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

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

Mike Y. Chen
UC Berkeley
Berkeley, CA 94720.

May 2, 2004

Abstract

We develop a methodology for extracting small investor sentiment from stock message boards. Five distinct classifier algorithms coupled by a voting scheme are found to perform well against human and statistical benchmarks. Time series and cross-sectional aggregation of message information improves the quality of the resultant sentiment index. Empirical applications evidence a relationship with stock returns – visually, using phase-lag analysis, pattern recognition and statistical methods. Sentiment has an idiosyncratic component, and aggregation of sentiment across stocks tracks index returns more strongly than with individual stocks. Preliminary evidence suggests that market activity influences 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.

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 Marvin Minsky [1985].

People talk. Talk feeds on itself. Thus, 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 750,000 messages by early 2004. 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]

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) in 1999:

*We owe a special debt to the creative environments at UC Berkeley's Computer Science Division and Haas School, where this work was begun. 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, 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, and the European Finance Association Meetings, Barcelona, 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 the Price Waterhouse Cooper's Risk Institute, the Dean Witter Foundation, and a Research Grant from Santa Clara University. Please address all correspondence to Professor 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 in the Computer Science Division, UC Berkeley, mikechen@cs.berkeley.edu.

[1] Das, Martinez-Jerez and Tufano [2000] 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 may or may not have, making it even harder for a computer algorithm with limited context. 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 by human subjects. Despite these issues, we find the performance of our algorithms to be encouraging.

Recent empirical work suggests a link between small investor behavior and stock market activity. 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 the 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, et al [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. 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.

Overall, this paper comprises two parts: (i) methodology and validation, and (ii) the empirical relationship of market activity and sentiment. The first part describes the algorithms for creating the sentiment index. The second part relates sentiment to stock market activity. This done in many ways – visually through plots, temporally using lead-lag analysis, structurally using pattern recognition, and statistically using regressions. Next, we delve into the technical approach.

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.[4] 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.[5] The algorithms "learn" sentiment classification rules from the pre-classified data set, and then

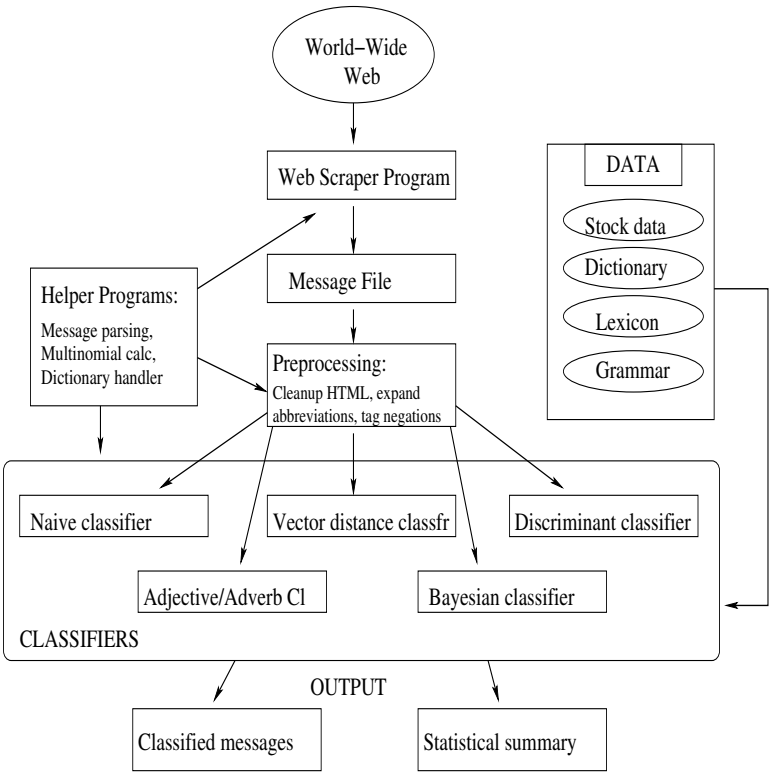
[2] Business Week, 23rd May, 2001 cites a Bear Stearns report that reports a huge spurt in volume, and a total number of day-traders in excess of 50,000.

[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] 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.

[5] 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**



apply these rules out-of-sample. A simple majority of the five rules 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 the 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.
- 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.
- The third database is the "grammar" or the training corpus. It is a base set of rules used in the statistical analysis.

These 3 databases are used by 5 algorithms (denoted "classifiers") to arrive at a 3-way classification of each message, and are described in the Appendix.

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 [NC]

This algorithm is based on a word count of positive and negative connotation words. It is the simplest and most intuitive of the classifiers. 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. If the net word count crosses a given positive (negative) threshold, we classify it as a buy (sell), else it is treated as neutral.

2.2.2 Vector Distance Classifier [VDC]

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.

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.

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

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 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 [DBC]

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 Fisher's discriminant function for each word (see Chakrabarti, Dom, Agrawal and Raghavan [1998]). Let the set

$C = \{null, sell, buy\}$ denote the categories for our messages. Let n_i be the number of times word w appears in category i. The average number of times word w appears in a message of category i is 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:

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

This equation assigns a score $F(w)$ to each word w in the lexicon, which is the ratio of the across-class (class i vs class k) variance to the average of within-class (class $i \in C$) variances. 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.

2.2.4 Adjective-Adverb Phrase Classifier [AAPC]

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.

2.2.5 Bayesian Classifier [BC]

The Bayesian classifier relies on a multivariate application of Bayes' theorem (see Mitchell [1997], Neal [1996], Koller and Sahami (KS) [1997], and Chakrabarti, Dom, Agrawal and Raghavan (CDAR) [1998]). Recently, it has been used for web search algorithms, for detecting web communities, and in classifying pages on internet portals. Our approach here is an adaptation of this technology for stock market sentiment.[6]

[6]Koller and Sahami 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 KS [1997] and CDAR [1998]. The exposition here briefly summarizes these approaches.

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.

Our 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 $F = \{w_k\}_{k=1}^{D}$. We use discriminant analysis (described in the previous subsection) to determine a good word set $\{w_k\}$.

