Language Models for Financial News Recommendation

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. Specifically, we describe the design and implementation of Ænalyst, a system that can recommend interesting news stories – stories that are likely to affect market behavior. Ænalyst operates by correlating the content of news stories with trends in financial time series. 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 and recommend news stories that are highly indicative of future trends. We evaluate the system in terms of its ability to recommend the stories that will affect the behavior of the stock market. We demonstrate that stories recommended by Ænalyst could be used to profitably predict forthcoming trends in stock prices.

Keywords

Application, recommender system, text mining, time series, piecewise linear regression, language models, evaluation.

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.

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

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. Then we can recommend the best stories to the user who is interested in detecting specific trends. For example, if the user is interested in stocks that are likely to go up in price, we would recommend the stories that are most likely to have come from a model of an upward trend.

Our task is a special case of the *Activity Monitoring* task introduced by Fawcett and Provost[7]. The task involves monitoring a stream of *data* and issuing *alarms* that signal *positive activity* in the stream. In our case, the data is represented by news stories and financial time series; unusual trends in the time series signify positive activity; and alarms take the form of recommended stories. This recommendation task is similar to the task of *Information Filtering* [3]. What sets us apart from conventional text filtering is our treatment of relevance. We are not attempting to model the notion of relevance to the user's interest. Instead, we estimate relevance to a trend – a probability that a story will be followed by that trend in the time series. This is a significant departure from the traditional view of relevance: rather than trying to learn what stories the user is interested in, we learn what stories are likely to influence the stock market, and then suggest those stories to the user.

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

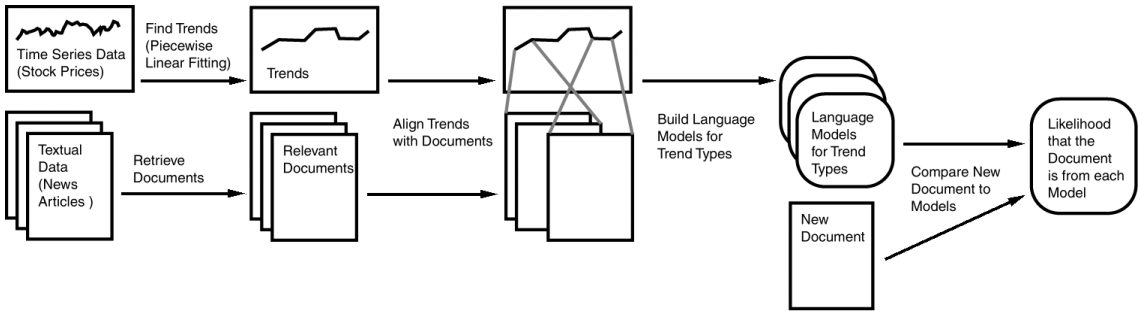

Figure 1: System Design

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. 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 recommendations, we rank the articles by the likelihood that they will be followed by a trend of interest.

Many news articles discuss not one but multiple stocks, sometimes addressing an entire market sector. With such stories it may not be sufficient to just recommend the story as a precursor of a particular trend, we may need to highlight the portion of the story which discusses a company of interest to the user. This goal can be achieved in Ænalyst by learning a model of language typical for a particular company, and then highlighting the passages that most closely match the company's language model.

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 three or more observations, during which the changes from observation to observation are predominantly positive (an *increasing* trend) or negative (a *decreasing* trend). 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 then be classified into those that are 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. A common technique for tackling this problem is *piecewise linear regression*, sometimes called *piecewise segmentation*. Many algorithms for piecewise segmentation were pioneered by Pavlidis and Horowitz [10].

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[2] over the time series, sucessively breaking the window into two halves, and recording the slope of each half. The point that maxmizes the difference in slope is tested to make sure that there is a statistically significant difference between the slopes. The algorithm is recursive and continues until the window can no longer be split.

