Mining of Concurrent Text and Time Series

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie,
David Jensen, and James Allan
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

We present a unique approach to identifying news stories that influence the behavior of financial markets. We describe the design and implementation of Ænalyst, a system for predicting trends in stock prices based on the content of news stories that precede the trends. We identify trends in time series using piecewise linear fitting and then assign labels to the trends according to an automated binning procedure. We use language models to represent patterns of language that are highly associated with particular labeled trends. Ænalyst can then identify news stories that are highly indicative of future trends. We evaluate the system in terms of its ability to predict forthcoming trends in the stock prices. We perform a market simulation, demonstrating that Ænalyst is capable of producing profits that are significantly higher than random.

1. INTRODUCTION

People read the news to both understand what is happening and what might happen in the future. News stories could explain why a presidential candidate is doing well in the polls or report on events that will adversely affect future polling results. Other stories may suggest why current economic performance is poor or predict an upturn in the economy in the coming months. Both approval ratings and economic performance can be viewed as time series because they are real valued data that change over time. News releases influence human behavior, and so may indirectly affect the fluctuations in these time series. Conversely, new stories may be written in response to fluctuations in a time series. We develop Ænalyst,[1] a system which models the dependencies between news stories and time series.

Ænalyst is a complete system, which collects two types of data, processes them, and then attempts to find the relationships between them. The two types of data are financial time series and time-stamped news stories. Once collected, the time series are redescribed into high-level features which we call trends. We then align each trend with time-stamped news stories, and learn language models of the stories that are correlated with a given trend. A language model determines the statistics of word usage patterns among the stories in the training set. Once we have learned a language model for every trend type, we can monitor a stream of incoming news stories and estimate which (if any) of our trend models is most likely to have generated the story. These estimates could be used by an investor or an automated trading system as recommendations to buy or sell a particular stock.

Our task is a special case of the *Activity Monitoring* task introduced by Fawcett & Provost[6]. The task involves monitoring a stream of *data* and issuing *alarms* that signal *positive activity* in the stream. In our case, the data are news stories and financial time series; unusual trends in the time series signify positive activity; and alarms take the form of recommended stories.

In the following section, we describe the system design of the Ænalyst and the technology used. Section 2.1 outlines our processing of time series. Sections 2.2 and 2.3 describe the process of learning the models of each trend. We evaluate the system in Section 3. Finally, we discuss related and future work.

2. SYSTEM DESIGN

Ænalyst is an implementation of a general architecture for the task of associating news stories with trends. Figure 1 illustrates our approach. A general system uses textual documents and numerical data over a time series. In Ænalyst, our numerical data is a history of stock prices, while the text is the news about the companies. We have a collection of 38,469 news articles for 127 stocks collected from Biz Yahoo! from October 15, 1999 to Feb 10, 2000. We also have stock prices for the 127 stocks over the same period of time. Using stock price information for a given company, we generate trends. The news articles are aligned with the price trends according to when the articles were released and when each trend occurred. Using the articles aligned with the trends of each type, we generate language models for the trend types. We can then use these models to correlate new articles with trends. We use these correlations as an indication of the articles' influence on the future of the time series. To perform predictions, we select the trend type whose model is most likely to be the source of the news article.

[1] From e-Analyst, pronounced "analyst".

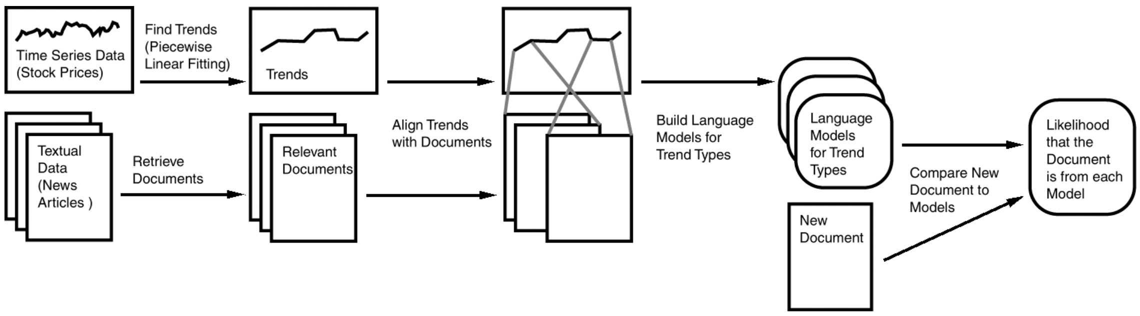

Figure 1: System Design

2.1 Redescribing Time Series

Stock traders do not base their decisions on single observations of a stock's price. Rather, decisions are made with respect to higher level trends in the stock price.

We define a trend as an interval in time, consisting of 3 or more observations, during which the changes from observation to observation are predominantly positive (an *increasing* trend) or negative (a *decreasing* trend). Due to latencies inherent in stock trading, trends have obvious importance relative to single observations or pairs of observations, and consequently news articles that predict trends are of particular relevance.

