Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web

Sanjiv R. Das*
Santa Clara University
Santa Clara, CA 95053.

Mike Y. Chen
Intel Research
Seattle, WA 98105.

January 5, 2006

Abstract

Extracting sentiment from text is a hard semantic problem. We develop a methodology for extracting small investor sentiment from stock message boards. The algorithm comprises different classifier algorithms coupled together by a voting scheme. Accuracy levels are similar to widely used Bayes classifiers, but false positives are lower and sentiment accuracy higher. Time series and cross-sectional aggregation of message information improves the quality of the resultant sentiment index, particularly in the presence of slang and ambiguity. Empirical applications evidence a relationship with stock values – aggregate tech sector sentiment is found to predict stock index levels, but not at the individual stock level. The algorithms may be used to assess the impact on investor opinion of management announcements, press releases, third-party news, and regulatory changes.

*We owe a special debt to the creative environment at UC Berkeley's Computer Science Division, where this work was begun. The comments of the referees on the paper contributed immensely to many improvements. Thanks to David Levine for many comments and for the title. We are grateful to Vikas Agarwal, Chris Brooks, Yuk-Shee Chan, David Gibson, Geoffrey Friesen, David Leinweber, Asis Martinez-Jerez, Patrick Questembert, Priya Raghubir, Sridhar Rajagopalan, Ajit Ranade, Mark Rubinstein, Peter Tufano, Raman Uppal, Shiv Vaithyanathan, Robert Wilensky and seminar participants at Northwestern University, UC Berkeley-EECS, London Business School, University of Wisconsin, Madison, the Multinational Finance Conference, Italy, the Asia Pacific Finance Association Meetings, Bangkok, European Finance Association Meetings, Barcelona, Stanford University for helpful discussions and insights. Danny Tom and Jason Waddle were instrumental in delivering insights into this paper through joint work on alternative techniques via support vector machines. The first author gratefully acknowledges support from a Breetwor Fellowship, the Dean Witter Foundation, and a Research Grant from Santa Clara University. Please address all correspondence to Sanjiv Das, Breetwor Fellow & Associate Professor, Santa Clara University, Leavey School of Business, Dept of Finance, 208 Kenna Hall, Santa Clara, CA 95053-0388. Email: srdas@scu.edu. Mike Chen is at Intel Research, Email: mikechen@cs.berkeley.edu.

1 Introduction

> *"Language is itself the collective art of expression, a summary of thousands upon thousands of individual intuitions. The individual gets lost in the collective creation, but his personal expression has left some trace in a certain give and flexibility that are inherent in all collective works of the human spirit" – Edward Sapir, cited in "Society of Mind" by Minsky (1985).*

We develop hybrid methods for extracting opinions in an automated manner from discussions on stock message boards, and analyze the performance of various algorithms in this task, including that of a widely used classifier available in the public domain. The algorithms are used to generate a sentiment index and we analyze the relationship of this index to stock values. As we will see, this analysis is efficient, and useful relationships are detected.

The volume of information flow on the web has accelerated. For example, in the case of Amazon Inc., there were cumulatively 70,000 messages by the end of 1998 on Yahoo's message board, and this had grown to about 900,000 messages by end 2005. There are almost 8000 stocks for which message board activity exists, across a handful of message board providers. The message flow comprises valuable insights, market sentiment, manipulative behavior, and reactions to other sources of news. Message boards have attracted the attention of investors, corporate management, and of course, regulators.[1]

In this paper, "sentiment" takes on a specific meaning, that is, the net of positive and negative opinion expressed about a stock on its message board. Hence, we specifically delineate our measure from other market conventions of sentiment such as deviations from the expected put-call ratio. Our measure is noisy, since it comprises information, sentiment, noise and estimation error.

Large institutions express their views on stocks via published analyst forecasts. The advent of stock chat and message boards enables small investors to express their views too, frequently and forcefully. We show that it is possible to capture this sentiment using *statistical language techniques*. Our algorithms are validated using revealed sentiment on message boards, and from the statistical relationship between sentiment and stock returns, which track each other.

Posted messages offer opinions that are bullish, bearish, and many that are confused, vitriolic, rumor, and spam (null messages). Some are very clear in their bullishness, as is the following message on Amazon's board (Msg 195006):

[1]Das, Martinez-Jerez and Tufano (2005) present an empirical picture of the regularities found in messages posted to stock boards. The recent case of Emulex Corp highlights the sensitivity of the internet as an sentiment channel. Emulex's stock declined 62% when an anonymous, false news item on the web claimed reduced earnings and the resignation of the CEO. The Securities Exchange Commission (SEC) promptly apprehended the perpetrator, a testimony to the commitment of the SEC to keeping this sentiment channel free and fair. In relation to this, see the fascinating article on the history of market manipulation by Leinweber and Madhavan (2001).

```
The fact is.......
The value of the company increases because the
leader (Bezos) is identified as a commodity
with a vision for what the future may hold.  He
will now be a public figure until the day he
dies.  That is value.
```

In sharp contrast, this message was followed by one that was strongly bearish (Msg 195007):

```
Is it famous on infamous?  A commodity dumped
below cost without profit, I agree.  Bezos had
a chance to make a profit without sales tax and
couldn't do it.  The future looks grim here.
```

These (often ungrammatical) opinions provide a basis for extracting small investor sentiment from discussions on stock message boards.

While financial markets are just one case in point, the web has been used as a medium for information extraction in fields such as voting behavior, consumer purchases, political views, quality of information equilibria, etc., (see Godes and Mayzlin (2001), Lam and Myers (2001), Wakefield (2001), Admati and Pfleiderer (2000) for examples). In contrast to older approaches such as investor questionnaires, sentiment extraction from web postings is relatively new. It constitutes a *real-time* approach to sentiment polling, as opposed to traditional *point-in-time* methods.

We use statistical and natural language processing techniques to elicit *emotive* sentiment from a posted message; we implement five different algorithms, some language-dependent, others not, using varied parsing and statistical approaches. The methodology used here has antecedents in the text classification literature (see Koller and Sahami (1997) and Chakrabarti, Dom, Agrawal and Raghavan (1998)). These papers classify textual content into natural hierarchies, a popular approach employed by web search engines.

Extracting the *emotive* content of text, rather than *factual* content, is a complex problem. Not all messages are unambiguously bullish or bearish. Some require context, which a human reader is more likely to have, making it even harder for a computer algorithm with limited background knowledge. For example, consider the following from Amazon's board (Msg 195016):

```
You're missing this Sonny, the same way the
cynics pronounced that ``Gone with the Wind''
would be a total bust.
```

Simple, somewhat ambiguous messages like this also often lead to incorrect classification even by human subjects. We analyze the performance of various algorithms in the presence of ambiguity, and explore approaches to minimizing its impact.

The technical contribution of this paper lies in the coupling of various classification algorithms into a system that compares favorably with standard Bayesian approaches, popularized by the phenomenal recent success of spam-filtering algorithms. We develop metrics to assess

algorithm performance that are well-suited to the finance focus of this work. There are unique contributions within the specific algorithms used as well as accuracy improvements overall, most noticeably in the reduction of false positives in sentiment classification. A new approach for filtering ambiguity is also devised and shown to be useful in characterizing algorithm performance.

Recent evidence suggests a link between small investor behavior and stock market activity. Noticeably, day-trading volume has spurted.[2] Choi, Laibson and Metrick (2002) analyze the impact of a web-based trading channel on the trading activity in corporate 401(k) plans, and find that the "web effect" is very large – trading frequency doubles, and portfolio turnover rises by over 50 percent, when investors are permitted to use the web as an information and transaction channel. Wysocki (1998), using pure message counts, reports that variation in daily message posting volume is related to news and earnings announcements. Lavrenko, Schmill, Lawrie, Ogilvie, Jensen, and Allen (2001) use computer algorithms to identify news stories that influence markets, and then trade successfully on this information. Bagnoli, Beneish and Watts (1999) examined the predictive validity of whisper forecasts, and found them to be superior to those of First Call (Wall Street) analysts.[3] Antweiler and Frank (2004) examine the bullishness of messages and find that while web talk does not predict stock movements, it is predictive of volatility. Tumarkin and Whitelaw (2001) also find similar results using self-reported sentiment (not message content) on the Raging Bull message board. Antweiler and Frank (2002) argue that message posting volume is a priced factor, and higher posting activity presages high volatility and poor returns. These results suggest the need for algorithms that can rapidly access and classify messages with a view to extracting sentiment – the goal of this paper.[4] The analyses presented in this paper, confirm many of these prior empirical findings, and extend them as well. Granger-cause tests using a tech-sector sentiment index developed here show prediction of stock values, which is not evidenced at the individual stock level, suggesting both, that cross-sectional aggregation of sentiment improves the quality of the tech index, and the presence of cross-autocorrelations in sentiment.

Overall, this paper comprises two parts: (i) methodology and validation, in Section 2, which presents the algorithms used, and their comparative performance. (ii) The empirical relationship of market activity and sentiment, in Section 3. Section 4 contains discussion and conclusions.