We regard each segment in the piecewise fit to be a trend. The significance of the trend is defined by its regression statistics: slope and r^2. The slope of the line indicates whether the trend is of interest. Very steep slopes characterize opportunities for maximizing profit by buying and selling when they are found in stock price data, while flatter segments, in general, recommend nothing. Likewise, high r^2 (goodness of fit) values for regression lines indicate strong confidence in the slope, and these trends are more trustworthy as a basis for decision.

2.1.2 Discretizing Trends

The second step to redescribing our time series is to discretize the 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 correlating trends with news stories.

Initally, we implemented a distance based agglomerative clustering algorithm to automatically cluster stock trends based on their slopes and confidence; however, it is not necessary to go to the length of clustering segments. While clustering is the principled approach to discretizing the stocks, the structure present in nearly all the stocks we followed

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

made a simple binning procedure as effective at identifing the most interesting trends. 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 with no recommendation.

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. For instance, if we are interested in Red Hat, we do not want to learn a model from articles about Intel. While stories about Intel can certainly affect Red Hat stock price, this influence has a very complex nature, and would be extremely hard to capture with language models. With a limited amount of training data that we have, using stories about companies other than Red Hat when training Red Hat's trend models will introduce unnecessary noise into the data. 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 the stock's price. In order to learn models that might 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 2 illustrates this.

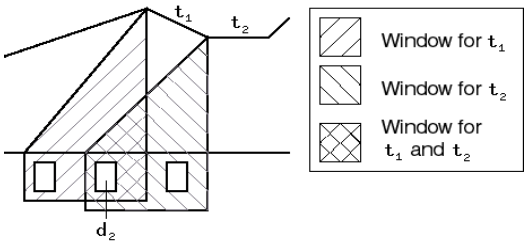

Figure 2: Current Alignment

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 length of 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. 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 looking for correlations between articles and future behavior of the time series to looking for correlations between articles and current behavior of the time series. The recommendations produced by such alignment would not be useful for predicting the future trends, but they may be very valuable in spotting the stories that explain what is happening to the stock price.

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[11, 15] provides a formal framework for text classification with respect to a set of targets (trends in our task). In particular, 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 $\{D_1 ... D_m\}$ was generated by the model M_t. Specifically, for a given set of documents we are interested in the model that is most likely to generate the set of observations $\{D_1 ... D_m\}$:

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

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 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

[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 3.3). 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.

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 and Pazzani [5], 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 backoff, allocates a non-zero probability mass to the terms that do not occur in M_t. We set λ_t to the Witten and 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, Wutrich, and Zhang [4]. Another example is the Activity Monitoring Operating Characteristic (AMOC), suggested by Fawcett and Provost [7]. 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 and Provost [7], but we use a more traditional ROC-style measures in place of AMOC. We carry out two types of evaluation. First, we attempt to evaluate the discriminating power of our language models using the classical classification framework. Then we address the issue of selecting news stories with the goal of making a profit. 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 $M_1 \ldots M_n$. We use Detection Error Tradeoff (DET) curves (see Figure 3) instead of the more traditional Recall Precision curves. The motivating factor is that DET curves are less influenced by "richness" (the a-priori probability of on-target item in the dataset). For a more detailed description of DET curves, see Martin *et al.* [8].

In our analysis we focused on a set of 127 stocks over the period of October 1999 - February 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 of time. 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 in our experiments. 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 4 demonstrates how well the language model identifies news stories that are followed by a surge in the stock price within 10 hours from the story. From the distributions on the left, we see that our language model assigns higher beliefs to the stories that are in fact followed by a surge. A DET curve on the right of Figure 4 allows us to analyze the errors of our system for users with different needs (by looking at different points on the curve). 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 a user would be alerted to 10% of stories that precede a surge in the stock price, while the system would correctly

[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.

