Computing Polarity in Sentiment Analysis Applications

Andrew Latterner
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
latte048@morris.umn.edu

ABSTRACT

As data mining techniques improve, the demand for large amounts of data is ever increasing. While some of this data is strictly numeric, there exists a large amount of textual data. By using natural language processing techniques, parsing sentiment and emotion from data is a reasonable task. However, when introducing complex elements to text, parsing sentiment from text accurately becomes a difficult process. In this paper, I will describe sentiment modifiers and why they are difficult to parse using algorithms. I will also describe a lexicon-based algorithm and a classifier-based algorithm that attempt to handle modifiers in an accurate way. Following the introduction of these algorithms and the process required to use them, the results of applying these algorithms to real-world data is described. While comparing the two algorithms is difficult given that they are being applied to different data, an analysis on similar algorithms being applied to equivalent data is performed and analyzed.

Keywords

Sentiment Analysis, Machine Learning, Natural Language Processing

1. INTRODUCTION

As modern language has developed, human beings have found a multitude of ways to express and understand sentiment, emotion, and opinions through written word. A portion of this expression can be found in the context of which the writer has written their thoughts. Another portion of this sentiment can be explained through the words that the writer uses to describe a topic. These words can be formulated in varying sentence structure complexity depending on the intent of the writer. Even more interestingly, writers can express their thoughts and feelings through the use of constructs such as sarcasm or hypothetical statements. Even though these thoughts and emotions can be expressed in a variety of ways, other human beings are generally able to parse the sentiment from thoughts and ideas very easily. This allows for efficient communication between individuals.

However, when linguists and computer scientists attempt to systematically parse the written work of humans, difficulty arises. This difficulty stems from trying to algorithmically recognize sentiment in complex sentence structures and abstract concepts such as hypothetical situations that do not reflect reality. Given the massive amount of written information found in books, journals, and web pages, it would be beneficial to have a system that is able to accurately analyze sentiment from such information. For example, in the field of economics, researchers such as Brett Drury and J.J. Almeida have found benefit in analyzing large amounts of real-time information found in sources such as financial blogs due to "the possibility of inferring future prospects of economic actors based upon information contained in news" [2].

Many researchers have attempted to solve this problem using an array of algorithms and techniques. These methods range from algorithms that count words of varying sentiment and calculating the net sentiment to using machine learning techniques to create a model in hopes of classifying sentiment using past data. While some of these techniques have had reasonable success, there still exist other problems when analyzing sentiment.

One such difficulty is determining how to handle aspects of complex sentences in a way that reflects the true sentiment expressed. These aspects could include modifiers (e.g. the word "not") that change the sentiment in certain words or phrases. In section 2, I explain some background required for understanding sentiment analysis. In section 3, I discuss modifiers and how their usage creates difficulty for sentiment analysis algorithms. In section 4 and 5, I describe two ways that differing systems handle modifiers: a lexicon-based approach and a classification approach. Lastly, I describe the results of testing the two systems in section 6.

2. BACKGROUND

Natural Language Processing is a field in which algorithms are implemented in a way that human input (e.g. human language) can be parsed and analyzed [7]. A sub-field within Natural Language Processing is *Sentiment Analysis* which, according to [1], "aims at detecting the expressions of sentiment in text and subsequently classify them, according to their polarity ... among different categories" [1]. This classification occurs in many contexts such as identifying if an instance of text's sentiment is positive or negative (e.g. happy or unhappy, calm or angry). The polarity that an instance is classified into is also known as its *semantic orientation.*

UMM CSci Senior Seminar Conference, December 2013 Morris, MN.

3. MODIFIERS

The process of parsing semantic orientation is somewhat trivial when using very basic sentence structures. However, when one introduces more complex sentence structures and word usages, parsing semantic orientation proves to be a more daunting task [5].There are many different difficulties that arise due to sentence complexity. The three complexities that are focused on in this paper are intensifiers, negators, and irrealis statements.

