based sentiment analysis.

2.1 Sentiment-dictionary-based Methods

One basic characteristics of these approaches is to do sentiment orientation on dictionary words first. Previous study explored the use of mutual information and co-occurrence. [12] is a representative work based on this technique. Another aspect people would like to research is utilizing WordNet. SentiWordNet [3] and Pageranking WordNet [4] are works in this area.

2.1.1 Semantic Orientation based on Mutual Information and co-occurrence

Torney’s work [12] introduces an unsupervised learning algorithm for classifying a review as positive or negative. Three steps are conducted. First, phrases in data that contain adjectives or adverbs are identified by using a part-of-speech tagger. Second, each extracted phrase is labeled with
semantic orientation. And third, the given review is classified as positive or negative (which Turney calls "recommended" or "not recommended") based on the average semantic orientation of phrases in that review.

The reason to extract phrases containing adjectives and adverbs is that adjectives can indicate subjective and evaluative sentences. Using a phrase instead of a single word can help identify context to improve semantic orientation. For instance, "unpredictable" might be negative in certain domains, but "unpredictable plots" should be positive in the movie domain.

The method of extracting phrases is based on utilizing part-of-speech patterns. For example, two adjacent words are extracted if the first word is an adverb and the second word is an adjective, while the third word should not be a noun.

The most interesting procedure is the second step where the semantic orientation of phrases is labeled. An algorithm PMI-IR is used here. PMI-IR uses the measure PMI which indicates Pointwise Mutual Information. It is a measure of statistical dependence between words and defined as follows [15].

\[
\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \left( \frac{p(\text{word}_1 \& \text{word}_2)}{p(\text{word}_1) p(\text{word}_2)} \right)
\]

(3)

\(p(\text{word}_1 \& \text{word}_2)\) is the probability that \(\text{word}_1\) and \(\text{word}_2\) co-occur.
The SO (Semantic Orientation) of a phrase based on PMI is defined as follows.

\[
SO(phrase) = PMI(phrase, \text{“excellent”}) - PMI(phrase, \text{“poor”})
\]  

(4)

PMI-IR combines PMI with information retrieval techniques. PMI-IR computes PMI by issuing queries to a search engine and recording the number of hits. The Semantic Orientation of a phrase based on PMI-IR is defined as follows.

\[
SO(phrase) = \log_2 \left( \frac{\text{hits}(phrase \text{ NEAR “excellent”})}{\text{hits}(phrase \text{ NEAR “poor”})} \right)
\]  

(5)

Once the SO of each phrase is calculated, the polarity for a given review is computed as the average SO of phrases in that review.

Turney conducted experiments on data which covered automobiles, banks, movies and travel destinations, with an accuracy ranging from 66% to 84%.

2.1.2 SentiWordNet

SentiWordNet is a sentiment dictionary that assigns each WordNet synset \( s \) with three numerical scores \( \text{Obj}(s) \), \( \text{Pos}(s) \) and \( \text{Neg}(s) \) to describe how
objective, positive and negative the terms contained in the synset are. Each of the three scores ranges from 0 to 1, with the following relationship:

\[ \text{Obj}(s) + \text{Pos}(s) + \text{Neg}(s) = 1 \quad (6) \]

One benefit of considering the sentiment of synsets instead of terms is that the various senses of the same term may have different polarities. SentiWordNet relies on [16][17] to train a set of ternary classifiers. Each ternary classifier is generated by semi-supervised methods. Separately, each classifier can determine a given synset's polarity: whether it is positive, or negative, or objective. These three classifiers have different training sets and learning devices, thus generating different classification results on synsets. Polarity scores of a synset are decided by the proportion of ternary classifiers that assigns corresponding label to it. If all classifiers agree on the same label for a given synset, that label will have the maximum score. Otherwise, each label will have a score proportional to the number of classifiers that assign that label.

[3] also has an interesting visualization of SentiWordNet as follows (Figure 2). \( \text{Obj}(s) \), \( \text{Pos}(s) \) and \( \text{Neg}(s) \) are displayed in a triangle. The vertices of the triangle indicate the maximum possible values of \( \text{Obj}(s) \), \( \text{Pos}(s) \) and \( \text{Neg}(s) \).
Figure 2. Visualization of SentiWordNet

2.1.3 Pageranking WordNet

Pagerank algorithm[18] is a random walk model for ranking web search results made famous due to the success of Google. Pageranking WordNet’s idea is based on this. We brief an introduction to Pagerank algorithm and then describe how this technique is used to do polarity mining.
Let a directed graph be $G = <N, L>$, where $N$ is the set of nodes while $L$ is the set of links in that graph. $W_0$ is a $|N| \times |N|$ adjacency matrix of $G$ such that $W_0[i, j] = 1$ if and only if there is a link from node $n_i$ to node $n_j$. Let $F(i) = \{n_j | W_0[i, j] = 1\}$ be the forward neighbours of $n_i$. $W$, the row-normalized adjacency matrix and defined as follows: $W[i, j] = 1/|F(i)|$ if $W_0[i, j] = 1$, otherwise $W[i, j] = 0$. $W$ is the input to PageRank and the output is a vector $a = <a_1, ..., a_{|N|}>$ in which $a_i$ is a score indicating the authoritativeness of a web page, which corresponds to $n_i$. The following formula determines how to compute $a$.