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 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

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

This measure has a corresponding probability: $\theta(c_i, w_k)$ is the probability with which word w_k appears in all messages m in class c:

$$\theta(c_i, w_k) = \frac{\sum_{m_j \in c_i} n(m_j, w_k)}{\sum_{m_j \in c_i}\sum_k n(m_j, w_k)} = \frac{n(c_i, w_k)}{n(c_i)} \tag{4}$$

We require that $\theta(c_i, w_k) \neq 0$, $\forall c_i, w_k$. Hence, an adjustment is made to equation (4) via Laplace's formula which is

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

The probability $\theta(c_i, w_k)$ is unbiased and efficient. 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. $\Pr[m_j|c_i]$:

$$\begin{aligned}\Pr[m_j|c_i] &= \left(\begin{array}{c} n(m_j) \\ \{n(m_j, w_k)\} \end{array} \right) \prod_{k=1}^{D} \theta(c_i, w_k)^{n(m_j, w_k)} \\ &= \frac{n(m_j)!}{n(m_j, w_1)! \times n(m_j, w_2)! \times ... \times n(m_j, w_D)!} \\ &\quad \times \prod_{k=1}^{D} \theta(c_i, w_k)^{n(m_j, w_k)}.\end{aligned}$$

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):

$$\Pr[c_i|m_j] = \frac{\Pr[m_j|c_i].\Pr[c_i]}{\sum_{i=1}^{C} \Pr[m_j|c_i].\Pr[c_i]}. \quad (5)$$

For each message, equation (5) 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.

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", typically of size 300-500 messages. The number of messages is deliberately kept small so as to assess whether the classifiers are amenable to a minimal amount of training. 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 Algorithm Performance

Our metric for classifier accuracy is the number of correct classifications divided by the number of attempted classifications. Therefore, if we have 50 test messages, discard 10 as ambiguous and classify correctly 20 of the remaining messages, the accuracy is 50%.[7]

As a first step in assessment of algorithm performance, we collected and hand-classified a few hundred messages in both training and testing datasets. We tuned the classifiers on the training set, and then compared the classification on the testing set to the human-coded one. A random classifier would result in a 33% accuracy level, since there are 3 categories of messages. Instead, we obtained accuracy levels of 62%. High ambiguity makes higher levels of accuracy difficult to achieve, and we found that humans also agreed on message classification to the extent of 72%, against which the accuracy level of 62% bears comparison.

Next, we carried out a more extensive test, using revealed sentiment on the boards by posters to cross-check the algorithm's interpretive ability. Yahoo!'s message board now allows posters to check a box in their message with their opinion on the sentiment of the company. We use this information to check the extent to which the algorithm's interpretation of each message matches that of the poster's stated classification. This test was run on the set of thirty-five Morgan Stanley High-Tech (MSH35) stocks for the period July 1, 2001 to August 31, 2001. During this period, these stocks experienced high message posting volumes. A total of 234,888 messages were downloaded of which 66,212 messages contained revealed sentiment by the posters. For these messages, we assessed the accuracy of the algorithm. Results are presented in Table 1.

The usual 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 top panel in Table 1 has 3 rows and 4 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 4 categories: Null, Sell, Buy, Not Rated. 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.

$$\chi^2(4) = \frac{1}{9}\sum_{i=1}^{9} \frac{(O_i - E_i)^2}{E_i}, \quad (d.o.f = 4) \quad (6)$$

The value of the statistic is 220.81 (P.val = 0.0), i.e. there is strong statistical evidence that the algorithm is able to detect sentiment in a manner consistent with the intention of posters.

[7] Messages in chat rooms are often highly ambiguous, implying that quite often, two 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. Of course, the worst-case benchmark, i.e. random classification may also be used. With three sentiment categories, this benchmark is 33%.

Table 1: **Algorithm Performance**

This table presents the confusion matrix and analysis of the performance of the algorithm. Our test statistic demonstrates that the greater weight on-diagonal is statistically different from a situation in which the rows and columns of the matrix are independent. The tickers used in the analysis are as follows: ADP AMAT BRCM CA CPQ CSCO DELL EDS EMC ERTS HWP IBM INTC INTU JDSU JNPR LU MOT MSFT MU NT ORCL PALM PMTC PSFT SCMR SLR STM SUNW TLAB TXN XLNX YHOO. This results in a total of 234,888 messages posted over the period July 1 to August 31, 2001.

Confusion Matrix

Board Type	Algorithm Classification Nulll	Sell	Buy	UnRated
Hold	1825	1223	1236	718
Sell	11621	10083	5945	4563
Buy	14942	7584	11753	5215

Observed Matrix: $O(.)$

As Posted	Classification Null	Sell	Buy	Total
Null	1825	1223	1236	4284
Sell	11621	10083	5945	27649
Buy	14942	7584	11753	34279
Total	28388	18890	18934	66212

Expected Matrix: $E(.)$

As Posted	Classification Null	Sell	Buy	Total
Null	1837	1222	1225	4284
Sell	11854	7888	7907	27649
Buy	14697	9780	9802	34279
Total	28388	18890	18934	66212
χ^2-statistic:	=	220.81		
(dof=4)		Pval = 0.0		

3 Relating the Index to Stock Market Data

In the previous section, we assessed the ability of the index to match human classification. In this section, we analyze the link of sentiment to stock market data, a necessary condition for the sentiment index to embed economic content.

3.1 Data

We created two datasets: (a) intra-day sentiment data, compiled from the last quarter of 2000, and (b) daily sentiment data, comprising messages over the period June-August, 2001, from stocks in the Morgan Stanley High-Tech index.

The intraday sample is compiled via a real-time download of messages and concurrent stock prices for 8 stocks during the last three months of 2000. Since this exercise was run in real time, there were periods for which some boards were technically inaccessible.[8] After accounting for these problems, we were able to collect a total of over 300 stock-days. Of these, more than 100 stock days contained active postings and no missing time periods, amounting to over 25,000 messages. This data set is used to examine intraday phase lags between the sentiment index and the stock time series.

The daily data set comprises messages from 35 stocks in the Morgan Stanley High-Tech Index (MSH35) over the period June 1 to August 27, 2001. For these 88 days we collected every message for 35 stocks, resulting in close to 400,000 messages. Using this data we cross-sectionally aggregated sentiment to build a daily high-tech sentiment index.

