A Simple Probabilistic Approach to Ranking Documents by Sentiment

Andrew Lacey
Department of Computer Science
Swarthmore College
Swarthmore, PA 19081
lacey@cs.swarthmore.edu

Abstract

The problem of determining the sentiment of documents has often been approached via highly human-structured or highly complex methods. These approaches have generally been constrained to dividing an input corpus into two categories, positive and negative. I present a simple and straightforward probabilistic algorithm that attempts to rank an entire corpus in increasingly-positive order of sentiment, which is a more useful output. The output can be easily reinterpreted to solve a two-category sentiment classification task, in which case the proposed algorithm performs nearly as well as existing approaches to that problem.

1 Introduction

Humans generally have little difficulty in determining the sentiment – that is, overall "positiveness" or "negativeness" – of documents. However, this has proved to be a rather difficult task for machines. Furthermore, much of the research in this direction has involved classifying the documents in a corpus into two categories, as if the task were essentially the same as topic classification, which it is not. In the case of topic classification, a document is generally, for example, about baseball or not about baseball. The output of a program ranking documents in order of the extent to which they are about baseball would probably seem strange to a human reader – a document that uses the term "home run" to describe the sales of the Chrysler 300, which might rank near the middle of such a scale, is really not about baseball at all. But this is not the case with sentiment analysis. Sentiment of documents is essentially a continuous spectrum. One imagines that the concept of "half stars" in movie reviews was introduced because human reviewers found the choice between a mere five ratings to be constraining. Thus, it makes sense to approach the sentiment analysis task in a way that naturally lends itself to ranking an entire corpus in order of sentiment, rather than simply (or, in fact, not so simply at all) making a decision between two categories, for documents that are clearly in one of those categories. The algorithm presented here takes such an approach, and is also easily modified to provide output that can be compared to that of two-category sentiment classifiers.

2 Background and Explanation

The problem of determining sentiment of documents really consists of two subproblems. The first is generating a list or dictionary of some kind containing some sort of sentiment-term data. This could take the form of a list of sentiment terms, a list of sentiment phrases, and/or a list of grammatical structures that assign the sentiment of a term in one position to a noun in another, to name a few approaches. The second subproblem is that of actually using this sentiment-term data to assign sentiment to documents in a test set.

It is possible to essentially skip the first subproblem, as far as machine NLP is concerned. That is, one can manually generate a list of sentiment data by

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

referring to a dictionary, a thesaurus, and common grammatical knowledge. This might be an acceptable approach if it could be done definitively once. However, sentiment data for varying domains can be quite different. In automotive reviews, a reference to "spongy" brake pedal feel seems to be a case of negative sentiment. But the term "spongy" is probably irrelevant to sentiment in the context of movie reviews. There are many similar examples. Thus, a human would be required to manually develop a list of sentiment data for each new domain of documents. This is a significant drawback.

It is also possible to design an algorithm – often, in existing research, a rather complex one – to develop a body of sentiment data from a training set of documents. The drawback of such methods is that they can be relatively complicated, difficult to implement, and may suffer from long running times. If they work well, that is a price worth paying. But they do not work exceedingly well, as explained in the following section. I show that similar results can be achieved using a more straightforward approach than those attempted in previous research.

3 Previous Work

A key piece of research on sentiment classification involves several methods of classifying movie reviews into positive and negative categories (Pang et al., 2002). This paper limits the domain to documents that humans have classified as clearly positive or negative. It does not attempt to rank documents on a spectrum. The methods include two probabilistic approaches, both more involved than that presented here, and a support vector approach that creates vectors describing training documents and finds a hyperplane that best separates them. The best accuracy reported by these authors is 82.9% correctly classified.