Furthermore, one can (and should) only attend to some types of trends. Relatively flat trends are of no importance to a trader who wants to make profit, while trends of steep increase are of importance for traders looking for a good buy, and trends of steep decrease are important for a trader who wishes to sell before taking a loss. We believe that the concept "slope is important" generalizes across domains.

Our system for redescribing time series builds from these two ideas: that a series of observed stock prices may be broken up into trends, and that trends may be classified into increasing, decreasing, or relatively flat.

2.1.1 *Identifying Trends*

Breaking time series of real-valued observations into component trends is not a new problem. An obvious technique for tackling this problem is *piecewise linear regression*, sometimes called *piecewise segmentation*, and many algorithms for piecewise segmentation were pioneered by Pavlidis & Horowitz[11].

The idea behind most piecewise fitting algorithms is to redescribe a time series by a sequence of regression lines which minimize some error metric (typically mean square error) over the length n of the series. Popular fitting algorithms work either top-down, starting with a single regression line for the whole series, and greedily splitting the sequence until some stopping criterion is met, or bottom-up, starting with $\frac{n}{2}$ segments, and greedily merging sequences until the stopping criterion is reached. A typical stopping criterion might be a threshold for the summed mean square error or a fixed number of segments, both specified in advance. Our piecewise fitting algorithm uses a top down procedure with an automatic stopping criteria based on the t-test.

The algorithm works by passing a window of length δ over the entire sequence of length n.[2] At each step of its pass over the data, the window is broken into two halves, about point i, and a regression line is fit to each half. The difference between $before_i$, the slope of the linear fit for the points in the window before i, and $after_j$, the slope of the linear fit for the points in the window after i is recorded. After the window has been passed over the complete time series, the point j which maximizes the difference in slope ($|before_i$ - $after_j|$) is considered as a candidate location for segmenting the time series.

The next step of the algorithm is to test whether the series should be split about j. Precisely, we test the hypothesis that the slope of the segment fitting the $\frac{\delta}{2}$ points before j is equal to the slope of the $\frac{\delta}{2}$ points after j. The following function computes the t statistic for the difference in slopes.

$$t = \frac{before_j - after_j}{\hat{\sigma}_{b_1 - b_2}}, \quad (1)$$

where $\hat{\sigma}_{b_1 - b_2}$ is the estimate of the standard error for the difference of slopes from two regression lines. This estimate is computed from the pooled variance, or sums of squares of residuals from each least-squares line:

$$\hat{\sigma}_{b_1 - b_2} = \sqrt{\hat{s}^2_{pooled}\left(\frac{1}{SS_{X_1}} + \frac{1}{SS_{X_2}}\right)} \quad (2)$$

$$\hat{s}^2_{pooled} = \frac{SS_{residual_1} + SS_{residual_2}}{df} \quad (3)$$

$$SS_{residual} = \left(1 - r^2\right) SS_Y \quad (4)$$

In Eqs. (2)-(4), SS_X and SS_Y are the sums of squares for the independent and dependent variables (respectively, time and stock price), and r^2 is the goodness of fit for the regression line. Finally, the t statistic is compared to the t distribution with $n_1 + n_2 - 4$ degrees of freedom to see whether the hypothesis of equal slope should be rejected[8]. If the hypothesis is rejected, the sequence is split at point j, and then the fitting algorithm is executed recursively on each half of the sequence. This test is also the stopping

[2] For results reported here, the window size is length 10, roughly an hour's worth of observations in each half of the window.

criteria for the segmentation, though, and if the hypothesis is not rejected, the algorithm terminates.

We regard each segment in the piecewise fit to be a trend. The significance of the trend is defined by its regression statistics. The slope of the line indicates whether the trend is of interest.

Results typical of the piecewise fitting algorithm when applied to stock data are shown in figure 2. The unmodified data are shown at left for ticker symbol YHOO, with piecewise linear fits shown at right. At this scale (observations taken every 10 minutes during open market hours, from October 14, 1999 to December 7, 1999), the difference is difficult to ascertain. As noted in Keogh & Pazzani[7], there is little loss of resolution in exchange for a reduction in the amount of data. A redescription allows our system to make decisions that are based on variable length intervals lasting hours or days, rather than minute-by-minute or closing prices.

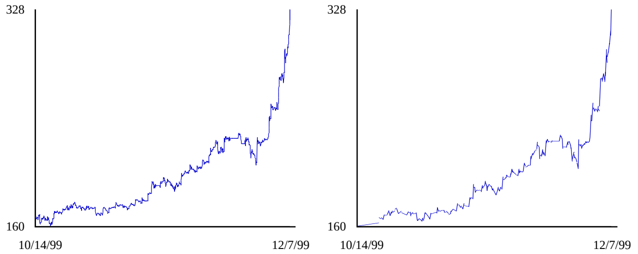

Figure 2: On the left, unprocessed stock data for ticker symbol YHOO, over a portion of October to December in 1999. On the right, the same data's piecewise segmentation.