$$a_i^{(k)} \leftarrow \alpha \sum_{j \in R(i)} \frac{a_j^{(k-1)}}{|F(j)|} + (1 - \alpha)e_i$$

(7)

$a_i(k)$ indicates the value of $a_i$ at the $k$-th iteration. $\alpha$ is a parameter and $0 \leq \alpha \leq 1$. $e$ is constant such that $\sum_i e_i^{[N]} = 1$.

PageRank WordNet takes glosses in WordNet as a graph of terms. The principle behind the method is that the positivity and negativity of WordNet synsets can be decided by analyzing the relationship among glosses. The hypothesis is that the glosses of positive (negative) synsets are more likely to contain terms that belong to positive (negative) synsets. Specifically, the binary relation $s_i \Rightarrow s_j$ (the gloss of synset $s_i$ contains at least a term appearing in synset $s_j$) is as directed link in the graph of WordNet synsets. Each synset is as a node in the graph. One adaption is now we have two
different vectors of \( a \): one for positive scores while the other for negative scores. The whole algorithm is conducted in the following steps: Let \( N \) be the total synsets in WordNet. \( L \) is all relations which satisfy \( s_i \Rightarrow s_j \) as described before. Remove "self-loops" links which go from a synset to itself. And generate \( W \) for \( G \). They conducted several versions of \( e \) and \( \alpha \). Then iterations start until a predefined termination condition is satisfied. Specifically, the condition is the cosine of the angle between \( a^{(k)} \) and \( a^{(k+1)} \) is above a threshold. Finally, all synsets are ranked according to a score. This process was run twice: one for positive scores while the other for negative scores.

In this work, it is interesting to note the relationship between the assignment of \( e \) and performance (\( p \)-normalized Kendall tau distance [18] is used). They tested five different settings of \( e \). \( e_1: 1/|N| \); \( e_2: \) uniform non-null \( e_i \) scores assigned to the adjective "good" ("bad") synsets and null scores for other synsets. \( e_3: \) uniform non-null \( e_i \) scores assigned to the synsets containing at least one of the "paradigmatic" adjectives ("good", "nice", "excellent", "positive", "fortunate", "correct", "superior" for positive adjectives and "bad", "nasty", "poor", "negative", "unfortunate", "wrong", "inferior" for negative adjectives) and null for other synsets. \( e_4: \) scores proportional to the positivity (negativity) scores assigned to that synset in SentiWordNet 1.0. \( e_5: \) similar to \( e_4 \) except using SentiWordNet 1.1. \( e_4 \) shows to have the best performance.

2.2 Machine-learning-based Methods
Traditional machine learning approaches still have their strength in sentiment analysis. Here we introduce two commonly used techniques: Naive Bayes and Support Vector Machine.

2.2.1 Naive Bayes

A Naive Bayes classifier [25] [26] is a probabilistic model based on Bayes' theorem. Bayes' theorem is formulated as follows.

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{8}
\]

P(A) is the prior probability of event A and P(A|B) is the conditional probability of A for given B.

When applying Naive Bayes classifier to text classification, the problem becomes for a given document d, its corresponding class c is the class which has biggest P(c | d). In the context of sentiment analysis, there are two types of classes: positive class and negative class. Specifically,

\[
P(c | d) = \frac{P(c)P(d | c)}{P(d)} \tag{9}
\]
Let \( d \) contain \( m \) features \( f_1, f_2, \ldots, f_m \). \( n_i(d) \) indicates the number of times of \( f_i \) appearing in \( d \). We assume that all features are independent. Then we can have:

\[
P_{NB}(c \mid d) := \frac{P(c) \left( \prod_{i=1}^{m} P(f_i \mid c)^{n_i(d)} \right)}{P(d)}
\]

(10)

\( P(c) \) and \( P(f_i \mid c) \) can be estimated by frequency. It does not matter what \( P(d) \) is in deciding the class of \( d \).

2.2.2 Support Vector Machine

The essence of Support Vector Machine (SVM) is to find a hyperplane that separates document vectors from one class to the other as much as possible. In other words the separation should be as large as possible. Figure 3 from [19] is an example of margin in a 2-dimensional space with support vectors marked grey.

![Figure 3. Example of SVM](image_url)
Let \( x \) be the vector of features and \( y \) be the corresponding class label. In sentiment analysis we have two classes: positive and negative. So \( y \in \{-1, 1\} \).