(Turney, 2002), working on a similar task, tries an interesting method: using a Web search engine to find associations between various words and the words "poor" and "excellent," classifying words that co-occur frequently with "poor" and infrequently with "excellent" to be negative sentiment terms, and vice versa. Although he achieves impressive 84.0% accuracy on automotive reviews, his attempt at classifying movie reviews logged a lackluster 65.8% accuracy. Turney mentions that "descriptions of unpleasant scenes" could be hampering the movie-review results. This is not surprising, because his sentiment data is gleaned from a Web search of general documents, where words might be used very differently than in movie reviews – not to mention the dubious choice of the word "poor" as the flag for negative sentiment, when the word is frequently used in the economic sense.

(Yi et al., 2003) reports an interesting variation on sentiment analysis. They developed a method for mining the sentiment about particular attributes of an item from a document, rather than classifying the sentiment of the entire document. While this is not directly related to the work presented here, it is interesting because it goes beyond the common task of binary document-level classification. This work is an example of a highly structured approach to sentiment analysis – the researchers used predefined dictionaries of terms and sentiment-phrase structures. They found that accuracy was quite good – 85.6 % – on product reviews, but deteriorated rapidly when the corpus contained general Web documents where sentiment phrases were sparse.

4 Algorithm

The proposed approach to the first subproblem – extracting a list of sentiment terms from a training set – functions entirely on a unigram level, with one exception, to be discussed later. A training set of documents, each of which is associated with a numerical score (the range of possible scores is specified as a parameter), is used to construct a list of words. Each word is associated with two numerical values: the number of documents in which it was seen, and the sum of the scores of all documents in which it was seen. A given word is counted only once per document. When all the documents have been processed and the table has been filled, the table data is used to compute an average score for each word. This word-scoring formula appears in equation (1), where d represents a document, w represents a word, D represents a set of documents, and $S(\mathrm{x})$ represents the score of item x. The words are then ranked in order of their final scores, and the list of words and scores constitutes the output.

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

$$S(w) = \frac{\sum[S(d) \mid w \in d]}{\mid (D : d \mid w \in d) \mid} \quad (1)$$

I added one heuristic that uses a bigram model. The program contains a predefined list of negation words, such as "no" and "didn't". These words are never inserted into the word table. Instead, they indicate that the next word should be flagged as negated. This is done by prepending a '!' to the lexeme. For example, the phrase "didn't satisfy" would result in the word token !SATISFY being inserted in the word table (or its count being incremented, if it were already there).

It is worth noting that this approach produces a list containing a large number of terms that are not sentiment terms by any reasonable standard. One could argue that using such a list would be a case of overfitting the data. However, in a certain sense, all intelligent approaches to sentiment-data gathering involve overfitting the data. As alluded to previously, a good set of sentiment data for movie reviews would probably perform poorly when used to classify automotive reviews. A generic set of sentiment data would probably perform poorly on most domains, as (Turney, 2002) discovered. Either it would be so small as to miss most of the sentiment terms in each document, or it would be so large that it would assign sentiment to terms that were not sentiment-oriented within the domain. Thus, a good set of sentiment data for a given domain is defined functionally – as a set of data that performs well at judging sentiment for that domain. Whether the words in the list "look like" sentiment words is not particularly relevant. Incidentally, this implementation does ignore words containing capital letters, on the grounds that assigning sentiment to proper names might indeed be a case of excessive overfitting. Reviews of 1990 Hyundais are much more negative than reviews of 2004 Hyundais, so a corpus with a large number of early Hyundai reviews might result in negative sentiment being assigned to the word "hyundai", which is clearly undesirable even within the domain. But, in general, I do not view the inclusion of non-sentiment terms in the word table as a serious problem.

There are two key parameters that can be adjusted to change the results of the sentiment-term-discovery process. The parameter *f* is the number of times that a word must be seen in order to be included in the final word list. For example, a word that was only seen in one review might be rare, spurious or irrelevant. The parameter *n* is the number of words on each end of the sentiment spectrum that will be included in the final word list. One approach would be to essentially set this parameter to infinity, leaving the entire word list intact. Since words near the center of the spectrum are probably not sentiment words and may be irrelevant, I hypothesized that removing a portion from the middle of the word list might give better results when scoring test reviews. A sample of resulting words and scores from running this algorithm on a training set of movie reviews appears in Table 1. Note that scores are automatically normalized to a zero-to-one scale for consistency across corpora.