[2]Business Week, 23rd May, 2001 cites a Bear Stearns report that reports a huge spurt in volume, and a total number of day-traders in excess of 50,000.

[3]The "whisper" number, an aggregate of informal earnings forecasts self-reported by individual investors is now watched extensively by market participants, large and small. Whispers are forecasts of the quarterly earnings of a firm posted to the web by individuals in a voluntary manner. The simple average of these forecasts is presented on the whisper web page, along with the corresponding forecast from First Call, which is an aggregate of the sentiment of Wall Street analysts.

[4]In contrast, Antweiler and Frank (2005) recently used computational linguistic algorithms to sort news stories into topics, instead of sentiment, and uncovered many interesting empirical regularities relating news stories and stock values.

2 Methodology

2.1 Overview

The first part of the paper is the extraction of opinions from message board postings to build a sentiment index. Messages are classified by our algorithms into one of 3 types: bullish (optimistic), bearish (pessimistic) and neutral (comprising either spam or messages that are neither bullish nor bearish). We use five algorithms, each with different conceptual underpinnings, to classify each message. These comprise a blend of language features such as parts of speech tagging, and more traditional statistical methods.[5] Before initiating classification, the algorithms are tuned on a training corpus, i.e. a small subset of pre-classified messages used for training the algorithms.[6] The algorithms "learn" sentiment classification rules from the pre-classified data set, and then apply these rules out-of-sample. A simple majority across the five algorithms is required before a message is finally classified, else it is discarded. This *voting* approach results in a better signal to noise ratio for extracting sentiment.

Figure 1 presents the flowchart for the methodology and Appendix A contains technical details. The sequence of tasks is as follows. We use a "web-scraper" program to download messages from the internet, which are fed to the five classification algorithms to categorize them as buy, sell or null types. Three supplementary databases support the classification algorithms.

- First, an electronic English "dictionary", which provides base language data. This comes in handy when determining the nature of a word, i.e. noun, adjective, adverb, etc.
- Second, a "lexicon" which is a hand-picked collection of finance words (such as bull, bear, uptick, value, buy, pressure, etc). These words form the variables for statistical inference undertaken by the algorithms. For example, when we count positive and negative words in a message, we will use only words that appear in the lexicon, where they have been pre-tagged for sign.
- The third database is the "grammar" or the pre-classified training corpus. It forms the base set of messages for further use in classification algorithms. These pre-classified messages provide the in-sample statistical information for use on the out-of-sample messages.

These 3 databases (described in the Appendix) are used by 5 algorithms (denoted "classifiers")

[5]This paper complements techniques such as support vector machines that are optimization methods that classify content. See the papers by Vapnik (1995), Vapnik and Chervonenkis (1964), Joachims (1999) for a review. A recent paper by Antweiler and Frank (2004) uses SVMs to carry out an exercise similar to the one in this paper. These approaches are computationally intensive and are often run on parallel processors. Moreover, they have been used for more than 30 years, and the technology is well-developed. In this paper we did not employ support vector machines, choosing to focus on purely analytic techniques that did not require optimization methods in the interests of computational efficiency.

[6]The training corpus is kept deliberately small to avoid over-fitting, which is a common ailment of text classification algorithms.

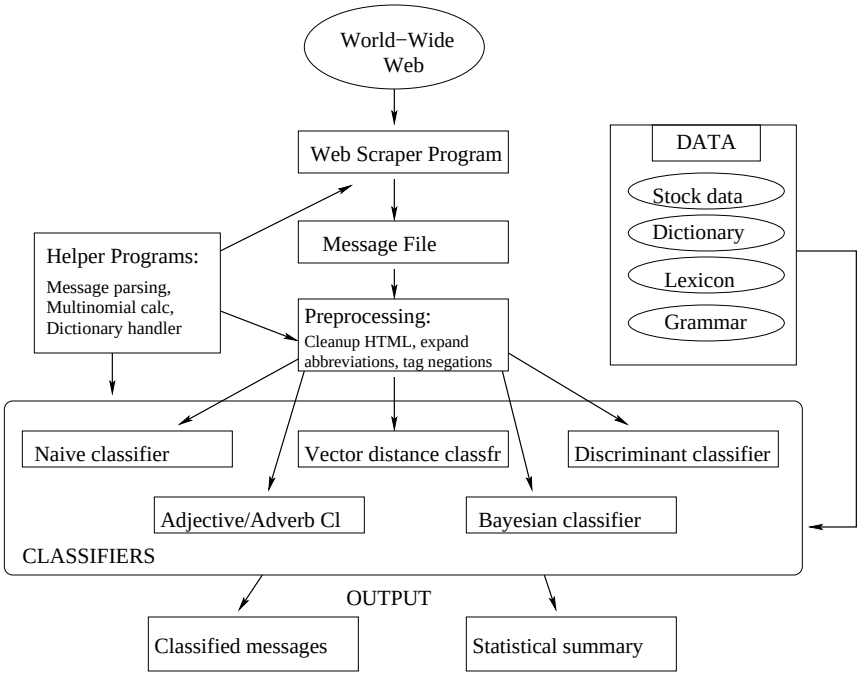

Figure 1: Schematic of the Algorithms and System Design used for Sentiment Extraction.

to arrive at the 3-way classification of each message.

2.2 Classifiers

Each of our five classifier algorithms relies on a different approach to message interpretation. Some of them are language-independent, and some are not. Each approach is intuitive. They are all analytical, and do not require any lengthy optimization, or convergence issues, hence they are computationally efficient, making feasible the processing of huge volumes of data in real time. We describe each one in turn.

2.2.1 Naive Classifier

This algorithm is based on a word count of positive and negative connotation words. It is the simplest and most intuitive of the classifiers. Recall that the lexicon is a list of hand-picked words that we found to be commonly used to describe stocks. Each word in the lexicon is identified as being positive, negative or neutral. Each lexical entry is matched by a corresponding counterpart with a negation sign (see the Appendix for an example). Before processing any message, it is treated by a parsing algorithm that negates words if the context of the sentence requires it; for example, if the sentence reads "this stock is not good", the word `good` is replaced by

`good__n`

signifying a negation. It would then be counted as a sell word, rather than a buy one.

Each word in a message is checked against the lexicon, and assigned a value $(-1, 0, +1)$ based on the default value (sell, null, buy) in the lexicon. After this assignment, the net word count of all lexicon matched words is taken, and if this value is greater than 1, we sign the message as a buy. If this value is less than 1, the message is taken to be a sell. All other messages are treated as neutral. The threshold value of 1 was chosen by experiment, and this may be adjusted for other applications. We did not subsequently attempt to improve the threshold value, so as to eliminate data-snooping bias. This classifier depends critically on the composition of the lexicon. By choosing words carefully in the lexicon this approach may be adapted to other uses.

2.2.2 Vector Distance Classifier

If there are D words in the lexicon, and each word is assigned a dimension in vector space, then the lexicon represents a D-dimensional unit hypercube. Every message may be thought of as a word vector ($m \in R^D$) in this space. The elements in the vector take values in the set $\{0, 1, 2, ...\}$ depending on how many times a word appears in the message. Suppose the lexicon contains about 300 words. Then, each message may be characterized as a vector in 300-dimension space. As would be expected, each message contains a few lexical words only, and is therefore represented by a sparse vector.

A hand-tagged message (or grammar rule) in the training corpus (grammar) is converted into a vector G_j, and occupies a location in this D-dimensional Euclidian space. Each new message is classified by comparison to the cluster of pretrained vectors in this space. The angle θ_j between the message vector (m) and the vectors in the grammar (G_j) provides a measure of closeness, i.e.

$$\cos(\theta_j) = \frac{m.G_j}{|m|.|G_j|} \in [0, 1], \quad \forall j \tag{1}$$

where $|X|$ stands for the norm of vector X. Each message is assigned the classification of the grammar rule with which it has the smallest angle, i.e. that of $\min[\cos(\theta_j)]$ (variations on this theme could use sets of top-n closest vectors). Since $\cos(\theta_j) \in [0, 1]$, the vector distance classifier provides a measure of proximity in the form of percentage closeness – when the angle is small, $\cos(\theta_j)$ is closer to 1.

2.2.3 Discriminant-Based Classifier

The naive classifier (NC) weights lexical words equally. However, lexical words may have differential importance for classification. Some words, such as “buy” may be more indicative

of sentiment than words such as "position". Using the training corpus, we compute a measure of the discriminating ability of each word in our lexicon. We then replace the simple word count in the naive algorithm (NC) by a weighted word count.

The weights are based on a simple discriminant function for each word (see Chakrabarti, Dom, Agrawal and Raghavan (1998) and Chakrabarti, Roy and Soundalgekar (2003) - the latter paper demonstrates the usefulness of using the so-called Fisher discriminant statistic). Let the set $i = \{null, sell, buy\}$ index the categories for our messages, and n_i be the number of messages in each category. Let the average number of times word w appears in a message of category i be denoted μ_i. The number of times word w appears in a message j of category i is denoted m_{ij}. The discriminant formula for each word is:

$$F(w) = \frac{\frac{1}{3}\sum_{i\neq k}(\mu_i - \mu_k)^2}{\left(\sum_i \sum_j (m_{ij} - \mu_i)^2\right) / \left(\sum_i n_i\right)}, \quad \forall w. \tag{2}$$

This equation assigns a score $F(w)$ to each word w in the lexicon, which is the ratio of the mean across-class squared variation to the average of within-class squared variation. The larger the ratio, the greater the discriminating power of word w in the lexicon. A good discriminant word maximizes across-class variation and minimizes within-class variation. Appendix C provides examples of the discriminant values of some of the words in the lexicon.

