Discovering Patterns on Financial Data Streams

WU, Di

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

in

Systems Engineering and Engineering Management

The Chinese University of Hong Kong

September 2010

Thesis/Assessment Committee

ABSTRACT

With the increasing amount of data in financial market, there are two types of data streams attracting a lot of research and studies, time series index stream and related news stream. In this thesis, we focus on discovering patterns from these data streams and try to answer the following challenging questions, I) given two co-evolving time series indices, what is the co-movement dependency between them. II) given a set of evolving time series, could we detect some leaders from them whose rise or fall impacts the behavior of many other time series? III) could we integrate the news stream information into stock price prediction? IV) could we integrate the news stream information into stock risk analysis? and V) could we detect what are those events that trigger time series index movement. For each of the question, we design algorithms and address three technique issues I) how to detect promising patterns from the noisy financial data II) how to update the old patterns when new data arrives in high frequency. III) how to use the pattern to support the financial applications.

We start from investigating the co-movement relationship of multiple time series. We propose techniques to study two aspects of this problem. First, we propose a co-movement model for constructing financial portfolio by analyzing and mining the co-movement patterns among two time series. Second, we presents an efficient streaming algorithm to discover leaders from multiple time series stream. Both of the algorithms are evaluated using real time series indices data and the result proves that co-movement patterns and detected leaders are promising and can support various applications including portfolio management, high frequency trading and risk management.

Then, we consider the patterns between news stream and time series indices stream. We first transform the news stream into a set of bursty feature (keywords) time series streams and propose three technique to study their relationship to time series index. First, we explore a Non-homogeneous Hidden Markov Model (NHMM) to predict the stock market process which takes both stock prices and news articles into consideration. Second, we propose a risk analytical model to predict the volatility of price indices by integrating news information. Finally, we devise an algorithm to detect the priming event from text and a time series index. The evaluation on real world dataset suggests the significant correlation exists between news stream and time series stream and our pattern discover algorithm can detect promising patterns from this relationship to support real world applications effectively.

摘要

当今金融市场，信息与日俱增。在这些海量信息中，有两种类型的数据流受到研究界最多的关注，他们是时间序列指数流以及相关的新闻数据流。在这篇论文里，我们专注于从这些数据流中挖掘出有用的模式从而来回答以下挑战性的问题。I) 给定两条数据流，在共同移动中，他们之间到底有什么相互影响的关系? II)给定一系列数据流，我们是否可以从他们中找到领头的数据流，该数据流的上升下降会影响很多其他数据流的移动。III) 我们能否把外部的新闻信息整合到对股票价格趋势的预测中去? IV) 进一步，我们能否把新闻信息整合到对股票价格波幅的预测中去? V) 我们能否自动发现和组织那些对时间序列流移动产生重大影响的事件。我们为每一个具体问题设计算法并试图解决以下三个技术难点，I) 如何从充满扰乱的金融数据中找到那些有用的模式? II)如何在新的数据到来的时候尽快更新这些模式? III)如何利用这些模式去解决金融应用中的实际问题。

我们从调查多条数据流的相互移动影响关系开始。我们提出了技术去解决两个方面的问题。第一，我们通过分析两条时间序列的相互移动关系提出了同移模型来构建金融投资组合，第二，我们提出了一种有效的流算法从多条时间序列中找到领头的时间序列。我们用真实的时间序列数据测试了这两种方法。实验证明，我们的同移模式和领头时间序列能够很好的支持包括投资组合管理，高频交易以及风险分析在内的多种金融应用。

另外，我们分析新闻事件流和时间序列流之间的关联模式。我们首先把新闻转化为一系列爆发关键字流，然后提出了三种方法来分析这些爆发关键字流同时间序列指数流之间的关系。首先，我们使用异质的马尔可夫模型来整合新闻和股价信息，从而对股票市场的变化过程进行建模。通过这种方法

我们可以预测股价的未来走势。其次，我们提出了能够整合新闻信息的风险分析模型，该模型可以自动找到与股价波幅相关的新闻信息从而来判断股价波幅的变化。 最后，我们设计了一套方法用来从新闻流中找到对时间序列指数产生重大影响的事件。 我们在真实数据中测试了上述方法，证明新闻事件流跟时间序列指数流间存在着明显的关系，而我们的模式挖掘方法能够有效的找出这些关系， 并能够很好的支持实际金融应用。

ACKNOWLEDGEMENTS

I am extremely grateful to my adviser, Professor Yu. From my first day of Ph.D life, Professor Yu required me to think deep and tackle the most challenging problem with the most precise solution. This is not easy. But Professor Yu guided me through with great patience and insight. He encouraged me never to hide from a problem but should face and solve it directly. It has been my fortune to meet such a great supervisor who not only enlightened me on how to do great research but also inspired me on how to live with difficulties in the life. I will apply this attitude and philosophy throughout my future career.

I would also like to thank all my committee members, Lam Wai, Lam Kai Pui, and Longbing Cao. I appreciate your suggestions throughout my Ph.D study.

I thank all my collaborators. Philip S. Yu and Lei Chen who gave me a lot of great suggestions in my research, Gabriel Fung who guided me to write the first research paper, Yiping Ke who required me to pay attention to every details, Zheng Liu who inspired me with new ideas whenever I discussed with him and Qi Pan who not only assisted me on research but also helped me on global business plan challenge. Without their invaluable support, my journey would not have been completed.

Thanks to all the people who shared my life in CUHK: Lu Qin, Bo Chen, Wenting Hou, Lijun Chang, Binyang Li, Lidong Bing, Fan Yang, Miao Qiao, Yuanyuan Zhu, Yang Zhou, Feng Zhang, Bolin Ding, Hong Cheng, Nan Tang, Jiefeng Cheng, Hong Pan. I remember all the joy and pain in every day's activities, either in the lunch time or in the fitness room. Finally, my special gratitude to the Department of Systems Enginnering and Engineering Management at the Chinese University of Hong Kong. I feel extremely fortunate to have studied at such a distinguished place.

Most of all, I am grateful to my parents, Jian Wu and Huajia Mei, without whose love and support none of this would be possible.

CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

With the increasing amount of data in financial market, there are two types of data widely observed by the investors and government officers. I) Price Index Stream. In the stock market, at each time tick, every equity has a quoted price for trade. And the buy and sell actions from investors will alter this price from time to time and generate an evolving price time series stream. II) News Stream. The buy/sell decision of investors are largely influenced by the external information related to the market, Most of external information is interpreted in the form of market news. As soon as an event emerges, the reporter will write a story and publish it as a news article. Therefore, the news stream is characterized by the arrival of the news articles. However, the influence of news articles over the market largely depends on its content. For example, a good news about company earnings is likely to drive the price index up. While a news report about company's regular operation may have little impact on the price movement.

Here, we use figure 1.1 to illustrate these two data streams, as well as the interaction between them. In this figure, there are four time series index streams and a stream of news articles. Each of the stream is evolving over time. And we can see there are relationship between a pair of price indices, i.e., the link between $T1$ and $T2$. On the other hand, a time series indices is also related to some news articles. For example, news articles $N1$ and $N2$ are related to price

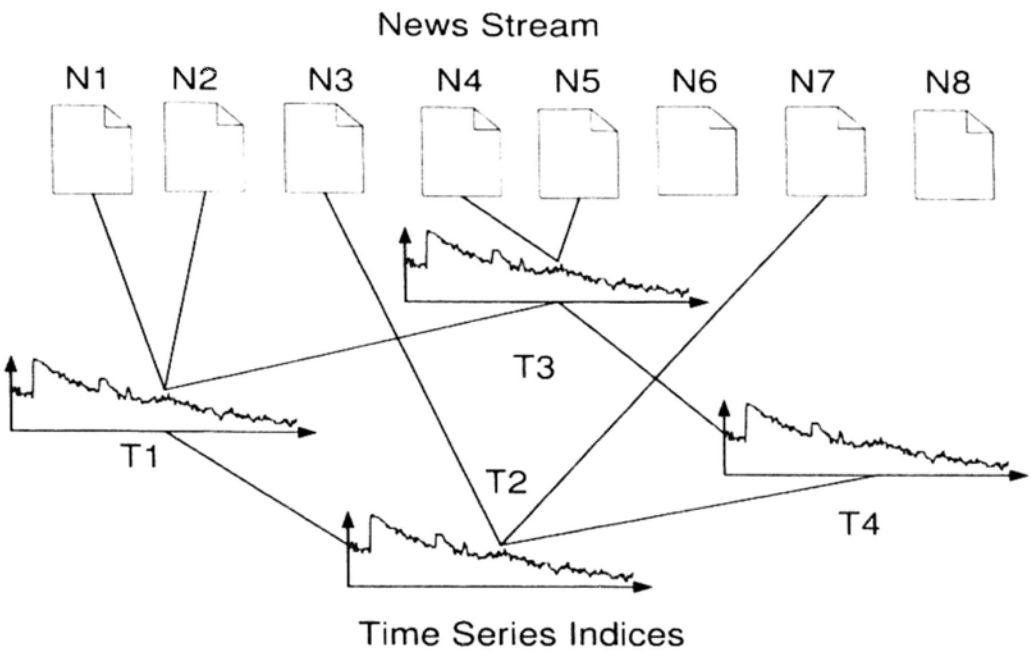


Figure 1.1: Financial Data Streams

index $T1$, news articles $N3$ and $N7$ are related to price index $T2$.

Based on the above data modeling, some pattern discovery questions arise naturally. First, from the co-movement of time series indices,

- What is the co-movement dependency between two price indices [79]?
- Given a set of evolving time series, could we detect some leaders from them whose rise or fall impacts the behavior of many other time series [82, 81]?

Second, from the evolving time series index and news streams,

- Could we utilize the impact of news articles to predict the trend of the price index [78, 80]?
- Could we utilize the impact of news articles to predict the volatility of the price index? Moreover, can the risk generated from news stream transfer from one price index to another if they are highly correlated [63]?
- Could we detect and organize these news events which trigger the movement of time series index?

These questions are exactly what we want to answer in this thesis. And for each of them, we want to address the following technique problems: I) How to detect promising patterns from the noisy financial data II) How to update the old patterns when new data arrives in high frequency. III) How to use the pattern to support the financial applications. We will formally define these problems and discuss the challenges in next section.

1.1. Problems and Challenges

In this section, we will introduce following patterns discovery problems which are challenging and have many practical financial applications.

1.1.1. Co-movement of Two Time Series

Many financial applications can converge to the problem of analyzing the co-movement of two time series. For example, in the modern financial engineering discipline, while the underlying principle of governing how the portfolios should be constructed is simple and intuitive – minimize the expected risk and maximize the expected return, the definition of expected risk and the definition of expected return are usually ambiguous and questionable.

For instance, given two financial assets, A and B, where we know that whenever the price of A drops, the price of B will drop, and vice versa. Intuitively, it may not be appropriate to construct a portfolio by including both A and B concurrently, as the exposure of loss will be increased. Yet, such kind of relationship can not always be captured by co-variance(i.e traditional statistics).

In fact, this problem is exactly what lead to the major differences among different portfolio management theories. Little consensus exists among different theories regarding to their meanings and their measurements [56, 4, 51, 31, 41].

Therefore, our task is to design a new co-movement model to capture the dependency relationship between two time series indices and measure the expected return and risk of the portfolio.

1.1.2. Leaders of Multiple Time Series

In the literature, the lagged correlation between two streams has been well studied in empirical research [15, 5, 68] and efficient algorithms to discover lagged correlations have also been developed [69]. However, the study on summarizing the relationships across multiple data streams is still lacking. The comprehensive relationships among multiple data streams are very helpful in financial applications to monitor and control the overall movement of the market where the price indices streams are generated.

For example, the stock market can be modeled as a financial network, in which each stream represents the price of a stock. The lead-lag effect between two streams implies that the price change of one stock influences that of another [68]. In finance crisis, when the market goes down dramatically and the government plans to launch finance bailout, the regulators desire to know the subset of stocks which poses risks (influences) on others and triggers the movement of the whole market. They can then apply a program to these market leaders and control the overall systemic risk.

However, the problem of finding the leaders among multiple time series poses great challenges. First, the observations of stock price usually change rapidly over time, which implies that the leaderships among them may also change from time to time. Therefore, the lagged correlations between pairs of time series, which are used for leadership identification, must be re-computed for every new time tick, while the correlation computation at each time tick is already costly. This high computational complexity makes the design of an efficient solution difficult. Second, after computing the lagged correlation between each pair of price indices, how to define and extract useful leaders out of the whole set of time series is also a big challenge.

Therefore, our task is to design an algorithm to detect the leaders among the multiple correlated price indices effectively and efficiently.

1.1.3. Integrating News Stream for Stock Prediction

In many real world applications, decisions are usually made by collecting and judging information from multiple different data sources. Let us take the stock market as an example. We never make our decision based on just one single piece of advice, but always rely on a collection of information, such as the stock price movements, exchange volumes, market index, as well as the information from the news articles, expert comments and special announcements (e.g., the increase of stamp duty).

Yet, modeling the stock market is difficult because: (1) The process related to market states (up and down) is a stochastic process, which is hard to capture by using the deterministic approach; and (2) The market state is invisible but will be influenced by the visible market information, such as stock prices and news articles.