Word	*Score*
!FUNNY	0.415323
SLOG	0.426471
DISMAL	0.431818
REDEEMING	0.4375
UNFUNNY	0.446429
...	...
ENCHANTING	0.897727
TRANSCENDS	0.902778
RESIGNED	0.902778
BLESSED	0.908333
HEARTBREAKING	0.910256

Table 1: Sample output of sentiment-term extraction

Once the list of scored words has been generated, the second subproblem – ranking a test set of documents – can be attacked, essentially in the inverse way. Each test document receives the score equal to the average of the scores of the words appearing in the document. Words in the document that are not in the list of sentiment words are ignored. The output is a list of pairs of numbers – the score assigned to the document by the program, and the document's actual score. The document-scoring formula appears in equation (2), where *L* is the list of words resulting from the term-extraction process and *W* is a set of words.

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

$$S(d) = \frac{\sum[S(w) \mid (w \in d) \wedge (w \in L)]}{\mid [W : (w \in d) \wedge (w \in L)] \mid} \quad (2)$$

It should be noted that the assigned scores output by this algorithm tend to be packed very close to the middle of the score range, and thus are not directly usable as meaningful scores. They could obviously be normalized to the scale if meaningful scores for individual reviews were desired. The goal of this work was to output a ranking of documents, which could be compared to the true human-determined ranking for a clear picture of the overall performance of the algorithm. For a large body of documents, this seems to be closest to the sort of task we would want to perform in a real-world situation. It also allows for some telling visual representations of the results. Much previous work on sentiment classification has involved classifying documents merely as positive or negative. Aside from the fact that this seems not terribly useful in practical applications, a few statistics showing the percent of documents correctly classified in a two-bucket model does not exactly provide a nuanced look at the performance of the algorithm.

5 Results

I ran trials on two corpora: a very large set of Roger Ebert's movie reviews (approximately 4,000,000 words) and a smaller set of Consumer Reports Magazine's automotive reviews (approximately 86,000 words). I divided each corpus into a training set and a test set of approximately equal size. For each corpus, I tried several settings of the two parameters described previously. Results are presented in graphical form. The horizontal axis represents the ranking this algorithm assigned to a document. The vertical axis represents the human-assigned score of a document. Thus, completely correct output would be a nondecreasing y-value as x increases. In the case of the movie reviews, where the number of documents is large and the number of possible scores is small, this would look like a step function. Lines connecting the points have been omitted from these graphs for clarity.

While the graphs clearly deviate significantly from an ideal result, they do provide some promising evidence. The Ebert graphs in Figures 1 and 2 show an obvious trend of increasing y-values as the x-values increase, which is a desired result. The algorithm works rather well with reviews that received scores at an extreme, particularly those that received very low scores (0 and 0.5). The middle of the spectrum is somewhat muddled, though an upward trend is still visible. It should be noted that the graph of an ideal result would not show steps of an equal width, because there are fewer 0-score reviews than 2.5-score reviews, for example. Thus, the fact that the 2.5-score reviews form a wide cluster is not entirely unexpected. The higher setting of both parameters seems to give better performance at the lower and upper ends of the scale, as is evident in the graphs. I tried a number of different parameter settings not shown here. There was not a great deal of variation in the results from different parameter settings.

In addition to generating the graphs showing the ranking of all reviews, I also tried ignoring all documents with human-assigned scores other than 0, 0.5, 3.5 and 4, supposing that the goal was to correctly classify strongly negative and positive documents as such. I counted a document that appeared in the lower half of the overall ranking as being ranked negative, and a document appearing in the upper half as being ranked positive. On this scale, 81% of strongly negative and strongly positive documents were correctly classified in the best trial. The best accuracy achieved by (Pang et al., 2002) on a similar task using much more complex approaches was 82.9%, which was achieved through the use of support vector machines. Thus, this algorithm appears to be within striking distance of a known benchmark on the binary classification task.