2.1.2 Discretizing Trends

Secondly, we redescribe our time series as discrete trends. This step is a subjective one in which we assign labels to segments based on their characteristics: length, slope, intercept, and r^2. These labels will be the basis for prediction and assigning relevance to news stories.

A stock trader is most likely to focus primarily on two trend characteristics: slope and r^2 (hereafter referred to as *confidence*). These inform the trader of the sharpness of price change, and the confidence that the trend is a good fit to the actual behavior of stock price.

We might discretize a sequence of trends, described by slope and confidence, in a number of ways. One might simply bin the trends, manually selecting cutoff values, and assigning labels like "surge, high confidence" to the bins. Another approach would be to use an unsupervised clustering algorithm to automatically generate clusters of trends. That way, any structure inherent in the segmented time series would be exploited automatically, and labels could be assigned as with the binned data to reflect that structure.

We implemented a distance based agglomerative clustering algorithm to automatically cluster stock trends based on their slopes and confidence. The algorithm, adapted from Everitt[5], works as follows.

First, compute a distance matrix for the trends of a particular stock. The distance between two trends is simply the Euclidean distance between their standardized slopes and standardized confidences. Next, create a cluster for each segment. Finally, select the two closest clusters C_1 and C_2 based on the *group average distance* between segments in C_1 and those in C_2:

$$GAD(C_i, C_j) = \frac{\sum_{c_x \in C_i} \sum_{c_y \in C_j} D_{x,y}}{|C_i||C_j|} \tag{5}$$

The minimum-distance clusters C_1 and C_2 are candidates for merging. We run a t-test of the inter-cluster segment distances versus the intra-cluster segment distances under the hypothesis that all these distances are drawn from the same distribution (and thus the clusters should be merged). If the hypothesis is not rejected by the t-test, we merge C_1 and C_2, and repeat for the next minimum distance pair. If the hypothesis is rejected, we also move on to the next minimal distance pair. If no pair is merged after one whole pass through the clusters, the clustering algorithm stops.

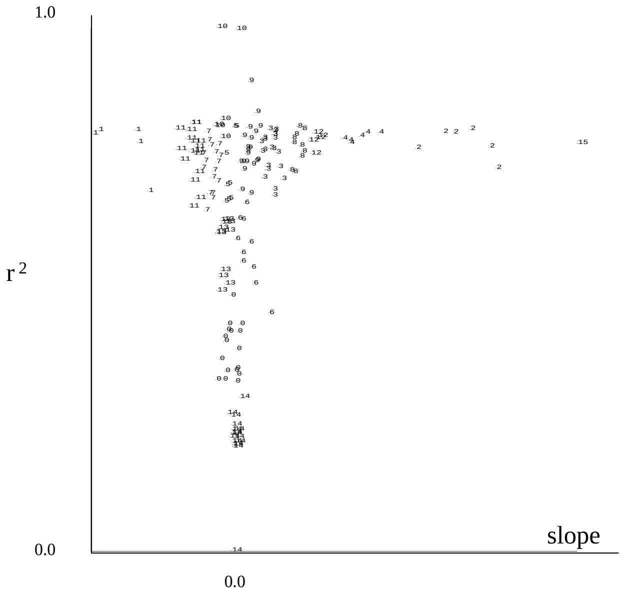

Figure 3: A plot of confidence versus slope for stock symbol YHOO. Different symbols correspond to different clusters generated by the agglomerative clustering algorithm.

The plot in figure 3 shows confidence plotted against slope for the stock YHOO. There are a total of 16 different clusters. Perhaps more importantly, though, is the structure that the graph reveals. The "funnel" shape to the graph of confidence versus slope is typical of all our linear fits. We can exploit this structural property by simply ignoring r^2. As we noted before, segments of importance are those with steep slopes, which we might call *surges* (for positive slopes) and *plunges* (for negative slopes). Since all such segments necessarily have relatively high confidence values, we need not worry.

In practice, then, it is not particularly important to go to the length of first clustering segments and then assigning labels. The structure present in the YHOO segmentation is typical of nearly all the stocks we followed. While clustering is the principled approach to discretizing the stocks, we have found that a simple binning procedure can effectively single out the trends that we find most interesting. Segments with slopes greater than or equal to 75% of the maximum observed segment slope are labeled SURGE, and those with slope greater than or equal to 50% of the maximum observed slope are labeled SLIGHT+. Similarly for negative slopes, we label PLUNGES and SLIGHT-. All other segments are labeled NR for NO RECOMMENDATION.