3.2 Sentiment Index creation using Message Aggregation

The quality of the High tech sentiment index is enhanced by two types of aggregation schemes, in time and in the cross-section.

The sentiment index for a single stock is a cumulative total of buy messages ($+1$) and sell messages (-1) from a specified starting point in time. Null messages are scored zero. Hence, the index is a *time-aggregated* number, which leads to a reduction in index error. The intuition for this is that classification errors are offsetting. To see this consider the following example. Suppose the message board has two messages, the first a buy and the second a sell, resulting in an index value of zero. Now, if the classifier is completely wrong, i.e. it classifies the first message as a sell and the second one as a buy, it still arrives at the same index value of zero. Thus, time aggregation for a single stock avails of error correction from offsetting errors.

The High Tech sentiment index is constructed by *cross-sectional* aggregation across stocks. By generating one index from 35 stocks in the MSH35, we avail of index-smoothing, wherein idiosyncratic mistakes by the classifier across stocks offset each other. We will

[8] This is caused by server rejection, machine downtime, network failure, message board repair, etc.

show that the sentiment of the index track returns better than in the case of sentiment for individual stocks.

We now proceed to look at the data.

3.3 Visual comparisons

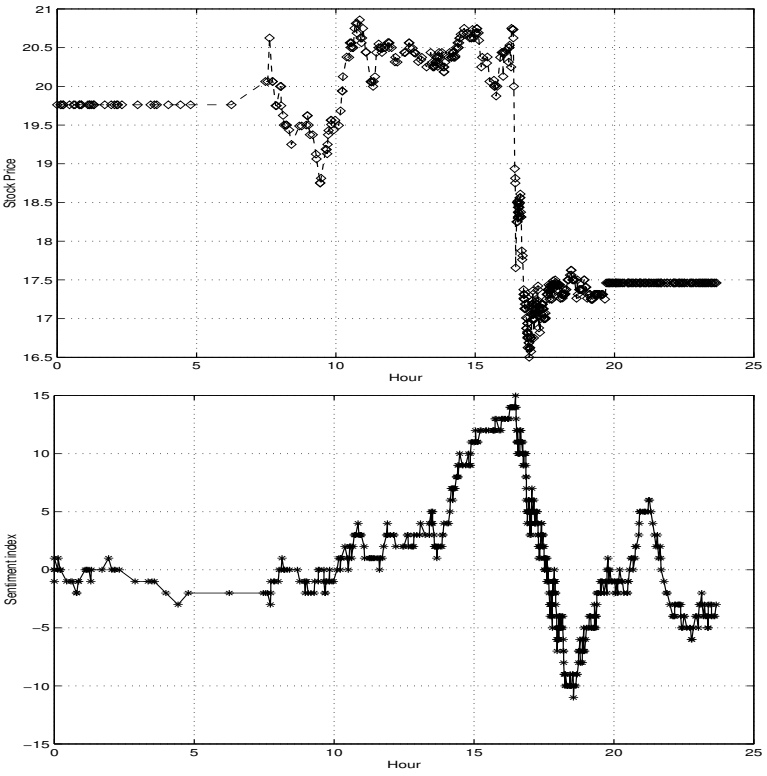
Empirical observation supports a link between sentiment and market returns. We first examine data at high frequency (i.e. intra-day). For the last quarter of 2000, we ran our message analysis programs on a real time basis on the following stock boards at Yahoo: AAPL (Apple Computer), AMZN (Amazon Inc), CDNW (CD Now), CSCO (Cisco Systems), DELL (Dell Computer), EBAY, ITWO (i2 Technologies), MSFT (Microsoft Corp), and YHOO (Yahoo). For every stock we recorded the message and the contemporaneous stock price in a separate file for each day for which we were able to access the board continuously for all trading hours.[9] Hence we collected about 340 (stock+day) combinations.[10] We classified the messages and used the time series of messages to create the sentiment index. Starting the index at zero at the beginning of the day, we update it with each message, adding 1 for positive messages and subtracting 1 for pessimistic ones. This provides a time series of sentiment for the 24 hour period.

In Figures 2-6, we present plots representing the variety of relationships between the stock price and our algorithm-generated sentiment index. A variety of plot shapes was evidenced, from strong relationships to none. On some days, message board sentiment responds almost contemporaneously to stock price movements. The plot for Apple Computer on 18th October, 2000 (Figure 2) reveals that the sentiment index reacts very quickly to a sharp drop in Apple's stock price. In contrast, on October 20, 2000, Apple Computer's stock (Figure 3) reflects the built-up positive sentiment from before the opening of trading. In Figure 4, on 7th December, 2000, Amazon's sentiment shows conflicting lead-lags, since in some portions of the plot, sentiment appears to lead, and in others it lags the stock price move. On other days, such as for Amazon on 11th December, 2000 (Figure 5), there appears to be almost no relationship between sentiment and stock price. was a precursor to stock price change. Sentiment appears to lead the price change in Figure 6, the graph of Dell Computer on 13th November, 2000. These plots suggest a contemporaneous relationship between the sentiment index and the traded stock price on an intra-day basis.

Figure 7 shows the sentiment index for the 3 days around an earnings event, i.e the day before, of and after the event. On September 28, 2000 Apple Computer announced reduced earnings. The announcement was made at 4pm Eastern time. From the graphs for the Apple sentiment index, it is apparent that the message boards had not anticipated the announcement. A day

Figure 2: **Apple Computer, 18-October-2000**

The two plots below depict the stock price and sentiment for the 24 hours of the day. Each point corresponds to the arrival of a message on the stock board. The stock price graph is usually flat in the region outside regular trading hours. The stock price is contemporaneously collected whenever a message arrives on the stock board.



[9]The time stamp of stock price quotes was corrected if the "real-time" source was time-lagged. In some cases, there was a 15 minute delay in quote posting on Yahoo boards. We made the required correction to ensure the synchronicity of sentiment and stock quotes.

[10]Of these, depending on the use to which the data is to be put, the valid sets ranged from 100-200. This number may have been higher if we had run the programs for a longer period, or if the downtimes (from server breakdowns, message-board rejections, network failures, etc) had been less frequent. Nevertheless, the files we obtained contained more than 50,000 messages.