3.1 Intensifiers

Intensifiers are words or phrases that modify a word in such a way that either amplify its original semantic orientation or act as down-toners and decrease it. These two intensifiers have the potential of modifying an entire statement's polarity drastically [5, 9].

In the case of *amplifiers*, a modified word's semantic orientation is increased by a percentage of its initial value.

1. The film was good.
2. The film was very good.
3. The movie was depressing.
4. The movie was incredibly depressing.

In the first two examples, we see a phrase with a positive semantic polarity being influenced by the amplifier "very" modifying the word "good." This amplifies the statement's already positive polarity to a much higher orientation. Likewise, we see a negative statement being amplified by the word "incredibly" in the latter two examples. This causes the orientation of the statement to become increasingly negative. Amplifiers increase the orientation's magnitude of the modified word.

The second example of intensifiers are *down-toners*, or words that reduce their word's semantic orientation [5].

1. The film was good.
2. The film was somewhat good.
3. The movie was depressing.
4. The movie was slightly depressing.

For the first two examples, we see a positively oriented statement being modified by a down-toner. In the first sentence, the orientation is positive. While the second orientation is still positive, the down-toner "somewhat" modifies the orientation in such a way that it is less positive than it was previously. In the second two examples, we see a negatively oriented sentence being modified in such a way that it's less negative than it was previously. Down-toners act similarly to amplifiers in that they modify a word's semantic orientation as a percentage of its initial value. However, unlike amplifiers, down-toners lessen the orientation's magnitude [5].

3.2 Negation

Negators are words or phrases that modify a word in such a way that the initial orientation of certain words is reversed [9]. For example, in the sentence "the film was not good", we see an initially positive statement being negated by the negator "not". This modifies the orientation to be negative instead of the initial positive orientation. This negator changes the semantic orientation of a word so that it is close to the opposite of the initial orientation [5].

There are two difficulties with identifying and applying negators that frequently come up when attempting to parse semantic orientation. The first difficulty is identifying a negator and applying that negator to the correct word.

1. The movie was not invigorating.
2. I can not say that I am happy with this movie.

In the first example, we see a very basic use of a negator. There is a word that has a positive semantic orientation ("invigorating") and a negator that appears just previous to it in the sentence. Given that there are no other semantically oriented words around the "not" statement, we can make the judgment that the negator is modifying the word, "invigorating". However, in the second example, we see a negator ("not") and a positively oriented word ("happy") with no immediately clear relation between the two words. The difficulty in this case is being sure that an algorithm is applying the negator to the correct word.

The second difficulty in applying a negator is the figuring out how much a negator should influence a word. For example, consider two words with extremely strong semantic orientations, "terrible" and "terrific". Each of these words represent a very positive or very negative semantic orientation. If we apply the negator, "not", to these words, we would be modifying their orientations considerably. However, assuming these two words have equivalent magnitudes of polarity, these negations are far from equal to their counterparts. The phrase "not terrific" would not imply a polarity of the same level as "terrible". The difficulty in applying these negators is considering how much to modify each word given the context of the negation.

3.3 Irrealis Statements

The last important modifier that is analyzed in this paper is the use of *irrealis statements*. An irrealis statement is one where hypothetical situations occur that don't actually occur in reality [9].

1. This movie should have been the best film of the summer.
2. The actor had the potential to improve this film drastically.

We see that in both of the examples above, the hypothetical situations for each are both positive. However, the two statements describe little about the reality of the situation. Instead, they focus on non-factual situations. A proper semantic analysis would classify both of these situations as having little or no effect on the orientation of the overall statement, given that these situations are only hypothetical and not a proper reflection of reality.

The difficulty arises in parsing and classifying a statement as an irrealis statements. Some may look for certain keywords associated with irrealis statements (e.g. "could", "should") and try to identify a statement as being an irrealis statement or not based on those keywords. However, even if someone were to correctly identify these blocks, the next difficulty is analyzing how these blocks modify the existing orientation. These problems are not easy to answer.