There exists a weight vector \( w \) and a scalar such that

\[
\begin{align*}
    w \cdot x_i + b & \geq 1 \quad \text{if} \quad y_i = 1 , \\
    w \cdot x_i + b & \leq -1 \quad \text{if} \quad y_i = -1 ,
\end{align*}
\]

(11)

So we can get

\[ y_i (w \cdot x_i + b) \geq 1 \quad (12) \]

Then we can get the optimal hyperplane as

\[ w_0 \cdot x + b_0 = 0 \quad (13) \]

Specifically,

\[
\bar{w} := \sum_j \alpha_j c_j d_j, \quad \alpha_j \geq 0,
\]

(14)

\( \alpha \) can be computed by solving a dual optimization problem.
2.3 Context-based Sentiment Analysis

Context is important in opinion mining and can be leveraged to improve performance. [8] investigates various aspects of contextual influence. [13] explores contextual polarity at the phrase level.

2.3.1 Contextual Valence Shifter

[8] argues that an author's attitude cannot be simply decided by individual terms. They discussed about two kinds of shifters: sentence-based contextual valence shifters and discourse-based contextual valence shifters.

Sentence-based contextual valence shifters includes negatives, intensifiers, modals, presuppositional items and irony. Negatives (such as "not", "never", "none"...) can flip the polarities of terms following them. For instance, "brilliant" is positive while "not brilliant" is negative. Intensifiers (such as "rather" and "deeply") can weaken or strengthen the original polarity of terms modified. Examples could be like "deeply sad" has more negative meaning than that of "sad". Modal operators (such as "could" and "ought to") provide a context of possibility that actually does not express the real attitude of the author. A sentence "If they lose this game, that will be a disaster" which does not express a negative meaning since "if" only indicates a possibility and the game may not actually be lost. Presupposition items (such as "barely") can also shift term polarity. "sufficient" is positive while "barely sufficient" is negative. Irony is common in daily life. For instance, "this outstanding student
cannot answer the simple question" might have the positive term "outstanding". But it indicates a negative polarity.

With regard to discourse-based contextual valence shifters, the authors explored connectors, discourse structure, multi-entity evaluation, genre and genre constraints, reported speech and subtopics. Connectors (such as "although", "but" ... ) can introduce new information or mitigate the force of information. For instance, in "Although they are an excellent team, they failed in that game", the focus of the sentence does not lie on "excellent" but "failed".

Discourse structure includes lists and elaborations. Sentiment scores may be modified by term positions in a hierarchical discourse structure. Multi-entity evaluation investigates multiple aspects of an discussed object. For instance, if a reviewer has negative attitude on only one feature of a product, but very satisfied with other features of that product, the overall sentiment is positive. Genre can have complex relationship with term polarities to impact the author attitude. Genre constraints can also impact on deciding the author attitude. For instance, text segments at the beginning or the end of a movie review usually have more weights than those of other parts in sentiment analysis. Reported speech could be some example like "Mike told me the student was stupid". In fact the student may be smart rather than stupid. But if the sentence becomes "Mike told me the student was stupid and that is a truth", then we can confirm that the new sentence has a negative sentiment. Subtopics are similar with multi-entity evaluations. A review could be
composed of multiple subtopics and different subtopics might have different sentiment scores.

2.3.2 Contextual Polarity in Phrase Level

[13] proposed an approach to identify the contextual polarity of expressions, using a neutral-polar classifier to determine whether a given expression is neutral or polar in context. If an expression is polar, a polarity classifier will be used to decide whether the expression is positive, negative or both.

For neutral-polar classification, 28 features consisting of 5 types (word, modification, sentence, structure and document) were used, as listed in Figure 4 from [13]. For example, one of modification features is checking whether a word is preceded by an adjective.

<table>
<thead>
<tr>
<th>Word Features</th>
<th>Sentence Features</th>
<th>Structure Features</th>
<th>Document Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>word token</td>
<td>strongsubj clues in current sentence: count</td>
<td>in subject: binary</td>
<td>document topic</td>
</tr>
<tr>
<td>word part-of-speech</td>
<td>strongsubj clues in previous sentence: count</td>
<td>in copular: binary</td>
<td></td>
</tr>
<tr>
<td>word context</td>
<td>strongsubj clues in next sentence: count</td>
<td>in passive: binary</td>
<td></td>
</tr>
<tr>
<td>prior polarity: positive, negative, both, neutral reliability class: strongsubj or weaksubj</td>
<td>weaksubj clues in current sentence: count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>weaksubj clues in previous sentence: count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>weaksubj clues in next sentence: count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>adjectives in sentence: count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>adverbs in sentence (other than not): count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cardinal number in sentence: binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pronoun in sentence: binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>modal in sentence (other than will): binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modified by strongsubj: binary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modified by weaksubj: binary</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. 28 Features Used for Neutral-Polar Classification

In modification features, dependency parse tree [23] [24] for a given sentence is used. Each node in the tree is a term while the edges between nodes are
grammatical relationship. [13] gives an example tree (Figure 5) for the sentence "The human rights report poses a substantial challenge to the US interpretation of good and evil".

