Paper 152-2008

Coming to a Theater Near You! Sentiment Classification Techniques Using SAS® Text Miner

Jake Bartlett and Russ Albright, SAS Institute Inc., Cary, NC

ABSTRACT

Many Web sites, including blogs, online stores, and some database Web sites, give users the ability to state their opinions about the products they buy and use. This is a serious concern for many companies because of the potential damage of reputation. Companies want to monitor and gauge customers' opinions, but they do not have the time or resources to manually gather statistics on this ever-expanding content source. Can companies reliably predict whether a customer comment is positive or negative without actually reading each posted comment? In this paper, we demonstrate how different techniques for classifying movie reviews can be implemented in SAS® Enterprise Miner™ and SAS® Text Miner. Movie reviews are classified not by category, but by sentiment.

INTRODUCTION

A large amount of textual content is subjective and reflects opinions. With the rapid growth of the Web, more people write online reviews for all types of products and services. It is becoming a common practice for a consumer to learn why others like or dislike a product before he or she buys it, or for a manufacturer to track customer opinions on its products to improve customer satisfaction. However, as the number of reviews for a product grows, it becomes harder to understand and evaluate customer opinions about a specific product.

Sentiment classification, also referred to as polarity, tone, or opinion analysis, can track changes in attitudes toward a brand or product, compare the attitudes of the public between one brand or product and another, and extract examples of types of positive or negative opinions. In this paper, we explore several techniques for improving sentiment classification with SAS® Text Miner. Standard techniques in SAS® Text Miner such as weightings, synonyms, and part-of-speech tagging are presented, along with more complex extraction and feature manipulation techniques.

In the next section, we provide the background on sentiment classification approaches, a description of the data set that we use for our experiments, and some technical information necessary to understand our techniques. In the next sections, we present several methods for improving the effectiveness of sentiment classification:

1. Baseline SAS® Text Miner – running SAS® Text Miner with the default settings
2. Baseline SAS® Text Miner with MI – running SAS® Text Miner with the default settings and using mutual information (MI) for weighting terms
3. Sentence Filtering – building a model using relevant sentences and removing noisy sentences
4. Synonym Lists – assisting the prediction with a knowledge base of positive and negative terms
5. Sentiment Reversal of Terms– manipulating features to track the occurrence of "not" and "n't" in text

Following these sections are experimental results applied to the data set described in the background section. Then, we present our conclusions.

BACKGROUND

SENTIMENT CLASSIFICATION APPROACHES

There are two basic approaches to sentiment classification. The first approach is to use a model that is based on the frequency of occurrence of each term in each document. Documents are represented as vectors of term frequencies, and the information about the sequence in which the terms appear is disregarded. The terms themselves become features. Techniques such as stop lists, SVD, and term weightings are used to select and enhance the features. This statistical-based approach is at the heart of SAS® Text Miner and has significant advantages because the algorithms can learn the important characteristics from the collection as a whole. Surprising and unexpected terms might actually prove useful for prediction accuracy. Unfortunately, while this bag-of-words approach can be effective for classifying documents according to a topic or category, it is often less successful in classifying sentiment.

The second approach is to semantically parse each sentence into formal concepts that can then be understood and interpreted with custom knowledge bases. This approach relies on a consistent and uniform body of text and requires an extensive set of rules and background information that disambiguate complex constructs and diverse vocabulary. When you and I read and interpret a phrase such as "sunken ship of a movie," we apply our background knowledge to help us understand that this is a negative phrase. A semantic-based system must have this background knowledge about sunken ships to know that the phrase is indicating that the movie is bad. Without the background knowledge, semantic systems are ineffective.

Although the semantic approach sounds good, it is difficult to implement and performance can be much worse than statistically-based systems except in the most idealized situations. For this reason, many researchers and the approaches presented in this paper incorporate aspects of the semantic approach while still using an underlying vector-based approach to represent each document.