4. LEXICON-BASED APPROACH

One approach to deciphering a sentiment from text is by analyzing the structure of individual words. In general, this requires the use of a defined *lexicon*, or a dictionary with mappings between words and emotional ratings, in order to parse emotions from text. These lexicons are usually predefined before any sentiment analysis is performed. With the introduction of a lexicon-based approach, the potential for more accurate parsing of polarity becomes a simpler task [5].

In the lexicon-based approach outlined in [5], it is apparent that the introduction of weighing modifiers provides benefit to the sentiment analysis process. This approach involves using a manually created lexicon of word-sentiment pairings. The lexicon is based on a scale ranging from 5 (positive) to -5 (negative) based on a word's *prior polarity*, or the meaning of the word in most contexts. This lexicon is built out of a 400 text *corpus*, or a collection of written texts, that originate from reviews from the site Epinion. These reviews consist of eight review sources (e.g. book reviews, film reviews). The polarity of the texts is based on whether or not the reviewer recommended the piece of media. The system that [5] based their sentiment analysis on relies on handling modifiers and other difficulties in order to achieve an accurate analysis: the handling of different parts of speech, intensifiers, negators, and irrealis statements [5].

4.1 Parts of Speech

The difficulty with analyzing different parts of speech for contextual polarity is that different parts of speech can modify the meaning of the word drastically. In the context of adjectives, one would be able to decipher the adjective's polarity fairly easily given that many adjectives already imply their context. For example, the adjective "terrible" tends to imply a negative connotation with very few contradictory cases. However, the use of nouns, verbs, and adverbs carry connotations that are more difficult to describe. In certain cases, verbs tend to have both a neutral and non-neutral connotation. [5] uses the example of the word "inspire" to illustrate this difference.

1. The teacher inspired her students to pursue their dreams.
2. This movie was inspired by true events.

In example 1, we see that the word "inspire" implies a fairly heavy positive connotation whereas example 2 implies a neutral connotation. The way that the authors' system handles these differences is by choosing the average case of polarity for a word, assuming that the words do not hold drastically different polarities (e.g. heavy positive and negative) [5].

4.2 Intensifiers

In the context of intensification, [5] handles each intensifier depending on its weight: positively weighted amplifiers and negatively weighted down-toners. Each weight for the intensifiers is handled multiplicatively in that they are multiplied by the polarity of the word that they modify.

1. The man wore a somewhat tattered suit for his interview.
2. The man greatly succeeded in the task he was given.

In example 1, we see "somewhat" (a down-toner) being applied to "tattered" (a negatively weighted word). Assuming tattered has a semantic orientation of -2, one can see that applying the down-toner to the word is able to change the word to imply its true semantic orientation using the weights. If we assume the down-toner has a modification value of 25%, one can calculate the new semantic orientation of the word modified by the down-toner: -2 * (100%-25%) = -1.5. In example 2, we see the same modification happening using "greatly" (an amplifier) and "succeeded" (a positively oriented word). Assuming "succeeded" has a semantic orientation of 3 and "greatly" has a modification value of 100%, one can recalculate the new orientation of the word: 3 * (100%+100%) = 6. These calculations rely on having a preconception of the modification values of certain intensifiers. In the model described in [5], a new lexicon is created for each potential intensifier. This allows the model to handle intensifiers in a way that better reflect the expected orientation of modified words.

4.3 Negators

When handling negators, there are two difficulties that arise: finding what words the negator is modifying and deciding how much the negator modifies the orientation of the words.

In [5], the process of finding a word's negator can occur according to two different processes. The first process involves finding a semantically oriented word and searching backwards in the sentence for a negator. This process ends at certain clause boundaries such as end of statement punctuation (e.g. periods) or certain sentential connectives (e.g. words such as "but"). This ensures that negators in unrelated statements do not modify incorrect words. The second process entails backward searching by skipping unrelated words in hopes of finding a negator. However, the list of words to be skipped is small which makes this process somewhat conservative [5].