![Example of Dependency Parse Tree](image)

Figure 5. Example of Dependency Parse Tree

The polarity classifier uses another set of features to decide whether polar expression is positive, negative or both. Figure 6 is the list of features used by the polarity classifier.
<table>
<thead>
<tr>
<th>Word Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>word token</td>
<td></td>
</tr>
<tr>
<td>word prior polarity: positive, negative, both, neutral</td>
<td></td>
</tr>
<tr>
<td>Polarity Features</td>
<td></td>
</tr>
<tr>
<td>negated: binary</td>
<td></td>
</tr>
<tr>
<td>negated subject: binary</td>
<td></td>
</tr>
<tr>
<td>modifies polarity: positive, negative, neutral, both, notmod</td>
<td></td>
</tr>
<tr>
<td>modified by polarity: positive, negative, neutral, both, notmod</td>
<td></td>
</tr>
<tr>
<td>conj polarity: positive, negative, neutral, both, notmod</td>
<td></td>
</tr>
<tr>
<td>general polarity shifter: binary</td>
<td></td>
</tr>
<tr>
<td>negative polarity shifter: binary</td>
<td></td>
</tr>
<tr>
<td>positive polarity shifter: binary</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6. Features for Polarity Classification

Their experiment results show their approach has significant performance improvement over baselines. However, their work focuses on contextual polarity of expressions instead of complete documents. And the recognition of expressions is manual.
Chapter 3: Methodology

3.1 Sentiment Confidence Indicator

We define Sentiment Confidence Indicator as follows:

Definition 3. The Sentiment Confidence Indicator (SCI) of a given algorithm on a given review is a measure of the trust than can be placed on the analysis of the review for the algorithm.

We now consider how to design SCI. When an algorithm can retrieve both the review positive score and the review negative score, we define SCI, for a given algorithm A and a given review R, as:

\[
SCI(A, R) = \text{Absolute}(POS_R - NEG_R) \tag{15}
\]

Thus, for a given review and an algorithm, the larger the SCI value indicates greater trust on the result.

Let us look at an example. Assume we have an sentiment analysis algorithm A_1 and two reviews R_1 and R_2. Based on A_1 we get \(POS_{R1} = 0.6\), \(NEG_{R1} = 0.1\), \(POS_{R2} = 0.8\) and \(NEG_{R2} = 0.9\). SCI(A_1, R_1) = Absolute (0.6 - 0.1) = 0.5. SCI(A_1, R_1) = Absolute (0.8 - 0.9) = 0.1. Using the same algorithm A_1, the result of R_1 is more reliable as it has a larger SCI.
The intuition behind SCI is that when an algorithm is "confident" about its result, the result will either be a large review positive score and a small review negative score or a small review positive score and a large review negative score. This statement is proved to be correct on our experiment data in the next section which we will show later. Note that when an algorithm is confident with an analysis result, it does not mean the result is correct. For the above example, \( R_1 \) should be positive (as it has a larger positive score) and \( R_2 \) should be negative. Based on their SCIs, we can only say that the probability that \( R_1 \) is actually positive is larger than that of \( R_2 \) to be negative. However, it is possible that both \( R_1 \) and \( R_2 \) are actually negative, and the algorithm is incorrect in the analysis.

Certain algorithms do not provide positive and negative scores. For example, in training data Support Vector Machine (SVM) gives each positive review a numeric label "+1", and each negative review a label "-1". In testing data, a decision function value (also numeric) is computed for each review. If the value is positive, the document is labelled as positive; Otherwise, the document is labelled as negative. In this case, we define SCI as:

\[
SCI(A,R) = \text{ABS}(\text{DecisionFunctionValue})
\]  
(16)
3.2 Sentiment Confidence Triple

Sentiment Confidence Indicator (SCI) is the foundation of Sentiment Confidence Triple (SCT). SCT studies the relation between an algorithm and a review dataset, and aims to identify the right space in a dataset for the algorithm - meaning that the algorithm performs well on documents within that space. Formally, we define SCT as follows:

Definition 4. Sentiment Confidence Triple (SCT) is a triple in the form of 
<Threshold, Confidence, Coverage> and is a measure of the relationship between an algorithm and a dataset.