Each word in a message is checked against the lexicon, and assigned a signed value $(-d, 0, +d)$, based on the sign (sell=-1, null=0, buy=$+1$) in the lexicon multiplied by the discriminant value $d = F(w)$. After this assignment, the net word count of all lexicon matched words is taken, and if this value is greater than 0.01 (threshold), we sign the message as a buy. If this value is less than -0.01, the message is taken to be a sell. All other messages are treated as neutral. Again, the threshold was not improved subsequently so as to keep the empirical experiments in the sequel fair.

2.2.4 Adjective-Adverb Phrase Classifier

This classifier is based on the assumption that adjectives and adverbs emphasize sentiment and require greater weight in the classification process. This algorithm uses a word count, but restricts itself to words in specially chosen phrases containing adjectives and adverbs. Hence, the goal here is to focus only on the emphatic portions of the message.

We wrote program logic for a parts of speech "tagger" which, in conjunction with the dictionary, searches for noun phrases containing adjectives or adverbs (i.e. in its simplest form, this would be an adjective-noun pair). Whenever this is detected, we form a "triplet", which consists of the adjective or adverb and the two words immediately following or preceding it in the message. This triplet usually contains meaningful interpretive information because it contains the adjective or adverb, both of which are parts of speech that add emphasis to

the phrase in which they are embedded. This simple heuristic identifies significant phrases, and the lexicon is used to determine whether these connote positive or negative sentiment. If the net count in these phrases was greater than equal to 1 (-1), a positive (negative) tag is assigned to the message, else it is neutral.

2.2.5 Bayesian Classifier

The Bayesian classifier relies on a multivariate application of Bayes' theorem (see Mitchell (1997), Neal (1996), Koller and Sahami (1997) (KS), Chakrabarti, Dom, Agrawal and Raghavan (1998) (CDAR)). Recently, it has been used for web search algorithms, for detecting web communities, and in classifying pages on internet portals.[7] These methods are now also widely used for spam filters, and the following description summarizes the ideas of prior work as specifically applied here.

The classifier comprises three components: (i) lexical words, (ii) message text, and (iii) classes or categories (bullish, bearish, or neutral), resulting in a word-message-class (w, m, c) model. The Bayesian classifier uses word-based probabilities, and is thus indifferent to the structure of the language. Since it is language-independent, it has wide applicability, which enables investigation of message boards in other financial markets, where the underlying language may not be English.

The notation is as follows. The total number of categories or classes is $C(=3)$, $c_i, i = 1...C$. Each message is denoted $m_j, j = 1...M$, where M is the total number of messages. We define M_i as the total number of messages per class i, and $\sum_{i=1}^{C} M_i = M$. Words (w) are indexed by k, and the total number of lexical words is D. The set of lexical words is $F = \{w_k\}_{k=1}^{D}$.

Let $n(m_j, w_k)$ be the total number of times word w_k appears in message m_j. We maintain a count of the number of times each lexical item appears in every message in the training data set. This leads naturally to the variable $n(m_j)$, the total number of lexical words in message m_j including duplicates. This is a simple sum, $n(m_j) = \sum_{k=1}^{D} n(m_j, w_k)$.

An important quantity is the frequency with which a word appears in a message class. Hence, $n(c_i, w_k)$ is the number of times word w appears in all $m_j \in c_i$. This is $n(c_i, w_k) = \sum_{m_j \in c_i} n(m_j, w_k)$. This measure has a corresponding probability: $p(c_i, w_k)$ is the probability

[7]Koller and Sahami (1997) develop a hierarchical model, designed to mimic Yahoo's indexing scheme. Hence their model has many categories and is more complex. On the other hand, their classifier was not discriminating emotive content, but factual content, which is arguably more amenable to the use of statistical techniques. Our task is complicated because the messages contain opinions, not facts, which are usually harder to interpret. The reader may obtain details of the hierarchical scheme by referring to the technical descriptions in Koller and Sahami (1997) and Chakrabarti, Dom, Agrawal and Raghavan (1998) for the document model approach of a naive Bayes classifier. The exposition here briefly summarizes these approaches. We modify but try to retain as closely as possible the notation of the naive Bayes classifier of Chakrabarti, Dom, Agrawal and Raghavan (1998).

with which word w_k appears in all messages m in class c:

$$p(c_i, w_k) = \frac{\sum_{m_j \in c_i} n(m_j, w_k)}{\sum_{m_j \in c_i} \sum_k n(m_j, w_k)} = \frac{n(c_i, w_k)}{n(c_i)} \tag{3}$$

We require that $p(c_i, w_k) \neq 0$, $\forall c_i, w_k$. Hence, an adjustment is made to equation (3) via Laplace's formula which is

$$p(c_i, w_k) = \frac{n(c_i, w_k) + 1}{n(c_i) + D}.$$

If $n(c_i, w_k) = 0$ and $n(c_i) = 0, \forall k$, then every word is equiprobable, i.e. $\frac{1}{D}$. We now have the required variables to compute the conditional probability of a message j in category i, i.e. $\Pr[m_j|c_i]$:

$$\Pr[m_j|c_i] \quad = \quad \frac{n(m_j)!}{\prod_{k=1}^{D} n(m_j, w_k)!} \times \prod_{k=1}^{D} p(c_i, w_k)^{n(m_j, w_k)}.$$

We also compute $\Pr[c_i]$, the proportion of messages in the training set classified into class c_i.

The classification goal is to compute the most probable class c_i given any message m_j. Therefore, using the previously computed values of $\Pr[m_j|c_i]$ and $\Pr[c_i]$, we obtain the following conditional probability (applying Bayes' theorem):

$$\Pr[c_i|m_j] = \frac{\Pr[m_j|c_i].\Pr[c_i]}{\sum_{i=1}^{C} \Pr[m_j|c_i].\Pr[c_i]}. \tag{4}$$

For each message, equation (4) delivers three posterior probabilities, $\Pr[c_i|m_j], \forall i$, one for each message category. The message is classified as being from the category with the highest probability.

2.3 Voting amongst Classifiers

All the classifier methods used here are analytical and do not require optimization or search algorithms. Hence, there are no issues of numerical convergence. Given the huge data sets involved, this is an important consideration in the overall algorithm design. The numerical speed of the algorithms is complemented by enhancing statistical reliability using a voting scheme, based on the intuition that all available information is not exploited when classifiers are used in isolation, instead of in conjunction. We will see that the primary benefit of the voting scheme lies in reducing false positives.

Final classification is based on achieving a simple majority vote amongst the five classifiers, i.e. 3 of 5 classifiers should agree on the message type. If a majority is not obtained the message is not classified. This approach marginally reduces the number of messages classified, but enhances classification accuracy.

2.4 Training and Evaluation

The classification algorithms are initially trained using a portion of the data, which we designate as the "training set", which we typically wanted to restrict to a size of less than 1,000 messages. The number of messages is deliberately kept small so as to assess whether the classifiers are amenable to a minimal amount of training. Hence, our approach is biased against performing well in-sample versus commercial Bayes classifiers. But, the small training set also prevents overfitting of the data (leading to poor out-of-sample performance), a common ailment in text classification algorithms.

2.5 Metrics

Our goal is to develop a sentiment index formed from a time-series accumulation of the sentiment from individual messages. The quality of this index will depend on the performance of our classification algorithms. We developed various assessment metrics, and also compared our models to a widely-used algorithm in the public domain.

First, a standard approach to measuring the performance of classifiers is to examine the "confusion matrix" for statistical significance. The confusion matrix is a tableau that presents a cross-classification of actual message type versus classified message type. The confusion matrix has 3 rows and 3 columns. Each of the rows signifies the sentiment (Hold, Sell or Buy) posted by the author of a message to the stock board. The columns detail how many of these messages were classified in each of 3 categories: Null, Sell, Buy. The greater the weight of the diagonal of the confusion matrix, the lesser the confusion experienced by the algorithm. The null hypothesis for our test postulates no classification ability of the algorithm, i.e. the rows and columns of the confusion matrix are independent. We checked this using a standard χ^2 test.

$$\chi^2(4) = \frac{1}{9}\sum_{i=1}^{9}\frac{(O_i - E_i)^2}{E_i}, \quad (d.o.f = 4) \tag{5}$$

where O_i are the elements of the observed confusion matrix, and E_i are the elements of the matrix when no classification ability is present.

As a first step in assessment of algorithm performance, we collected and hand-classified a few hundred messages to comprise a training dataset. We tuned the classifiers on the training set, and then undertook classification on testing sets. The quality of text on message boards is very poor, resulting in a hard problem even for human classification. Our ultimate goal lies in developing a sentiment index formed from the cumulated classification of messages over time, where buys, holds and sells are $\{+1, 0, -1\}$ respectively.