The results from the Consumer Reports corpus, in Figures 3 and 4, were similar in character to the Ebert results, though appear less informative due to the small size of the corpus. While there is a correct upward trend in the graphs of results, there is quite a bit of fluctuation. However, it is notable that, if the task is recast as a binary decision task, where reviews with a human-assigned score of 80 points or higher are considered positive, those with a human-assigned score of 40 points or lower are considered negative (a larger section of low scores was used because almost no reviews in the corpus have human-assigned scores below 20 points), and all other reviews are ignored, the classification ac-

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

curacy is 87%, which is quite good. In fact, this is better than the accuracy numbers reported by any of the similar binary classification experiments noted in Section 3. This is, no doubt, at least partially due to the nature of the Consumer Reports documents, which are written in a very straightforward and unembellished style that does not wander off on tangents not directly related to the product being reviewed, as movie reviews sometimes tend to do. None of the other sentiment-classification research I have examined used Consumer Reports documents, so the accuracy number presented here is not directly comparable. If a larger Consumer Reports corpus were compiled, and some additional heuristics possibly added to the algorithm, I would not be surprised to see accuracy approaching 90%. Further experimentation in this direction is warranted.

6 Future Work

Assembling a larger corpus of Consumer Reports documents for further experimentation is an obvious next step. I would expect better results for a larger training set. Some further heuristics, perhaps including additional bigram analysis or part-of-speech sensitivity, may be worth trying. However, I would be careful to add such features gradually and conservatively. The results presented here show that other researchers who began with fairly complex approaches may be over-thinking the problem.

7 Conclusions

Automatic sentiment classification is a relatively difficult problem. However, when a large corpus is available, a simple probabilistic approach yields results that are nearly comparable to those derived from substantially more complicated algorithms. Furthermore, this simple approach lends itself naturally to creating an ordered ranking of documents by sentiment, rather than merely classifying documents into two buckets. This may be a much more useful approach in real-world applications. A human who is trying to read reviews of products will likely want to read the best few reviews, not the best half, particularly when thousands of reviews are available. The approach presented here is inherently suited for generating such results.

References

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of the Conference on Empirical Methods in NLP, July 2002, pp. 79-86*.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proc. of the ACL*

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, Wayne Niblack. 2003. Sentiment analyzer: extracting sentiment about a given topic using natural language processing techniques. In *Third IEEE International Conference on Data Mining, Nov 2003*.