THE MOVIE COLLECTION

For our experiments, we use a movie review collection that is well established in the research community. The collection is available from http://www.cs.cornell.edu/people/pabo/movie-review-data/. It is described and used in the papers by Pang et al. (2002) and by Pang and Lee (2004). The collection contains 2000 reviews by a large variety of individuals. Each review has an associated rating of positive (the reviewer recommends the movie) or negative (the reviewer does not recommend the movie).

Each review is several paragraphs long, and was case-normalized by the researchers who provided the collection. (This is unfortunate because we believe SAS® Text Miner's part-of-speech tagger would perform better on text that is not case-normalized). An actual paragraph from one of the reviews follows:

> i saw it at a matinée and took quite a while to adjust to the sunny skies afterward . often the jekyll and hyde archetypes are viewed as splitting a whole person into components of good and evil or perhaps cerebral and emotional parts. here the doctor is intellectual and perhaps good, but certainly powerless. his alter ego is forceful and totally without conscience. two parts that make up a whole. neither are capable of functioning without the other and once separated, disaster is inevitable. the differences between the two characters would be more effective if it were accomplished by demeanor and attitude. when we finally see the transformation, the special effects thrust the story into the realm of science fiction, not the psychological horror that the tale demands. roberts and malkovich are skillful in their roles. both are suitably melancholy fitting in with the rest of the film

As you might expect from the paragraph, sentiment analysis is a difficult machine learning problem. Much of the content in the movie review actually describes the plot, not the reviewer's opinion of the movie. Because a reviewer could easily write negatively about an aspect of the plot, but still give the overall movie a positive rating, the challenges are significant.

Consider the following negative and positive phrases pulled from various reviews:

> "sunken ship of a movie"
> "falls flat on its face"
> "not worth your time or your hard-earned dollars"
> "shows us how to get it right"
> "gives us faith that Hong Kong cinema is alive and flourishing"

In all of these phrases, and these were typical phrases, the statements were the clearest positive or negative comments that we could find in the reviews. Because the sentiment polarity is only indirectly stated, the sentiment analysis challenge is accentuated.

SAS PERL REGULAR EXPRESSIONS

The sentence selection and sentiment reversal approaches that we performed involved preprocessing the text of the data set using SAS DATA step code. A common technique was to search for and replace (or manipulate) portions of text. The SAS implementation of Perl regular expression functions was invaluable for this purpose.

A Perl regular expression is a string that describes or matches text based on syntax rules. The expressions provide a language for searching and manipulating text. SAS provides INDEX, SCAN, and SUBSTR functions for simple search and replace operations, and Perl regular expression functions when more advanced search and replace operations are needed.

All Perl regular expressions are created between forward slashes. For example, in the expression /abc/, the only text that will match is text that contains the literal combination of “abc.”

The true value of Perl regular expressions lies in the metacharacters. The Perl regular expression syntax provides metacharacters that can represent digits (\d), words (\w), the starts of lines (^), and the ends of lines ($). You can enumerate the number of occurrences that a pattern can match. For example, the regular expression /abc\d/ will match abc followed by a single digit.

The function PRXPARSE causes a regular expression to match a certain pattern. In the following example of a regular expression, we are searching for sentences:

```
re=prxparse("/([^.?!]*?)([.?!])/m");
```

The function PRXMATCH lets us know when the pattern is matched. It returns a 1 when the pattern is matched, and a 0 when it is not.

```
matched=prxmatch(re, text);
```

The regular expression is looking for any number of characters except a period, a question mark, or an exclamation point. When it finds one, it stops searching.

The function PRXPOSN lets us extract the content that the match found. The parentheses are special characters that define search groups. In the regular expression, two groups are defined. One group contains the sentence, and the other contains the ending punctuation. We can use PRXPOSN to return the contents of either or both of these groups.

In the following code, we use the DATA step DO WHILE loop to process a text variable by repeatedly searching for sentences longer than 10 characters that match the regular expression. When one is found, PRXPOSN extracts the sentence and sends it to the target data set.

```
do while (matched);
    call prxposn(re, 2, start, len);
    sentence=substr(text, 1, start+1);
    if length(sentence) > 10 then output;
    text=substr(text, start+1);
    matched=prxmatch(re, text);
end;
```

TECHNIQUE 1 AND 2: BASELINE SAS® TEXT MINER AND BASELINE SAS® TEXT MINER WITH MI

To compare the effectiveness of the different techniques, we established a baseline by using SAS® Text Miner with its default settings. We used a sample node in our training data set, setting aside 70% for training, 20% for validation (needed in the support vector machine (SVM) model), and 10% for testing. Then, we ran SAS® Text Miner with its default settings. There were three runs with three different test samples and the results were averaged.