Second, in addition to the χ^2 metric described above, we gain a useful understanding of the algorithm by examining the percentage of messages correctly classified. Note that the low clarity of messages implies that quite often, people would be likely to disagree on the

classification. In order to gauge the extent of ambiguity, a re-classification of the training corpus was undertaken by a second human subject. This subject had nothing to do with the design of the study, and is not one of the authors. We believe that no bias existed, even for this informal test. Of the 374 training messages, and 64 test messages, the two human subjects agreed on on the classification of only 72.46% of the messages. We may like to think of the mismatch percentage of 27.54% (100.00-72.46) as the "ambiguity coefficient" of the message boards. A more stable version of this coefficient would be one obtained from many (say n) human subjects, for reasonably large n (approximately $n \sim 10$), where the agreement percentage is based on the consensus of all n people. This might well result in an ambiguity coefficient a little higher than from just a few subjects. It is also intuitive that as we increase n, the ambiguity coefficient will first rise rapidly and then taper off to an asymptote, as there will be a core set of messages on which there can be little disagreement. Hence there are two benchmarks of algorithm performance. One is perfect performance, i.e. a comparison with 100% accuracy rates, and the second is the human benchmark, i.e. an "agreement" coefficient, equivalent to 100 minus the ambiguity coefficient.

A third useful metric is the extent to which false positives occur. These are buy messages classified as sells and vice versa. We report the percentage of false positives to total messages. Note that the value of the sentiment index is doubly impacted if a false positive error is made, because sentiment is incremented by the wrong sign. Hence such errors tend to be more costly, than other types of misclassification.

Fourth, in addition to false positives, we compare the value of the aggregate sentiment given no classification error versus that obtained based on our classifier. Note that if the classification error has no bias, then it is likely to have a smaller effect on the index than if there is bias.

To summarise, we examine four different metrics of classification performance: percentage classification accuracy, percentage of false positives, percentage error in aggregate sentiment, and a χ^2 test of no classification ability. We report the results for all our five individual algorithms, and for the majority voting algorithm. This voting scheme is applied in two ways, one, where messages with no majority are treated as "hold" messages, reported as type "Vote" in the subsequent tables, and two, where such messages are discarded, reported as "Vote-d" type in the tables. Finally, we also applied a popular, well-known software tool, the `Rainbow` algorithm of McCallum (1996).[8] This is a highly optimized and widely-used algorithm and provides a good benchmark of performance.

First, in Table 1, Panel A, we run the classifier in-sample, i.e. setting the training data set to be the testing one. As is to be expected the algorithms perform quite well. Note that, unlike our algorithms that use a fixed lexicon, the Rainbow algorithm first undertakes a scan of words in the training set to determine which words are the best discriminants and then does

[8]This algorithm is available for download from `http://www.cs.umass.edu/~mccallum/code-data.html`.

Table 1: Tests of various algorithms. In this table, we present statistics of the performance of various approaches to message classification. Five basic algorithms as described in the text are used: (i) Naive, (ii) Vector distance, (iii) Discriminant weighted, (iv) Adjective-Adverb, (v) Bayes. The results of a majority vote are given, as are the same when no-majority votes are discarded ("Vote-d", "d" = discard). The results of the `Rainbow` algorithm are also presented. We note that this algorithm is a Bayes classifier that develops a word set from the training set; the one in our algorithm uses an independent word set that is hand generated from the training set and other messages. Panel A has in-sample results from a training set of 374 messages taken from 1999. Panel B uses this training set on a small out-sample of 913 messages taken from July-August, 2001. Panel C uses a large out-sample of about 50,000 messages, covering all messages for 25 tickers. The first three measures are expressed in percentage terms. Sentiment error is expressed without sign.

Panel A: In-sample

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	92.2460	0.2674	0.5348	64.2581	374
vecDist	49.1979	12.2995	23.5294	6.1179	374
DiscWtd	45.7219	16.0428	35.0267	4.4195	374
AdjAdv	63.3690	7.7540	20.3209	17.4351	374
BayesCl	60.6952	8.2888	14.4385	13.4670	374
Vote	58.2888	7.7540	17.3797	11.1799	374
Vote-d	62.8743	8.6826	17.0659	14.7571	334
Rainbow	97.0430	0.8065	3.2258	75.2281	372

Panel B: Out-of-sample

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	25.7393	18.2913	6.4622	2.0010	913
vecDist	35.7065	25.8488	5.8050	1.4679	913
DiscWtd	39.1019	25.1917	4.1621	2.6509	913
AdjAdv	31.3253	29.7919	51.3691	0.5711	913
BayesCl	31.5444	19.8248	12.2673	2.0873	913
Vote	30.0110	17.8532	8.3242	2.1955	913
Vote-d	33.1242	20.4517	7.7792	2.3544	797
Rainbow	33.1140	33.0044	38.7061	0.5458	912

Panel C: Out-of-sample

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	27.6024	20.8000	9.5031	33.8485	50952
vecDist	38.4224	25.7281	8.5728	103.1038	50952
DiscWtd	40.6049	25.0530	5.8172	131.8502	50952
AdjAdv	32.6366	30.2186	54.4434	35.2341	50952
BayesCl	33.2254	19.9835	14.1525	128.9712	50952
Vote	31.8614	18.0268	10.2665	136.8215	50952
Vote-d	35.8050	20.8978	11.0348	138.4210	43952
Rainbow	35.2335	33.0893	40.0484	24.3460	49575

Tickers used in Panel C for July-August 2001: AMAT, EDS, INTU, MU, PSFT, BRCM, EMC, JNPR, NT, SCMR, CA, ERTS, LU, ORCL, SUNW, CSCO, IBM, MOT, PALM, TLAB, DELL, INTC, MSFT, PMTC, TXN.

the classification. Therefore it performs very well in-sample in terms of classification accuracy. As can be seen, it behaves much like our Naive algorithm, which delivers similar performance. The other algorithms do less well, as they are based on a preselected smaller set of words, all of which may not be ideal for this specific data set. However, the somewhat weaker in-sample performance is likely to be offset by a reduction in overfitting when working out-of-sample.

In Panels B and C of Table 1, we see that the out-of-sample performance is much lower than in-sample, as is to be expected. Looking at the false positives, the Rainbow model now performs worse than the others. It also has a high error in aggregate sentiment. We notice that the voting algorithm produces the lowest false positive rate. The overall performance across all algorithms highlights the fact that classifying very dirty emotive text is indeed a hard problem. There are two routes to improving this baseline performance. First, we increase the size of the training set without making it too big to result in overfitting. Second, we screen messages for ambiguity before classification, discarding messages we find ambiguous. We take this up in the next section.

2.6 Ambiguity

Messages posted to stock boards are highly ambiguous. There is little adherence to correct grammar, and many words in the messages do not, and probably will never, appear in standard dictionaries. This makes the task of classification algorithms exceedingly hard, as opposed to say, spam filters, where the characteristics of spam versus non-spam emails are quite distinct. There is often only a subtle difference between a buy and a sell message on a stock board, resulting in many false positives. Ambiguity is related to the absence of "aboutness", in that classification based on word counts may not always correctly capture what a message is about (see Morville (2005)).

Ambiguity reduction therefore is an important consideration of a classification algorithm in this realm. To reduce ambiguity, we developed a method to exclude possibly ambiguous messages. We did this by using the General Inquirer, a computer-assisted approach for content analyses of textual data from Harvard University.[9] The algorithms in this approach return a count of optimistic and pessimistic words in text. We filtered messages for ambiguity using a simple metric, i.e. the difference between the number of optimistic and pessimistic words as a percentage of the total words in a body of text. We denote this the optimism score. Since we use a completely different dictionary and approach for the optimism score, we ensure that none of the results are biased in favor of our algorithm relative to others. As a first pass, we examined over 50,000 messages for which we had buy, sell and hold classifications provided by posters on Yahoo, and looked at the average optimism score in each category. These are as follows:

[9]See `http://www.wjh.harvard.edu/~inquirer/` for more details.

	Optimism Score	
Msg Type	Mean	StDev
Buy	0.032	0.075
Hold	0.026	0.069
Sell	0.016	0.071

Therefore, on average the optimism score does rank order the categories well. However, note that the standard deviation of these scores is quite large. Hence for example, if we would like to filter out ambiguous buy messages, we may do so by filtering in messages that were posted as buys and had optimism scores greater than one standard deviation away from the mean, ($> 0.032 + 1 \times 0.07$). Likewise to filter in sell messages with low ambiguity scores, we would want sell messages with ambiguity scores lower than a standard deviation from the mean ($< 0.016 - 1 \times 0.07$). For hold messages, we reduce ambiguity by taking ever smaller intervals around the mean of 0.026.