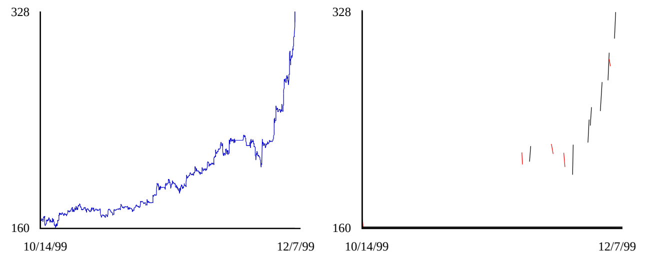

Figure 4: On the left, a plot of the stock prices of YHOO for October and December 1999. On the right, the segments of interest generated by our system.

The graph to the right of figure 4 shows the segments of interest for stock symbol YHOO from October to December of 1999. There are 11 such segments, each of which is very steep in slope.

2.2 Aligning the trends with news stories

Selecting an appropriate set of news stories is a very important step in our modeling. If done carelessly, it can be a source of noise when generating the language models. To ensure that all training stories are at least marginally relevant to the companies we use external relevance assignments obtained from our source of news (http://biz.yahoo.com/). For every stock symbol, Yahoo! maintains a list of stories that are considered to be relevant to that stock symbol. Note that availability of these assignments are not crucial for Ænalyst: in the absence of external relevance assignments, we could use traditional information retrieval techniques to select documents relevant to a given company.

Once we have selected the stories that are relevant for a particular stock, we can associate groups of these stories with trends. In order to learn models that assist Ænalyst in suggesting future behavior of a time series, we associate a document with a trend if its time stamp is h hours or less before the beginning of the trend. For instance, using a five-hour alignment, we would associate all documents released from 10:00AM to 2:59PM with a trend that began at 3:00PM. Figure 5 illustrates this.

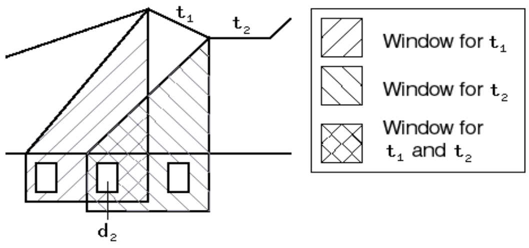

Figure 5: Current Alignment

The alignment method we use can result in a document being associated with more than one trend. Document d_2 in the figure is associated with both trends t_1 and t_2. While this may seem contradictory, it is possible for d_2 to influence both trends t_1 and t_2. We can reduce the amount of overlap between time windows associated with trends by decreasing h, the number of hours included in the window. However, this can reduce the number of documents that are associated with each trend, yielding fewer examples with which to train our models. In the experiments we performed, we determined that using a window of 5 to 10 hours tends to work best. We treat all documents aligned with a trend as having equal importance to a trend.

An alternative alignment method would be to align documents with the trend that was happening when the article was released. This would shift the system from a predictive system to an explanative system by looking for correlations between articles and current behavior of the time series. While this does not appear to be useful for predictions, we will demonstrate that using this type of alignment results in very high mean performance (see Section 3.3 for details).

2.3 Language Models

After aligning news documents with trends in the time series, we can estimate a language model that is characteristic of every trend type. For example, a language model may learn that words like *loss, shortfall, bankruptcy* are highly likely to precede a downward trend in the stock price, while *merger, acquisition, alliance* are likely to be followed by an upward trend.

Language modeling[12, 15] provides a formal framework for text classification with respect to a set of targets (trends in our task). After making certain assumptions about patterns of word occurrence in natural language, we can formally estimate the likelihood of a trend, given the observed distribution of words in a set of news stories. Given a set of stories $\{D_1 ... D_m\}$, we would like to estimate the probability that a trend of type t (e.g. a *surge*) will occur in the near future. To do this, we associate a language model M_t with every trend type t. A language model represents a discrete distribution over the words in the vocabulary. M_t specifies an expected usage of words on the onset of the trend t. We then estimate how likely it is that a set of documents was generated by the model M_t. For a given set of documents we are interested in the model that is most likely to generate the set of observations $\{D_1 ... D_m\}$:

$$\begin{aligned} M_{best} &= \arg\max_{t \in trends} P(M_t | \{D_1 ... D_m\}) \\ &= \arg\max_{t \in trends} \frac{P(\{D_1 ... D_m\} | M_t) P(M_t)}{P(\{D_1 ... D_m\})} \end{aligned}$$

In the current implementation of the system we assume a uniform prior $P(M_t)$, though in the future we will condition $P(M_t)$ on the frequency of the trend t in the time series.[3] We

[3] A uniform prior does not affect the evaluations based on the ROC/DET curves (Section 3.1), but it does affect the market simulation (Section 3.2). Since DET evaluation is done independently for each trend type t, $P(M_t)$ is constant in each individual evaluation and does not affect the ranking of documents with respect to M_t.

estimate a prior $P(\{D_1...D_m\})$, as the probability that the set of documents was generated by the background (General English) language model $P(\{D_1...D_m\}|GE)$. This is somewhat different from a common formulation for a prior: $\sum_i P(\{D_1...D_m\}|M_i)P(M_i)$, but conditioning on the background language model has proven effective in text categorization research[15].