To improve the misclassification rate of the baseline, we first adjusted the settings of the SAS® Text Miner node. Of all of the adjustments, we found that setting the weighting to use MI was the only adjustment with significant gains. MI weights the terms found in the training set in accordance with the target variable. Terms that tend to occur a high proportion of time in either documents that are rated with a positive sentiment or in documents that are rated with a negative sentiment will receive a high mutual information weight. Terms that are evenly distributed between both types of documents will receive a low MI weight.

This adjustment improved results more than any other, which is encouraging because it indicates that out of the box, SAS® Text Miner requires minimal experience to take advantage of the results in fields as challenging as sentiment analysis.

TECHNIQUE 3: SENTENCE FILTERING

Sentence filtering attempts to identify two types of sentences: objective (such as movie plot details and summary) and subjective (opinions). It uses the subjective sentences to build the model and make predictions. In effect, sentence filtering is designed to reduce the noise of a document.

Extracting relevant sentences is not a new approach. Beineke et al. (2004) built a predictive model to automatically extract the key statements made by a movie reviewer. This was done by using quotations from www.rottentomatoes.com as training data. This data contains 5000 movie review snippets (for example,"bold, imaginative, and impossible to resist"). The snippets served as examples of subjective sentences. Examples of objective sentences were obtained from 5000 sentences about plot summaries from the Internet Movie Database (www.imdb.com). Only sentences of at least 10 words or more were selected from plot summaries of movies released after 2001. This ensured that there was no overlap with the movie review data.

To implement a similar approach, we have a two-step process. First, we choose the relevant sentences that we want to keep using a predictive model. Then, we use the relevant sentences as our representative documents.

For the first step, we use SAS Perl regular expressions to break down our original positive and negative documents into individual sentences. The code that does this is shown in the example in the previous section. The individual sentences make up a new data set, which contains one sentence per observation. Using our predictive model, we score each sentence and decide whether it should be classified as subjective or objective.

For the second step, we combine all of the sentences that we decide to keep. Sentences that were classified as subjective are concatenated in a single document. This step allows us to reject sentences that are objective.

Finally, we replace the original documents with the new document containing the classified subjective sentences. We train an SVM model using these sentences to predict the sentiment of the entire document.

SUBJECTIVE PREDICTION THRESHOLD

We varied the threshold for choosing subjective sentences that were concatenated in documents. The decision to keep a sentence was based on the posterior probability that the sentence was subjective. We varied the threshold probability from 0.0 to 0.8. A threshold of 0.0 resulted in keeping every sentence from the original document and giving us results in line with our baseline model. For thresholds above 0.8, too few sentences remained to build a useful model. In Figure 1, you can see that using a threshold between 0.4 and 0.6 yielded the optimum misclassification rate. If we choose too few sentences, which is the result when the threshold is 0.8, the misclassification rate increases dramatically.