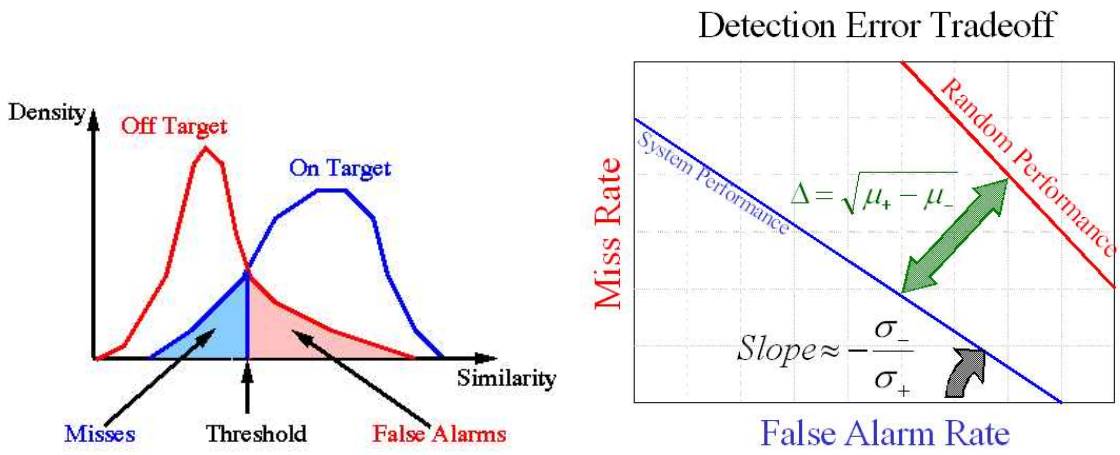

Figure 3: DET curves are a way to visualize tradeoff between misses and false alarms. Left: distributions of on-target and off-target scores, areas under the curve correspond to miss and false alarm rates. Right: corresponding DET curve, obtained by varying the threshold from $-\infty$ to ∞. *NOTE: lower means better.*

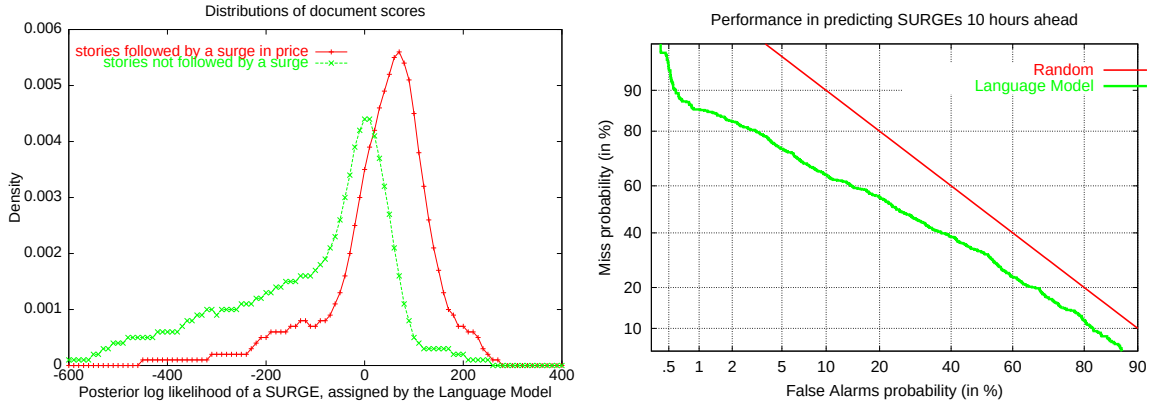

Figure 4: Language model can separate stories that are followed by a surge from stories that are not. Left: belief densities. Right: corresponding DET curve.

eliminate 99.5% of useless stories. This is a very significant reduction in the amount of news which a user would have to read. If not satisfied with low recall, the user 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 Language models vs. classical IR

This section demonstrates that Bayesian language modeling is a good framework for associating news stories with financial trends. To support this claim, we show that our approach significantly outperforms baseline IR approaches that could be applied to the same task. We compare the performance of language modeling to a more traditional vector-space (cosine similarity) approach commonly used in Information Retrieval for text classification [2]. Briefly, a vector space approach compares each document D in the test set to the centroid C_t of the training documents associated with trend t. Both C_t and D are viewed as vectors in vocabulary space, and cosine of the angle between the two vectors constitutes similarity. We used Okapi $tf \cdot idf$ weighting on both vectors. The weighting scheme is described in Robertson *et al.* [12], and has proven very successful in text classification research, e.g., Callan [3].