Assuming that individual documents in the set $D_1...D_m$ are random samples from a common distribution, we can expand the formulation as:

$$M_{best} = \arg \max_{t \in trends} \prod_{i=1}^{m} \frac{P(D_i|M_t)}{P(D_i|GE)}$$

To estimate $P(D_i|M_t)$ we make an assumption that words in D_i are generated independently of each other. The assumption of word independence is a common practice in text classification research. There is some evidence that preserving word dependencies does not improve the accuracy of probabilistic models of text (e.g. pairwise dependence model by van Rijsbergen[14]). There is also a more general result by Domingos & Pazzani[3], which demonstrates that in some cases, naive Bayesian classifiers give very good results even when dependencies exist in feature distributions. Note that after making the word independence assumption, we effectively arrive at a naive Bayesian classifier:

$$M_{best} = \arg \max_{t \in trends} \prod_{i=1}^{m} \prod_{w \in D_i} \frac{P(w|M_t)}{P(w|GE)}$$

Here w represent individual word occurrences in the document D_i. We could use a maximum likelihood estimator for $P(d_i|M_t)$, which is simply the number of occurrences of w in M_t divided by the total number of tokens in M_t; however, this turns out to be problematic. Since our models may be sparse, some words in a given document D_i may have zero probability under a given model M_t, resulting in $P(D_i|M_t) = 0$. To alleviate this problem we use a smoother estimate: $P(w|M_t) = \lambda_t P_{ml}(w|M_t) + (1 - \lambda_t)P(w|GE)$.

This formulation, commonly known as linear back-off, allocates a non-zero probability mass to the terms that do not occur in M_t. We set λ_t to the Witten & Bell[16] estimate $N_t/(N_t + U_t)$ where N_t is the total number of tokens and U_t is the number of unique tokens in the model M_t. Since modeling the market is a dynamic on-line task where language usage may change, we may encounter words that are not present in GE. To ensure that a new term does not force a zero probability for a document, we smooth GE in a similar fashion using a uniform model for the unseen words: $P(w|GE) = \lambda_{GE} P_{ml}(w|GE) + (1 - \lambda_{GE})/N_{GE}$.

3. EVALUATION

Associating patterns in text with patterns in time series is a fairly novel task. As such, it does not yet have accepted evaluation metrics. A number of metrics could be defined to address such evaluation. The most simple is classification accuracy, as used in a similar context by Cho *et al.* [2]. Another example is the Activity Monitoring Operating Characteristic (AMOC), suggested by Fawcett & Provost[6]. For the case of market prediction, they define the scoring function in terms of ability to predict a 10% or more jump in the market price within 34.5 hours of receiving relevant news. Our evaluation is similar to the work of Fawcett & Provost[6], but we use a more traditional ROC-style measures in place of AMOC. In this work we alternate between two types of evaluation. First, we attempt to evaluate the discriminating power of our language models using the classical classification framework. Second, we attempt to evaluate whether the system could be used to profitably buy and sell stocks. This dual evaluation will both exhibit the technological potential of language models for this task and satisfy the practical curiosity of investors.

3.1 Evaluating Language Models

The results in this section address the issue of how well each language model M_t discriminates between the documents followed by trend t from the documents that are not followed by trend t. We perform n binary evaluations, one for each model $M_1 \dots M_n$. We use Detection Error Tradeoff (DET) curves (see [9]), which are similar to ROC curves, common in classification literature.

In our analysis we focused on a set of 127 stocks over the period of Oct 1999 - Feb 2000. The stocks were selected based on two criteria: the average amount of news reporting about that stock, and how frequently that stock is traded. Our price data was sampled every 10 minutes during the market hours, resulting in over 3600 data points for every stock. The price data was re-described into trends as explained in Section 2.1 to produce an average of 450 trends per stock. Our news collection contains over 38,000 news stories, gathered online over the same period. Each story contains a reference to at least one of the stocks we are tracking. The documents (D) for each stock were then aligned with the future trends (t) for the same stock, to provide the labeled set of pairs $\{t, D\}$, which we use for training and testing the models.

To obtain a good estimate of mean performance, we use 10-fold randomization in the following way. We randomly split the set of pairs $\{t, D\}$ into a training set (90%) and a testing set (10%). In doing this, we ignore the temporal ordering of $\{t, D\}$ pairs. This does not present a problem since our evaluation is a binary classification task, and our model does not learn in any way from the pairs in the testing set. For each trend type t (e.g. a *surge*) we form a language model M_t using all the documents D that are labeled with t in the training set.[4] Each model M_t is then used to assign probabilities to all documents in the testing set. The procedure is repeated 10 times with a different random training and testing set each time. We use pooled averaging to produce a single DET curve for each trend type.