Therefore, our task is to develop a new model to integrate both news and price information into the stock market modeling and predict stock price movement.

1.1.4. Integrating News Stream for Risk Prediction

Modern risk management system has been strongly criticized in the recent financial tsunami, where the numerous different arguments converge to one single theme: the current system failed to accurately estimate the risks of financial instruments, which were considered *to be isolated, but in many cases seemingly challenge human understanding* [28]. In stock markets, risk means the uncertainty of future outcomes, and is the probability that an investment's actual return is different from the expected value. Volatility, which measures the variation or dispersion or deviation of an asset's returns from the mean value, is used to quantify risk over a time period. The risk of a financial instrument is commonly estimated by the stock price volatility, which measures the variation or dispersion or deviation of an asset's returns from the mean value.

Several models (e.g., ARMA [27], GARCH [9]) were proposed to predict the future volatility based on historical stock price (time series information). However, these models cannot fully capture the bursty behavior of stock prices, especially when there is some breaking news hitting the market.

Several studies [67, 27, 17] have discussed the GARCH forecast errors and related the errors to the arrival of asset specific news articles, i.e., the existing models cannot interpret the change of external environment (news) and therefore could not react accordingly. In [67], a classification model is designed to detect interesting news articles that would help understand the behavior of stock price volatility. However, except for some empirical studies, none of these methods attempted to incorporate news information into risk analysis, or in other words, volatility prediction and ranking.

Therefore, our task is to explore the possibility to predict and rank the risk of stocks by news.

1.1.5. Priming News Event Detection from Time Series

There are a lot of events related to the company or the stock market happening while only some of them would change the valuation of the company in investors' mind. The greedy/fear will drive them to buy/sell the stock and eventually change the stock index. Therefore, the investors will be eager to know and monitor these events as they happen.

This case indicates that there is a need to find out the priming events which drive an interested time series index. Such priming events are able to help people understand what is going on in the world. However, the discovery of such priming events poses great challenges: 1) with the tremendous number of news articles over time, how could we identify and organize the events related to a time series index; 2) several related events may emerge together at the same time, how can we distinguish their impact and discover the priming ones; 3) as time goes by, how could we track the life cycle of the priming events, as well as their impact on the evolving time series index.

Some existing work has focused on discovering a set of topics/events from the text corpus [2, 25] and tracking their life cycle [52]. But these methods make no effort to guarantee these events are influential and related to the index that people are concerned with. There is another stream of work considering the relationship between the keyword trajectory and interested time series [30, 78, 63]. However, these work can only identify a list of influential keywords for users and do not consider to organize these words into some high level meaningful temporal patterns (i.e., topics, events).

Here, our task is to address these challenges and design an algorithm to detect priming events from a time series index and an evolving news stream.

1.2. Contributions and Outlines

In this thesis, we present a series of tools to discover patterns from financial time series streams to enlighten the real world applications.

In Chapter 2, we present two techniques to analyze the co-movement of multiple time series streams. Section 2.2 presents a co-movement model for constructing financial portfolio by analyzing and mining the co-movement patterns among multiple time series. Different from traditional statistical approach in financial world, which takes the co-variance among the portfolio as risk and the summation of individual expect return as portfolio return, our approach models the risk from the co-movement patterns and computes portfolio returns by considering all dependency relationships among assets. Section 2.3 presents an efficient streaming algorithm to discover leaders from multiple time series stream. The main contributions are I) we formalize a new problem of discovering the leadership among multiple time series, which well captures the overall co-movements of time series. II) We devise an efficient solution that discovers the leaders in a real-time manner. Our solution utilizes an effective update strategy, which significantly reduces the computational complexity in a stream environment. III) we justify the efficiency of our solution, the effectiveness of our update strategy, as well as the usefulness of the discovered leaders by conducting extensive experiments over the real financial data.

Chapter 3 presents three different techniques to analyze the relationship between time series streams and news stream. Section 3.2 presents an approach to model the stock market process by using a Non-homogeneous Hidden Markov Model (NHMM). It takes both stock prices and news articles into consideration when it is being computed. A unique feature of our approach is event-driven. We identify associated events for a specific stock using a set of bursty features(keywords) with significant impact on the stock price changes when building the NHMM. We apply the model to predict the trend of future stock price and the encouraging results indicated that our proposed approach is practically sound and highly effective. Section 3.3 presents a new problem of predicting stock risks based on the predicted volatility by utilizing both time series information (stock price) and textual information (news articles). First, a new feature selection algorithm is proposed to select bursty volatility features which have co-occurring bursty patterns with the volatility bursts of stocks. A set of such selected bursty volatility features can accurately represent the stock volatility. Feature weights are learned from historical stock prices and news articles to measure the impact of bursty keywords on stock volatility. We further use random walk to propagate the impacts of news among stocks based on their correlation. The random walk approach can greatly improve the volatility prediction performance for those stocks with very limited news reports. The volatility prediction and ranking methods are built on top of the random walk model. Finally, we conducted extensive experimental studies using real datasets and demonstrated the superiority of our approach in comparison with existing approaches. Section 3.3 formulates the problem of detecting priming events from a text stream and a time series index. A priming event is an object which consists of three components: (1) Two timestamps to denote the beginning and ending of the event; (2) A sequence of local influential topic groups identified from the text stream; (3) a score representing its influence on the index. We design an algorithm that first discovers the influential topics at a global level and then drills down to local time periods to detect and organize the priming events based on the influential topics. We evaluate the algorithm on a real world dataset and the result shows that our method can discover the priming events effectively.

In summary, this thesis concerns two aspects of the pattern discovery from financial time series streams: First, we analyze the co-movement of multiple time series indices. Second, we exploit different techniques to analyze the relationship between time series stream and news stream. In particular, we apply the discovered patterns to practical financial application, including portfolio management, market events/trend detection, stock price trend prediction and risk analysis.

CHAPTER 2

Pattern Discovery on Multiple Time Series

Co-movement of multiple time series has received tremendous attention in financial engineering, due to many important applications:

- **Portfolio Management** The classical portfolio management model, mean-variance model requires to compute the expected risk of each pair of assets. Therefore, it is a big challenge to analyze the co-movement of two time series and estimate its corresponding risk.

- **High Frequency Trading** In the stock market, a lot of trading institutions look for arbitrage opportunity by analyzing the co-movement of time series stream. For example, given two stocks, A and B, where we know there are patterns that whenever the price of A drops, the price of B will drop, and vice versa. Then it is a good pair trading for traders, i.e., when A drops and B has not dropped yet, he can short B and long A by assuming this two index will eventually move together. Besides, if we are able to detect leaders of the market, then the price movement of stock leaders implicates the future market trend and thus has prediction power of market index (Dow Jones Index, Standard Poor 500, etc.). In other words, the market trend can be detected in an early stage. For example, traders can buy the call option of the market index when detecting stock leaders begin a rising trend. On the other hand, index arbitrage can be implemented by trading these stock leaders, the sample of which closely mirrors the market index

- **Empirical Finance** Co-movement patterns can also provide evidence and ideas for financial empirical research. For example, The stock leaders we exploit from the correlated time series graph provide the empirical finance researchers, financial analysts and investors with insightful understanding and tracking of the market behavior. These people are always curious about the cause of market movement and try to predict and monitor the market movement. Stock leaders present a new perspective to analyze the stock market: a revealed news event introduces some changes to stock leaders' prices, whose effect then propagates to related stocks by lagged correlations. As a result, analysts only need to monitor and analyze stock leaders in order to evaluate the whole market.

In this section, we first present some background and related work in Section 2.1. We then present a co-movement model to capture the dependency relationship of two time series and explore this model for portfolio management in Section 2.2. After that, we present our algorithm to detect the leaders from multiple time series stream in Section 2.3. We summarize this chapter in Section 2.4.

2.1. Multiple Time Series Co-movement Related work

In this section, we present some related work on co-movement patterns of the time series stream and corresponding applications in portfolio management.

2.1.1. Segmentation of Time Series

When we process time series for co-movement study, we often explore segmentation algorithm to discretize the time series [42, 54, 66, 26]. Keogh [42] gives a good survey of segmentation algorithm on time series, where three segmentation approaches are compared, the sliding window, the bottom-up and the top-down algorithm. From the experiment results of [42], the bottom-up algorithm, which starts from the finest possible approximation and merges segments until some criteria is met, can achieve best performance on the selected financial dataset.

2.1.2. Portfolio Management

Most of the modern portfolio theory is motivated by the mean-variance model [56], which is also know as Markowitz model. It uses the minimum variance strategy to select the portfolio which can be summarized as below:

$$\min \sum_{i,j=1}^{n} w_i w_j \sigma_{ij} \tag{2.1}$$

$$\text{subject to: } \sum_{i,j=1}^{n} w_i r_i = R \tag{2.2}$$

$$\sum_{i,j=1}^{n} w_i = 1 \tag{2.3}$$

where σ_{ij} is the covariance of the asset i and the asset j, w_i is the weight of asset i in this portfolio, R is the desired rate of return, and r_i is the expected return of asset i. Hence, if r_i and σ_{ij} are identified and R is specified, we can identify the weight, w_i, of each asset in this portfolio. Intuitively, given a specified return, a portfolio with the minimum risk can always be formulated. It is obvious that the covariance of two asset i and j is taken as the measure of risk for the allocation of the two assets.

Yet, this mean-variance model perceives the expected risk is a combination of and is equal for both the *down-side* (the possibility of earning *less* than the desired level of return) and *up-side* (the possibility of earning *more* than the desired level of return). Obviously, this cannot mimic the real-life situations in the financial markets, where the investors always concern with the down-side risk only rather than the up-side risk. Accordingly, many different theories are proposed which are targeting for modeling the expected risk as down-side risk only [31, 53, 51, 50]. Nevertheless, their concepts for modeling the down-side risk are the same - the variance of the portfolio's expected return below a specified desired level of return.

In respect to downside risk, even the pioneering work by Markowitz acknowledged the relevance of risk associated with failure to achieve a target return. Harlow and Rao [31] discussed asset allocation to minimize portfolio downside risk for any given level of expected return. The downside risk approach is more attractive than traditional mean-variance approach because of its consistency with the observation that investors are averse to downside results but not upside variability [19]. In this regards, Harvey et.al [18] indicated that the usual measure of correlation represents average co-movement in both up and down markets. Separate correlation estimates in different return environments would permit detection of whether correlation increases or decreases in down markets. Increased correlation in down markets reduces the benefit of portfolio diversification. However, different from the definition in our model, they define the down(up) market as the time with return below(upon) average return and compute the correlation in the same way as semi-variance measure [53].

2.1.3. Lagged Correlation

When we study the leadership of time series, we consider a set of N synchronized time series streams $\{S^1, S^2, \ldots, S^N\}$, where each time series $S^j = (s_1^j, \ldots, s_t^j)$ is a sequence of discrete observations over time, and s_t^j represents the value of time series S^j at the most recent time point t.

Given a sliding window of length w and a time point t, the sliding window for time series S^j, denoted as $s_{t,w}^j$, is the subsequence $(s_{t-w+1}^j, \ldots, s_t^j)$. Below, we discuss lagged correlation between two sliding windows of two time series streams and the transition probability in a Markov chain which play an important role in this work.

The lagged correlation between two sliding windows $s^i_{t,w}$ and $s^j_{t,w}$ of two time series S^i and S^j at lag l, denoted as $\rho^{ij}_{t,w}(l)$, is computed by considering the common parts of the shifted sequences.

$$\rho^{ij}_{t,w}(l) = \begin{cases} \frac{\sum_{\tau=t-w+1}^{t-l}(s^i_{\tau+l}-\overline{s^i_{t,w-l}})(s^j_{\tau}-\overline{s^j_{t-l,w-l}})}{\sigma^i_{t,w-l}\sigma^j_{t-l,w-l}}, & l \geq 0; \\ \rho^{ji}_{t,w}(-l), & l < 0, \end{cases} \tag{2.4}$$

where $\overline{s^i_{t,w-l}}$ and $\overline{s^j_{t-l,w-l}}$ are the mean values in the shifted sliding windows $s^i_{t,w-l}$ and $s^j_{t-l,w-l}$, respectively, and $\sigma^i_{t,w-l}$ and $\sigma^j_{t-l,w-l}$ are the standard deviations in $s^i_{t,w-l}$ and $s^j_{t-l,w-l}$, respectively. Here, $\rho^{ij}_{t,w}(0)$ is the correlation with zero lag (known as the local Pearson's correlation [64]). When $l > 0$, $\rho^{ij}_{t,w}(l)$ denotes the correlation between the sliding windows, $s^i_{t,w}$ and $s^j_{t,w}$, when S^i is delayed by a lag l. Therefore, $\rho^{ij}_{t,w}(l)$ is essentially the correlation between the common parts of the shifted windows, $s^i_{t,w-l}$ and $s^j_{t-l,w-l}$, with zero lag. The case when $l < 0$ can be easily handled symmetrically. Since $\rho^{ij}_{t,w}(l)$ is computed on the common parts of the two windows, l is less than the window length w, and in practice $|l| \leq w/2$ as suggested in [10].