Threshold, confidence and coverage are the three elements of SCT. The threshold is a value between the minimum SCI in the dataset and the maximum SCI in the dataset. Once the threshold is fixed, only reviews which have a SCI no smaller than the threshold are selected. We call these reviews the analysis subset for the algorithm. Confidence is the number of reviews within the analysis subset correctly identified by the algorithm divided by the total number of reviews in the subset. Confidence is a measure of the accuracy of the algorithm in the subset. Coverage is the number of reviews in the analysis subset divided by the total number of reviews in the complete review dataset. Coverage indicates how many reviews are covered for a given threshold. Obviously, threshold decides confidence and coverage.

For example, suppose we have a dataset which has 200 reviews. For a given algorithm, let us assume the minimum SCI in the dataset is 0.01 while maximum SCI is 0.85. Assume we set a threshold of 0.3. Then we select
reviews which have SCIs no smaller than 0.3 and form the analysis subset. Let the number of reviews in the subset be 100. Among these 100 reviews, let 66 of them be correctly identified. The confidence is $66/100 = 66\%$. The coverage is $100/200 = 50\%$ and the SCT is $<0.3, 66\%, 50\%>$. In order to empirically study the relationship between threshold, confidence and coverage, we used five datasets (the details of the data are described the experiment section 4.1). For each dataset, we ran six different baselines (described in the experiment section 4.2). For each algorithm, we plotted confidence v.s. coverage. All combination of datasets and algorithms showed the same tendency in the relationship between threshold and confidence (Figure 7), and the relationship between threshold and coverage (Figure 8).

![Graph](image.png)

Figure 7. Threshold and Confidence
Figure 8. Threshold and Coverage

One observation on Figure 7 is that when threshold increases, confidence increase. When threshold is large, documents in the subset tends to have a high accuracy which is presented as confidence. It means that the SCIs of these document have large values since their SCI should be no smaller than the threshold which is large. Thus, it verify the statement that when a review has a large SCI, the analysis results is more reliable.

Another observation is that as threshold increases, coverage decreases. Thus, reviews that satisfy the large threshold tend to have good performance, but at the cost of coverage. However, we need both large confidence and large coverage. Our approach to address this problem is using multiple algorithms, where each algorithm covers a different subset with minimal overlap, thus enlarge the overall coverage while retaining confidence. Our principle for algorithm selection is to look for algorithms that have different advantages which can complement each other.
Issues still remains, however. As mentioned earlier, if a review is selected as part of the subset by only one algorithm, we let the algorithm decide the polarity of the review. What if the review is selected by multiple algorithms? How should we decide the polarity? If a review is not selected by any algorithm, what is its polarity? The Selection Voting (SV) framework introduced in the following subsection addresses these issues.

3.3 Selection Voting (SV) Model

The framework integrates multiple sentiment analysis algorithms. The basic idea behind this framework is that for each review we identify the best individual algorithm for that review. This step is done based on the SCT. Thus, the first step is to identify the right space in the dataset for each algorithm. If a review is found to be good either on multiple algorithms or on no algorithm, a voting mechanism is used to generate the final result.

During execution two inputs are considered: a review R and a parameter C indicating coverage. C is a constant number within [0, 100%] and is decided manually. The output of the model is a label indicating whether R is positive or negative. The execution process consists of two phases: selection and voting, as follows:

Selection: How the right reviews for each algorithm are selected. Key here is used the management of the relationship between the three elements of SCT. As mentioned earlier, threshold determines confidence and coverage. However, different algorithms have different ranges of threshold, and one
single threshold value as the input parameter across all algorithms is difficult. Coverage is easier to tune as it is always within the range of [0, 100%]. Thus, we let the coverage decide the threshold. For each dataset we first generate figures similar with Figure 8. Once the coverage is fixed, we get multiple corresponding threshold values from the figure describing the relation between threshold and coverage (Figure 8), and choose the minimum threshold. For a given review and a given algorithm, small coverage indicates large threshold and large confidence. Reviews whose SCIs are larger than the large threshold form the analysis subset which is close to the right space of that algorithm. Thus, in this phase, we compute a corresponding threshold for the input coverage parameter for each algorithm. Then a review is examined in the following way: for a given algorithm, if the review's SCI is larger than the threshold of that algorithm, it is selected to be included in the subset of that algorithm.

Voting: This resolves the conflicts among the results of multiple algorithms. After performing the phase of selection, a review falls into one of the following three cases:

• Selected by only one algorithm: this case is simple. It means the algorithm is the best choice for the review, and we simply use the analysis result of the algorithm.

• Selected by multiple algorithms: it means that multiple algorithms are good at the review. We set two counters to indicate the positive votes and negative votes for the review. Among those algorithms which perform well at the
review, if an algorithm tells its result for the review is positive, the review gets one positive vote; otherwise the review gets one negative vote. After all these algorithms have done voting, if the number of positive votes is greater than that of negative votes, the review is assigned a positive label as the output; otherwise, the review is negative. Note that algorithms involved in the voting are those which selected the review in the phase of selection.