Figure 3: **Apple Computer, 20-October-2000**

The two plots below depict the stock price and sentiment for the first 16 hours of the day. Each point corresponds to the arrival of a message on the stock board. The stock price graph is usually flat in the region outside regular trading hours. The stock price is contemporaneously collected whenever a message arrives on the stock board.

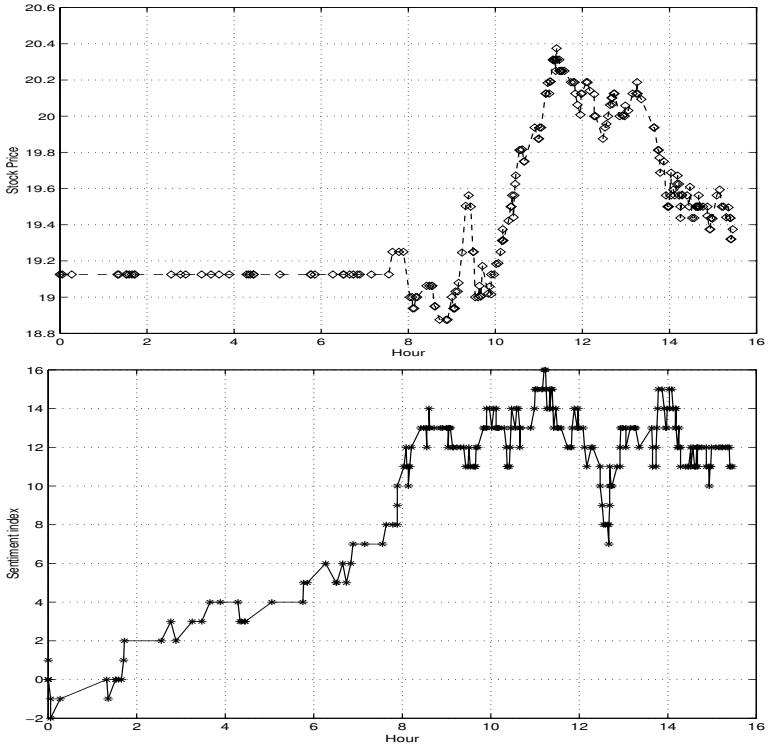


Figure 4: **Amazon, Inc., 07-December-2000**

The two plots below depict the stock price and sentiment for the 24 hours of the day. Each point corresponds to the arrival of a message on the stock board. The stock price graph is usually flat in the region outside regular trading hours. The stock price is contemporaneously collected whenever a message arrives on the stock board. Notice that the jump in price at the end of the upper plot comes from a late after-hours trade.

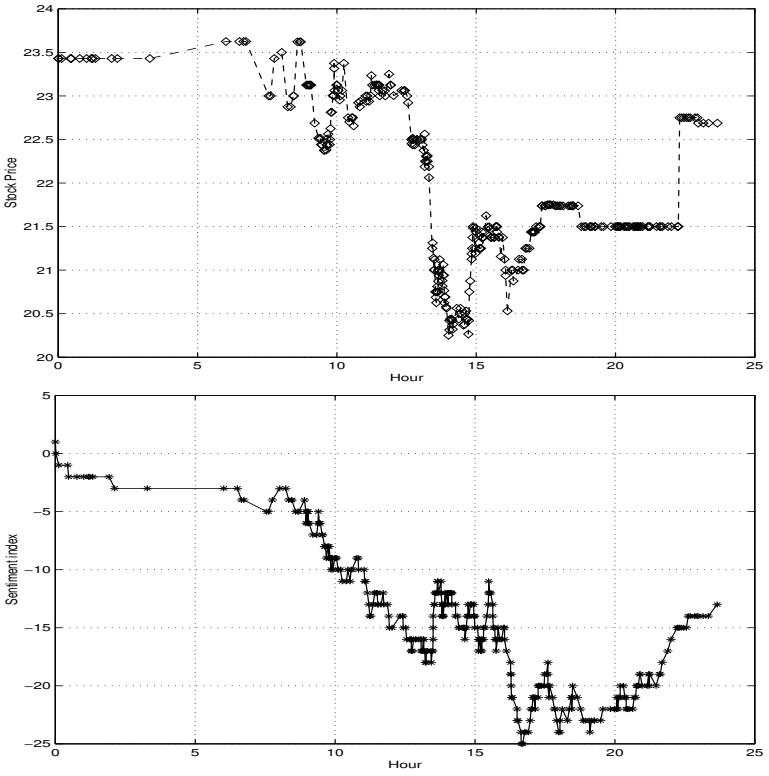


Figure 5: **Amazon, Inc., 11-December-2000**

The two plots below depict the stock price and sentiment for the 24 hours of the day. Each point corresponds to the arrival of a message on the stock board. The stock price graph is usually flat in the region outside regular trading hours. The stock price is contemporaneously collected whenever a message arrives on the stock board.