Figure 6 demonstrates how well Ænalyst identifies news stories that are followed by a surge in the stock price within 10 hours from the story. From the distributions on the top, we see that our Language Model assigns higher beliefs to the stories that are in fact followed by a surge. A DET curve

[4]This corresponds to forming a *universal* language model across all stocks. The intent is to capture uses of language that will affect any stock in a similar fashion (e.g., *takeovers*). In other experiments we form language models separately for every stock.

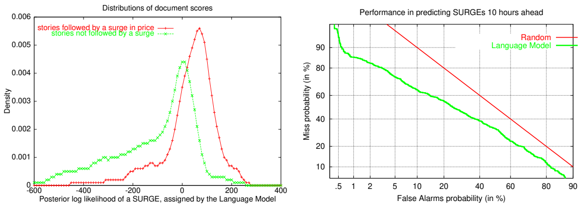

Figure 6: Language model can separate stories that are followed by a surge from stories that are not. Left: belief densities. Right: corresponding DET curve.

on the bottom of Figure 6 allows us to analyze the errors of our system at varying levels of risk. For example, a system is capable of achieving a 10% recall (90% miss), while keeping the false alarm rate around 0.5%. That means that we correctly identify 10% of stories that precede a surge in the stock price, while the system would correctly eliminate 99.5% of stories that do not precede a surge. An investor who is willing to take more risk, could lower the threshold of the system, and be alerted to 40% of stories preceding a surge, but the false alarm rate would increase to 15%.

3.2 Market Simulation

For the task of market prediction, the ultimate evaluation of performance is whether or not the system would be able to make a profit. In the following simulation we construct a strategy which mimics the behavior of a day trader who would use system predictions in a very simple fashion: if Ænalyst indicates that a story is likely to precede an upward trend for the stock, the system will invest in that stock; if the predicted trend is downward, the system sells the stock. To simulate this very simple strategy, we induced a separate language model for each stock for each trend type using 3 months worth of training data between October and December 1999. Then, for 40 days starting on January 3rd, we have our system monitor the news. Every time a new story appears for some company, Ænalyst determines which trend model is most likely to have generated it.

If the most likely trend is positive, our system will purchase $10,000 worth of the stock. We assume sufficient credit to purchase $10,000 worth of stock whenever we need to. Another assumption is zero transaction cost, which is common in similar evaluations (the transaction costs are easily absorbed by increasing the volume of each transaction, as long as we are making profit). After a purchase, the system will hold the stock for 1 hour. If during that hour we can sell the stock to make a profit of 1% ($100) or more, we sell immediately. At the end of the hour, we sell the stock at the current market price, and take a loss if necessary. We define 1-hour holding time to exclude non-market hours, so an hour at the end of the day can “spill” overnight.

If the most likely trend is negative, Ænalyst will *sell short* $10,000 worth of the stock (this means selling the stock we do not yet have in hopes of buying it later at a lower price).[5] Again, Ænalyst will hold the stock for 1 hour. If during the hour the system can buy the stock at a price 1% lower than *shorted*, the system buys the stock to cover. At the end of the hour, Ænalyst buys the stock at the current market price.

[5] We do not model stock exchange restrictions which prohibit shorting the stock during a down-tick.

Stock	Gain	Stock	Loss
IBM	$47,000	Disney	-$53,000
Lucent	$20,000	AOL	-$18,000
Yahoo	$19,000	Intel	-$14,000
Amazon	$14,000	Oracle	-$13,000

Table 1: Best and worst cumulative profits over 40 days of trading

This model represents an extremely simplistic and greedy day-trading strategy. One could easily improve the strategy using a plethora of indicators available to a real trading system, such as the general market conditions, recent history of trends for a particular stock, trends of related stocks etc. We choose to focus on our simplistic strategy to isolate the contribution of language models, separated from any other indicators that may be of assistance in trading. This simple strategy provides an absolute lower bound on how well a real trading system could perform if it took advantage of predictions provided by Ænalyst.

Table 1 summarizes cumulative profits that Ænalyst would make if it followed the strategy described above. The cumulative performance of our strategy varies widely from stock to stock. Out of over 100 companies that we tracked, we show only the best and the worst performing stocks. System predictions were most profitable for IBM, Lucent, Yahoo and Amazon, while most money was lost on Disney, AOL, Intel and Oracle.

The numbers mean that buying $10,000 worth of Yahoo every time that Ænalyst predicts an uptrend (and shorting $10,000 for every downtrend prediction) would result in net profit of $19,000 after 40 days of trading. Of course, we have to realize that the system traded Yahoo about 570 times during these 40 days, so we made less than $50 (0.5% gain) on an average transaction. A natural objection to our extremely active trading strategy is that perhaps investing in these stocks in January and holding them for 40 days would result in a larger (and certainly more effortless) profit. However, the four companies we earned the most profit from all either stayed the same, or decreased slighty, so buying and holding each stock over the same period would result in a cumulative loss. For instance, shares of Lucent traded at $75 in the first days of January, but dropped to below $50 by February.