Figure 5 suggests that the popular vector-space model does not adequately discriminate between the sets of documents associated with different trends. We present DET curves for all trend types: surges, plunges, slight rises and slight falls (recall section 2.1 for definitions of trends). The results are presented for predicting trends up to 5 hours in advance. From figure 5 we can see that a language modeling approach results in noticeably better prediction for all trend types (the false alarm rate is lower at all recall levels). The difference is especially remarkable at low levels of recall: for example at 20% recall, the false alarm is almost 4-10 times lower (2-5% for the language model vs. 20% for the vector space model). A notable exception is predicting plunges (steep downward trends), where both models perform similarly, but language models still outperform cosine by a small margin. We are not certain what is different about plunges that causes slightly worse performance of the language modeling approach, and at the same time allows vector-space model to perform reasonably well. With the

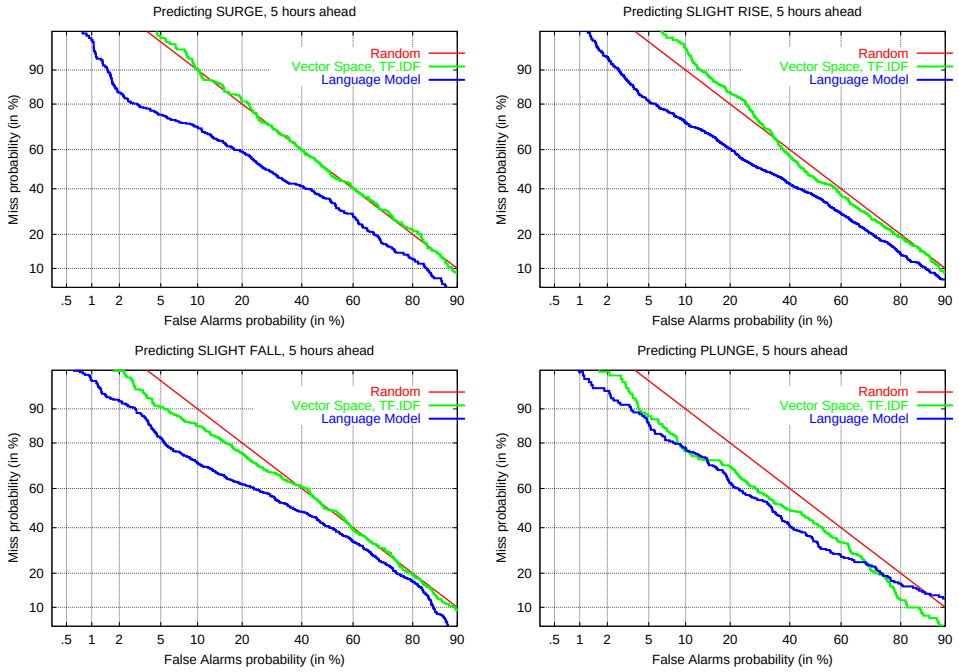

Figure 5: Bayesian language model consistently outperforms baseline text classification approaches on all trend types.

exception of plunges, the errors made by the cosine model are very close to the error tradeoff that would be produced by random ranking. This suggests that a baseline vector-space model does not adequately capture the relationships between news and trends in stock prices. However, we only examined the most popular formulation of the vector-space model. There may exist variations of the vector-space model that exhibit better performance.

3.3 Market Simulation

For the task of financial news recommendation, the ultimate evaluation of performance is whether or not the user would be able to make a profit by reading the stories suggested by the system and making informed trading decisions. In the following simulation we attempt to mimic the behavior of a day trader who would use the stories suggested by Ænalyst in a very simple fashion: if the system indicates that a story is likely to precede an upward trend for the stock, the user will invest in that stock; if the predicted trend is downward, the user 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 user 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 user will hold the stock for 1 hour. If during that hour she can sell the stock to make a profit of 1% ($100) or more, she sells immediately. At the end of the hour, the user sells the stock at the current market price, and takes 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, the user 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, the user will hold the stock for 1 hour. If during the hour the user can buy the stock at a price 1% lower than she *shorted*, she buys the stock to cover. At the end of the hour, the user buys the stock at the current market price, and takes a loss if necessary.