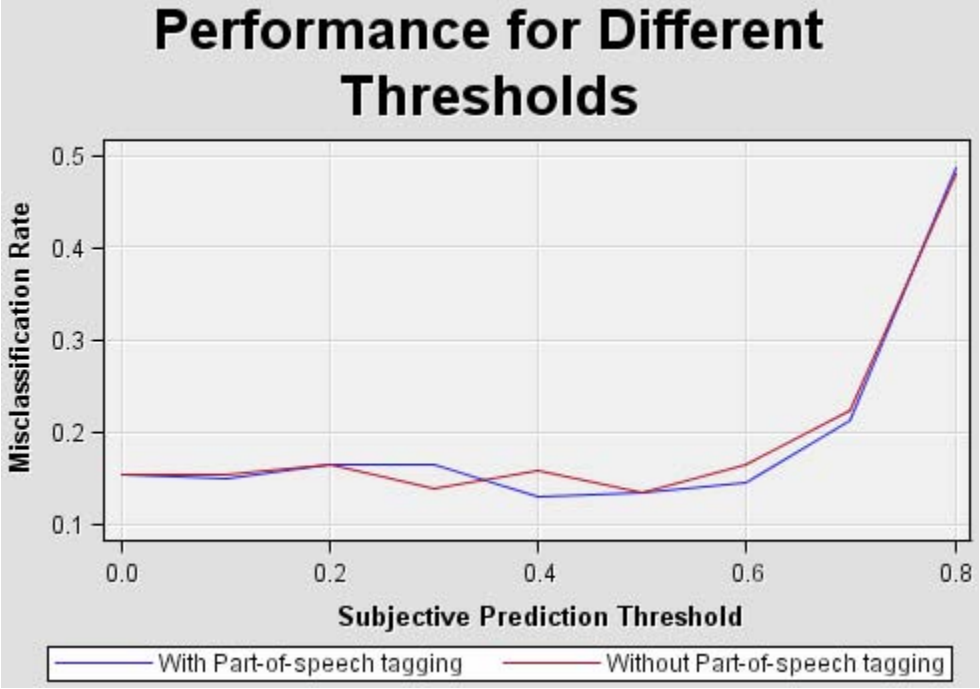

Figure 1: Misclassification Performance for Different Thresholds

TECHNIQUE 4: SYNONYM LISTS

In SAS® Text Miner, a synonym list can be viewed as a knowledge base that encodes domain-specific information and translates that information into the model being built. This motivates the creation of synonym lists that contain terms associated with positive and negative sentiment. Terms with a negative connotation are mapped to negative, and terms with a positive connotation are mapped to positive. Terms that are neutral are not included.

There are many sources for useful synonym lists. The Web provides many sites with taxonomies or term lists for different domains. Unfortunately, retrieving this information can be challenging. Frequently, scripts are required that traverse Web pages and extract the required data. In our case, we are interested in words that convey sentiment. The WordNet dictionary at wordnet.princeton.edu provides an extensive dictionary of terms, their meanings and usage, and their relationships to other words in the dictionary.

We wrote code to extract words from WordNet, add them to our synonym list, and categorize them as positive or negative. We created our synonym list by reading several movie reviews and using domain knowledge to add common positive and negative terms (good, wonderful, bad, terrible, and so on) to the synonym list. We enhanced the list by using WordNet to look up the synonyms and antonyms of these terms. We appended them to the list. This process was repeated when new words were found. For example, the word “good” has the terms “great,” “nice,” “solid,” and “superb” as synonyms. These terms were added to the synonym list with “positive” as the parent term. Some of the antonyms of “good” are “atrocious,” “awful,” “dreadful,” “painful,” and “terrible.” These terms were added to the synonym list with “negative” as the parent term. We then proceed to look up the synonyms and antonyms of each new word (e.g. “superb”) and add them to the list. This process has been used by several researchers (Kim and Hovy, 2004).

After we completed this process, we manually reviewed the synonym list to eliminate terms that did not convey the sentiment that was assigned. In addition, common words such as “great,” “strong,” “take,” and “get” occurred many times in both positive and negative documents. These terms were thrown out.

TECHNIQUE 5: SENTIMENT REVERSAL OF TERMS

By sentiment reversal, we mean the reversal of sentiment of individual terms caused by terms such as “not” or contractions ending in “n’t.” For example, “I did not like the movie” and “I liked the movie” are very close in the term space. Yet, they have significantly different meanings because the meaning of “like” has been reversed by the “not” that precedes it. This general approach has been used by several researchers (Godbole et al. 2007).

We investigated improving the sentiment classification by preprocessing the text before running it through SAS® Text Miner. In cases where “not” occurs as the parent term, we appended the string “not_” to selected terms in the remainder of the sentence. We chose to preprocess the original text, rather than post-processing the results of SAS® Text Miner. Post-processing includes the intricate task of post-processing the validation and training data. If the data needs only to be preprocessed, SAS® Text Miner can create the validation and testing data.