After the 40-day simulation, the cumulative earnings of our system, pulled over all stocks, totaled $21,000. We used a randomization test[4] to determine if these earnings are statistically significant. Specifically, we conducted 1000 trials of a system where buying and shorting decisions were made randomly, without reference to the actual content of news stories. Then we compared our actual earnings to the distribution of cumulative earnings produced from the randomized trials. The randomized system was constrained to buy and short particular stocks with the same probability per stock as the actual system, and decisions about buying and shorting a particular stocks were made at the same times as in the actual system. After a decision to buy or short, the randomized system followed the same strategy for selling as was followed in the actual system. The results of the randomized system equaled or exceeded $21,000 in only eight of the 1000 trials, and thus the performance of the actual system is significant at the 1% level. The mean over the randomized tests was -$9,300 and the standard deviation was $13,600.

The simulation we presented uses a very simple strategy. We do not suggest this strategy for real trading environments. Our simulation is a proof of concept, demonstrating that news releases indeed have a strong influence on the market, and suggesting a way of predicting and leveraging that influence.

3.3 How quickly news influence stock prices?

In the previous section we demonstrated that news releases do indeed have an influence on the trends in stock prices. A natural question to ask is how quickly the market responds to news releases. We could explore the issue by attempting to correlate news stories with trends further and further in the future and observing at which distance we get the best correlation. Recall from section 2.2, that in the training phase, we could align trends with stories that precede the trend by any number of hours, or we could align with stories that are released in the duration of a trend. We performed the following experiments: concurrent alignment, and with trends up to 1, 5 or 10 hours in the future. For aligning 1 hour in advance we get a labeled set of 10,863 $\{t, D\}$ pairs; for 5 hours we get 49,872 pairs; and for 10 hours we get 75,610 pairs.[6] When we align trends with the concurrent stories (simultaneous alignment), we get 35,681 pairs.

We performed a DET analysis, similar to the one in section 3.1, to examine the impact of alignment on all types of trends we considered. We observed that aligning stories with trends that happen at the same time gives the overall lowest DET curves, having lower false alarm rates at any level of recall. Aligning 1 hour in advance results in language models with little discrimination power: the error tradeoff curves are all very close to random performance at any false alarm rate. Aligning 5 hours in advance produces models that are close to random at the high false alarm rate, but are noticeably better than random at low false alarm rates (low false alarm rate is more important for many practical applications). Aligning 10 hours in advance provides relatively low errors in predicting surges (in fact better that any other alignment). However, for any other type of trend this alignment performs considerably worse, which leads us to doubt its usefulness.

Based on the DET analysis, we would favor simultaneous alignment, as the most stable for all trend types at all levels of recall. To verify whether this is a reasonable choice, we repeated our market simulation (Section 3.2) using different alignments to form the training set of our models. The results are summarized in the first four rows of Table 2. Rather than reporting the sum of cumulative gains over all stocks (as we did in Section 3.2), we report the mean gain, as well as the median, best, worst and standard deviation over all the

[6]Recall that a single story may be aligned with more than one trend.

Training	Align	Mean	stdev	Median	Best	Worst
Specific	Sim.	$3840	$9270	$0	$40460	-$24070
Specific	1 hr	$740	$13180	$840	$47870	-$53020
Specific	5 hrs	$290	$10480	$0	$47870	-$53020
Specific	10 hrs	$2360	$8440	$0	$47870	-$24530
Universal	Sim.	$2040	$5940	$350	$32920	-$7330
Universal	1 hr	$1170	$5940	$0	$27210	-$23610
Universal	5 hrs	$1920	$6440	$160	$33040	-$14110
Universal	10 hrs	$1650	$5510	$150	$34160	-$8460

Table 2: Impact of alignment window and universal vs. specific models

stocks. The simulation from Section 3.2 is presented in the third row (5 hour alignment window). This turns our to be our worst alignment ($290 average cumulative profit), and it also has the highest variance of all alignments. Surprisingly, even 1-hour alignment results in better performance. Our best alignment scheme is simultaneous alignment, giving us an average cumulative profit of $3840 per stock. We note that all alignment schemes in the first four rows have an extremely high variance. This implies that in an actual implementation of a trading system we would have to be very careful in selecting the set of stocks for which our Ænalyst performs reasonably well.