In a stream context, it is not desirable to compute $\rho^{ij}_{t,w}(l)$ from scratch at each time point t. As shown in [84, 69], the lagged correlation can be computed efficiently by tracking some statistics as follows.

$$\rho^{ij}_{t,w}(l) = \frac{\psi^{ij}_{t,w}(l) - \frac{\sum(s^i_{t,w-l})\cdot\sum(s^j_{t-l,w-l})}{w-l}}{\sigma^i_{t,w-l}\sigma^j_{t-l,w-l}}, \tag{2.5}$$

where $\psi^{ij}_{t,w}(l)$ is the inner product between the shifted windows $s^i_{t,w-l}$ and $s^j_{t-l,w-l}$, $\sum(s^i_{t,w-l})$ and $\sum(s^j_{t-l,w-l})$ are the sum over the two shifted windows, respectively, and $\sigma^i_{t,w-l}$ can be computed as follows.

$$\sigma^i_{t,w-l} = \sqrt{\sum(s^i_{t,w-l})^2 - \frac{(\sum(s^i_{t,w-l}))^2}{w-l}}, \tag{2.6}$$

where $\sum(s^i_{t,w-l})^2$ denotes the sum of the squares of the shifted window $s^i_{t,w-l}$. The value of $\sigma^j_{t-l,w-l}$ can be computed similarly. It implies that as long as the inner product, the sum of squares and the sum of the shifted windows are kept track of, the correlation value at each lag can be computed quickly.

Finally, we introduce some empirical study on lead-lag effect in finance domain. One of the theoretical analysis was based on the study of informativeness [68]. If the price of a security is informative for prices of other securities, its return will lead those of other securities. The inside mechanism is that trading reveals information that causes price revisions of securities with correlated underline values or information. On the other hand, more empirical work [11] concentrated on the speed of price adjustment. In this mechanism, a security is said to lead other securities if its price adjustment to a common factor is earlier than that of other securities. The leaders detected by our approach is independent of these two mechanisms and can provide an insight analysis for the source of the lead-lag effect across the graph $\mathcal{G}$.

2.1.4. Matrix Analysis

In a Markov chain, the transition probability matrix $H = \{h_{ij}\}$ describes the state transition property, where h_{ij} is the probability of being in state j at the next step given that the chain is in state i at the current step. The stationary probability distribution vector for a Markov chain with H is a probability vector $\tilde{\pi}$ such that $\tilde{\pi}H = \tilde{\pi}$. The Markov chain defined by H has a unique stationary probability distribution if H is aperiodic and irreducible (primitive) [3]. Besides, the k-th step probability distribution vector for a chain with N states is defined to be $\pi^k = \{\pi^k(1), \ldots, \pi^k(N)\}$, where $\pi^k(j)$ is the probability of being in state j at the k-th step. Given π^k, the classical power method can be applied to compute π^{k+1} as follows:

$$\pi^{k+1} = \pi^k H. \tag{2.7}$$

For any starting vector, as long as the transition matrix H is stochastic and primitive, the power method applied to H converges to a unique stationary probability distribution vector $\tilde{\pi}$.

However, the stochastic and primitivity properties are often violated in large graphs, for example, WWW. Therefore, to measure the importance of Web pages in a large WWW graph, Brin and Page applied the stochasticity adjustment and primitivity adjustment to convert the original induced matrix H to the irreducible Markov matrix G as follows.

$$G = \alpha H + (1/n)(\alpha a + (1-\alpha)e)e^T, \tag{2.8}$$

where n is the number of pages (i.e., states), e is a size-n vector with all ones, a is a size-n vector with $a_i = 1$ if page i is a dangling node (i.e., a node with zero out-degree), and α controls the proportion of time that the random teleportation follows the hyperlinks as opposed to teleporting to a random new page (α is set to be 0.85 in [13]).

Recently, there are also a stream of work that constructs the weighted graph by connecting two time series based on their correlation and attempt to detect abnormal behavior among multiple time series. Tsuyoshi [38] computed the anomaly score of a node based on analyzing its k-neighborhood time series. [37] reported an anomaly detection method which was based on analyzing the eigenspace of the dependency matrix.

2.2. Mining Co-movements of Multiple Time Series

In this section, we will present our co-movement model to compute the expected return and risk of the portfolio. Section 2.2.1 will define four different kinds of co-movements between two time series, whereas Section 2.2.2 and Section 2.2.3 will respectively present how the expected return and expected risk are computed based on the idea of co-movements. Section 2.2.4 will show how a portfolio will eventually be constructed by applying these novel measures and concepts. Section 2.2.5 evaluates the effectiveness of our proposed model and reports our

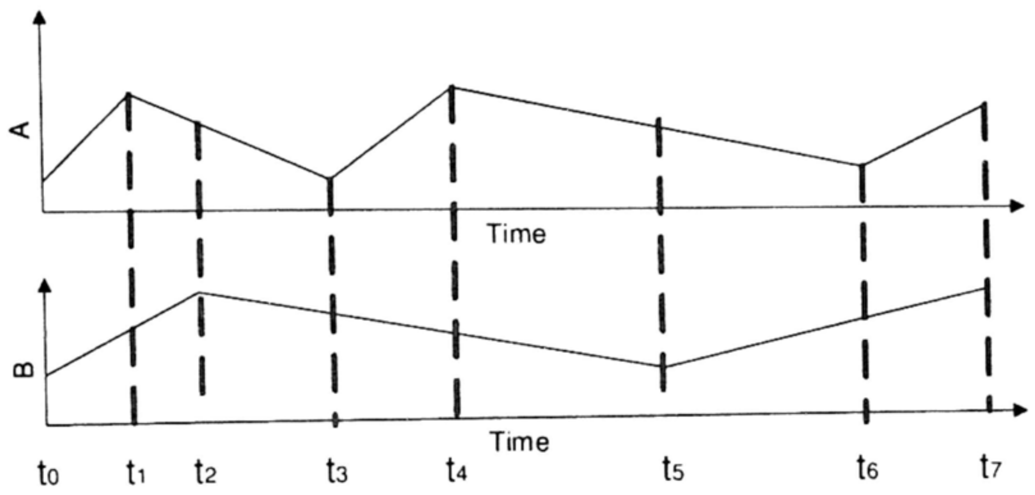


Figure 2.1: The definition of co-movement. In the figure, there are two financial assets: A and B. With respect to A, there are four types of co-movements: (1) up-up ($t_0 - t_1$, $t_6 - t_7$); (2) up-down ($t_3 - t_4$); (3) down-up ($t_1 - t_2$, $t_5 - t_6$); (4) down-down ($t_2 - t_3$, $t_4 - t_5$)

findings. Finally, Section 2.2.6 summarizes this section and discusses its possible extensions.

2.2.1. Four Different Types of Co-movements

In order to account for different types of co-movements, let us first refer to figure 2.1. In Figure 2.1, it shows two time series: A and B. For time series A, there are totally five segments, whereas for time series B, there are totally three segments. With respect to time series A, four different types of co-movements can be identified: (1) up-up; (2) up-down; (3) down-up; and (4) down-down. Let us explain this in the next paragraph.

In figure 2.1, the co-movement between i and j forms a total number of seven co-movement patterns (1) Two up-up co-movements, from t_0 to t_1 and t_6 to t_7; (2) Two down-up co-movements, from t_1 to t_2 and t_5 to t_6; (3) Two down-down co-movements, which has the longest cumulative duration, from t_2 to t_3 and t_4 to t_5; and (4) one up-down co-movements, from t_3 to t_4.

2.2.2. Co-movement Based Expected Return

As discussed previously, the existing literatures compute the portfolio's expected return when they will ignore the dependence relationships among assets in the portfolio. Its computation is based on a linear combination of the multiplication between the expected return of each individual asset and its contribution in the portfolio. In this work, we try to incorporate the dependence relationships among assets by using the concept of co-movement, which has been defined in the previous section. Given two financial assets, i and j, conceptually, our co-movement based expected return is as follows:

1. If whenever the price of i rises(drops), the price of j will rise(drop) (i.e. a lot of up-up (down-down) co-movements), then their co-movement based expected return should be higher(lower) than the sum of their independent expected return.

2. If whenever the price of i rises(drops), the price of j will drop(rise) (i.e. a lot of up-down(down-up) co-movements), then their co-movement based expected return should a mixture of their independent expect return considering their contribution to portfolio.

As a result, there are four different kinds of returns, which in-turn depends on the four different kinds of co-movements. Intuitively, the overall expected return of a portfolio should be the summation of these four kinds of returns. In the followings, we will provide the mathematical details for this formulation step by step.

In the most simplest case, let us consider there are only two financial assets, i and j, in the portfolio. The expected return of the portfolio is therefore:

$$r_{ij} = \sum_{s} r_{ij}^{s} \cdot P_{ij}^{s}, \tag{2.9}$$

where s denotes different types of co-movements (up-up, up-down, down-up and down-down); r_{ij}^{s} denotes the expected return between asset i and asset j when their type of co-movement is s; P^s_{ij} is the probability that the type of co-movement between asset i and asset j is s. Hence, Eq. (2.9) computes the portfolio's expected return by summing the expected return of each type of co-movement with its probability of occurrence.

The probability, P^s_{ij}, in Eq.(2.9) is computed as follows:

$$P^s_{ij} = \frac{T^s_{ij}}{T}, \tag{2.10}$$

where T^s_{ij} denotes the accumulative length of which the type of co-movement between asset i and asset j is s and T is the total length of the time series. Let us illustrate the idea of Eq. (2.10) with the help of Figure 2.1. In the figure, there are two periods for the up-up co-movement between asset A and asset B: from t_0 to t_1 and from t_6 to t_7. Assume further that the differences between t_0 and t_1 is 10, between t_6 and t_7 is 12 and between t_0 and t_7 is 50. Then, $P^s_{AB} = T^s_{AB}/T = ((t_1 - t_0) + (t_7 - t_6))/(t_7 - t_0) = (12 + 10)/50$.

The expected return, r^s_{ij}, in Eq.(2.9) is computed as follows:

$$r^s_{ij} = \frac{1}{N^s_{ij}} \sum_{k \in s} \left(w_i r^k_i + w_j r^k_j \right), \tag{2.11}$$

where w_i and w_j are respectively the weights of asset i and asset j in the portfolio; N^s_{ij} denotes the number of segments of type s in the time series; k is a segment belongs to the co-movement of type s; r^k_i and r^k_j are respectively the expected return of asset i and asset j in segment k. Again, let us use Figure 2.1 to illustrate the idea of this equation. Consider for the case where s denotes for the up-up co-movement. In this situation, there are two segments, from time t_0 to t_1 and from time t_6 to time t_7, belong to s. Assume further that the expected return of asset A in the two segments are respectively $r^1_A = \$5$ and $r^2_A = \$4$. Similarity, let $r^1_B = \$6$ and $r^2_B = \$3$. Hence, $r^s_{AB} = (1/2)\left((w_A \cdot \$5 + w_B \cdot \$6 + w_A \cdot \$4 + \$w_B \cdot \$3)\right)$.

Now, we extend this simple portfolio with two assets only into a portfolio with n multiple assets. In the multiple assets situation, we can obtain the expected return of the portfolio, r_Θ, by a pairwise linear summation of every two assets:

$$r_\Theta = \sum_{i=1}^{n-1} \sum_{j \neq i}^{n} r_{ij}, \tag{2.12}$$

where r_{ij} is preciously defined in Eq. (2.9). In order for efficient computation by using linear programming [74], Eq. (2.12) can be formulized in this form:

$$r_\Theta = \sum_{i=1}^{n} w_i r_i, \tag{2.13}$$

$$r_i = \frac{1}{(n-1)} \sum_{j=1, j \neq i}^{n} \sum_{s} \frac{1}{N_{ij}^s} (\sum_{k \in s} r_i^k) \cdot P_{ij}^s, \tag{2.14}$$

where the component $1/(n-1)$ in Eq. (2.14) is added for the reason of normalization and r_i is the return generated from asset i in its co-movement with all the other assets.

2.2.3. Co-movement Based Expected Risk

In this section, we will present how we quantify and derive the expected risk of the portfolio, β_Θ. Unlike the approaches used in the modern portfolio theories where the portfolio's expected risk is usually defined as the variance of the portfolio's expected return (a statistical perspective), in this work, we define the portfolio's expected risk as *the chance of exposure to loss*, by considering the co-movement among the time series of the assets in the portfolio (a data mining perspective).

In the beginning, let us use Figure 2.1 again to illustrate our idea. For simplicity, assume we only have two choices for formulating the portfolio: either buy asset i solely (i.e. $w_i = 1$ and $w_j = 0$) or buy equal shares of asset i and asset j (i.e. $w_i = 0.5$ and $w_j = 0.5$). Let β_1 and β_2 be the risk associated with buying asset i solely (the first option) and risk associated with buying asset i and asset j simultaneously (the second option), respectively. Suppose we have identified β_1. Now, if we switch to the second option (buy asset i and asset j simultaneously), would the chance of exposure to loss be increased or decreased? That is, would $\beta_2 > \beta_1$ or $\beta_2 \leq \beta_1$?

In Figure 2.1, from t_2 to t_3, the type of co-movement between asset i and asset j belongs to down-down. If this type of co-movement occurs frequently in the entire time series, we might have to expect the price drops of asset i would usually accompany with the price drops of asset j. In other words, *the chance of exposure to loss* is *increased*. So if two assets frequently involved in the down-down co-movement, they should generate a higher level of risk with respect to having either of themselves alone. To conclude, if the down-down co-movement frequently appears among the two time-series, then $\beta_2 > \beta_1$.