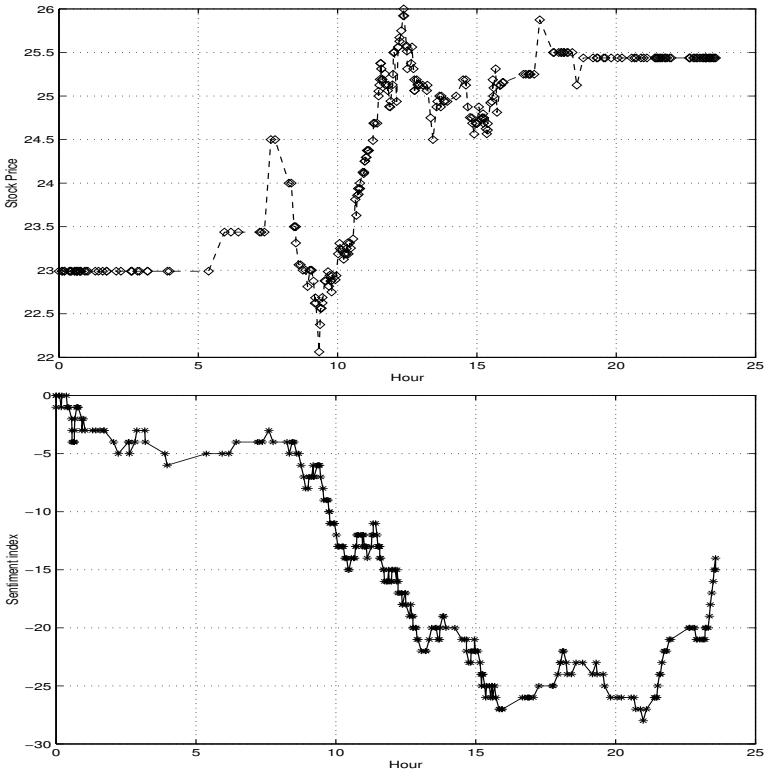


Figure 6: **Dell Computer, 13-November-2000**

The two plots below depict the stock price and sentiment for the 24 hours of the day. Each point corresponds to the arrival of a message on the stock board. The stock price graph is usually flat in the region outside regular trading hours. The stock price is contemporaneously collected whenever a message arrives on the stock board.

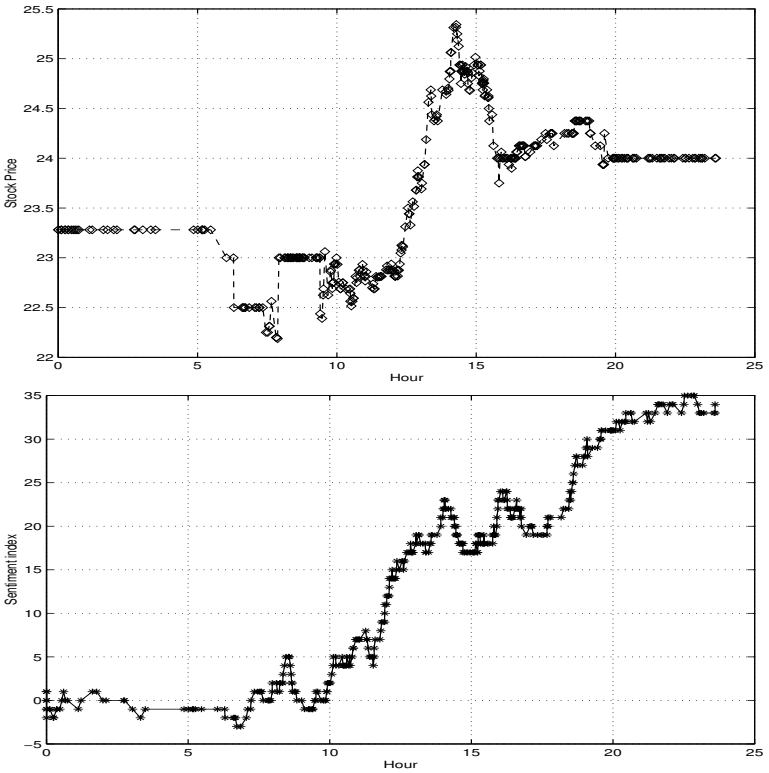
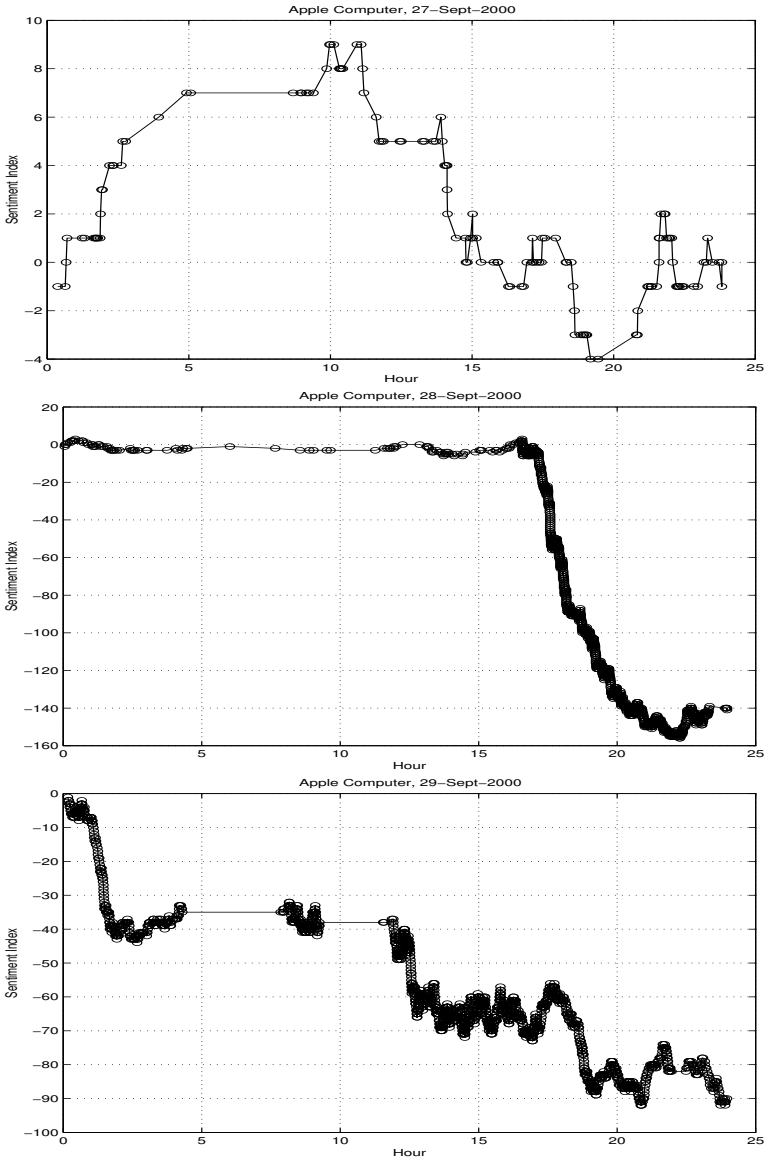


Figure 7: **Apple Computer, Event Effects and Sentiment**

The three plots below depict the sentiment index of Apple Computer for September 27, 28, 29, 2000. The earnings announcement was made a little after 4pm on September 28, and the immediate and persistent reaction of sentiment is evident from the plots.



earlier, on September 27, the sentiment index tracked up and then down, but revealed no persistent direction. The density of sentiment arrival may be inferred from the thickness of the plotted sentiment line on the graph (a thin line shows low message volume). A day later, when the announcement was made, just after 4pm, the sentiment index crashed and continued to take a beating for the next few days. The public announcement created an immense volume spurt on the message board (notice that the plot gets substantially thicker). Between 4pm and midnight on September 28, there were more than 2000 messages posted to the Yahoo board. There was clearly plenty of disagreement too, as the net index was minus 150, which is the net of many buy and sell messages. The reaction was persistent: the negative pattern after 4pm on September 28 extends well into September 29.