3.4 Specific or universal models?

Another issue of particular importance in training our models is whether we opt to use stock-specific models or universal models. The distinction is highlighted by Fawcett & Provost[6] in their work on activity monitoring. In stock-specific models, we train a separate set of models for each stock. In universal models, we train the same set of models across all stocks. The market simulations in Section 3.2 and Section 3.3 used stock-specific models. These models have the advantage that they can learn the specific model of language that affects each stock. The main disadvantage of stock-specific models is the small size of their training sets: since we train a separate set of models for each stock, companies that are rarely covered by news releases are at a disadvantage. Universal models overcome that difficulty: we train one set of models for all stocks at once. The idea behind universal models is learning the patterns of language that affect all (or most) stocks in the same way. Universal models are not prone to shortage in training data because all news from all stocks is used in training. The price is inability of the universal models to distinguish the specific effect of news on a particular company.

Table 2 highlights the difference in performance between universal and stock-specific models. We observe that while the absolute best performance is certainly lower than stock-specific models ($2040 vs. $3840), universal models exhibit drastically lower variance across different stocks. This is not surprising, since we use the same model (but a different testing set of documents) for each stock. Universal models are also much less sensitive to the alignment conditions. Table 2 suggests that stock-specific models are characterized by lower bias (provided appropriate alignment) but very high variance, while universal models exhibit larger bias and lower variance. This observation leads us to consider a mixture of universal and stock-specific models, combining the best of both worlds.

4. RELATED WORK

The Ænalyst draws on progress made in several areas: text classification[1], language models[12], and time series analysis and clustering[13]. Ænalyst's strengths come from its use of language models and redescriptions of time series as trends. Our treatment of stock prices is different from the work of Fawcett & Provost[6], who focus on raw time series when trying to predict a shift of at least 10% by interpreting the problem as an *activity monitoring* problem. Furthermore, by using language models, our system can incorporate the entire vocabulary used in the text, rather than concentrating on feature phrases selected by experts as was done by Cho *et al.*[2] in their work of trying to predict the closing price of the Hang Seng stock index. There have been numerous other works including McCluskey[10] that attempt to predict stock behavior by relying purely on the analysis of time series data.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated how to use language models to successfully associate stories and trends in time series. We conclude that piecewise linear regression is a useful tool for describing time series, particularly in this task where we are interested in high-level view of stock fluctuations. We also demonstrated that language models represent a good framework for associating news stories with forthcoming trends. In our market simulation, we showed that a simple greedy strategy allows Ænalyst to profit from market fluctuations. Finally, we addressed in detail the issues of aligning trends with news stories to be used for training, and highlighted the differences between stock-specific and universal trend models.

For future work, we would like to experiment with richer document features. In this work we treat articles as though they are a bag-of-words. These features could be augmented with associations among trends, pairs of related words that are significant across many documents, and also relations between objects within a document that could add interesing knowledge and make our models more complete. We further plan to investigat the use of mixture language models for predicting trends in stock prices.

6. ACKNOWLEDGMENTS

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement numbers EEC-9209623 and EIA-9820309. This material is also based on work supported in part by Defense Advanced Research Projects Agency/ITO under DARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235, in part by SPAWARSYCEN-SD grant number N66001-99-1-8912, and in part by DARPA/AFOSR contract F49620-97-1-0485. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] J. Allan, J. Callan, F. Feng, and D. Malin. INQUERY and TREC-8. In D. Harman, editor, *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[2] V. Cho, B. Wutrich, and J. Zhang. Text processing for classification. Technical report, The Hong Kong University of Science and Technology, 1998.

[3] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[4] E. Edgington. *Randomization Tests, Third Edition.* Marcel Dekker, New York, 1995.

[5] Brian Everitt. *Cluster Analysis.* John Wiley & Sons, Inc., 1993.

[6] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the 5th International Conference on KDD*, 1999.

[7] Eamonn Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *AAAI-98 workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, pages 44–51, 1998.

[8] S. Kotz and N.L. Johnson. *Encyclopedia of Statistical Sciences.* Wiley, New York, 1982-1989.

[9] A. Martin, G. Doddington, T. Kamm, and M. Ordowski. The det curve in assessment of detection task performance. In *EuroSpeech*, pages 1895–1898, 1997.

[10] Peter C. McCluskey. Feedforward and recurrent neural networks and genetic programs for stock market and time series forecasting. Master's thesis, Brown University, 1993.

[11] T. Pavlidis and S. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, C-23(8), 1974.

[12] Jay Ponte. *A Language Modeling Approach to Information Retrieval.* PhD thesis, Dept. of Computer Science, University of Massachusetts, Amherst, 1998.

[13] Matthew D. Schmill, Tim Oates, and Paul R. Cohen. Learned models for continuous planning. In *Proceedings of Uncertainty 99: The Seventh International Workshop on Artificial Intelligence and Statistics*, pages 278–282, 1999.

[14] C. J. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33:106–119, 1977.

[15] F. Wallis, H. Jin, S. Sista, and R. Schwartz. Topic detection in broadcast news. In *Proceedings of the DARPA Broadcast News Workshop (HUB4)*, 1999.

[16] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Trans. Information Theory*, 37:1085–1094, 1991.