In Tables 2 and 3, we reduce ambiguity as we proceed from Panel A down to Panel C and examine the changes in classification metrics. In Panel A, the cut off for filtering in a message is based on 1 standard deviation as above. In Panel B, we raise the cut off to 2 standard deviations, and in Panel C, the cut off is 3 standard deviations. Naturally, we sample fewer messages as we proceed to the least ambiguous case. For hold messages, Panel A is uses a spread of 0.01 around the mean, Panel B is at 0.005, and Panel C is at 0.0025.

Table 2 uses a small training set (size 374) and the one used in Table 3 is larger (size 913). The testing sets are the same in both tables. Accuracy levels increase as the training set increases in size. However, this does not impact the percentage of false positives. The false positives decline dramatically with a reduction in ambiguity as expected. The Rainbow algorithm persistently returns a higher number of false positives than the others, as it may be overfitting the data. Overall, the algorithms increase in performance as we tune down the ambiguity of the messages. We achieve an accuracy range of between 60-70% which is good for classification of sentiment (see also Pang, Lee and Vaithyanathan (2002) for an application to sentiment parsing for movies).

In the rest of the paper we explore the sentiment index generated by the voting classifier. We note that this model has fairly low error in sentiment index generation, especially out-of-sample. It also has lower incidence of false positives than most of the remaining classifiers. The rest of our analyses explore whether the sentiment index offers return predictability or not, and its relation to various stock market variables.

3 Experiments: Sentiment and Stock Markets

In the previous section, we assessed the ability of the index to match human classification. In this section, we analyze the relationship of sentiment to stock market data, so as to understand the connection of message board discussion to market economics.

Table 2: Classification as a function of ambiguity. The training set is of size = 374. In this table, we present statistics of the performance of various approaches to message classification. Five basic algorithms as described in the text are used: (i) Naive, (ii) Vector distance, (iii) Discriminant weighted, (iv) Adjective-Adverb, (v) Bayes. The results of a majority vote are given, as are the same when no-majority votes are discarded ("Vote-d", "d" = discard). The results of the `Rainbow` algorithm are also presented. We note that this algorithm is a Bayes classifier that develops a word set from the training set; the one in our algorithm uses an independent word set that is hand generated from the training set and other messages. The extent of ambiguity in the test data set declines as we proceed from Panel A to Panel C. Accuracy, false positives, and sentiment error are expressed in percentage terms. Sentiment error is expressed without sign.

Panel A

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	25.4777	20.3556	9.5011	35.7433	7536
vecDist	49.7479	15.9501	3.1714	138.5787	7536
DiscWtd	54.8832	13.1900	10.0318	204.3926	7536
AdjAdv	29.8301	32.8822	53.3439	4.7059	7536
BayesCl	43.0467	10.6290	9.0366	159.1631	7536
Vote3	41.7596	8.8110	5.6661	169.2776	7536
Vote3-d	47.7541	10.1794	6.8067	168.9292	6523
Rainbow	34.8511	34.7309	39.5246	3.0299	7489

Panel B

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	23.9664	17.5711	14.0181	15.2671	1548
vecDist	53.2946	9.5607	2.7778	47.9415	1548
DiscWtd	58.1395	8.5917	9.4961	58.2234	1548
AdjAdv	26.8734	32.8165	52.9716	2.0661	1548
BayesCl	45.2196	6.3307	9.0439	45.9316	1548
Vote3	44.1860	4.7804	5.9432	49.0391	1548
Vote3-d	49.5549	5.4896	4.8961	49.1041	1348
Rainbow	34.1952	32.2560	43.5036	1.3772	1547

Panel C

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	20.0000	14.8276	10.0000	5.5572	290
vecDist	55.8621	3.7931	0.3448	12.6265	290
DiscWtd	57.2414	5.5172	8.2759	11.7723	290
AdjAdv	24.8276	34.1379	55.5172	0.5376	290
BayesCl	48.9655	2.0690	6.8966	11.6353	290
Vote3	49.3103	1.3793	5.1724	12.2487	290
Vote3-d	52.7778	1.5873	1.9841	12.1598	252
Rainbow	37.5862	24.1379	33.4483	1.0625	290

Table 3: Classification as a function of ambiguity. The training set is of size = 913. In this table, we present statistics of the performance of various approaches to message classification. Five basic algorithms as described in the text are used: (i) Naive, (ii) Vector distance, (iii) Discriminant weighted, (iv) Adjective-Adverb, (v) Bayes. The results of a majority vote are given, as are the same when no-majority votes are discarded ("Vote-d", "d" = discard). The results of the `Rainbow` algorithm are also presented. We note that this algorithm is a Bayes classifier that develops a word set from the training set; the one in our algorithm uses an independent word set that is hand generated from the training set and other messages. The extent of ambiguity in the test data set declines as we proceed from Panel A to Panel C. Accuracy, false positives, and sentiment error are expressed in percentage terms. Sentiment error is expressed without sign.

Panel A

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	45.5016	34.2224	1.8843	16.7918	7536
vecDist	61.0934	14.2118	10.2309	236.2058	7536
DiscWtd	54.8832	13.1900	10.0318	204.3926	7536
AdjAdv	64.1056	33.5987	41.2818	74.0030	7536
BayesCl	57.4443	12.3275	7.8025	238.8515	7536
Vote3	53.3838	10.1778	14.3577	242.0414	7536
Vote3-d	63.2705	12.1534	12.4703	227.2089	6311
Rainbow	64.8818	32.6479	13.0191	86.8046	7489

Panel B

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	46.9638	30.7494	1.0982	6.2881	1548
vecDist	64.5995	8.6563	8.6563	69.9800	1548
DiscWtd	58.1395	8.5917	9.4961	58.2234	1548
AdjAdv	65.7623	28.4884	41.5375	23.2180	1548
BayesCl	61.4341	7.7519	8.0749	67.8975	1548
Vote3	58.3979	6.3307	13.5659	68.8180	1548
Vote3-d	66.7671	7.3851	11.6805	65.8436	1327
Rainbow	65.4816	28.9593	10.7304	27.4832	1547

Panel C

Algo	Accuracy	False Postv	Sent. Error	χ^2	No. Msgs
NvWtd	46.5517	25.5172	9.3103	1.9822	290
vecDist	66.8966	3.7931	7.9310	16.9034	290
DiscWtd	57.2414	5.5172	8.2759	11.7723	290
AdjAdv	61.3793	24.1379	40.0000	4.7444	290
BayesCl	64.4828	4.4828	8.2759	15.2331	290
Vote3	63.4483	4.1379	12.4138	15.3550	290
Vote3-d	66.7939	4.5802	11.0687	14.5289	262
Rainbow	67.5862	18.2759	12.0690	9.0446	290

3.1 Data

Our data comprises 25 tech sector stocks, present in the Morgan Stanley High Tech Index (MSH). These stocks were chosen so as to focus on the tech sector, and also because their message boards showed a wide range of activity. For a period of two months, July and August 2001, we downloaded every message posted to these boards. This resulted in a total of 145,110 messages. These messages were farmed by a lengthy process of web scraping.

We then employed our voting algorithm with the larger training set (found to provide the best performance in the previous section) to assess each message and determine its sentiment. The messages upto 4pm for each day were used to create the aggregate sentiment for the day upto close of trading for each of the stocks. Buy messages increment the index by one and sell messages decrement the index by one. We also aggregated messages across all sample stocks so as to obtain a sentiment index formed from our tech portfolio.

We downloaded the MSH index for the same period. The following measures are constructed for further analysis:

1. Normalized indexes. We normalized both, the MSH index and the aggregate sentiment index by subtracting the mean value and dividing by the standard devation of the data series. We also did this for the sentiment of each individual stock series. This scaling does not impact the correlations between the various series.

2. Disagreement. Following the work of Das, Martinez-Jerez and Tufano (2005) we created a disagreement measure which is as follows:

$$\text{DISAG} = \left|1 - \left|\frac{B-S}{B+S}\right|\right|,$$

   where B is the number of buy messages and S the number of sell messages. This measure lies between zero (no disagreement) and one (high disagreement). It may be computed for any time period; in our analyses, we computed it daily.

3. Volatility. We measure intra-day volatility as the difference between the high and low stock prices for the day divided by the average of the open and closing price.

4. Volume. This is the trading volume in number of shares traded in the day. We also keep track of message volume for each stock.

We first use the data to look at the relationship of aggregate sentiment to the stock index. The analysis is undertaken using normalized time series.