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

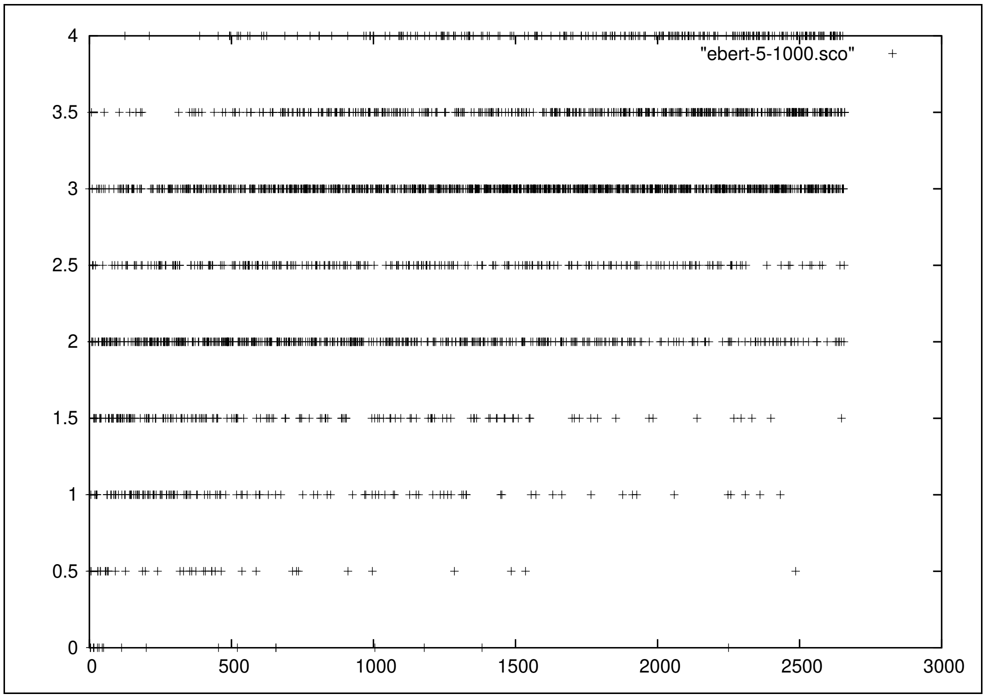

Figure 1: Results for Ebert corpus with *f*=5 and *n*=1000

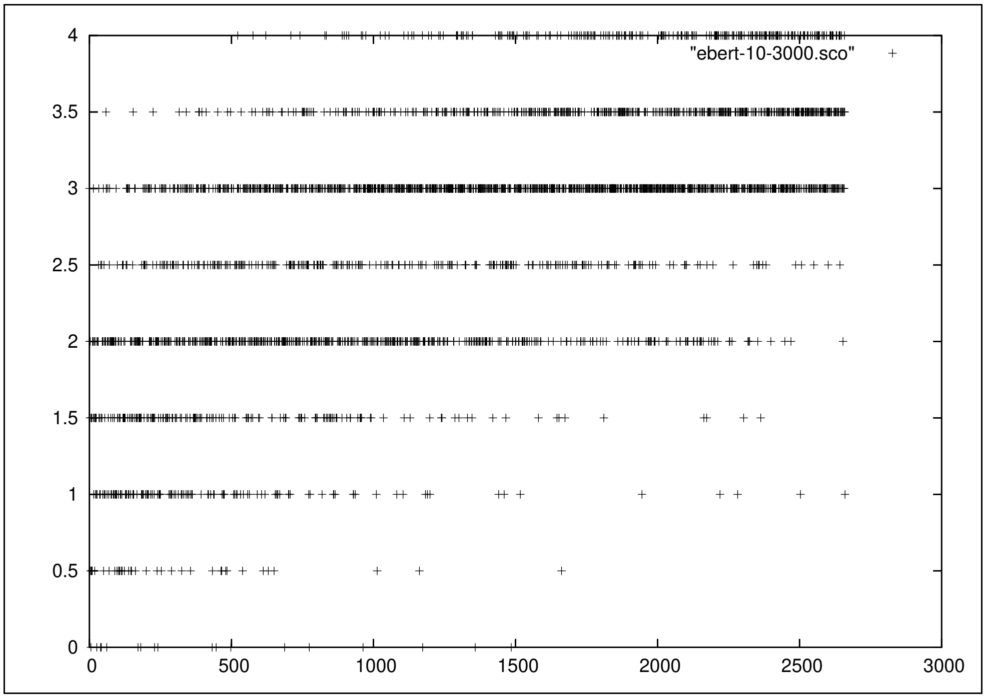

Figure 2: Results for Ebert corpus with *f*=10 and *n*=3000

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