The process of applying a negator to a word is not a simple process as different contexts may cause a negator to modify a word differently. For example, simply flipping the polarity of a negated word doesn't necessarily provide the best results.

1. The cake wasn't terrible.

In the previous example, one could simply flip the polarity of the word "terrible." This would result in the initial polarity (-4) flipping to the opposite polarity (4). However, the phrase "not terrible" would not be equivalent to an equally positive word (e.g. fantastic). [5] handles this by deciphering the modifying value of the negation phrase and applying that value in an additive way to the prior polarity of the modified word. This would lessen the extremity of the polarity of the modified word [5].

4.4 Irrealis Terms

As previously mentioned, irrealis terms are phrases that do not reflect reality. The model outlined in [5] handle irrealis sentences and phrases by simply ignoring the semantic orientation of words that are within an irrealis phrase. This is done by parsing the sentence for irrealis words (e.g. should, could) and finding what words are affected by the irrealis statement [5]. However, in some cases, a hypothetical statement would reflect a part of reality. For example, if the statement "he can get away with marketing this amateurish crap and still stay on the bestseller list?" is a hypothetical statement, however still holds a negative orientation. The authors' system looks for definitive words that reflect reality within a hypothetical statement (e.g. "this"), and would therefore ignore the irrealis phrase's nullification, causing the phrase to be treated as any other real phrase [5].

5. CLASSIFICATION BASED METHOD

Aside from algorithms that analyze sentences based on the preassigned polarities of the words or sets of words within an instance of text, there are other algorithms that utilize machine learning techniques to create models that use sets of data to learn how to classify new sentences' polarity. Instead of computing the polarity of the sentences based solely on the prior polarities of each word, these algorithms are able to create models that can learn to classify, or label new textual instances as being of a certain polarity based on previous experiences with similar datasets [9].

5.1 Text Classification Approaches

Using a text *classifier*, or an algorithm that uses previous textual instances to decide what category (e.g. positive or negative) a new textual instance belongs to, is a commonly used approach to textual polarity classification. In order to create classifiers, one must use a large set of *training data*, or instances that are used to help teach the classifier rules by which to classify new instances. This training data must be related to the problem in order to create a useful and robust group of features (e.g. initial polarity of a word, if a word occurs in the subject of a sentence) on which to classify new instances on. Unlike the lexicon-based approach, a classifier-based approach's features would not necessarily represent just the word, but elements of the word's sentence and surrounding sentences. After the labeled data-set is created and it is known what to analyze for classification, a classification algorithm can be used to create the classifier. There exist many algorithms from which classifiers are built [9]. Each of these algorithms differ in the way they create rules or heuristics in which to classify new instances of data. In the context of this paper, we will not be focusing on the classification algorithms themselves, but instead the focus will be on how features to base classification on are created.

5.2 Process

One approach to recognizing contextual polarity using textual classification approaches is the two-step approach outlined in [9]. This two step process involves using potentially subjective words inferred from a previously defined lexicon to decide if an instance is neutral or polar. In the second step, polar instances are classified as positive or negative.

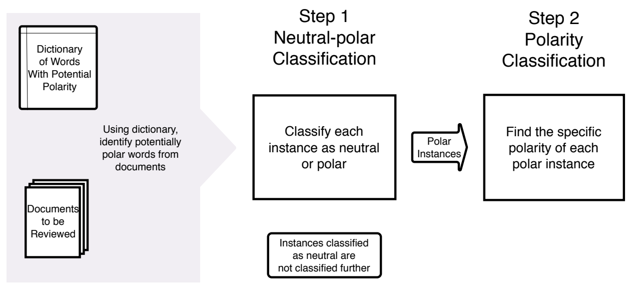
Dictionary of Words With Potential Polarity
Documents to be Reviewed
Using dictionary, identify potentially polar words from documents
Step 1 Neutral-polar Classification
Classify each instance as neutral or polar
Polar Instances
Step 2 Polarity Classification
Find the specific polarity of each polar instance
Instances classified as neutral are not classified further

Figure 1: The two step process described in [9]. Edited from [9].