The visual relationship between sentiment and stock prices suggests that the classifier algorithms do not generate noise. In order to confirm this more formally we undertake three more analyses. Our first analysis explores the lead or lag at which the correlation between the stock and sentiment is maximized. A second analysis approach uses a "phase-lag" measure to probe further the lead-lag relationship. The third set of analyses consists of estimating formal statistical relationships between sentiment and stock returns.

3.4 Correlation Analysis

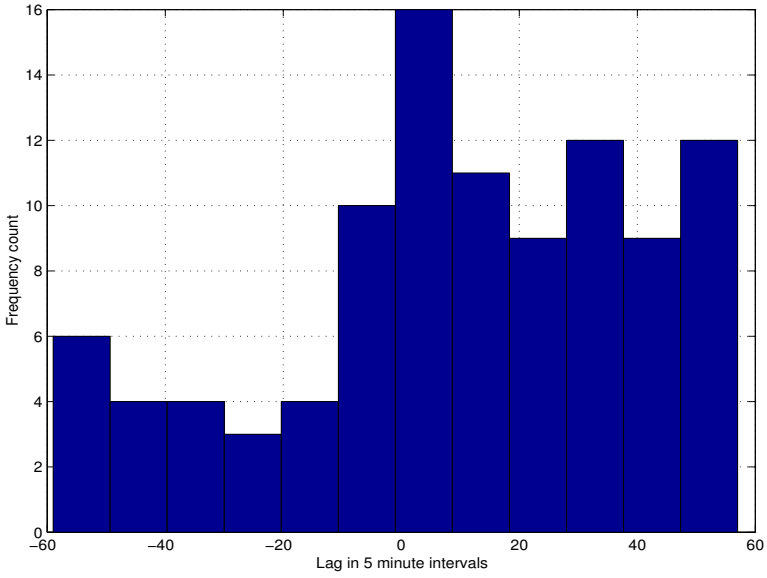
Using intra-day data, we computed the correlation between stock price and sentiment when both series were contemporaneous (zero lag), and for lags -5 hours (sentiment leads stock price) to +5 hours (sentiment lags stock price) hours at intervals of 5 minutes each. The lags are numbered from -60 to $+60$, each representing a 5-minute block. Hence, for each stock-day we computed 121 correlation numbers. We recorded the lag or lead at which correlation was maximum. After this was done for all sets in the data, we plotted the histogram of the results, presented in Figure 8. It is apparent from the figure that a majority of times the stock price leads the sentiment index, suggesting that small investor sentiment on message boards is reactive rather than predictive. The average number of lags is 10.04, i.e. message board sentiment reflects the stock price change with an average delay of 50 minutes.

Interestingly, it was pointed out to us that even if the sentiment index lags stock returns, there may be information in it.[11] For example, assume there are 3 posters A,B,C. A,B have no information whatsoever, and only post messages that follow the previous price change. C has perfect information and posts messages that predict the next price move. Therefore, the probability that the sentiment index moves in the same direction as the previous price change is 5/6 (A,B confirm for sure, and C does so half the time). The probability that a message is in the same direction as the following price change is only 2/3 (C is right, and A,B are half-right), yet there is informed trading information in the index. Hence, even though correlation is maximized when sentiment lags the stock price change, there may be information in the index.

[11] This example is not our idea, and we are grateful to a previous anonymous referee for pointing it out to us.

Figure 8: **Correlation based lag**

This figure presents a correlation analysis of the lead-lag relation between sentiment and the stock price. For each stock-day, the correlation between stock price and sentiment is computed when both series are contemporaneous (zero lag), and for lags -5 hours (sentiment leads stock price) to +5 hours (sentiment lags stock price) hours at intervals of 5 minutes each. The lags are numbered from -60 to $+60$, each representing a 5-minute block. Hence, for each stock-day we computed 121 correlation numbers. The the lag or lead at which correlation was maximum is reported in the histogram for all stock-day combinations in the data from the last two months of 2000.



3.5 Pattern-Recognition Analysis

Comparison of time-series relationships of sentiment to economic data is only one way to assess whether the classifier algorithms generate meaningful information. We now describe a different approach in which we compare graph patterns for lead-lag relationships. Rather than look at autocorrelation minutae, we employ a pattern recognition approach that examines the overall trend of the sentiment index relative to the stock series.

The algorithm is based on a small and simple set of graph patterns. Figure 9 displays the 8 canonical patterns which we ascribe to any graph. For example, the "min-max" pattern is said to be present in a graph if the graph begins at its minimum value, and ends at its maximum value. The "up-down" graph is one where the maximum and minimum are not the end-points of the graph, and the maximum value comes before the minimum. The "down-max" graph is one where the graph ends at its maximum, but the minimum is not an end-point. The other five types are self-explanatory and are depicted in Figure 9.

The type of chart is easily determined, since it is a function of exactly 4 points of the plot: the starting and end points, the maximum and minimum points. Therefore, each graph may have from 1 to 3 predominant up or down swings. For example, in the "max-min" graph the main change is a downward one, while in the "down-up" graph it could be either the up or down swing.