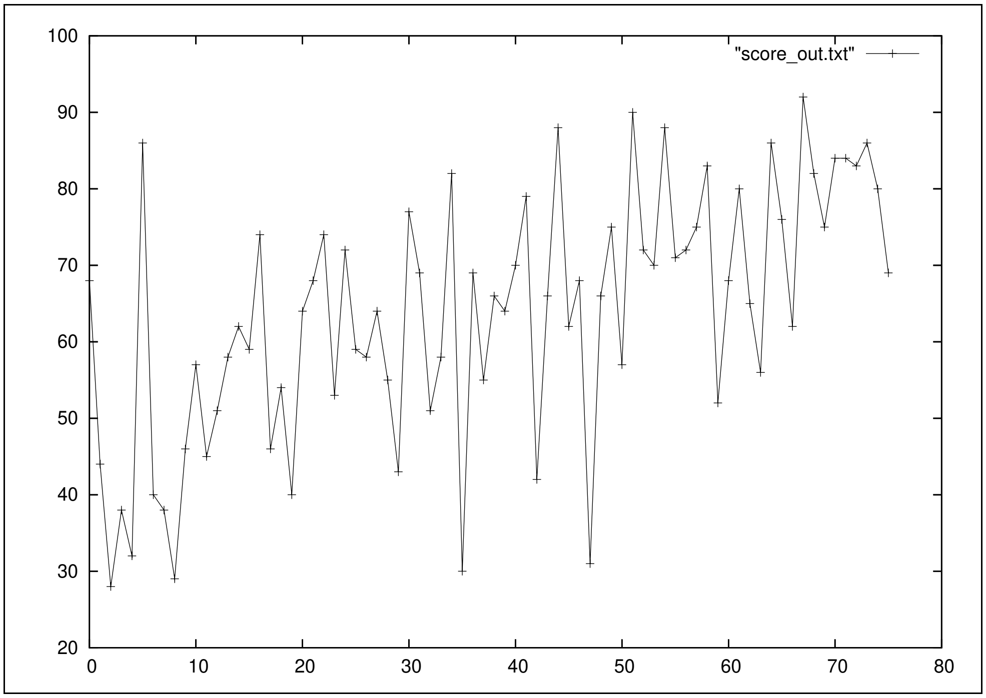

Figure 3: Results for Consumer Reports corpus with f=2 and n=500

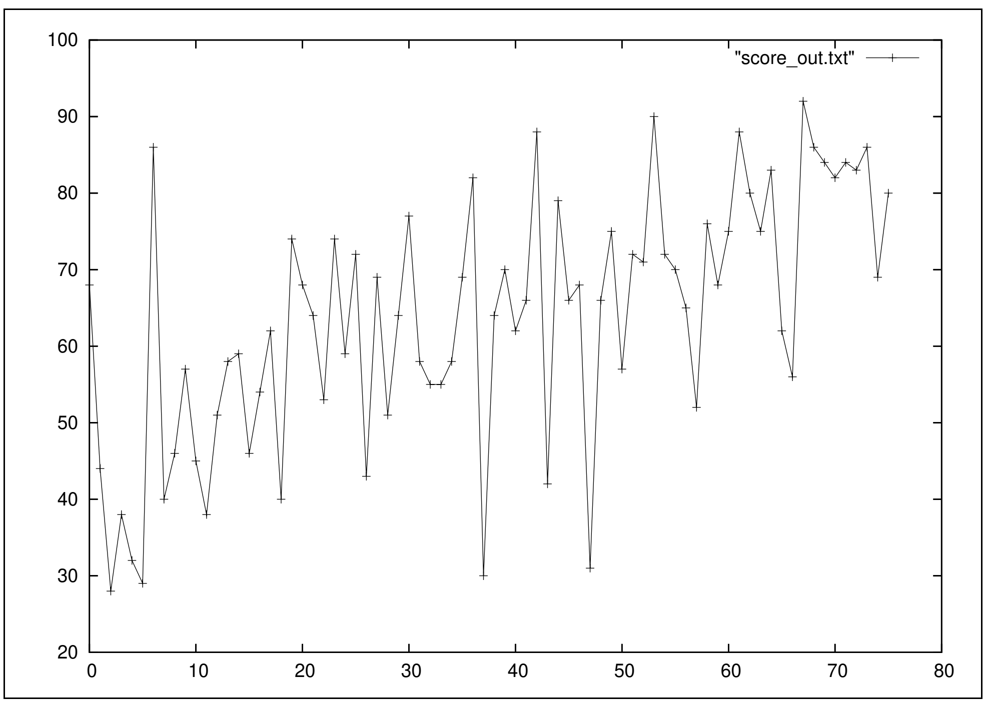

Figure 4: Results for Consumer Reports corpus with f=2 and n=∞

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College