Before the two step process occurs, a lexicon of potentially subjective words is created. This lexicon is compiled from both a previously defined lexicon and new potentially subjective words derived from a list of positive and negative words. Words that are subjective in almost every case are labeled as strongly subjective (e.g. happy), whereas words that are subjective in some cases but not others are labeled as weakly subjective (e.g. venomous). Lastly, each individual word is labeled with its prior polarity, or the polarity that the word tends to have in the majority of situations. These can be labeled as positive (e.g. happy), negative (e.g. sad), both (e.g. bittersweet), or neutral. Using this lexicon, the two step process can receive instances of words that are potentially subjective and, as such, can begin to decipher the polarity [9].

5.3 Neutral-Polar Classification

After receiving an instance of text that is marked as being potentially subjective, the first step in deciphering polarity is to determine if an instance is polar or neutral. In order to do this, a list of features for the instance are created to be used in the classification algorithm [9]. These features are representations of attributes of the word, modifying features of the word, and general sentence or document features.

The first set of features is related to the word and the words around it. The features regarding the word itself include the part of speech of the word (e.g. adjective), the polarity of the word according to the lexicon, whether it is a strongly or weakly subjective word, and its location within the sentence. The other features regard the words around the target word such as the surrounding words' parts of speech and whether the word is preceded by an intensifier.

The second set of features relate to how the word modifies or is modified by other words. These features are more difficult to gather as they are parsed through a dependency tree. A dependency tree is a tree in which relationships between nodes (in this case, words) can be described and analyzed in relation to each other. In a dependency tree, each node represents a word, with children representing modifiers for the parent node [9]. Each word's features include:

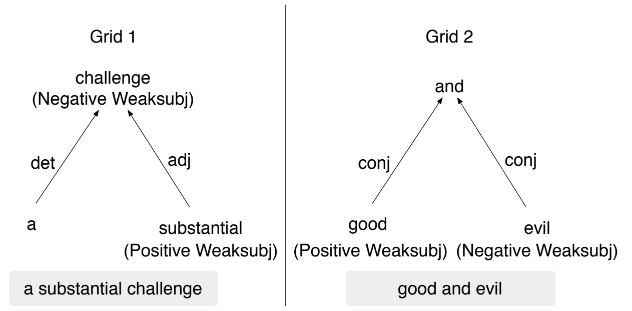

Figure 2: Dependency trees representing the phrases "a substantial challenge" and "good and evil". Edited from [9].

- Whether it modifies or is modified by a subjective term. This is parsed from the decision tree by checking if the node shares an adjective or a modifier relationship with its parent or child and that the parent or child is potentially subjective. In the tree in figure 2, the word *challenge* would be modified by the word substantial.

- If it is a modifier, the polarity of the parent node. In figure 2, the word *substantial* is a modifier and the polarity of the parent is negative.
- If it's child is a modifier, the polarity of the child. In figure 2, the word *challenge's* child is a modifier, and the child's polarity is positive.
- If it is in a conjunction, the polarity of the sibling. In the tree, the two adjectives *good* and *evil* are in conjunction.

Lastly, there are general features related to the sentence or document in which the word appears. Many of these were chosen based on external experiments judging the features as useful [9]. These include the subject of the instance (e.g. politics) and whether there is a cardinal number (e.g. 4, 13), a modal (e.g. can, should), or a pronoun in the sentence. There is also a count of how many subjective words, adverbs, and pronouns are in the current and surrounding sentences.

A classification algorithm can use a large grouping of feature sets described above to create a model that would be able to classify new instances as either polar or neutral. The results of using these features are described in section 6.2.