On the other hand, from t_3 to t_4 in Figure 2.1, the type of co-movement with respect to asset i is up-down. This means that when the price of asset i goes up, the price of asset j will goes down. From asset i's stand point of view, *the chance of exposure to loss* is also *increased* in this situation. Thus, the more frequently this situation happens, the higher the risk should be expected. Yet, it is worth noting that this type of risk is *not* symmetric. From asset j's angle, *the chance of exposure to loss* is *decreased*. This is because when the price of asset j decreases, the price of asset i will increase. This compensate effect will surely reduce the loss from asset j's perspective. As a result, the expected risk of the portfolio should take the considerations of both sides. Therefore, in our co-movement model, the underlying idea is to consider the frequency of this up-down co-movement as well as the relative magnitude of their movements. The mathematical details of how the expected risk should be computed will be discussed in the later paragraphs.

Finally, the co-movements with respect to asset i for the remaining time periods in Figure 2.1 are either up-up co-movement or down-up co-movement. Both of these co-movements imply *the chance of exposure to gain* increase (a kind of up-side risk). Hence, these two types of co-movements do not need to be included in computing the portfolio's expected risk, as our definition of risk is *the chance of exposure to loss* (a kind of down-side risk).

To summarize, following these discussions, the expected risk of a portfolio is computed by having a pairwise comparison of the assets in the portfolio against the frequencies and magnitudes of this down-down co-movement or up-down co-

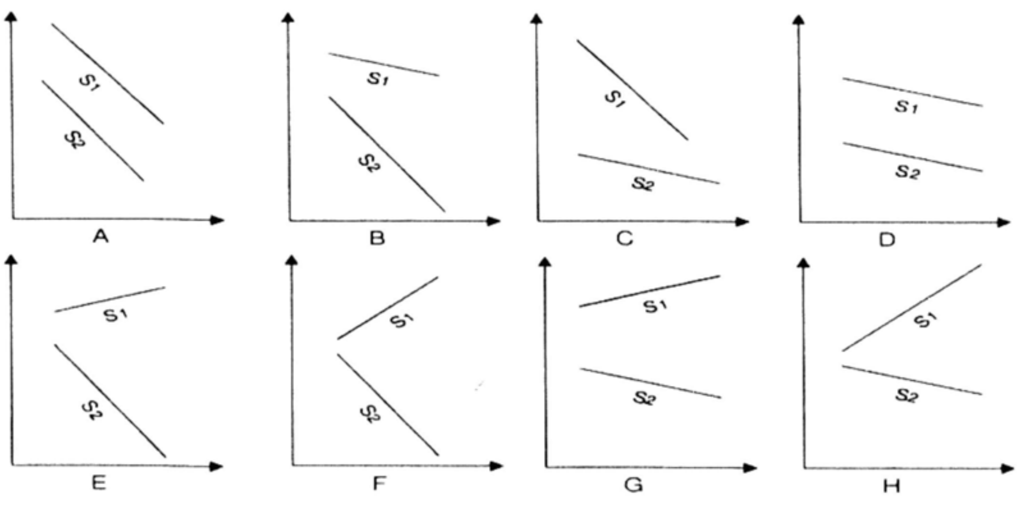


Figure 2.2: Different level of risks generated by different kinds of co-movement.

movement. Mathematically, the risk of the portfolio, β_Θ, is computed as follows:

$$\beta_\Theta = \sum_{i=1} w_i \sum_{j=1} w_j \beta_{ij}, \tag{2.15}$$

where β_{ij} is the risk of buying asset j with respect to asset i. Note that $\beta_{ij} \neq \beta_{ji}$ according to the previous discussion about the asymmetric relationship of risk. Note that the summation in Eq.(2.15) includes β_{ii}, which can be interpreted as risk of buying the asset i itself.

Now, the only remaining question here is how to quantify β_{ij}. Unfortunately, it is not a trivial task. To account for it, let us refer to Figure 2.2 which shows the down-down and up-down co-movements with different kinds of slopes. In the figure, the first row (A, B, C and D) compares four different cases of down-down co-movements, whereas the second row (E, F, G and H) compares four different cases of up-down co-movements.

In case A, both time series, $S1$ and $S2$, go down straightly, whereas in case B, only the time series $S2$ goes down straightly. Comparing case A and case B, one should agree that the risk (chance of exposure loss) associated with case B is *higher* than that of case A, because case B outlines a small drop of $S1$ will lead to big drop of $S2$. Similarly, in case C, a big drop of $S1$ will only lead to a small drop of $S2$, whereas in case D, a small drop in $S1$ will immediately accompany with the same amount of drop of $S1$. Accordingly, the risk (chance of exposure loss) associated with case C should be less because the relative magnitude of dropping in case C is less. Lastly, it is obvious that the risk associated with case D is less than that of case A, meanwhile case C is associated with less risk than case B. To summarize, the impact of β_{ij} in down-down correlation for these four cases should be: $B > A > D > C$.

By using an analysis similar to the above discussion, the impact of β_{ij} in the up-down co-movement for case E, case F, case G and case H should be: $E > F > G > H$. To conclude, in order to quantify the risk, we need to consider the relative magnitude of the slopes in each segment within the entire time series. Eventually, β_{ij} (the risk of buying asset j with respect to asset i) is formulated as follows:

$$\beta_{ij} = \sum_{k \in s} \delta_i^k \cdot \theta_j^k \cdot (\frac{\theta_j^k}{\theta_i^k}) \cdot (c_i^k c_j^k) \cdot P_i^k \tag{2.16}$$

where s belongs to either down-down or up-down co-movements. θ_i^k and θ_j^k are respectively the slope of asset i and asset j in segment k. Hence, the second component θ_j^k represents the magnitude of slope of asset j, the third component θ_j^k/θ_i^k accounts for the relative slope difference between asset i and asset j. c_i^k and c_j^k are respectively the coefficient of determination[60] for asset i and asset j in segment k, which denote how confident is the co-movement relationship of that segment for the two time series. P_i^k is the probability of segment k appears in asset i. $\delta_{i,k}$ is defined as:

$$\delta_{i,k} = \begin{cases} -1 & \theta_i^k < 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2.17}$$

which identifies the type of co-movements (up-down or down-down) between i and j.

2.2.4. Co-movement Based Portfolio Construction

According to the discussions of the previous sections about how the portfolio's expected return and the portfolio's expected risk are derived, we now present how a portfolio could be constructed in our proposed co-movement model in this section.

Underlying for all of the portfolio management theories is that we should choose a portfolio that provides an expected return equals to a desired level of return, R, meanwhile it accumulates the lowest level of expected risk. Mathematically, we are trying to solve the following optimization problem by identifying all w_i based on a given R:

$$\min \quad \beta_\Theta = \sum_{i=1} w_i \sum_{j=1} w_j \beta_{ij}, \tag{2.18}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} w_i r_i = R. \tag{2.19}$$

where $\forall i, j$, $w_i \geq 0$,$w_j \geq 0$ and their summation equals to 1. The r_i in Eq. (2.19) is defined in Eq. (2.14). Finally, if short selling is allowed, then constraint $w_i \geq 0$ and $w_j \geq 0$ can be omitted.

2.2.5. Evaluation

In this section, we evaluate the effectiveness of our proposed co-movement model by using a real life dataset, the Morgan Stanley Capital International G7 index (MSCI-G7).[1] This dataset, MSCI-G7, consists of equity indices of seven countries (Canada, France, Germany, Italy, Japan, United Kingdom and United States) that are recorded in every working day (5 working days per week). The period that we have archived is from 2003/1/3 to 2007/4/30 (around 4.3 years). Two experiments are conducted to compare and evaluate our proposed co-movement model:

[1]http://www.mscibarra.com

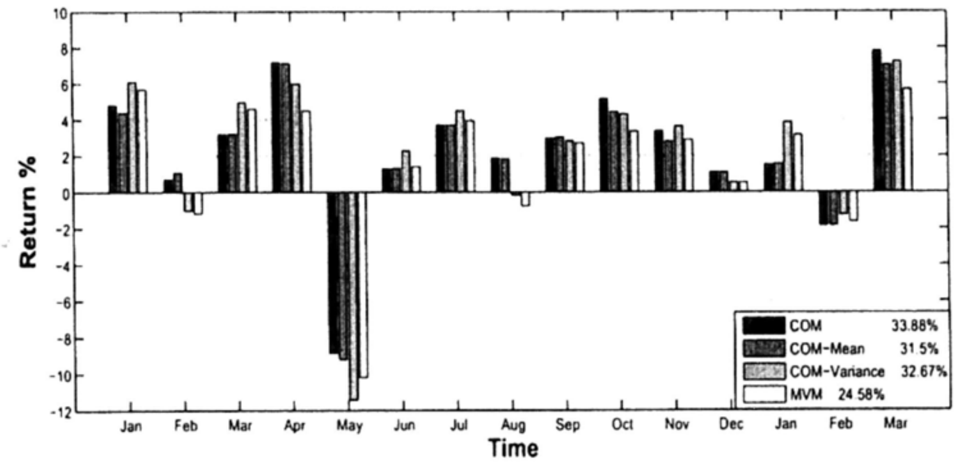


Figure 2.3: The returns obtained by different portfolio models

1. **Different Portfolio Models Comparison** We compare our proposed co-movement model with three different kinds of portfolio models. This experiment tries to identify the necessity of the assets' dependency relationships when computing the portfolio's expected return, and the possibility of having the non-traditional view of the portfolio's expected risk as the exposure of loss.

2. **Sensitivity of Segmentation** As discussed in the previous section, since all financial time series contain high levels of noise, we need to smoothen the time series by using bottom-up segmentation [42] and regression analysis. In this experiment, we try to evaluate the sensitivity of our proposed co-movement model with different number of segments.

Different Portfolio Models Comparison

In this section, we compare the performance of different portfolio models with our proposed co-movement model. Here is a list of models that we have implemented:

1. **COM**: This is the proposed model that we have discussed in this work.

2. **MVM**: This is the traditional mean-variance model [55] for portfolio construction. It serves as the benchmark approach in our experimental study. Mathematically details of this model is discussed in Section 2.1 – Related Work.

3. **COM-Mean**: This model is similar to our proposed co-movement model, except that the portfolio's expected return is computed by the traditional method, i.e. the dependency of the expected return of the assets in the portfolio is ignored. Mathematically, the r_i in Eq. (2.19) is simply the expected return of asset i in the portfolio, but *not* the one defined in Eq. (2.12).

4. **COM-Variance**: This model is similar to our proposed co-movement model, except that the portfolio's expected risk is computed by using the traditional method, i.e. the portfolio's expected risk, β_Θ, is the variance of the the portfolio's expected return. Mathematically, we change β_Θ in Eq. (2.18) with the following equation:

$$\beta_\Theta = \sum_{i=1}^{n-1} \sum_{j=i}^{n} Var(r_{ij}), \tag{2.20}$$

   where $Var(r_{ij})$ is the variance of the expected return.

In order to evaluate these four alternative allocation strategies in terms of their expected returns and risks, we evaluate their difference on realized portfolio performance. We use the first three years' data (784 days) of MSCI-G7 as training data, so as to compute the optimal weights, w_i, for each asset, i, in the portfolio. The rest of the data, 15 months, are served as the testing data for evaluation. The portfolio will be re-constructed each month in these 15 months, so as to capture the latest movements of the time series. The oldest month would be discarded whenever for the re-construction, so as to denote for ignoring the outdated information. The average segment number is set to 90 in this experiment. The sensitivity of number of segments will be discussed in the next section. The desired level of return, R, in Eq. (2.19) is set as the mean return of their corresponding efficient frontiers [71].

Figure 3.5 shows the returns that we get from each model in each month, as well as the cumulative return for the four approaches over the 15 months in the bottom right corner in the box. Although our proposed co-movement model, COM, is not a sole winner for all of the months, the cumulative return of COM does outperform the other approaches significantly. The highest cumulative return that we get is came from COM, with a profit of 33.88%, which is 11% higher than that of MVM (the traditional Mean-Variance model, the benchmark approach), which obtains only 24.5%.

Furthermore, it is worth noting that all the approaches that are related to co-movements (COM, COM-Mean and COM-Variance) outperform the MVM (the benchmark approach). It demonstrates clearly that the importance of identifying different types of co-movements is justifiable.

By comparing between COM and COM-Variance, one can justify the possibility of regarding risk as *the exposure of loss* rather than the traditional point of view - *variance of return.* Although COM-Variance performs better than COM in a few months, it performs inferior than COM most of the time. Similarly, COM-Mean performs inferior than COM, which demonstrates the importance of considering the dependency relationship when computing the portfolio's expected return.

Sensitivity of Segmentation

Since our proposed co-movement model requires data pre-processing by using bottom-up segmentation and regression analysis for identifying the trends of the time series, a question commonly asked would be: how sensitivity is the co-movement model with the number of segments? As a result, we try to evaluate how the average number of segments will affect the performance of our approach in this experiment.

The settings of this experiment is the same as the previous one (Section 2.2.5), except that we alter the number segments from 10 to 300, and then we record down the cumulative returns. Figure 2.4 shows the empirical result of this experiment. We address some interesting findings below.