This model represents an extremely simplistic and greedy day-trading strategy. We expect real users to make much better decisions based on the semantic content of the recommended stories, as well as any other information they may have access to. So this simple simulation provides an absolute lower bound on how well an experienced trader might perform if she had access to the stories recommended by Ænalyst.

After the 40-day simulation, the cumulative earnings of our user, pulled over all stocks, totaled $280,000. This is a very modest gain, considering the number of trades the user would have to execute (average gain per transaction was around 0.23% or $23). We used a randomization test [6] to determine if these earnings are statistically significant.

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

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 a buying and shorting particular stocks were made at the same times as in the actual system (when a story dealing with that stock was retrieved). 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 $280,000 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.

It is worth noting that alignment of stories with trends has a very strong effect on performance in the market simulation. We obtained best results when aligning trends with news that are released during the duration of the trend (i.e. simultaneous alignment). Aligning trends with documents that precede the trend by 1, 5 or 10 hours resulted in much lower average gains per transaction: 0.03%, 0.02% and 0.16% respectively.

The simulation we presented uses a very simple strategy. Actual users should be able to perform much better when presented with stories recommended by Ænalyst. 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.

3.4 Specific or universal models?

An 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 and Provost[7] 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 simulation in Section 3.3 and 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.

Experimentally we observed that stock-specific models indeed achieve higher profits, but at the cost of much higher variance of cumulative profit from company to company. Universal models are more stable, but give lower expected profits in the market simulation. For example, a universal model with simultaneous alignment achieved 0.15% average gain per transaction.

This observation leads us to consider mixture models when performing recommendations. Specifically, we would use stock-specific models as the primary trend model, and smooth it by the universal model for that trend. Recall that in Section 2.3 we used a smooth estimate:

$$P(w|M_t) = \lambda_t P_{ml}(w|M_t) + (1-\lambda_t)P(w|GE)$$

Here we smoothed the trend model with the background model of General English. Instead, we would use the universal trend model for smoothing:

$$P(w|M_t) = \lambda_t P_{ml}(w|M_t^{stock}) + (1-\lambda_t)P(w|M_t^{univ})$$

4. RELATED WORK

Other researchers have used data mining techniques to predict the stock market. Cho, Wutrich, and Zhang [4] at the Hong-Kong Institute of technology implemented a system that predicts the closing price of Hang Seng stock index, based on a fixed set of news sources. From these news stories, they looked for the occurrence of 400 keywords that were provided by market experts. Fawcett and Provost [7] developed a system that used news stories to predict when a stock would shift at least 10% by interpreting the problem as an *activity monitoring* problem. McCluskey [9] and numerous other works attempt to predict the stock behavior by relying purely on the analysis of time series data.

Our system is different in that rather than predicting the market, it suggests what stories the trader should read – the stories that are most likely to signal an upcoming trend. 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 [4] and [7], who focus on raw time series. Furthermore, by using language models, our system can incorporate the entire vocabulary used in the text. This elevates the use of human judgments and allows us to find associations that are only meaningful to a particular company's stock performance. The Ænalyst draws on progress made in several areas: text classification [1], language models [11], and time series analysis and clustering [13].

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. Finally, in our market simulation, we showed that using Ænalyst's story recommendations would allow a trader to do significantly better than random in terms of cumulative profit.

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.

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 Eighth Text REtrieval Conference (TREC-8)*, 1999.

[2] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *Proceedings on the 18th annual international ACM SIGIR conference*, pages 351–357, 1995.

[3] J. P. Callan. Document filtering with inference networks. In *Proceedings on the 19th annual international ACM SIGIR conference*, pages 262–269, 1996.

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

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

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

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

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

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

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

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

[12] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. OKAPI at TREC-3. In D. Harman, editor, *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, 1996.

[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.