TURNING OFF PART-OF-SPEECH TAGGING

When part-of-speech tagging is off, we appended “not_” to every term that follows “not” up to a punctuation character. For example, the sentence, “Stalked does not provide much suspense.” would become “Stalked does not_provide not_much not_suspense.”

The SAS code that performs this action uses PRX functions. A regular expression searches for the occurrences of the terms that trigger negation:

```
re=prxparse("/([^.?!]*?)(n't | not)(.*?)([.?!])/m");
matched=prxmatch(re,trim(left(temp1)));
```

In a DO loop, every term between the punctuation and following the match is appended with “not_” by using the TRANWRD function:

```
do while (matched);
    call prxposn(re,1,start, len);
    call prxposn(re,2,start2,len2);
    call prxposn(re,3,start3,len3);
    newtext=trim(left(newtext))||substr(trim(left(temp1)),1,start);
    temp2=compbl(substr(trim(left(temp1)),start2,len2));
    temp2=tranwrd(trim(left(temp2))," "," not_");
    newtext=trim(left(newtext))|| trim(left(temp2))||" ";
    temp2=substr(trim(left(temp1)),start2+len2);
    temp1=trim(left(temp2));
    matched=prxmatch(re,trim(left(temp1)));
end;
```

TURNING ON PART-OF-SPEECH TAGGING

When the documents being analyzed have well-formed sentences, part-of-speech tagging is effective and smarter about which terms get negated. Rather than negating all terms that follow “not,” we can reverse the sentiment of only terms that typically convey sentiment, such as adverbs, verbs, and adjectives. Other types of terms, such as nouns and determiners, do not typically convey sentiment.

The overall approach is to parse the text to determine the terms, tags, and the offset positions in which the terms occur in the observations. We identify which terms to negate based on whether they follow a term stemmed to “not” and have an appropriate part-of-speech tag. Finally, we append “not_” to the terms and replace them in the original documents by using the offset positions.

The implementation requires a preprocessing call to PROC DOCPARSE, the SAS® Text Miner parsing procedure. Two new options, ADDPARENT and ADDOFFSET, cause the procedure to generate an output observation for every occurrence of every term in every document.

```
proc docparse data=rsent.posnegmovie key=NEWTERMS out=_parseout1 stop=sashelp.stoplst
    addparent addoffset
    stemming=yes
    tagging=yes
```

```
    ng=std
    entities=no
    stop=sashelp.stoplst
    language="english"
    encoding="cp_1252"
    addterm
    addparent
    addtag
    addoffset;
var TEXT;
run;
```

The output reports the offset position of the occurrence, along with the term, its parent term, and its tag:

WORK._PARSEOUT1

	TERMNUM	_DOCUMENT_	_COUNT_	_OFFSET_	_TERM_	_PARENT_
1	43	1	1	0	plot	plot
2	44	1	2	5	:	:
3	45	1	1	7	two	two
4	46	1	4	11	teen	teen
5	1	1	1	11	teen coup...	teen couple
6	47	1	1	16	couples	couple
7	48	1	2	24	go	go
8	49	1	3	27	to	to
9	50	1	13	30	a	a
10	51	1	1	32	church	church

Figure 2: Output of Each Occurrence of Each Term

Once the call is made to PROC DOCPARSE, the terms to be negated, their tags, and their offset positions are placed in arrays:

```
if _TAG_ in ('Verb','Adj','Adv') and length(_parent_)>2 then do;
    numwords=numwords+1;
    words[numWords]=left(trim(_parent_));
    offsets[numWords]=_offset_;
    wLengths[numWords]=length(_term_);
end;
```

The number of terms to be negated, indicated by NUMWORDS, is stored in a variable.