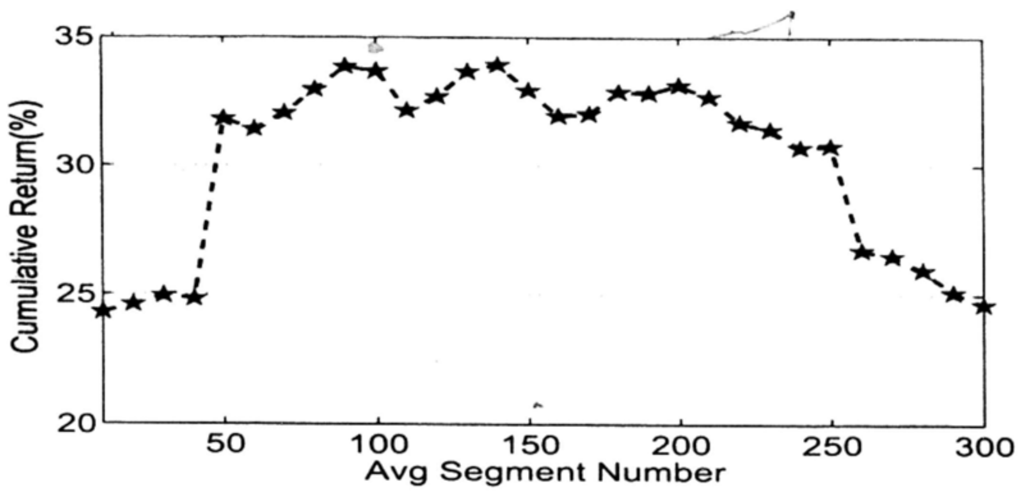


Figure 2.4: The cumulative return of different number of segments.

1. The cumulative returns for all kinds of segments are always positive. This is a very encouraging sign, since it implies the number of segments will not change the cumulative return from positive to negative or vice versa.

2. The overall shape of the graph is concave. This suggests that the cumulative return and the number of segments have some kinds of dependency relationships, where the cumulative return is *not* distributed randomly with respect to the number of segments. Moreover, a concave shape also suggested that there should exist a "best" range against the number of segments, which is around 100 to 150, i.e. each segment lasts for around 5 to 7 days (a week).

3. When the number of segment is too few (less than 50) or too many (more than 300) the cumulative return would drops to around 25%. Nevertheless, it still performs better than MVM (the benchmark model, 24.5%).

We also conduct segment number analysis on the combination types of other co-movement models and come out the similar result.

2.2.6. Summary

We describe a co-movement model for constructing financial portfolio by analyzing and mining the co-movement patterns among multiple time series. Different from traditional statistical approach in financial world, which takes the co-variance among the portfolio as risk and the summation of individual expect return as portfolio return, our approach models the risk from the co-movement patterns and computes portfolio returns by considering all dependency relationships among assets. As the first step of this area, the promising experiment results on real financial data show the new formulation of our return and risk can bring great benefits for effective asset allocation.

2.3. Detecting Leaders from Correlated Time Series

In this section, we present our algorithm to detect leaders from multiple correlated time series streams.

2.3.1. Leadership Discovery

In this section, we first define the problem of leadership discovery.

Problem Definition The problem of leadership discovery is to find the leaders among N synchronized time series, $S^1, S^2, \ldots, S^N$, that exhibit significant lead-lag relations over the set of time series in a real-time manner, where the lead-lag relation is measured by the concept of lagged correlation.

In this work, we focus on high-frequency time series, i.e., the time series whose data points are generated in high frequency (seconds, minutes) from the finance market. Although most of the studies in financial literature [14, 16] deal with low-frequency data (i.e., daily or hourly), we find that high-frequency data carries more information about lead-lag that is not revealed by low-frequency

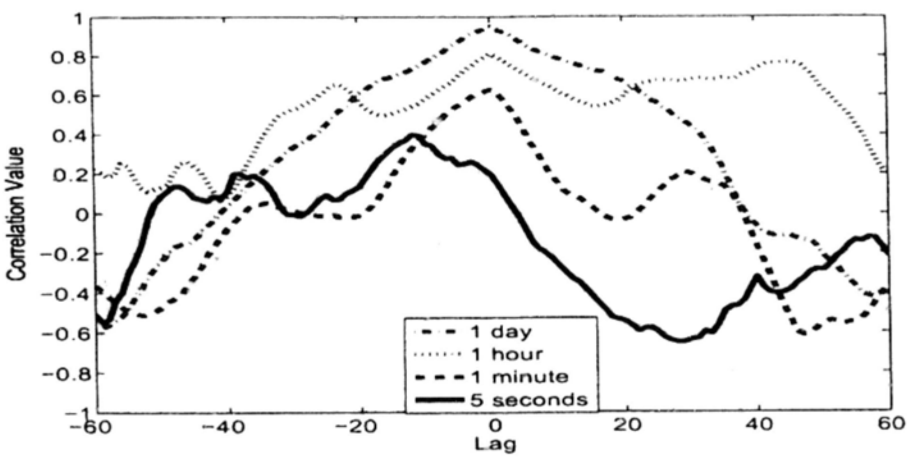
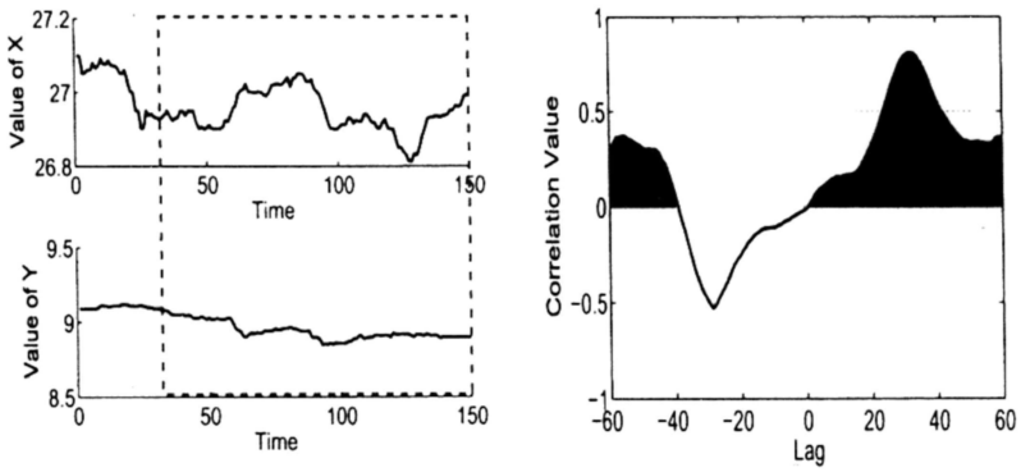


Figure 2.5: Lagged Correlation at Different Data Frequency

data. We demonstrate this point by the following real example.

Fig. 2.5 shows the lagged correlation computed on two stock price time series at four different data frequency (5 seconds, 1 minute, 1 hour and 1 day) but starting from the same date 01/02/2004. As shown in the figure, the low-frequency data (1 day and 1 hour) has a maximum correlation value at zero lag and exhibits positive correlation at most of the lags, which implies that the two stocks co-move with the same trend at zero lag from a macro perspective. On the other hand, the correlation computed on high-frequency data (1 minute and 5 seconds) has its peak value at a non-zero lag and exhibits more negative correlations, which means that the two stocks actually undergo a co-movement in an opposite trend with some delay from a micro perspective. Therefore, it is necessary and interesting to investigate the high-frequency data since it indicates the microstructure of the finance market (e.g., the process of practitioners making their trading decision and generating stock price).

Solution Overview Our solution to the problem of leadership discovery has three main steps: (1) compute the lagged correlation between each pair of time series; (2) construct an edge-weighted directed graph based on lagged correlations to analyze the lead-lag relation among the set of time series; (3) detect the leaders by analyzing the leadership transmission in the graph. We now discuss each step



(a) Two Time Series (b) Lagged Correlation

Figure 2.6: Two Time Series and the Lagged Correlation Plot over their Local Sliding Windows

in detail.

Lagged Correlation Computation

The first step is to compute the lagged correlation between each pair of time series. Existing work [69] on computing lagged correlations cannot be directly applied to our problem, since i) it tries to capture lag correlation in the whole history of streams while our objective is to obtain the local lags in the current sliding window, and ii) the approximation in their updating algorithm has accuracy preference to the points with small lags and may generate a large error for large lags, which is not desirable for our problem. Therefore, we propose to aggregate the effects of various lags and define an *aggregated lagged correlation*. Without loss of generality, we focus on positive correlation, while negative correlation can be handled similarly. We explain how to compute the aggregated lagged correlation by the following example. Fig. 2.6(a) shows two time series X (top) and Y (bottom) with a length of 150. The window length is set to be 120 and we consider the window marked by the dotted rectangle. Fig. 2.6(b) shows the lagged correlation at each lag l computed by Eq. (3.25) over the two windows. The maximum lag $m = 60$, i.e., $|l| \leq 60$. When $l < 0$ (i.e., Y is delayed), the positive correlation only exists for $l \in [-60, -39]$ (the shadowed area). When $l \geq 0$ (i.e., X is delayed), starting from $l = 1$, we observe a strong increase in positive correlation and it achieves a peak value of 0.81 at $l = 32$. In order to identify the leadership (X leads Y or Y leads X), we need to aggregate all the observed correlation values over the entire lag span and take the expected correlation value given the two cases of l. The aggregated lagged correlation between two time series S^i and S^j, denoted as $E^{ij}(\rho)$, is then defined as the larger expected correlation value:

$$E^{ij}(\rho) = \max(E^{ij}(\rho|l \geq 0), E^{ij}(\rho|l < 0)). \tag{2.21}$$

We say that S^i *leads* S^j *if* $E^{ij}(\rho) = E^{ij}(\rho|l < 0)$, *and* S^i *is led by* S^j *otherwise if* $E^{ij}(\rho) = E^{ij}(\rho|l \geq 0)$. Such leadership ($S^i$ leads S^j or vice versa) is also called the *lead-lag* relation between S^i and S^j. The value of $E^{ij}(\rho|l \geq 0)$ is computed as

$$E^{ij}(\rho|l \geq 0) = \sum_{l=0}^{m} \max(\rho^{ij}(l), 0) \cdot p(l|l \geq 0), \tag{2.22}$$

where $\max(\rho^{ij}(l), 0)$ takes only positive correlations and $p(l|l \geq 0)$ takes the value of $1/(m+1)$ since the contribution of each lag is equal. $E^{ij}(\rho|l < 0)$ can be computed symmetrically. In Fig. 2.6, by Eq. (2.22), $E^{XY}(\rho|l < 0) = 0.1056$ and $E^{XY}(\rho|l \geq 0) = 0.4017$. Thus, $E^{XY}(\rho) = max(0.1056, 0.4017) = 0.4017$ indicating X is led by Y.

Graph Construction

In order to model the leadership relationships among a set of time series, we construct a simple edge-weighted directed graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the set of nodes $\mathcal{V} = \{S^1, S^2, \ldots, S^N\}$ represents N time series, and the set of directed edges $\mathcal{E}$ represents lead-lag relations between time series. An edge (S^i, S^j) indicates that S^i is led by S^j and its weight is set as $E^{ij}(\rho)$. Since we are interested in significant lead-lag relations, we set a *correlation threshold* γ such that only those pairs S^i and S^j with $E^{ij}(\rho) > \gamma$ have edges in $\mathcal{G}$. It is important to note that, when the window slides, the edges and their weights in $\mathcal{G}$ will change dynamically.

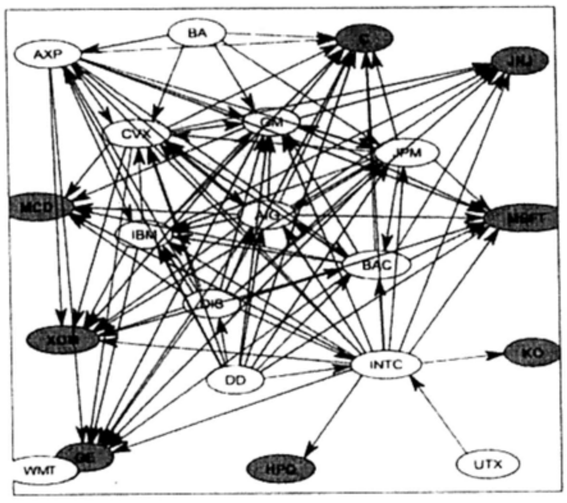


Figure 2.7: Graph $\mathcal{G}$ on Component Stocks of Dow Jones Industry Average

Figure 2.7 shows the graph $\mathcal{G}$ constructed on 30 component stocks of Dow Jones Industry Average. Each node in the graph represents a stock, each edge represents a significant lagged correlation between two stocks and the arrow on the edge points to the leading stock (we omit the edge weight for clear visualization). For example, there is an edge from Intel Corporation (INTC) to Hewlett-Packard Company (HPQ), which indicates that there is a significant lagged correlation between the world's largest semiconductor company and the largest worldwide seller of personal computers in this sampled period. In the graph, there are some nodes that have relatively more in-links than others, which indicates that these stocks are the leading centers. Citigroup Inc. (C), the major American financial service company, has the largest in-degree of 11, leading 1/3 of Dow Jones component stocks at the time. On the other hand, there are some dangling nodes in the graph, which are not led by any other stocks. Wal-Mart Inc. (WMT), the world's largest public corporation, is an isolated stock without out-link or in-link.