3.2 Sentiment and the stock index

We aggregated sentiment across all the stocks, and examined how closely this sentiment related to the MSH index. This is represented in Figure 2 which indicates that these two series do

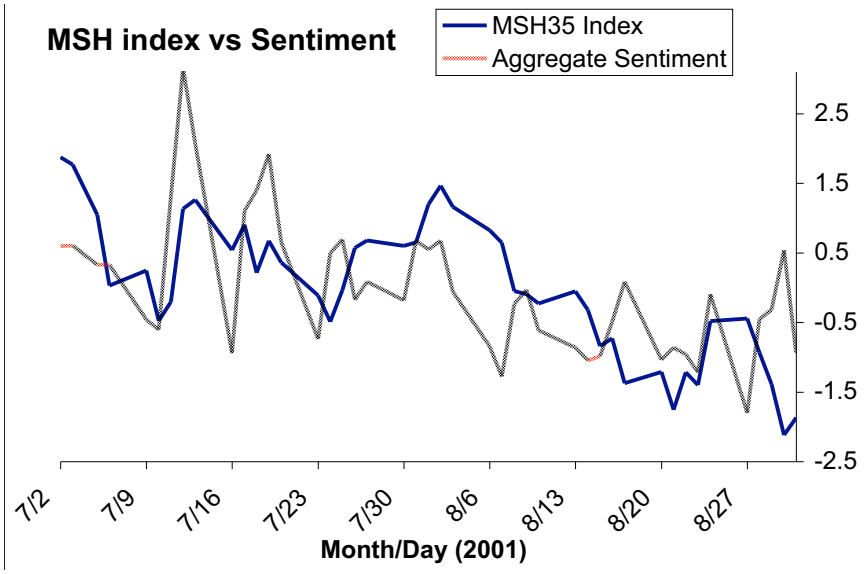

Figure 2: MSH index and aggregate sentiment, daily, July-August, 2001.

track each other closely, implying that the extracted sentiment is not excessively noisy. The correlation between the stock index and the sentiment time series is 0.48. From Table 4 we see that normalized sentiment is quite persistent as evidenced by the high autocorrelations. Individual stock sentiment autocorrelations are far lower. Hence, this is consistent with positive cross-autocorrelations of sentiment, and similar results are known for cross-autocorrelations of stock returns, as in Lo and MacKinlay (1990).

3.3 Causality

The natural next step lies in examining whether the stock series (MSH) is predictive of the sentiment series (SENTY) or vice versa. To do so, we examine the following standard pair of regressions for Granger-type causality.

- Regression of MSH on lagged values of MSH and SENTY.

$$MSH_{t+1} = -0.08(-1.12) + 0.79(9.28)MSH_t + 0.15(1.86)SENTY_t, \quad R^2 = 0.77$$

- Regression of SENTY on lagged values of MSH and SENTY.

$$SENTY_{t+1} = -0.03(-0.21) + 0.10(0.62)MSH_t + 0.45(2.90)SENTY_t, \quad R^2 = 0.25$$

The coefficients in the regressions are followed by the t-statistics in brackets. The first regression shows that tech index is strongly related to its value on the previous day, and is also significantly related to the sentiment index value from the previous day at the 10% level.

Table 4: Autocorrelation of normalized sentiment. The autocorrelation is significant at all lags as is seen from the high Box-Ljung statistics. Ten lags using daily data are computed corresponding to 2 trading weeks. The last column shows the average autocorrelation of the individual stock sentiment series, which is considerably lower than that of the index.

Lag	Auto-correlation	Box-Ljung stat	Avg Stock AutoCorr
1	0.792	29.5532	0.178
2	0.574	45.4350	0.081
3	0.410	53.7239	0.059
4	0.343	59.6681	0.040
5	0.322	65.0398	0.006
6	0.350	71.5739	-0.007
7	0.324	77.3214	0.002
8	0.268	81.3489	0.027
9	0.200	83.6698	0.020
10	0.147	84.9567	-0.004

From the second regression we see that the sentiment index on a given day is significantly related to its prior day's value, but not that of the stock index. The interpretation of this pair of equations is that sentiment does offer some explanatory power for the level of the index. Conversely, since the lagged MSH index does not explain sentiment, if there is any causality, it flows from sentiment to index levels.

3.4 Sentiment for individual stocks

Given that there appears to be a link from sentiment to the index at the aggregate level, we now drill down to the individual stock level and conduct an examination for causality. Our analysis uses the normalized stock price and normalized sentiment index for each stock.

As a first step we computed the contemporaneous correlations between each stock price and stock index to examine if the high level of correlation obtained at the aggregate level exists at individual ticker level. Figure 3 presents the histogram of correlations for all the stock tickers. We can see that the levels of correlation are strongly positive for most stocks in our sample.

Given a strong contemporaneous relationship between each stock's price series and sentiment, we examine if any causal relationship exists at the individual stock level. For each stock, we undertook two regressions. First, we regressed the stock price on the previous day's price and previous day's sentiment. Second, we regressed the sentiment on the previous day's price

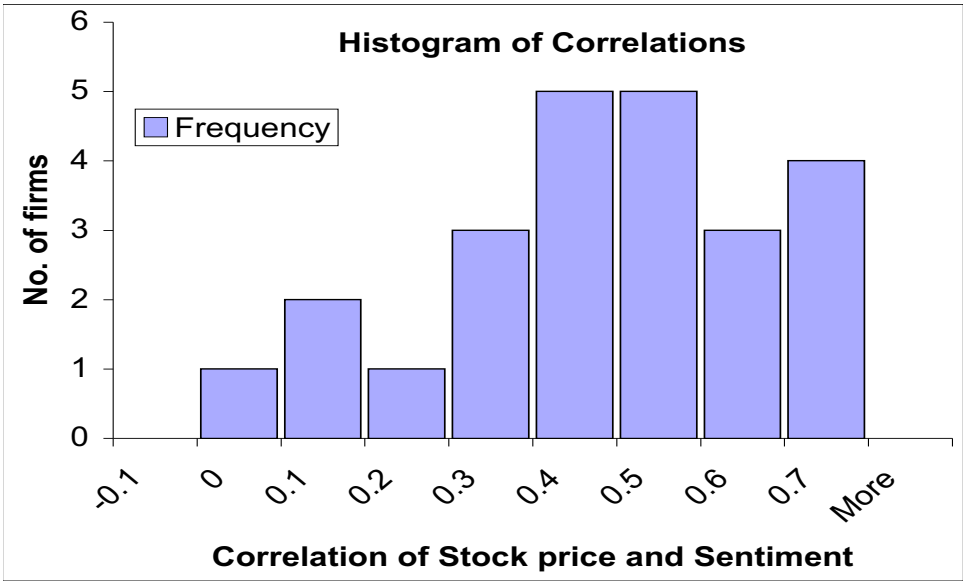

Figure 3: Histogram of correlations between individual stock price daily series and sentiment, July-August, 2001. Correlations are strongly positive across all these tech stocks.

and previous day's sentiment. In order to summarize the results across the separate stock regressions, we report the mean t-statistics across the stocks as well as the standard deviation of the t-statistic across all stocks for both regressions (see Table 5).

We can see from an examination of the t-statistics, that there is no causal relationship on average across the individual stocks. Neither regression is significant on average. While a few stocks do seem to indicate a causal relation from sentiment to stock value, mostly there is none. On the other hand, in section 3.2 for the aggregate index, we found a causal relation from sentiment to stock prices. An implication of these results is that aggregation of individual stock sentiment may be resulting in a reduction of idiosyncratic error in sentiment measurement, giving significant results at the index level. Further, this evidence is also consistent with the presence of cross-autocorrelations of sentiment amongst stocks.

3.5 Sentiment, disagreement, volume and volatility

We now examine the relationship of our sentiment measure to trading volume and volatility, both defined previously in Section 3.1. Table 6 presents the results of a regression of sentiment on volume and volatility. There is a strong contemporaneous relationship between sentiment and trading volume, and none between sentiment and volatility. This relationship is significant for 16 of the stocks in the sample. Higher trading volume correlates with positive sentiment. Further, we examined whether we could explain trading volume on any day with the previous day's sentiment level, and found that no significant explanatory relationship.

Table 5: Statistical significance of Granger causality regressions for individual stocks. In this table we report summarized t-statistics for all the individual regressions. The list of tickers is: MSH, AMAT, BRCM, CA, CSCO, DELL, EDS, EMC, ERTS, IBM, INTC, INTU, JNPR, LU, MOT, MSFT, MU, NT, ORCL, PALM, PMTC, PSFT, SCMR, SUNW, TLAB, TXN.

Dependent Variable	Independent Variables		
Stock Price(t+1)	Intercept	Stock Price(t)	Sentiment(t)
Mean of T-stats	-0.034	1.193	-0.630
Std Dev of T-stats	0.107	0.747	1.371
Number significant (at 10%)	0	8	3

Dependent Variable	Independent Variables		
Sentiment(t+1)	Intercept	Stock Price(t)	Sentiment(t)
Mean of T-stats	0.021	-0.106	1.222
Std Dev of T-stats	0.118	1.357	1.389
Number significant (at 10%)	0	4	10

Table 7 presents the raw correlations of our normalized variables. All the expected relationships are shown to hold in the data. First, stock price and sentiment are positively related to each other. Second, sentiment is inversely related to disagreement. Hence, when disagreement increases, sentiment drops. Alternatively, there is greater disagreement when sentiment is falling rather than when it is rising. Third, sentiment is correlated to high posting volume, suggesting that increased discussion indicates optimism. It hints at the fact also that people prefer making posts when they are bullish on a stock (Antweiler and Frank (2004) therefore name their extracted sentiment a "bullishness" index). Fourth, there is a strong relationship between message volume and volatility, consistent with the findings of Antweiler and Frank (2002), who find that sentiment extracted from message boards is not predictive of stock movements, but activity on these boards may presage increases in volatility. Finally, trading volume and volatility are strongly related to each other.