- Selected by no algorithm: this review can not find an appropriate algorithm to do sentiment analysis. A voting similar with the second case is conducted here, with all algorithms taking part in the voting procedure.

The SV framework can integrate any set of algorithms. However, the selection of algorithms is quite important. A guiding principle is to choose algorithms which cover as many and as different spaces as possible of the dataset, with the overlap between different spaces as small as possible. Assuming m algorithms, we empirically pose that the value of the coverage should fall in [0, 1/m]. We justify this statement in our experiment.

Voting on multiple algorithms is quite common in machine learning [18]. Our framework differs from other voting mechanism by selection. In the experiment we demonstrate our model performs better than simply voting on multiple algorithms.

In this thesis we choose three components to form the model - ASV using SentiWordNet, Support Vector Machine using term presence and Common-Substring-based Sentiment Analysis. As there are two ways to implement
CSSA, there are actually two implementations of SV in our experiments. We will describe the components.

3.3.1 SentiWordNet

WordNet [11] is an electronic dictionary utilized in the generation of sentiment lexicons. Words have similar meaning are grouped together as synsets in WordNet. SentiWordNet [3] is one of the state of the art sentiment lexicons. It uses a set of ternary classifiers to assign each WordNet synset S with three numerical scores OBJ_S, POS_S and NEG_S to describe how objective, positive and negative the terms contained in the synset are. We discard OBJ_S since it does not affect POS_S and NEG_S. If a term T is contained in S, the term positive score POS_T is POS_S and the term negative score NEG_T is NEG_S. ASV is used to decide the polarity of a review as described in section 1.1. This method is used as a component of the SV model. It is also one of the baselines in the experiment.

3.3.2 Support Vector Machine

Support Vector Machine (SVM) has proved to be quite effective in sentiment analysis [9]. The essence of SVM is to find a hyperplane that separates document vectors from one class to the other as much as possible. Here the two classes are positive class and negative class separately. Based on the experiment in [9], SVM using term presence in unigram as the feature has the best performance. That is the same way we use SVM as a component of the
SV model. This method is also used as one of the baselines as individual algorithm in the experiment.

3.3.2 Common-Substring-based Sentiment Analysis (CSSA)

Using word subsequences is not a novel idea ([17]). However, CSSA is easier to implement. CSSA consists of two steps, as described below.

In the first step, we begin with a training dataset where each document labeled as positive or negative. From this training set, our algorithm pair-wise compares every two sentences, and extracts the common expressions (sets of common words) in both sentences. We remove stop words (such as "a", "the" etc.) but keep negation words (such as "not"). Obviously the length of each expression should be at least two; otherwise it is a term rather than expression.

In the second step, we use two different ways to decide the review polarity. The first is called CSSA_ASV (CSSA using Average Sentimental Value). We compute polarity scores for expressions extracted from the first step. For an expression E, let the number of E appearing in positive training reviews be PT_E and the number of E appearing in negative training reviews be NT_E. The positive score of E is POS_E and negative score of E is NEG_E.

\[
POS_E = \frac{PT_E}{PT_E + NT_E} \quad (17)
\]

\[
NEG_E = \frac{NT_E}{PT_E + NT_E} \quad (18)
\]
For each of the given reviews in the test dataset we extract all expressions appearing in that review. Then, the average of positive scores of expressions is the positive score for the review and the average of negative scores of expressions is the negative score for the review. If the review positive score is larger than the review negative score, the review is positive; otherwise, it is negative.

The other method is called CSSA_Unigram. We use an SVM implementation which takes the presence of expressions as the features. These two methods are implemented in separated SV models. They are also used as baselines. It is worth of noting that the introduction of CSSA is not to use it individually, but to enhance the overall performance of the SV model.
Chapter 4: Experiments

4.1 Data

For our experiments, we used five different datasets which cover five domains: book, DVD, MP3 player ("MP3" for short), video game ("game" for short) and restaurant. The first four datasets were randomly extracted from reviews in amazon.com. The restaurant dataset is from [10]. All datasets are rated by users using the same rating system (a scale of 1-5 star(s)). If a review received an overall rating equal to or larger than 4, we considered it a positive rating. If a review received a rating equal to or smaller than 2, we considered it negative. We excluded neutral comments from our experiment. For each domain except the restaurant domain we have 200 positive reviews and 200 negative reviews. Our restaurant data (which was larger) had 400 positive and 400 negative reviews.

4.2 Experimental Designs and Results

The purpose of the experiments is to validate the following two statements: (1) The SV model can outperform other existing methods, especially better than machine learning methods with diverse information sources. (2) Two separate experiments were conducted to support the above statements.
The first experiment is comparing the performance of six baseline methods with those of SV models. All algorithms involved in the experiment are described as follows.