The content of the offset position arrays can be used to find the specific terms in the original text:

```
do i=1 to numwords;
    whichword=numwords-i+1;
    wordlength=wLengths[whichword];
    if length(temp) > offsets[whichword]+wordlength+1 then do;
    temp2=substr(trim(left(temp)),offsets[whichword]+wordlength+1);
end;
```

By stepping through the data set, it is possible to identify and append “not_” to terms in the original data set with the SUBSTR function and DO loop. The full implementation of the previous code is available by contacting the authors.

EXPERIMENTAL RESULTS

The misclassification results of the various techniques are presented in Table 1. Two columns display the results. The first column shows the results when part-of-speech tagging is turned on. The second column shows the results when part-of-speech tagging is turned off.

Technique	Part-Of-Speech Tagging On	Part-Of-Speech Tagging Off
Baseline SAS® Text Miner	0.208	0.217
Baseline SAS® Text Miner with MI	0.144	0.144
Subjective/Objective with MI	0.144	0.154
Synonym Lists with MI	0.155	0.149
Negation with MI	0.149	0.144

Table 1: Misclassification Rates of the Techniques

The rows correspond to the five techniques presented in this paper. After obtaining the results, we experimented with combining techniques to determine whether results could be improved. In these cases, the misclassification rates increased.

In some ways, the results are discouraging. The significant gains from the baseline (using SAS® Text Miner with its default settings) are solely attributed to adding MI. The other techniques that focus on sentiment cannot improve on the significant gain made by adding MI.

On the other hand, we should not understate the importance and effectiveness of statistical techniques, such as MI weighting. MI weighting learns details about the training data which we could not improve on with our various techniques. For baseline SAS® Text Miner with MI, the misclassification rate of 0.144 is almost identical to results in outside literature (such as 0.129 by Lee and Pang (2004)), but ours was obtained with a lot less work. Lee and Pang used a complicated sentence filtering approach without MI weighting.

CONCLUSION

Sentiment classification has many challenges that are difficult to meet. We presented several techniques for preprocessing your original text and modifying your synonym list. For your particular data in your domain or application, some of these techniques might improve misclassification rates. Our test data set was relatively small, having only 1800 observations for training and validation, and 200 observations for testing. With a larger test data set, more patterns might arise that improve classification accuracy.

There is always a trade-off in manipulating features in a model. All of the suggested techniques in this paper actually had negative or insignificant effects on the final results. This is frequently the case when modeling the nuances of text. For every action we take, such as the subjective/objective, synonym list, and negation approaches, there are negative consequences that we might not have considered. For some documents, we improve the probability of predicting correctly. For other documents, we make the probability of predicting correctly worse. In the end, the best approach might be to let the algorithms learn the patterns.

REFERENCES

Beineke, Philip, et al. 2004. Exploring Sentiment Summarization. In Exploring Attitude and Affect in Text: Theories and Applications: Papers from the 2004 Spring Symposium, ed. Yan Qu, James Shanahan, and Janyce Wiebe, 12-15. Technical Report SS-04-07. American Association for Artificial Intelligence, Menlo Park, California.

Kim, S., and E. Hovy. 2004. “Determining the Sentiment of Opinions.” *Proceedings of the Twentieth International Conference on Computational Linguistics*, 1367-1373.

Pang, B., and Lillian Lee. 2004. “A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.” *Proceedings of the Forty-second Annual Meeting of the Association for Computational Linguistics*, 271-278.

Pang, B., Lillian Lee, and Shivakumar Vaithyanathan. 2002. “Thumbs up?: Sentiment Classification Using Machine Learning Techniques.” *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, 79-86.

Secosky, Jason. 2004. “The DATA Step in SAS® 9: What’s New?” *Proceedings of the Twenty-ninth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc.

Yu, Hong, and Vasileios Hatzivassiloglou. 2003. “Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences.” *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 129-136.

Godbole, Namrata, Manjunath Srinivasaiah, and Steven Skiena. 2007. “Large-Scale Sentiment Analysis for News and Blogs.” International Conference on Weblogs and Social Media.

ACKNOWLEDGMENTS

The authors thank James Cox and Manya Mayes for reviewing drafts of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Jake Bartlett
SAS Institute Inc.
jake.bartlett@sas.com

Russ Albright
SAS Institute Inc.
russell.albright@sas.com