It has been suggested that disagreement leads to both, increases in trading volume and volatility. Therefore we check whether our disagreement metric is predictive of these variables. Table 8 shows that this hypothesis is not supported in the data. This is in contrast to the findings of Antweiler and Frank (2004) who find a positive relationship of agreement and reduced trading volumes.

Table 6: Statistical significance of regressions for individual stock sentiment on trading volume and volatility. In this table we report summarized t-statistics for all the individual regressions. The list of tickers is: MSH, AMAT, BRCM, CA, CSCO, DELL, EDS, EMC, ERTS, IBM, INTC, INTU, JNPR, LU, MOT, MSFT, MU, NT, ORCL, PALM, PMTC, PSFT, SCMR, SUNW, TLAB, TXN.

Dependent Variable	Independent Variables		
Sentiment(t)	Intercept	Volume(t)	Volatility(t)
Mean of T-stats	0.000	2.656	-0.005
Std Dev of T-stats	0.000	2.500	1.764
Number significant (at 10%)	0	16	5

Table 7: Correlations of normalized variables. The table shows the correlations of six variables of interest, i.e. stock price, sentiment, disagreement, message volume, trading volume, and volatility. A correlation matrix is computed for each stock using the time series of data in the sample, and averaged across all individual stocks in the sample to give the table below. Normalized variables are used for the analysis.

	STKPR	SENTY	DISAG	MSGVOL	TRVOL	VOLATILITY
STKPR	1.000	0.389	0.029	0.448	0.704	0.713
SENTY	0.389	1.000	-0.517	0.738	0.448	0.370
DISAG	0.029	-0.517	1.000	-0.123	0.038	0.070
MSGVOL	0.448	0.738	-0.123	1.000	0.577	0.504
TRVOL	0.704	0.448	0.038	0.577	1.000	0.756
VOLATILITY	0.713	0.370	0.070	0.504	0.756	1.000

4 Conclusion

We developed a methodology for extracting small investor sentiment from stock message boards. Five distinct classifier algorithms coupled by a voting scheme are evaluated using a range of metrics. Time series and cross-sectional aggregation of message sentiment improves the quality of the sentiment index. Sentiment aggregated across stocks tracks index returns more strongly than with individual stocks. Whereas our methodology results in classification accuracy levels similar to that of widely used Bayes classifiers, the noise-reduction approaches we employ substantially reduces the number of false positives generated in the classification, as well as improves the accuracy of the index value itself. We submit that our suite of techniques presents an effective approach to classifying noisy stock message board postings in order to develop an index of sentiment.

An innovation of this paper is the creation of a simple filter for message ambiguity. We showed that classification improves with the application of our ambiguity filter.

Table 8: Statistical significance of regressions for volatility and trading volume on lagged volatility and disagreement. In this table we report summarized t-statistics for all the individual regressions. The list of tickers is: MSH, AMAT, BRCM, CA, CSCO, DELL, EDS, EMC, ERTS, IBM, INTC, INTU, JNPR, LU, MOT, MSFT, MU, NT, ORCL, PALM, PMTC, PSFT, SCMR, SUNW, TLAB, TXN. The implication of this table is that our disagreement measure is not predictive of either volatility or trading volume.

Panel A: Volatility

Dependent Variable		Independent Variables		
Volatility(t+1)	Intercept	Volatility(t)	Disag(t+1)	Disag(t)
Mean of T-stats	-0.047	0.930	0.357	0.136
Std Dev of T-stats	0.115	1.348	1.556	1.029
Number significant (at 10%)	0	8	6	2

Panel B: Trading Volume

Dependent Variable		Independent Variables		
TrVolume(t+1)	Intercept	TrVolume(t)	Disag(t+1)	Disag(t)
Mean of T-stats	-0.044	1.422	0.190	-0.080
Std Dev of T-stats	0.151	1.107	1.588	1.445
Number significant (at 10%)	0	14	5	2

As an application, we created a tech sector sentiment index from a representative set of twenty-four stocks. This index Granger causes stock index levels. However, causality is not found at the individual stock level; therefore, aggregation of sentiment reduces some of the noise from individual stock board postings. Whereas our sentiment index has expected contemporaneous relationships with various market variables, the disagreement measure we create evidences very little correlation to other variables.

The overall evidence suggests that market activity is strongly related to small investor sentiment. Thus, the algorithms developed in this paper may be used to assess the impact on investor opinion of management announcements, press releases, third-party news, and regulatory changes.

These algorithms may be extended to other applications. First, there is a limited understanding of the microstructure of tech stocks. Since these stocks have the most active message boards, the sentiment classifier may support empirical work in this domain. Second, the algorithms may be used to investigate the mechanics of herding. A third application is that of

monitoring market activity. Regulators are concerned about market manipulation that goes undetected amongst the millions of messages posted to message boards every day. Fourth, firms may use the classifier to monitor their message boards for investor reaction to management actions. Finally, the sentiment index may be applied to testing theories in the domain of behavioral finance.

Appendices

A Overview of the Methodology Flow

This a brief overview of the model components, which complements the model schematic presented earlier in Figure 1. The programs were coded in Java.

1. Message data is collected from the web using a scraper program, for example, it may be titled (`YahooScraper.java`).
2. Stock data is obtained using a web download, stored in e.g. `TICKER-stk.dat`.
3. The raw data is converted into meta-format using routines in a message handling java class (`Message.java`).
4. The CUVOALD dictionary (`thesaurus.txt`) for parts of speech processing is obtained from the University of London. Object classes for dictionary handling and parts-of-speech tagging were written in java (`Dicttable.java`).
5. Some useful mathematical routines for probabilistic computations are developed in a helper program (`MultiNom.java`).
6. The lexicon is stored in `lexicon.dat`. The grammar is maintained in `Grammar.dat`.
7. The main program is called, for example, `MsgFilter.java`. It performs many tasks such as (i) preprocessing the data, i.e. clean up, expansion of abbreviations, and negation of sentence meaning. (ii) classification using all 5 algorithms, (iii) implementation of the voting schemes.
8. Output consists of two files: (i) a dataset of full message text and classification information (`*.classified`), and (ii) classification statistics by date, embedding the sentiment index (`*.stats`).

A.1 The Dictionary

Our data includes auxiliary information on the English language. To exploit parts-of-speech usage in messages, a dictionary was used to detect adjectives and adverbs for the classifier algorithms. This dictionary is called CUVOALD (Computer Usable Version of the Oxford Advanced Learner's Dictionary).[10] It contains parts-of-speech tagging information, and we wrote appropriate program logic to use this dictionary while analyzing messages for grammatical information.

[10]The dictionary was downloaded from Birkbeck College, University of London. It is the creation of Roger Mitton of the Computer Science Department. It contains about 70,000 words, and covers most of the commonly used words

A.2 The Lexicon

Words are the heart of any language inference system, and in a specialized domain, this is even more so. The sentiment classification model relies on a lexicon of "discriminant" words, which comprise the lexicon. The lexicon is designed using domain knowledge and statistical methods. A discriminant function is used to statistically detect which words in the training corpus are good candidates for classifier usage (the details of the discriminant function are provided in Section 2.2.3). Therefore, the lexicon is essentially a collection of words relevant to the classification problem, which will be used by the classifier algorithms to discriminate buy messages from sell messages. Hence, we exercised care in creating the lexicon, so as to include many useful words that would enable the algorithms to discriminate positive from negative sentiment. Clearly, a different lexicon will result in different classifications; this injects flexibility and the ability to tune the algorithm, but also requires domain expertise. We had to read thousands of messages to cull the set of words that now comprise the lexicon. The user's goal is to populate the lexicon with words of high discriminant value, and this is where the application of domain expertise is valuable. Over time, more words may be added to the lexicon, which improves in this evolutionary manner. More details on the lexicon are presented in Appendix B.

A.3 The Grammar

A grammar may be defined as a set of functions or rules applied in conjunction with the lexicon to extract sentiment from text. Correspondences between word sets, language features and classification types comprise the grammar. In our setting, the training corpus is the grammar. This set of messages, once hand-tagged, may be thought of as a set of rules that govern the classification of other messages. One way to approach classification of any message is to search the grammar for a rule that may be applied to the message. For example, a distance function under a carefully chosen metric may be used to identify the applicable rule. Suppose we wish to analyze message M. We compare, using some metric, the relationship of this message M to a set of other messages G, and find the one that is its closest look-alike. We then equate the properties of message M to those of the proxy. The set of pre-classified messages G is denoted the grammar, and the rule that finds the proxy message or a proxy set of messages is codified in a classification algorithm. The classification algorithm implements a rule that finds closest messages in a grammar, using the words in the lexicon as variables. Some of the algorithms use only the grammar, or the lexicon, and some use both.[11]

in the English language. Informal tests of the dictionary showed that about 80-90 percent of the words in a message were found in the dictionary.