- **SentiWordNet_ASV:** SentiWordNet using average sentimental value as described in 3.3.1. Baseline.
- **CSSA_ASV:** CSSA using average sentimental value as described in 3.3.3. Baseline.
- **CSSA_Unigram:** CSSA using unigram as described in 3.3.3. Baseline.
- **SVM (Term_Unigram):** SVM using presence of term unigram as the feature as described in 3.3.2. Baseline.
- **Hybrid_SVM_1:** SVM using presence of term unigram, SentiWordNet average sentimental value and CSSA average average sentimental value as its features. It represents machine learning using diverse information sources. Baseline.
- **Hybrid_SVM_2:** SVM using presence of term unigram, SentiWordNet average sentimental value and CSSA unigram as its features. Similar with Hybrid_SVM_1, it also represents machine learning using diverse information sources. Baseline.
- **SV_1:** SV model using presence term unigram, SentiWordNet average sentimental value and CSSA average average sentimental value as features. As this method using the same information sources with Hybrid_SVM_1, we will compare it with Hybrid_SVM_1.
• SV_2: SV model using presence of term unigram, SentiWordNet average sentimental value and CSSA unigram. We will compare it with Hybrid_SVM_2.

We used the toolkit SVM light [6] to implement SVM with all parameters set to default values. SV has only one parameter coverage C that needs to be assigned manually. C falls in [0, 100%]. We then used 5-fold cross-validation to remove sample bias. That is, the data set was randomly divided into 5 samples with equal sizes of both positive and negative reviews. One of the samples was used as training data (if the algorithm needed training data) while the others became test data. We then repeated this process for each sample and used the average accuracy as the performance measure. Table 1 is an accuracy comparison between baseline and SV models. Both SV models are tuned to have the best performances (the corresponding C is also presented). For each SV, we also present its improvement over two methods. The first improvement row shows the method which has the best performance among baselines. The second improvement row shows the comparison between corresponding SVM implementations. Thus, Hybrid SVM (specifically, SV_1) is compared with Hybrid_SVM_1 and SV_2 is compared with Hybrid_SVM_2. The table shows that SV_1 outperforms the other methods in all data sets and its improvement is significant. Although SV_2 is not as good as SV_1, it is also better than all baselines. We can see in 90% cases, SV model gets its best performance when C has value smaller than 33% (within the empirical value range for coverage).
As we stated earlier, using as many as possible features may get worse performance than that of using appropriate features. This can be justified by comparing SVM (Term_Unigram) and Hybrid_SVM_2. In more than half cases, Hybrid_SVM_2 is worse than SVM (Term_Unigram).

Let us look into more details about how C can affect the SV performance. Figure 4 and 5 show the performance of SV_1 and SV_2 with various coverage. Table 2 presents the best/worst performance when C falls in [0, 33%] and [0, 100%]. It also presents the performance when C is zero. We call the case when C is zero "simple voting" since in this case no selection happens and all algorithms within SV take part in voting process. As we mentioned earlier, if C falls in [0, 33%], 90% cases can achieve the same best performance when C falls in [0, 100%]. Even in the rest 10% cases (SV_2, restaurant), the best performance when C falls in [0, 33%] is quite close with that when C falls in [0, 100%]. This empirical evidence appears to justify the statement that if a SV model has m algorithms as its components, the coverage with best performance falls in [0, 1/m]. Another observation is that simple voting has a performance quite close to the worst performance when C falls in the empirical range [0, 33%]. In 70% cases they are the same. In the rest 30% cases, the difference of accuracy is no larger than 0.004 (which is quite small). This phenomenon demonstrates that the selection phase in the SV model is important. If C is assigned an empirical value, the corresponding performance usually does much better than that of simple voting.
Table 1. Accuracy Comparison between Baselines and SV

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>DVD</th>
<th>MP3</th>
<th>Game</th>
<th>Restaurant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet _ASV</td>
<td>0.578</td>
<td>0.629</td>
<td>0.653</td>
<td>0.625</td>
<td>0.723</td>
</tr>
<tr>
<td>CSSA_ASV</td>
<td>0.590</td>
<td>0.586</td>
<td>0.614</td>
<td>0.589</td>
<td>0.689</td>
</tr>
<tr>
<td>CSSA_Unigram</td>
<td>0.571</td>
<td>0.570</td>
<td>0.595</td>
<td>0.571</td>
<td>0.670</td>
</tr>
<tr>
<td>SVM(Term_Unigram)</td>
<td>0.643</td>
<td>0.609</td>
<td>0.660</td>
<td>0.584</td>
<td>0.741</td>
</tr>
<tr>
<td>Hybrid_SVM_1</td>
<td>0.655</td>
<td>0.624</td>
<td>0.689</td>
<td>0.601</td>
<td>0.770</td>
</tr>
<tr>
<td>Hybrid_SVM_2</td>
<td>0.639</td>
<td>0.613</td>
<td>0.642</td>
<td>0.583</td>
<td>0.748</td>
</tr>
<tr>
<td><strong>SV_1</strong></td>
<td><strong>0.678</strong> (+3.5%/+5.5%)</td>
<td><strong>0.678</strong> (+8.0%/+8.7%)</td>
<td><strong>0.725</strong> (+5.2%/+5.2%)</td>
<td><strong>0.668</strong> (+6.9%/+11.1%)</td>
<td><strong>0.783</strong> (+1.7%/+1.7%)</td>
</tr>
<tr>
<td>SV_2</td>
<td>0.655</td>
<td>0.674</td>
<td>0.693</td>
<td>0.663</td>
<td>0.778</td>
</tr>
</tbody>
</table>