5.4 Polarity Classification

Using the polar instances produced in the first step, the second step is to classify the polarity as being positive or negative. This is done similarly to the neutral-polar classification in that there are initially features that are generated based on the text instance. These features do not rely on a dependency tree in order to be parsed. There are some features that are taken from step one: the features related to the word and the polarity features in relation to other words [9]. These features include:

- Whether or not the word is negated. This is done by looking at the four words preceding the instance and determining whether or not there is a negating word.
- If the subject is negated. This would potentially imply that some of the words following the subject are negated as well.
- If there are any modifiers that are influencing the word.

After parsing these features, a classification algorithm can then be used to create a model to classify new instances as being positive or negative. After this step, the polarities of each word are classified as positive, negative, or neutral.

6. RESULTS

6.1 Lexicon-based Method

As previously noted, the lexicon-based method put forth in [5] evaluates the system using a series of reviews on varying media outlets (e.g. movies, music) from the website Epinions. Using the lexicon-based system that was outlined previously, [5] attempt to predict whether a product review was positive or negative against what the reviewer scored the product. They ran the experiment using many different forms of their system: a simple system with switch negation, intensification, and other dictionaries, a system with only the adjective dictionary, a system with only single-word values, and a system with all modifier support. The results show that using all forms of modifiers is beneficial given that the accuracy for a full support and simple support (modifier support) system had a higher value than the adjective only system by around 1% (statistically significant) [5].

Also, when scoring their system in its full form to classify, in general, the review's polarity, the general accuracy is quite high. These results average around an 80% accuracy over all predictions. One interesting aspect of the results is that the system tended to perform better when predicting a positive polarity over a negative one as seen by the higher F-measures for positive classification [5]. An F-measure is a measure of accuracy (ranging from 0 to 1) that takes into account both precision (percentage of true positives per true and false positives) and recall (percentage of true positives per true positives and false negatives), both of which are influenced by an ability to make correct classifications. This F-measure will help to compare accuracies of different classification models. The equation is as follows: $F = 2 \times \frac{precision \times recall}{precision + recall}$ [6].

6.2 Classifier-based Method

The classifier-based method of sentiment analysis outlined in [9] creates a list of features based on the text instance and, using machine learning techniques, creates a classification model to categorize new instances. This process requires the use of a classification algorithms that will take the list of features, and generate the necessary classification model [9]. For experimentation, [9] use a series of classification algorithms in order to generate the model.

In order to explain the results, we discuss boosting and support vector learning. For the techniques, the specific algorithms that are used are BoosTexter and SVM [9]. In the BoosTexter algorithm, many weak, partially accurate classification models are created. These weak models are then combined to create a stronger, more accurate model [4]. Support Vector Machines (SVM) create classification models using linear classification. Each data point in a training set is represented by a single point on a hyperplane. The goal for the SVM is to differentiate between categories or classifications by creating lines that separate different classification categories. New classification occurs by plotting the point on the already existing plane and classifying based on which side of the line(s) the data point falls under [8]. Using these algorithms, an attempt is made to classify polarity from 10,287 sentences in Multi-Perspective Question Answering (MPQA) corpus documents [9].

When comparing a single-step algorithm (not separating polar-neutral and polarity classification) to a two-step algorithm outlined above, there seems to be mixed results. As seen in table 1, the two-step algorithm outperforms the one-step algorithm in accuracy. However in actual F-measure, it performs equal to or worse than the single-step algorithms. The only field in which most of the classification algorithms outperform the two-step process is for neutral values. Otherwise, with the exception of SVMs, the two-step process tends to falter [9].

6.3 Comparing Models

While these models both show improvement in their respective systems, it is difficult to compare the two. One variable that makes this comparison nearly impossible is that the two methods of handling modifiers rely on completely

	Acc	Pos-F	Neg-F	Neutral-F
BoosTexter				
two-step	74.5	47.1	57.5	83.4
one-step	74.3	49.1	59.8	82.9
SVM				
two-step	73.1	46.6	58.0	82.1
one-step	71.6	43.4	51.7	81.6

Table 1: Accuracy and F-measure for positive, negative, and neutral classifications. Edited from [9].

different models. If one were to outperform the other, it could be due to the system's choices for lexicons, algorithms, or a multitude of other differences in the model and not with the modifier support.