For each stock price graph, we examine the matching sentiment graph and assess the number of hours by which the detected pattern reflects a prediction of or reaction to the pattern in the stock graph. For example, suppose the sentiment graph is an "down-up" chart, then there are 3 phases: (i) a down phase, (ii) up phase, and (iii) a final down phase. The stock price plot may be a "max-down" chart. It has 2 phases: (i) a down phase and (ii) a final up phase. In a comparison of the 2 charts, the down phases start at the same time on both plots. Hence, this does not result in a phase difference. However, the up phase will most likely begin at a different time on each plot, and from these times, the phase difference is easy to compute. If the graph pair contains more than one pattern match, then all are assessed for the lead-lag computation. We look for matching patterns only after the opening of the trading day.[12]

For each stock-day pair of stock and sentiment graphs, we determine the amount of phase difference. The results are plotted in the histogram in Figure 10, which is a frequency distribution of phase-lags (in units of hours). On average, the message boards lag by 0.92 hours, i.e. approximately 55 minutes. By looking only at predominant patterns, the algorithm detects lead-lag relationships *conditional* on an information change or stock move of reasonable size. Most of the time, the phase difference is zero, implying that the sentiment index is contemporaneous to the stock price.[13]

[12] We wrote an analyzer to detect patterns. While the pattern recognition algorithm appears complex, it is actually very simple, since it relies on simple logic over just 4 inflexion points on the chart.

[13] The nature of these results appears to be confirmed in recent work by Antweiler and Frank [2004] on a different data set. See also Tumarkin and Whitelaw [2001].

Figure 9: **Canonical Graph Patterns**

This figure depicts the eight canonical graph patterns that are used for comparing the stock graph with the sentiment graph. The graphs are based on the idea that by treating the start and end points and the maximum and minimum values as key features, we get exactly eight possible graph types.

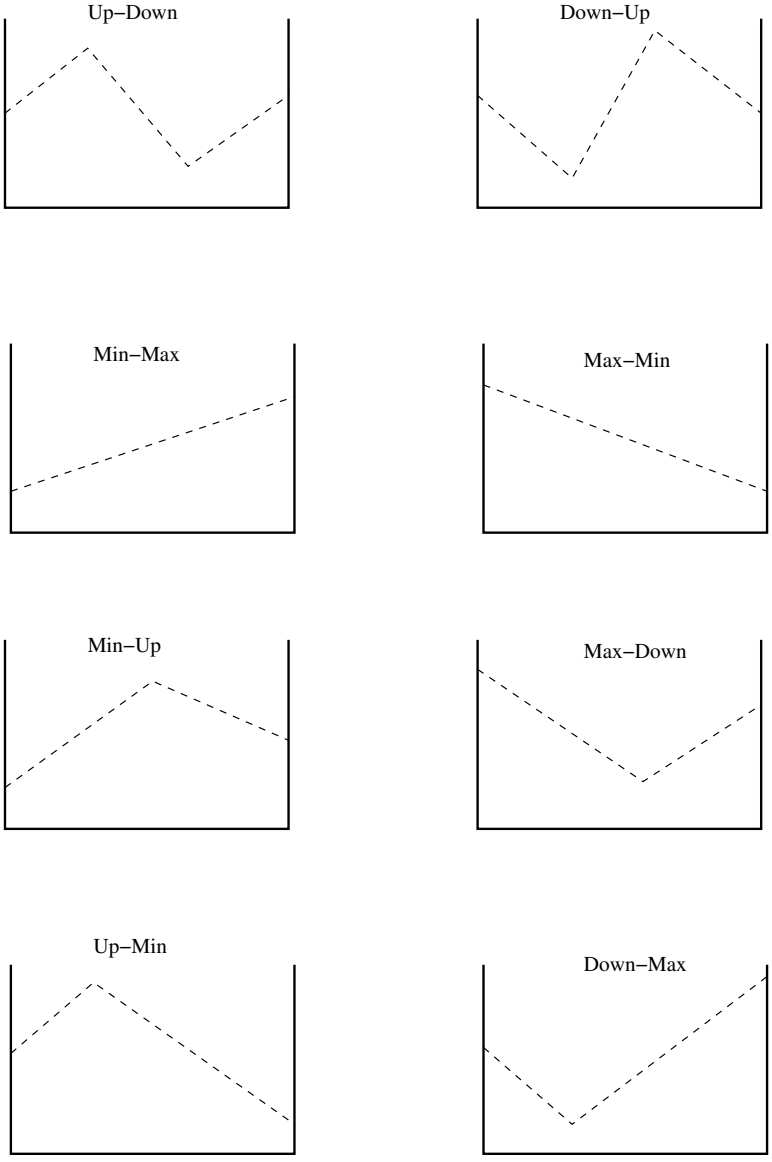
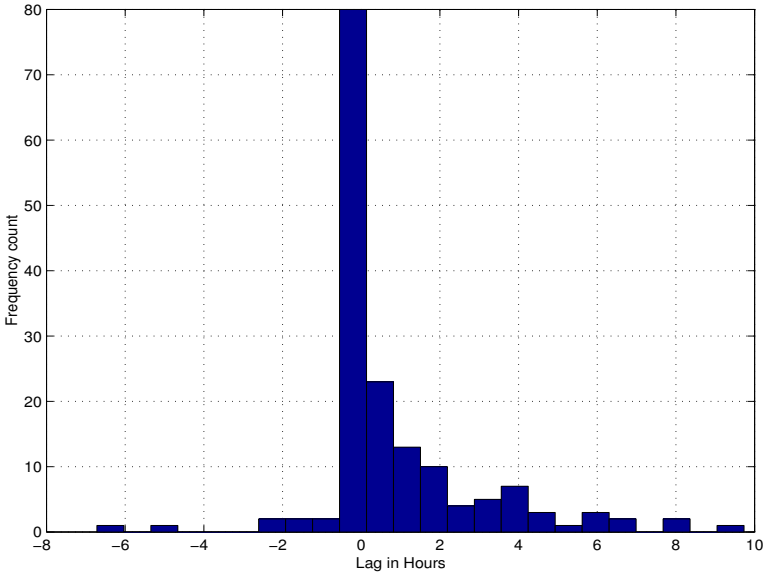


Figure 10: **Trading Phase-based Lag**

This figure presents the results of the lead-lag analysis obtained from a pattern-matching algorithm. Lead-lags are reported in hours based on a comparison of the stock and sentiment graphs for major directional changes, based on patterns arising from the simple set in Figure 9. Positive lags indicate that the stock price leads the sentiment index, vice-versa for negative lags.