Table 2. Performance of SV with Different Coverages

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>DVD</th>
<th>MP3</th>
<th>Game</th>
<th>Restaurant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV_1</td>
<td>Best (0% &lt;= C &lt;= 100%)</td>
<td>0.678</td>
<td>0.678</td>
<td>0.725</td>
<td>0.668</td>
</tr>
<tr>
<td></td>
<td>Best (0% &lt;= C &lt;= 33%)</td>
<td>0.678</td>
<td>0.678</td>
<td>0.725</td>
<td>0.668</td>
</tr>
<tr>
<td></td>
<td>Worst (0% &lt;= C &lt;= 100%)</td>
<td>0.637</td>
<td>0.656</td>
<td>0.686</td>
<td>0.638</td>
</tr>
<tr>
<td></td>
<td>Worst (0% &lt;= C &lt;= 33%)</td>
<td>0.660</td>
<td>0.664</td>
<td>0.696</td>
<td>0.648</td>
</tr>
<tr>
<td></td>
<td>Simple Voting (C = 0%)</td>
<td>0.662</td>
<td>0.668</td>
<td>0.696</td>
<td>0.648</td>
</tr>
<tr>
<td>SV_2</td>
<td>Best (0% &lt;= C &lt;= 100%)</td>
<td>0.655</td>
<td>0.674</td>
<td>0.693</td>
<td>0.663</td>
</tr>
<tr>
<td></td>
<td>Best (0% &lt;= C &lt;= 33%)</td>
<td>0.655</td>
<td>0.674</td>
<td>0.693</td>
<td>0.663</td>
</tr>
<tr>
<td></td>
<td>Worst (0% &lt;= C &lt;= 100%)</td>
<td>0.620</td>
<td>0.645</td>
<td>0.650</td>
<td>0.613</td>
</tr>
<tr>
<td></td>
<td>Worst (0% &lt;= C &lt;= 33%)</td>
<td>0.638</td>
<td>0.651</td>
<td>0.672</td>
<td>0.626</td>
</tr>
<tr>
<td></td>
<td>Simple Voting (C = 0%)</td>
<td>0.638</td>
<td>0.651</td>
<td>0.672</td>
<td>0.626</td>
</tr>
</tbody>
</table>

Figure 9. Performance of SV_1
In order to further show the potential of SV model, we conducted a second experiment. In this experiment, we selected two baselines, SentiWordNet_ASV and SVM (Term_Unigram). We define an notion Optimal SV as one that can always choose the right algorithm for a given review. That is, if there exists one algorithm which can correctly identify the polarity of the review, then this review will be correctly identified by the SV model. We call this model as Optimal SV. Table 3 shows the comparison between two baselines and the corresponding Optimal SV - which uses these baselines as its components. The improvement of Optimal SV over the baselines is significant, thus indicating the importance of the selection phase.

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>DVD</th>
<th>MP3</th>
<th>Game</th>
<th>Restaurant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentiWordNet_ASV</td>
<td>0.578</td>
<td>0.628</td>
<td>0.653</td>
<td>0.625</td>
<td>0.723</td>
</tr>
<tr>
<td>SVM (Term_Unigram)</td>
<td>0.643</td>
<td>0.609</td>
<td>0.660</td>
<td>0.584</td>
<td>0.741</td>
</tr>
<tr>
<td>OPT Accuracy</td>
<td>0.864</td>
<td>0.845</td>
<td>0.888</td>
<td>0.845</td>
<td>0.879</td>
</tr>
</tbody>
</table>

Table 3. Optimal Accuracy based on SentiWordNet_ASV and SVM (Term_Unigram)
Chapter 5: Conclusions and Future Work

In this paper we introduced two metrics to study the effectiveness of sentiment analysis algorithms. A model integrating multiple algorithms was also proposed and the corresponding performance shown to have significant improvement over existing methods in the experiment. Our planned future research consists of finding more algorithms to fit into the SV model. As the choosing of algorithms for current model is based on the understanding on contextual information, we seek other information to help the integration.
References