However, [3] attempts to compare similar algorithms by applying a lexicon-based method and various classifier-based methods on the same test data. While these methods are not exactly the same as the methods outlined previously, some reasonable comparison between the two classes of methods could potentially be made. The two algorithms that are the most similar to the algorithms discussed in this paper are the Linguistic Inquiry and Word Count (LIWC) tool and the SentiStrength algorithm [3]. These two algorithms represent what is close to a lexicon-based method for analysis and a classification based algorithm.

For comparison purposes, I will analyze the agreement, or how well the algorithms classify the text against what the preclassified value is. In order to compare the two, [3] uses two different data-sets: a series of Twitter messages reflecting events, reviews, and sports, and a data-set of miscellaneous text labeled as positive or negative. According to the results, the F-measure for LIWC is 0.689 in comparison to the F-measure of SentiStrength at 0.765. It seems that SentiStrength outperforms LIWC on average [3]. While this comparison does show some evidence for classification-based approaches to sentiment analysis, it still may be the case that the success of the classification-based approach may be domain specific. While we may be able to compare algorithms that are like the ones previously described, they are not the same algorithms. As such, making any sort of conclusion regarding the comparative effectiveness of the lexicon-based algorithm over the classification-based algorithm is still difficult.

Acknowledgments

I would like to thank Elena Machkasova for being my adviser and Nic McPhee, Skatje Myers, and Elijah Mayfield for reviewing my paper.

7. CONCLUSION

In this paper, we have discussed the usefulness of being able to parse sentiment from various documents. However, with the introduction of sentence modifiers, accurate classification becomes a difficult problem. A lexicon-based approach is able to identify modifiers by parsing the sentence and words within the sentence and applying the effect of the modifier using certain mathematical properties. The classification approach is able to create certain features of a text instance by analyzing relationships between certain words. Using these features, a classification algorithm can create a model that is able to categorize the words based on these generated features. In the classification-based approach that was described in this paper, the process first parses features that are able to identify whether a text's polarity is neutral or not. The second step involves finding features that can be used to build a classification model that will categorize the polar words as being positive or negative. When testing the lexicon-based and classification-based algorithms, they both seem to perform well in specific domains. However, the two-step classification approach seems to falter in comparison to a one-step classification approach. Given that these algorithms are being tested for different domains, it is difficult to compare the two. However, when comparing algorithms similar to the outlined algorithms, the lexicon-based approach outperforms a simple classification approach.

When analyzing the two algorithms, there is noticeable benefit to analyzing modifiers. With this additional ability, future sentiment analysis is improved and, as such, the field of natural language processing can parse sentiment in more accurate ways. With the large amounts of data being readily available with the increase in popularity of data mining, being able to process important pieces of information from text is an expectation of the future. These incremental improvements only help to foster an environment where complex analysis of textual data is possible and beneficial.

8. REFERENCES

[1] A. Balahur, J. M. Hermida, and A. Montoyo. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 53–60, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] B. Drury and J. J. Almeida. Identification of fine grained feature based event and sentiment phrases from business news stories. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, pages 27:1–27:7, New York, NY, USA, 2011. ACM.

[3] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha. Comparing and combining sentiment analysis methods. In *Proceedings of the first ACM conference on Online social networks*, COSN '13, pages 27–38, New York, NY, USA, 2013. ACM.

[4] R. E. Schapire and Y. Singer. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[5] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June 2011.

[6] Wikipedia. F1 score — wikipedia, the free encyclopedia, 2013. [Online; accessed 17-November-2013].

[7] Wikipedia. Natural language processing — wikipedia, the free encyclopedia, 2013. [Online; accessed 31-October-2013].

[8] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2013. [Online; accessed 25-November-2013].

[9] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Comput. Linguist.*, 35(3):399–433, Sept. 2009.