[11]We may think of the grammar as Schank (1975) did, i.e. it is a "conceptual processor". With stock market messages, the language is cryptic, and the grammar rules must work together so as to make sense of the "thought bullets" posted to the web. Schank states this particularly well: "People do not usually state all the parts of a given thought that they are trying to communicate because the speaker tries to be brief and leaves out assumed or inessential information. The conceptual processor searches for a given type of information in a sentence or a larger unit of discourse that will fill the needed slot." Our algorithms combine grammar rules and lexical items to achieve automated classification.

A.4 Message Pre-processing

Before applying the lexicon-grammar based algorithms, each message is preprocessed to enable cleaner interpretation. First, we carry out "HTML Cleanup", which removes all HTML tags from the body of the message as these often occur concatenated to lexical items of interest. Examples of some of these tags are: `<BR>,<p>,&quot`, etc. Second, we expand abbreviations to their full form, making the representation of phrases with abbreviated words common across the message. For example, the word "`ain't`" is replaced with "`are not`", "`it's`" is replaced with "`it is`", etc. Finally, we handle negation words. Whenever a negation word appears in a sentence, it usually causes the meaning of the sentence to be the opposite of that without the negation. For example, the sentence "`It is not a bullish market`" actually means the opposite of a bull market. Words such as "`not`", "`never`", "`no`", etc., serve to reverse meaning. We handle negation by detecting these words and then tagging the rest of the words in the sentence after the negation word with markers, so as to reverse inference. These three parsers deliver a clean set of messages for classification.

B Construction of the Lexicon

The features of the lexicon are as follows:

1. These words are hand-selected based on a reading of several thousand messages.
2. The lexicon may be completely user-specified, allowing the methodology to be tailored to individual preference. For example, if the user is only interested in messages that relate to IPOs, a lexicon containing mostly IPO-related words may be designed. (The grammar, i.e. the training set would also be correspondingly tagged).
3. For each word in the lexicon, we tag it with a "base" value, i.e. the category in which it usually appears. For example, the word "sell" would be naturally likely to appear in messages of type SELL, and we tag "sell" with base value 1. If the word is of BUY type, we tag it with value 3, and NULL words are tagged 0.[12] Every time a new word is added to the lexicon, the user is required to make a judgment on the base type.
4. Each word is also "expanded", i.e. appears in the lexicon in all its forms, so that across forms, the word is treated as one word. This process is analogous to stemming words, except that we exhaustively enumerate all forms of the word rather than stem them.[13]
5. Each word is also entered with its "negation" counterpart, i.e. the sense in which the word would appear if it were negated. Negation is detected during preprocessing (described later) and is used to flag portions of sentences that would be reversed in meaning.

An example of a lexical entry along with its base value, expansion and negation is provided below:

```
3 favorable favorite favorites favoring favored
1 favorable__n favorite__n favorites__n favoring__n favored__n
```

[12] These tag values seem odd, but are used in the algorithms; the numbers are an implementation detail, and may vary across algorithms. There is no special reason for the choice of the numbers used.

[13] Stemming is the process of mapping a word to its root word. For example, the root of "buying" is "buy".

All forms of the word appear in the same line of the lexicon. As can be seen, a tag is attached to each negated word in the second line above. The default classification value (the "base" value) is specified at the beginning of the line for each lexical item (i.e. a 0, 1 or 3).

The current size of the lexicon is approximately 300 distinct words. Ongoing, incremental analysis results in additions to the word set.

Based on the training corpus, we can compute the *discriminant value* of each item in the lexicon. This value describes the power of the lexical item in differentiating message types. For example, the word "buy" is likely to be a strong discriminator, since it would be suggestive of positive sentiment. The goal is to populate the lexicon with words that are good discriminators.

C Discriminant values

Example values for some words from the discriminant function are shown here (we report a selection of words only, not the entire lexicon). The last three words appear with their negation tags.

```
SAMPLE DISCRIMINANT VALUES
bad 0.040507943664639216
hot 0.016124148231134897
hype 0.008943543938332603
improve 0.012395140059803732
joke 0.02689751948279659
jump 0.010691670826157351
killing 0.010691670826157329
killed 0.016037506239236058
lead 0.003745650480005731
leader 0.0031710056164216908
like 0.003745470397428718
long 0.01625037430824596
lose 0.12114219092843743
loss 0.007681269362162742
money 0.15378504322023162
oversell 0.0
overvalue 0.016037506239236197
own 0.0030845538644182426
gold__n 0.0
good__n 0.04846852990132937
grow__n 0.016037506239236058
```

These values make for interesting study. For example, the word "lose" understandably has a high discriminant value. The word "oversell" is not used at all. One of the higher values comes from the negated word "good–n" which means that there is plenty of negation in the language used in the message boards. Compare this with its antonym "bad", which actually has a lower discriminant value! The word "joke" is a good discriminator, which is somewhat surprising, though not totally nonintuitive. The highest valued discriminant is the word "money".

References

Admati, A., and P. Pfleiderer (2000). "Noisytalk.com: Broadcasting Opinions in a Noisy Environment," working paper 1670R, Stanford University.

Antweiler, W., and M. Frank (2004). "Is all that Talk just Noise? The Information Content of Internet Stock Message Boards," *Journal of Finance*, v59(3), 1259-1295.

Antweiler, W., and M. Frank (2002). "Internet Stock Message Boards and Stock Returns," working paper, UBC.

Antweiler, W., and M. Frank (2005). "The Market Impact of News Stories," working paper, UBC.

Bagnoli, M., M. D. Beneish, and Susan G. Watts (1999). "Whisper forecasts of Quarterly Earnings per Share," forthcoming, *Journal of Accounting and Economics*, v28(1), 1999.

Chakrabarti, S., B. Dom, R. Agrawal, and P. Raghavan. (1998). "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies," *The VLDB Journal*, Springer-Verlag.

Chakrabarti, S., B. Dom, P. Indyk. (1998). "Enhanced hypertext categorization using hyperlinks," SIGMOD ACM, 1998.

Chakabarti, S., S. Roy, and M.V.Soundalgekar (2003). "Fast and accurate text classification via multiple linear discriminant projections," *The VLDB Journal*, Springer-Verlag.

Charniak, E. (1993). Statistical Language Learning, MIT Press, Cambridge, Massachusetts.

Choi, J., D. Laibson, and A. Metrick (2002). "Does the Internet Increase Trading? Evidence from Investor Behavior in 401(k) Plans," *Journal of Financial Economics*, v64, 397-421.

Das, S., A. Martinez-Jerez, and P. Tufano (2005). "e-Information," *Financial Management*.

Godes, D., and D. Mayzlin (2001). "Using Online Conversations to Study Word of Mouth Communication," forthcoming *Marketing Science*.

Joachims, T (1999). "Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning," B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press.

Koller, D., and M. Sahami (1997). "Hierarchically classifying documents using very few words," International Conference on Machine Learning, vol 14, Morgan-Kaufmann, San Mateo, California.

Lam, S.L., and J. Myers (2001). "Dimensions of Web Site Personas," working paper, UC Berkeley.

Lavrenko, V., M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allen (2001). "Mining of Concurrent Text and Time Series," proceedings of the KDD 2000 Conference Text Mining Workshop.

Leinweber, D., and A. Madhavan (2001). "Three Hundred Years of Stock Market Manipulation," *Journal of Investing*, v10, 7-16.

Lo, Andrew, W., and A. Craig MacKinlay (1990). "When are Contrarian Profits due to Stock Market Overreaction?" *Review of Financial Studies*, v3(2), 175-205.

McCallum, A., 1996, "Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering," School of Computer Science, Carnegie-Mellon University.

Minsky, M. (1985). "Society of Mind," Simon & Schuster, New York.

Mitchell, Tom (1997). "Machine Learning", McGraw-Hill.

Morville, Peter (2005). "Ambient Findability," O'Reilly, California.

Neal, R.(1996). Bayesian Learning for Neural-Networks, Lecture Notes in Statistics, v118, Springer-Verlag.

Pang, Bo., Lillian Lee and Shivakumar Vaithyanathan (2002). "Thumbs Up? Sentiment Classification using Machine Learning Techniques," proceedings of *Conference on Empirical Methods in Natural Language Processing*.

Schank, R. (1975). "Conceptual Dependency Theory," (Ch 3), in Conceptual Information Processing, North-Holland, 22-67.

Smola, A.J., and Scholkopf, B (1998). "A Tutorial on Support Vector Regression," NeuroCOLT2 Technical Report, ESPIRIT Working Group in Neural and Computational Learning II.

Tumarkin, R., and R. Whitelaw (2001). "News or Noise? Internet Postings and Stock Prices," *Financial Analysts Journal*, v57(3), 41-51.

Vapnik, V. and Chervonenkis (1964). "On the Uniform Convergence of Relative Frequencies of Events to their Probabilities," *Theory of Probability and its Applications*, v16(2), 264-280.

Vapnik, V (1995). The Nature of Statistical Learning Theory, Springer-Verlag, New York.

Wakefield, J. (2001). "Catching a Buzz," *Scientific American*, November, 30-32.

Wysocki, Peter (1998). "Cheap Talk on the Web: The Determinants of Postings on Stock Message Boards," working paper No.98025, University of Michigan Business School.