Leader Extraction

Given the graph $\mathcal{G}$, we now extract leaders from it. Since a good leader needs to capture both direct and indirect leaderships, we first analyze the leadership transmission in $\mathcal{G}$. Suppose that each time series has a leadership score, based

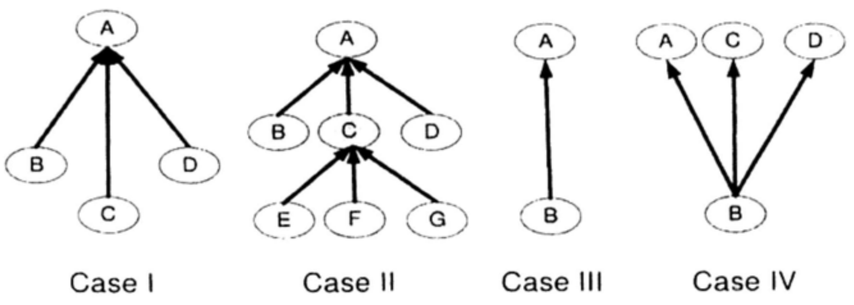


Figure 2.8: Comparison of Leadership Score on Different Graph Structures

on which a ranking among time series can be obtained. We now discuss how to assign a good leadership score.

Consider the leadership score of A under different graphs as shown in Fig. 2.8. In case I and II, A directly leads 3 time series, B, C, and D. In case I, all of the three have zero in-degree. In case II, C has an in-degree of 3, which implies that A indirectly leads the three that are led by C as well as the three directly led by A itself. It indicates that the leadership score of A in case II should be larger than that in case I. On the other hand, consider case III and case IV. In case III, B is exclusively led by A, whereas in case IV, B is led by A as well as the other two, C and D. The leadership score of A in case III should be larger than that in case IV. Therefore, we define leadership score as

$$score^j = \sum_{S^i \in L_{S^j}} \frac{score^i E^{ij}(\rho)}{d_{out}(S^i)}, \tag{2.23}$$

where L_{S^j} is the set of time series that are led by S^j, $score^i$ is the leadership score of S^i and $d_{out}(S^i)$ is the summation of out edge weights of S^i. This leadership score defined above is similar to that defined for the Web Graph on which PageRank score is computed to represent the popularity of web pages. In this thesis, we adopt PageRank [13] as the leadership score of a time series to quantify its importance in the graph $\mathcal{G}$.

Finally, based on the structure of $\mathcal{G}$ and the PageRank values of time series, we extract the leaders by eliminating redundant leaderships. The basic idea is to first sort the time series by the descending order of their PageRank values and then to remove iteratively the time series that is led either by previously found leaders or by the descendant of previously found leaders.

The Overall Algorithm

Our solution is presented in Algorithm 1. Given the latest values in time series at time point t, the algorithm first updates the statistics needed in computing lagged correlations as stated in Section 2.1. It then computes pairwise aggregated correlations (Lines 2-5). Graph $\mathcal{G}$ is then constructed (Line 6) and the power method computes the PageRank vector π (Line 7). Finally, the *ExtractLeaders* procedure (Algorithm 2) identifies leaders. In *ExtractLeaders*, time series are first sorted by the descending order of the rank π. Then starting from the time series with the highest rank, it checks the time series led by it and removes them as well as their descendants from the list. The procedure *RemoveDescendant* repeats the process recursively until all descendants of the current leader are removed. The remaining time series on the list are returned as leaders.

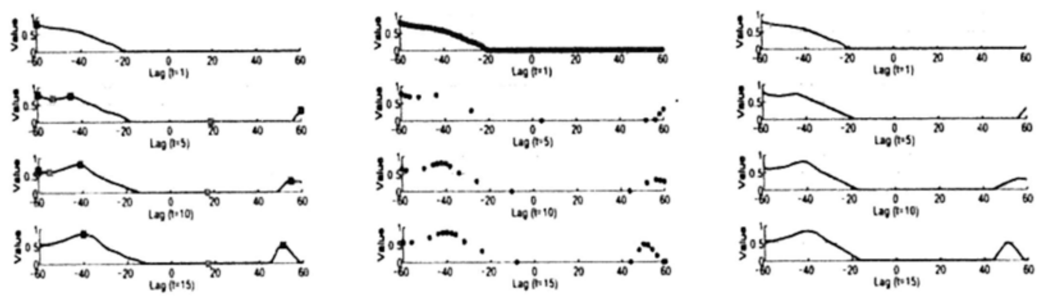
We now analyze the complexity of Algorithm 1. Correlation computation in Lines 2-5 needs to compute $(2m + 1)N^2$ correlation values, which involves complex mathematical calculation. PageRank computation and the *ExtractLeaders* procedure take $\mathcal{O}(kN^2)$ and $\mathcal{O}(N)$ time, respectively, where k is the number of power method iterations. Thus, the most time-consuming steps in Algorithm 1 are in computing correlations and PageRank. The space complexity of the algorithm is $\mathcal{O}(mN^2)$ for storing the correlation statistics and $\mathcal{O}(N^2)$ for storing the values in power method.

Algorithm 1 DiscoverLeaders

```
INPUT: $N$ time series, $S^1, \ldots, S^N$, up to current time $t$, sliding window length $w$, maximum lag $m$, correlation threshold $\gamma$
OUTPUT: *leaders*
1: Update statistics needed for correlation computation;
2: for every pair of time series $S^i$ and $S^j$ do
3:     Compute correlation $\rho^{ij}_{t,w}(l)$, for $|l| \leq m$;
4:     Compute aggregated lagged correlation $E^{ij}(\rho)$ by Eq. (2.21);
5: end for
6: Construct graph $\mathcal{G}$ with respect to $\gamma$;
7: Compute PageRank vector $\pi$ on $\mathcal{G}$;
8: $L \leftarrow ExtractLeaders(\mathcal{G}, \pi)$;
9: return $L$;
```

Algorithm 2 ExtractLeaders

```
INPUT: graph $\mathcal{G}$, rank vector $\pi$
OUTPUT: *leaders*
1: $L \leftarrow$ Sort time series in descending order by $\pi$;
2: for each time series $S^j$ in $L$ do
3:     $RemoveDescendant(L, \mathcal{G}, S^j)$;
4: end for
5: return $L$;
6: Procedure $RemoveDescendant(L, \mathcal{G}, S^j)$
7: for each time series $S^i$ in $L$ after $S^j$ do
8:     if $(S^i, S^j)$ is an edge in $\mathcal{G}$ then
9:         $RemoveDescendant(L, \mathcal{G}, S^i)$;
10:        Remove $S^i$ from $L$;
11:    end if
12: end for
```



(a) Exact Interesting Area (b) Probed Correlation Values (c) Approx. Interesting Area

Figure 2.9: Tracking the Interesting Area

In a stream environment, correlation computation becomes the bottleneck of Algorithm 1 since the implementation of PageRank is fast when the graph is small enough to store in the main memory (e.g., $N = 500$). Too many correlation values need to be computed at each time point and there are endless time points coming into the stream. In order to accomplish prompt leadership detection, we further propose an effective update approach that is able to reduce the number of correlation computations and meanwhile retaining high accuracy, which is described in the following section.

2.3.2. Real-Time Correlation Update

In order to speed up the computation of the aggregated lagged correlation for a pair of time series, we propose an efficient update approach by investigating the evolutionary characteristics of lagged correlations. Recall that in Eq. (2.22), all positive lagged correlation values are aggregated, i.e., we compute the area with positive correlations. Therefore, compared with the exact correlation value at each lag, the area formed by these positive correlations is more crucial to determine the lead-lag relation. We call this area the *interesting area*. The basic idea of our update approach is to track the interesting area. More specifically, at an initial time point, we compute the exact correlation value at each lag and record the interesting area. Then at the subsequent time point, we track and update this interesting area by computing the correlation for only a small number of lags. We then use this interesting area to approximate the aggregated lagged correlation.

We now discuss how to track and update the interesting area. Fig. 2.9(a) gives an example of the evolutionary shapes of the interesting area between two time series. The lagged correlation is computed at each lag $l \in [-60, 60]$. At time $t = 1$, the interesting area spans from $l = -60$ to $l = -20$ and the corresponding correlation value decreases gradually from 0.8 to 0. We call such continuous area a *wave*. When $t = 5$, we note that there are two waves of the interesting area. The first one spans from $l = -60$ to $l = -17$, which is obviously an evolution from the previous wave. Compared with the wave at $t = 1$, the boundary of this wave enlarges from $l = -20$ to $l = -17$. Hereafter, we call this type of wave an *existing wave*. The second wave spans from $l = 55$ to $l = 60$. Since this wave does not exist at $t = 1$, we call this type of wave a *new wave*. When $t = 10$ and $t = 15$, the existing wave changes slowly, while this new wave enhances its effect.

The above example shows that, in order to keep track of the interesting area, we need to capture the evolutionary pattern of two types of waves, existing waves and new waves. Our solution is based on two observations.

Observation 1. An existing wave at time t is relatively stable at subsequent time points after t.

Observation 1 can be explained as follows. For a specific lag l, the correlation $\rho^{ij}_{t,w}(l)$ at time t is computed on two shifted windows $s^i_{t,w-l}$ and $s^j_{t-l,w-l}$. When the time moves to $t+1$, correlation $\rho^{ij}_{t+1,w}(l)$ is computed on $s^i_{t+1,w-l}$ and $s^j_{t-l+1,w-l}$. Notice that there is a large overlap in these two sets of windows. Specifically, the difference between $s^i_{t,w-l}$ and $s^i_{t+1,w-l}$ (also between the other two windows) is only one point. As a result, the two correlations $\rho^{ij}_{t,w}(l)$ and $\rho^{ij}_{t+1,w}(l)$ cannot differ a lot. Therefore, we have the above observation of an existing wave.

Using Observation 1, we can track an existing wave as follows. The most important features of a wave are its magnitude and width. The magnitude of a wave can be characterized by its maximum points, while the width can be characterized by the minimum points. Therefore, we propose to approximate the area of an existing wave by tracking its peak points. Specifically, after we compute the exact correlation value for each lag at the initial time point, we record the peak points for the existing wave. Then, at the subsequent time point, we only compute the exact correlation value for the lag of each maximum peak point and conduct a geometric progression probing to both sides of the lag until the probe reaches the boundary. The boundary can be either the adjacent minimum peak point, the maximum lag $\pm m$ or the point with a negative correlation value. Then, we conduct a linear interpolation over the computed correlation points to approximate the area of the wave. Finally, the peak points are updated according to the probed correlation values so that they can be used for the subsequent time point.

Fig. 2.9 (b) shows the points, at which we compute (probe) correlation values. Suppose that $t = 1$ is an initial time point. We compute all the lagged correlation values for $l \in [-60, 60]$ and record a maximum peak point at $l = -60$. When $t = 5$, we probe from the maximum peak point $l = -60$ until reaching the boundary, where we detect a negative correlation. In this process, the probing step is increased exponentially so that the approximated wave has higher accuracy around the peak point. There are altogether 7 correlation values computed in the probing process. Then, as shown in Fig. 2.9(c), linear interpolation is applied to these 7 points to form the approximated existing wave. As further shown in $t = 10$ and $t = 15$, this existing wave can be well tracked.

Now, the remaining problem is to track a new wave. As there is no existent evidence of a new wave at the initial time point, we are not able to record its peaks for tracking purpose. Fortunately, we have the following observation of new waves.

Observation 2. A new wave at t only emerges at maximum lag values of $\pm m$.

Observation 2 can be explained as follows. We first consider the case when $0 \le l \le m$. At a specific time t, the correlation $\rho^{ij}_{t,w}(l)$ is computed on two

Algorithm 3 UpdateCorrelation

INPUT: new value at t for two time series S^i and S^j, sliding window length w, maximum lag m, the set of peak points $peak^{ij}_{t-1}$ at time $t-1$

OUTPUT: the lead-lag relation of S^i and S^j

```
1: if there is no existing wave at $l = \pm m$ then
2:     Compute $\rho^{ij}_{t,w}(m)$ and $\rho^{ij}_{t,w}(-m)$ to detect potential new waves;
3:     if there exists new waves then
4:         Add the corresponding $l$ to $peak^{ij}_{t-1}$;
5:     end if
6: end if
7: for each maximum peak point $ptMax$ in $peak^{ij}_{t-1}$ do
8:     $sampleWavePointSet = Probe(ptMax)$;
9:     $wavePointSet = Interpo(sampleWavePointSet)$;
10:    Add $wavePointSet$ to corresponding $\rho^{ij}_{t,w}(l)$;
11: end for
12: $peak^{ij}_t = detectPeak(\rho^{ij}_{t,w}(l))$;
13: Compute aggregated lagged correlation $E^{ij}(\rho)$ by Eq. (2.21);
14: Decide the lead-lag relation of $S^i$ and $S^j$;
```

windows of length $(w - l)$. Therefore, with the increase of l from 0 to m, the window length, on which $\rho^{ij}_{t,w}(l)$ is computed, decreases. On the other hand, compared with the previous time point $t-1$, each time series evolves by adding a new data point to and deleting an old data point from the sliding window. This causes the value of $\rho^{ij}_{t,w}(l)$ to be different from $\rho^{ij}_{t-1,w}(l)$. However, the effect of the time series evolvement on the value of $\rho^{ij}_{t,w}(l)$ is different for different lag l. With the increase of l, the windows, on which $\rho^{ij}_{t,w}(l)$ is computed, becomes smaller and thus the effect of the evolvement becomes larger, which results in larger difference of $\rho^{ij}_{t,w}(l)$ and $\rho^{ij}_{t-1,w}(l)$. This explains why a new wave may emerge at the largest lag $l = m$. Similarly, a new wave is also likely to emerge at $l = -m$.