3.6 Aggregating Sentiment into a High-Tech Index

In this section, we further validate the relationship of sentiment to stock data with two departures from the prior analyses. First, we examine daily sentiment, in contrast to the hourly aggregated data we used so far. Second, we look at both, individual stocks and an index created by aggregating sentiment across stocks, i.e. a cross-section of sentiment. This is useful to examine whether sentiment aggregates effectively in the cross-section. We used all messages posted for 35 stocks that comprise the Morgan Stanley High-Tech Index (MSH35) for the period June 1 to August 27, 2001. This results in 88 calendar days and 397,625 messages, an average of about 4,500 messages per day. For each day we determine the sentiment and stock return (close to close). Daily sentiment uses messages up to 4pm on each trading day, coinciding with the stock return close.

We also compute the average sentiment index of all 35 stocks, i.e. a proxy for the MSH35 sentiment. The corresponding equally weighted return of 35 stocks is also computed. These 2 time series permit an examination of the relationship between sentiment and stock returns at the aggregate index level. Table 2 presents the correlations between individual stock returns and full-day sentiment, and between the MSH index return and MSH sentiment. We notice that there is positive contemporaneous correlation between most stock returns and sentiment. The correlations were sometimes as high as 0.60 (for Lucent), 0.51 (PALM) and 0.49

Table 2: **Correlations of Sentiment and Stock Returns for the MSH35 stocks and the aggregated MSH35 index**
This table presents the correlations between daily sentiment and stock returns for 35 stocks in the Morgan Stanley High-Tech Index (MSH). Stock returns (STKRET) are computed from close-to-close. We compute correlations using data for 88 days in the months of June, July and August 2001. Return data over the weekend is linearly interpolated, as messages continue to be posted over weekends. Daily sentiment is computed from midnight to close of trading at 4pm (SENTY4pm).

Ticker	Correlations of SENTY4pm(t) with		
	STKRET(t)	STKRET(t+1)	STKRET(t-1)
ADP	0.086	0.138	-0.062
AMAT	-0.008	-0.049	0.067
AMZN	0.227	0.167	0.161
AOL	0.386	-0.010	0.281
BRCM	0.056	0.167	-0.007
CA	0.023	0.127	0.035
CPQ	0.260	0.161	0.239
CSCO	0.117	0.074	-0.025
DELL	0.493	-0.024	0.011
EDS	-0.017	0.000	-0.078
EMC	0.111	0.010	0.193
ERTS	0.114	-0.223	0.225
HWP	0.315	-0.097	-0.114
IBM	0.071	-0.057	0.146
INTC	0.128	-0.077	-0.007
INTU	-0.124	-0.099	-0.117
JDSU	0.126	0.056	0.047
JNPR	0.416	0.090	-0.137
LU	0.602	0.131	-0.027
MOT	-0.041	-0.014	-0.006
MSFT	0.422	0.084	0.210
MU	0.110	-0.087	0.030
NT	0.320	0.068	0.288
ORCL	0.005	0.056	-0.062
PALM	0.509	0.156	0.085
PMTC	0.080	0.005	-0.030
PSFT	0.244	-0.094	0.270
SCMR	0.240	0.197	0.060
SLR	-0.077	-0.054	-0.158
STM	-0.010	-0.062	0.161
SUNW	0.463	0.176	0.276
TLAB	0.225	0.250	0.283
TXN	0.240	-0.052	0.117
XLNX	0.261	-0.051	-0.217
YHOO	0.202	-0.038	0.222
Average correlation across 35 stocks			
	0.188	0.029	0.067
Correlation between 35 stock index and 35 stock sentiment index			
	0.486	0.178	0.288

(DELL). Only 6 stocks evidenced negative correlations, mostly in small magnitude. The average contemporaneous correlation is 0.188, which suggests that sentiment tracks stock returns in the high-tech sector. (We also used full-day sentiment instead of only till trading close and the results are almost the same – the correlations are in fact higher, as sentiment includes reactions to trading after the close).

Average correlations are weaker when one lag (0.067) and lead (0.029) of the stock return are considered. More interesting is the average index of sentiment for all 35 stocks. The contemporaneous correlation of this index to the equally-weighted return index is as high as 0.486. Thus, cross-sectional aggregation helps in eliminating some of the idiosyncratic noise, and makes the positive relationship between returns and sentiment salient. This is also reflected in the strong positive correlation of sentiment to lagged stock returns (0.288) and leading returns (0.178).[14]

4 Discussion

We developed a methodology for extracting small investor sentiment from stock message boards. Five distinct classifier algorithms coupled by a voting scheme are found to perform well against human and statistical benchmarks. Time series and cross-sectional aggregation of message information improves the quality of the sentiment index. Empirical applications evidence a relationship with stock returns (on a visual level, by phase-lag analysis, using pattern recognition and regression methods). Sentiment aggregated across stocks tracks index returns more strongly than with individual stocks.

Preliminary evidence suggests that market activity influences 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.

The algorithms in this paper may be used for other applications. 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. Third, our algorithms would enable monitoring of market activity. Regulators are concerned about market manipulation that goes undetected amongst the millions of messages posted to message boards every day. 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.

[14] We confirmed the statistical contemporaneous relationship of returns to sentiment by regressing returns on sentiment (T-statistics in brackets):

$$\underset{}{STKRET(t)} = \underset{(0.93)}{-0.1791} + \underset{(5.16)}{0.3866} SENTY(t), \quad R^2 = 0.24$$

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).[15] It contains parts-of-speech tagging information, and we wrote appropriate program logic to use this dictionary while analyzing messages for grammatical information.

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.[16]

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

[15] 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 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.

[16] We may think of the grammar as Roger 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.

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.[17] 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.[18]
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
```

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.

[17] 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.

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

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

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

[2004] 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.

[1999] 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.

[1998] 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.

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

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

[2002] 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.

[2000] Das, S., A. Martinez-Jerez, and P. Tufano (2000). "e-Information," working paper, Harvard Business School.

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

[1999] 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.

[1997] 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.

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

[2001] 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.

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

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

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

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

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

[1998] 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.

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

[1964] 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.

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

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

[1998] 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.