Algorithm 4 Probe

```
INPUT: a peak point $ptMax$
OUTPUT: $sampleWavePointSet$
1: $sampleWavePointSet \leftarrow$ Compute $\rho^{ij}_{t,w}(ptMax)$;
2: $step = 1$;
3: $index = ptMax \mp step$;            // + for right side probe
4: while $index$ is not a left(right) boundary point do
5:     $sampleWavePointSet \leftarrow$ Compute $\rho^{ij}_{t,w}(index)$;
6:     $step = step \times 2$;
7:     $index = ptMax \mp step$;        // + for right side probe
8: end while
```

According to Observation 2, we can track new waves by monitoring the correlation values at $l = \pm m$. As shown in Fig. 2.9(b), although there is no sign of a new wave at $l = 60$ when $t = 1$, we also compute its correlation at $t = 5$. This strategy successfully detects a positive correlation value at $l = 60$. Then, we take it as an existing wave and track it using the approach we have discussed above. In summary, at $t = 5$, we use 11 points to track the whole interesting area, saving 91% of correlation computation.

Our update approach, *UpdateCorrelation*, is presented in Algorithm 3. It first checks the correlation values at the two maximum lag points to detect potential new waves (Line 2). If there exists a new wave, the algorithm treats it as an existing wave (Lines 3-5). Then, the algorithm approximates each existing wave by two procedures *Probe* and *Interpo* (Lines 7-11). Procedure *Probe* is shown in Algorithm 4. After computing the correlation value at the maximum peak point, it probes the points on its two sides in a geometric progression style. The probing stops when the boundary is met, which we have discussed above. As for the procedure *Interpo*, we use the linear interpolation [58] to connect the probed values and form the approximated interesting area. We then detect and update peak points according to the probed correlation values (Line 12), which can be implemented by an existing peak detection algorithm [12]. Finally, we decide the lead-lag relation based on the approximated interesting area (Lines 13-14).

The *UpdateCorrelation* algorithm enables us to track the interesting area using only $\mathcal{O}(\log m)$ correlation computations instead of $\mathcal{O}(m)$ that a brute-force approach requires. Moreover, since we start probing from the maximum peak points and stop probing when detecting the boundary, the actual number of correlation computations is much smaller. We further study the efficiency improvement of *UpdateCorrelation* in Section 2.3.3.

2.3.3. Evaluation

In this section, we design a comprehensive set of experiments to answer the following questions:

(1) What are the effects of the parameters (e.g., the length of the sliding window, the correlation threshold) on the performance of our algorithm in term of discovered leaders?

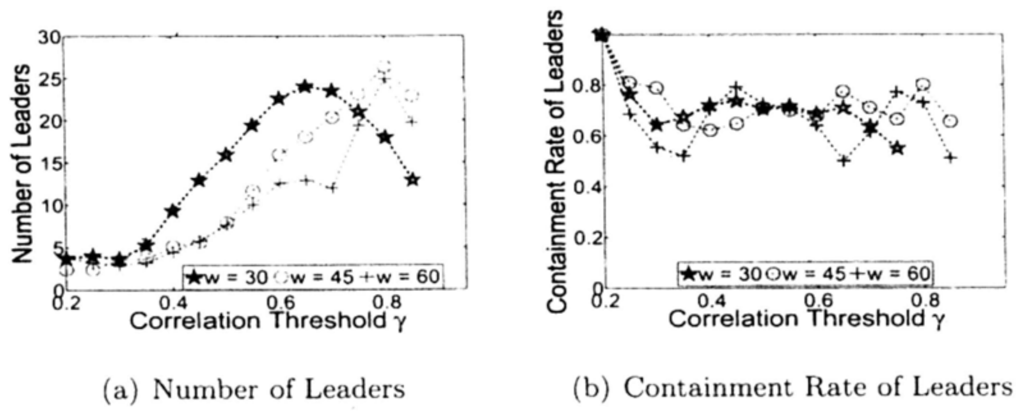
(2) How does the set of discovered leaders evolve as the sliding window moves forward? Does the set of leaders remain stable or evolve a lot with time?

(3) Whether the detected leaders really make sense and give insight to market modelling? How can we apply them to help the real investment?

(4) How much improvement we can gain by using *UpdateCorrelation*? What is the approximation accuracy? Does the accuracy degrade over time? How does *UpdateCorrelation* scale with the maximum lag?

We perform our experiments on a PC with a Pentium IV 3.4GHz CPU and 2GB RAM. We retrieve intraday transaction data for stocks from the NYSE Trade and Quote (TAQ) database[2]. We extract the tick data of stock prices

[2]The database covers all securities listed in the NYSE and American Stock Exchange (AMEX), as well as in Nasdaq National Market System (NMS) and SmallCap issues.



(a) Number of Leaders (b) Containment Rate of Leaders

Figure 2.10: Stock Leaders in S&P 500 when Varying Correlation Threshold

by computing the Volume Weighted Average Price (VWAP) for transactions at each tick as follows:

$$VWAP = \frac{NumberofShareBought \times SharePrice}{TotalShareBought}. \tag{2.24}$$

In this way, we obtain two real stock datasets:

- **DOW 30**. The selected stocks are 30 component stocks of the Dow Jones Industrial Average, which are the largest and most widely held public stocks in the United States. We compute their VWAP by setting one tick as 5 seconds.

- **S&P 500**. The selected stocks are component stocks of the S&P 500 index. These 500 stocks are Large-Cap corporations, which are heavily traded every day. We compute their VWAP by setting one tick as 1 minute.

Sensitivity of Parameters

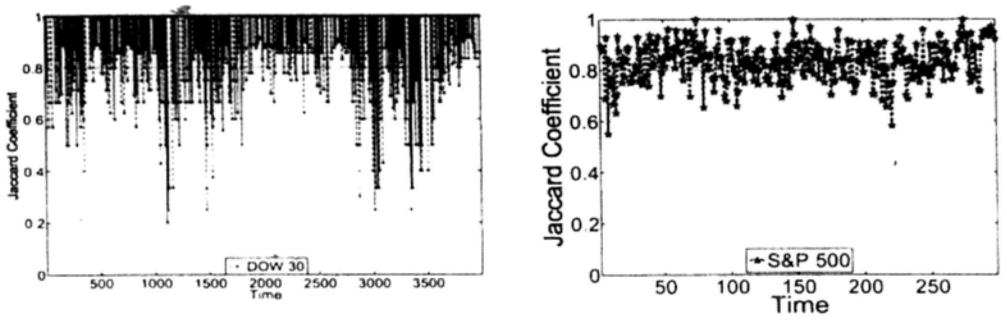
This set of experiments tests the sensitivity of parameters on the performance of our algorithm. As shown in Algorithm 1, there are three parameters: the length of the sliding window w, the correlation threshold γ and the maximum lag m. As recommended in [10], m is set to be $w/2$. Therefore, we only test different values of γ and w. We use the dataset S&P 500 and vary γ from 0.2 to 0.9 with a step of 0.05. We also test three values of window length: $w = 120$, $w = 180$ and $w = 240$.

Figure 2.10(a) presents the number of stock leaders we detect at each γ. For all values of w, we find a clear rise in the number of leaders when γ increases from 0.2 to 0.8. This is because the number of edges in the graph $\mathcal{G}$ decreases with the increase in γ. And as $\mathcal{G}$ becomes sparser, the stocks are less likely to be covered by the same leader, which results in more leaders. When γ exceeds 0.8, there is a drop in the number of stock leaders. This is because when γ is set too high, many stocks become isolated and are not led by any other stocks. Therefore, the number of leaders decreases when γ is high and reaches 0 when γ is set to be 1.

In order to study the evolution of the leaders when varying γ, we compute the containment rate of leaders between two consecutive γ values as $\frac{|Leaders(\gamma_i) \cap Leaders(\gamma_{i-1})|}{|Leaders(\gamma_{i-1})|}$. As shown in Figure 2.10(b), for all values of w, the containment rate for different values of γ remains to be high until γ reaches 0.8. This indicates that most of the leaders discovered at a low γ can also be discovered by a higher γ, as long as γ is not set to be too high. This gives us some hint for choosing γ: normally, γ can be set to be around 0.3 since it tends to select a small number of leaders. If users want to be more confident with the lead-lag relation, they can set γ higher and a higher γ also covers most of the results that are produced by the lower ones. From our experimental study, we found setting $\gamma = 0.3$ can generally comes out a good result.

Stability of Leaders Over Time A user may raise the following question: since the leaders are updated every time tick, can I trust the current detected leaders? This set of experiments studies the stability of detected leaders as time goes by. We adopt the Jaccard coefficient [72] to measure the similarity between the set of leaders extracted at two consecutive time ticks. Therefore, Jaccard coefficient is computed as $\frac{|Leaders(t_i) \cap Leaders(t_{i-1})|}{|Leaders(t_i) \cup Leaders(t_{i-1})|}$.

Figure 2.11 reports the result on the two datasets, DOW 30 and S&P 500. For DOW 30, we fix $w = 240$, $\gamma = 0.3$, $m = 120$ and extract leaders at 4000 consecutive time ticks in an entire trading day. As shown in Figure 2.11(a), there are 3095 time ticks when the leaders remain the same as those of the previous



(a) DOW 30 (b) S&P 500

Figure 2.11: Stability of Leaders

time tick (the similarity is 1) and the average similarity is 0.9536, which suggests that the detected leaders are quite stable for most of the time. This is because the interval between each consecutive time tick is only 5 seconds, which leads to little difference of the graph $\mathcal{G}$. However, since the size of graph is quite small, an altered edge related to the leaders may result in a significant change in the set of leaders. For example, the standard deviation of the similarity is 0.10 and at time tick 1113, the similarity is as low as 0.2.

On the other hand, for the 1-minute per tick data of S&P 500, we fix $w = 120$, $\gamma = 0.3$, $m = 60$ and extract leaders at 270 consecutive time ticks in an entire trading day. The result is shown in Figure 2.11(b) and we find that the similarity is quite stable (The standard deviation is 0.07 and the lowest similarity is 0.55). This is because the graph $\mathcal{G}$ of this dataset is large and a small number of altered edges is not likely to affect the whole PageRank vector π. However, since the interval between each consecutive time tick is 1 minute, it is more likely that there are several altered edges in the graph. As a result, the average similarity is 0.82, which is lower than that of DOW 30. In summary, this set of experiments suggests a certain degree of stability for the evolution of the leaders and we will further explore this nice property in the next section.

Leadership Index Vs. Market Index

In this section, we study the usefulness of the detected leaders in market investment. We construct a Leadership Index by the following procedure. I)

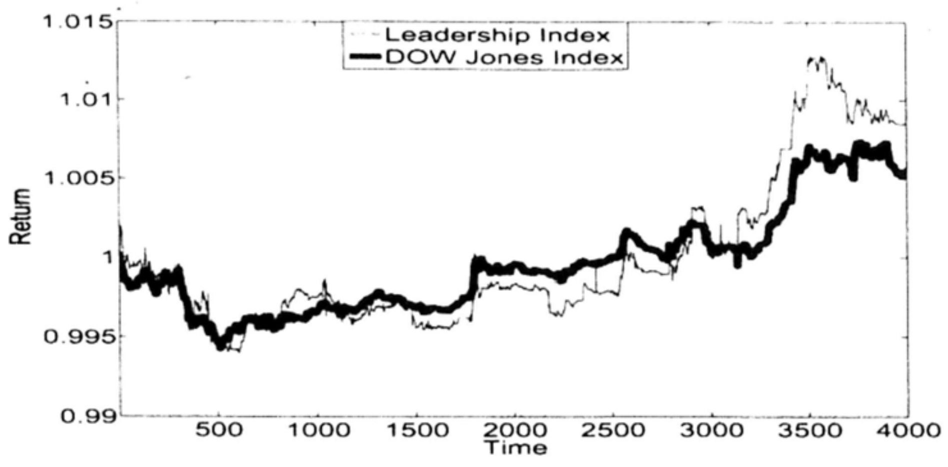


Figure 2.12: Leadership Index VS. DOW 30 Market Index

Initially at time $t_0 = w$, we form a portfolio using the extracted leaders. The weight β_i of each leader in the portfolio is decided by their relative leadership score, i.e., $\beta_i = \frac{\pi_i}{\sum_{j \in Leaders} \pi_j}$. The initial return of the portfolio $r_0 = 1$. II) At the following time tick t, we record the cumulative return of the portfolio r_t and update the portfolio using the newly detected leaders and their weights. The Leadership Index is then represented as a time series recording the cumulative return of the portfolio. We now present some numerical examples to demonstrate the application of our approach to portfolios of stocks in DOW 30 and S&P 500. While these examples by no means constitute a thorough numerical study, they are indeed representatives of a much larger set of experiments that we have run.

Figure 2.12 presents the Leadership Index using the leaders of DOW 30. We set $w = 240$, $\gamma = 0.3$ and extract an average of 6.5 leaders out of 30 stocks over the 4000 time ticks of a day. Besides, as a comparison, we also present the corresponding cumulative return of the DOW Jones Industry Average Index (DJI). As shown in Figure 2.12, Leadership Index formed by the small set of leaders vividly tracks the entire market index behavior (The cross correlation between the two indices is 0.949). This tracking ability of Leadership Index comes from the facts I) the detected leaders are the power source of the market movement and they can represent other stocks' behavior II) these leaders have a certain degree of stability so that holding them can track the portfolio of the entire market. This tracking property has mainly two usages. I) The financial analysts can easily detect the information source and evaluate the investment opportunity by studying these leaders. II) The money managers, who are always trying to make their funds beat or match market indices, can even form their portfolio using the leaders. The stability of leaders also suggests that they do not need to adjust the portfolio very often if the transaction cost is considered.

Figure 2.13 shows the Leadership Index we obtain on S&P 500. We set $w = 120$, $\gamma = 0.3$ and extract an average of 10.8 leaders out of 500 stocks in a trading day. In the comparison with the Standard & Poor 500 Index, we detect there are four phases for both indices: up, down, up and down. In the first rising phase, these two indices move together in a synchronous style. Then, at $t = 95$, the Leadership Index begins to go down first while the S&P 500 Index keeps rising and meets its first turning point until $t = 145$, 45 minutes later. After that, the Leadership Index rebounds at $t = 177$ with a first steady rise trend followed by a steep burst at $t = 209$. In contrast, S&P 500 Index starts the rise trend at $t = 197$ and meets the burst point at $t = 214$ which are both delayed with the Leadership Index. The final turning points for both indices is at $t = 220$ and they resume synchronous correlation until the end of the trading day. As a summary, in the first and fourth phases, the Leadership Index correlated with S&P 500 Index around zero lag point while in the second and third phase the Leadership Index lead S&P index and the lag decrease from 45 minutes in the beginning to 5 minutes in the end.

So the question arises naturally: why there are two types of tracking property, the zero-lag tracking property and the prediction tracking property? Why at some period, the prediction tracking property is stronger than the zero-lag tracking property but at some other periods in an opposite way. In order to answer these questions, we study again the shape of the *interesting area* and differentiate two types of waves, the zero-lag wave and non-zero-lag wave. The zero-lag-wave would be centered around zero lag point. And the two time series with the zero-lag wave would correlate closely with few time lag, i.e., they tend to

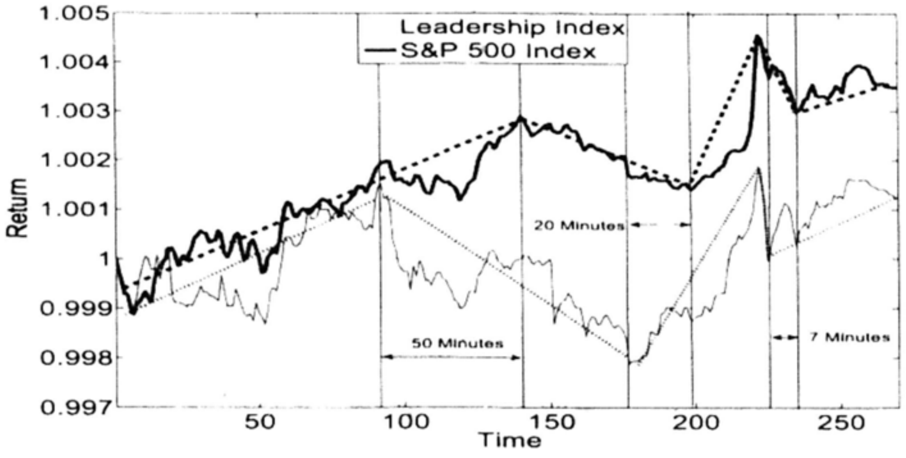
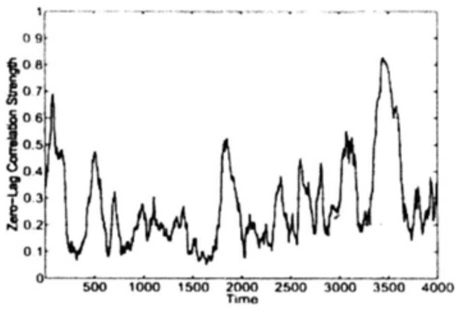
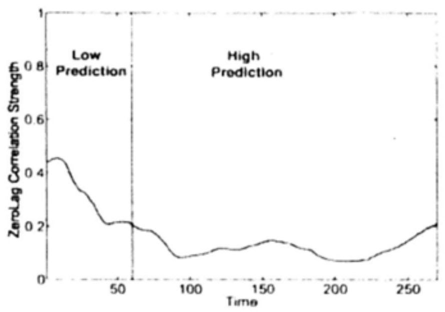


Figure 2.13: Leadership Index VS. S&P 500 Market Index



(a) DOW 30

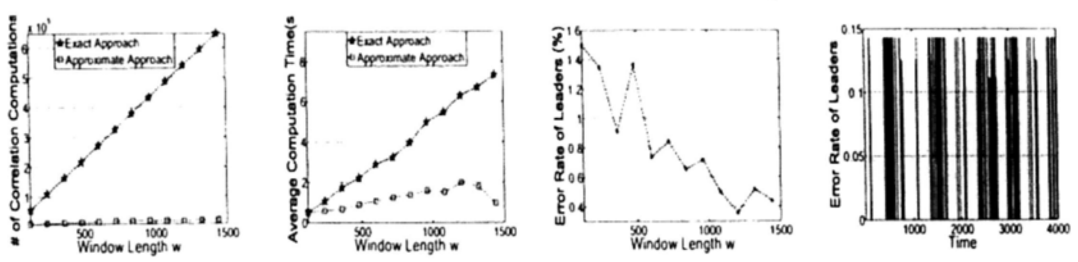


(b) S&P 500

Figure 2.14: Zero-Lag Correlation Strength

track with each other. We call this type of lead-lag relation zero-lag correlation.

We compute the strength of the zero-lag correlation on the total relations as $\frac{|E^{ij}(\rho^{ij}(0)>\epsilon,\rho>0)|}{|E^{ij}(\rho>0)|}$, where ϵ is set to be 0.8. The strength indicates how likely the Graph is contributed by the zero-lag correlation. Figure 2.14 (a) presents zero-lag correlation strength over time for DOW 30. And we can see the strength fluctuates and tends to be quite strong occasionally (reach 0.5) which implies the graph is largely contributed by zero-lag correlation. Therefore, the leaders detected in this graph tends to move closely with other time series and exhibits strong zero-lag tracking property and weak prediction tracking property. On the other hand, in figure 2.14 (b), from $t = 1$, we observe a strong but decreasing strength curve and it reaches 0.1 at $t = 95$ (The end of the first rise phase of



(a) Number of Correlation Computations (b) Running Time (c) Approximation Error Rate (d) Approximation Error Rate Over Time

Figure 2.15: Performance of Correlation Update

figure 2.14). It then stays very low until $t = 220$(the end of the third phase) and ris~~es~~ again afterwards. The evolution pattern of the strength coincides with the transition between the zeros-lag tracking and prediction tracking property well. Therefore, as a conclusion, when the zero-lag correlation strength is strong (i.e., when there are no significant leaders in the market) the Leadership index will have strong zero-lag tracking property while when it is weak (i.e., when there are significant leaders in the market), the Leadership index will show stronger prediction power over the market index.

Correlation Update

This set of experiments studies the effectiveness of the *UpdateCorrelation* algorithm, which approximates the lagged correlation in deciding the lead-lag relation between time series. We test on the dataset DOW 30 and vary w from 120 to 1440 (i.e., vary m from 60 to 720). For each w, we move forward the sliding window over that trading day and report the average result. We compare our approximate approach with the exact approach which computes the correlations for all lag values in a brute-force way.

Figure 2.15(a) reports the number of correlation computations. When $w = 120$, the exact approach needs around 54,000 correlation computations, while our approximate approach only needs 7571 computations. The number of correlation computations for the exact approach increases linearly with w, while our approximate approach grows very slowly with w. When $w = 1440$, our approximate approach only needs to compute 20,767 correlation values, which is over 30 times less than $648,000$ computations in the exact approach.

Figure 2.15(b) presents the average running time for the two approaches. The running time shares a similar trend to that of correlation computations presented in Figure 2.15(a). When $w = 1440$, the running time for our approximate approach is 0.94s, which is an order of magnitude faster than 9.3s for the exact approach. With the increase of w, the difference between the running time of the two approaches will continue to enlarge. Note that the fluctuation in the running time of our approximate approach is mainly due to the extra cost of conducting linear interpolations and updating peak values.

Figure 2.15(c) shows the accuracy of the approximation. The error rate is computed as the Jaccard distance between the two sets of leaders detected by the two approaches. We can see that the average error rate is less than 1.5% and decreases when w increases. In Figure 2.15(d), We also present the approximation error rate over time when we move forward the sliding window by setting $w = 360$, $\gamma = 0.3$. The result shows that the error does not continuously increase (always lower than 0.15) as time goes far away from the initial time tick, on which the exact interesting area is computed. This is simply because our approximate approach refines peak values at each time tick, so that the approximation at the following time ticks does not accumulate previous errors. This justifies the effectiveness of our correlation update approach in achieving good approximation accuracy.

2.3.4. Summary

In this section, we formalize a novel problem of discovering leaders from multiple correlated time series based on real time lagged correlation. A time series is identified as a leader if its movement triggers the co-movement of many other time series. Since the lagged correlation computation is very costly in a stream environment, we develop an effective update approach that is able to significantly reduce the number of correlation computations as well as to retain good accuracy. We typically study the leadership discovery problem in the financial domain and attempt to find leaders in the stock market. The experiments on real stock price data show that the discovered leaders demonstrate high tracking and predictive power in the market. Our experiments also show that the approximate update approach of correlations is up to an order of magnitude faster than the exact approach at a low approximation error rate.

2.4. Chapter summary: multiple time series co-movement

Co-movement patterns of time series stream have a variety of applications in financial domain, including portfolio management, risk analysis and high frequency trading. The key questions are the following: How to detect co-movement patterns from evolving time series streams? How to update them efficiently? How to use them to assist financial applications.

We introduced two techniques that exploit two aspects of co-movement pattern. First, we presented co-movement model which capture the dependency relationship of two time series. We explored this model to compute the expected return and risk for the portfolio. Second, we studied the lead-lag effect in financial market and proposed an algorithm which can detect leaders from multiple time series stream efficiently. Both of technique exhibits sound result in real financial data.

Next, we will present techniques which integrate news information into pattern discover.

CHAPTER 3

Pattern Discovery on Time Series and News Stream

Time-series analysis always plays a significant role in the financial markets, where predictions (the decision making processes) are made based solely on the historical movements of the stock prices (time-series). However, there are many factors that do not explicitly exhibit in the time-series but have large impact on the movements of these time series. As an evidence, in the stock market, the price changes are the consequences of the actions taken by the investors. Investors' actions, although occasionally irrational, are predominantly understandable and rational with respect to the social structure, social organization, perceptions and collective beliefs of this complex arena [1, 7, 21, 45]. A key issue is whether it is possible to interpret the time series movement better by using some additional, widely available and actionable information, together with the time-series data.

In this section, we try to answer this question by introducing three techniques to discover the relationship between the time series stream and news stream. Specifically, we first present some background and related work in Section 3.1. Then, we present an approach to model and predict the stock market process by using a Non-homogeneous Hidden Markov Model (NHMM) in section 3.2. After that, we present our technique for volatility prediction by utilizing both time series data (stock prices) and textual information (news articles)in section 3.3. Finally, we present the algorithm which can detect the priming events which pose great impact on time series movement in Section 3.4 We summarize this chapter in Section 3.5.

3.1. Time Series and News Stream Related work

In this section, we present some related work on mining news stream and time series stream.

3.1.1. News Sensitive Stock Prediction

The first systematic examination against the impacts of textual information on the financial markets is conducted in [46], which compares the movements of Dow Jones Industrial Average with general news during the period from 1966 to 1972. [20] formulates an *activity monitoring task* for predicting the stock price movements, which issues alarms based on the content of the news articles. [73] integrates the textual information into trading rules, where for the textual data, a maximum entropy text classification approach [61] is used for classifying the impacts of the posted messages on the stock prices.

[83] develops an online system for predicting the opening prices of five stock indices, where by combining the weights of the keywords from news articles and the historical closing prices of a particular index, some probabilistic rules are generated using the approach proposed by [77]. [49] proposes a system for predicting the intra-day stock price movements by analyzing the contents of the real-time news articles based on a language modeling approach, which is in turn proposed by [65]. [22] and [24] propose a model for mining the impact of news stories on the stock prices by using a t-test based split and merge segmentation algorithm for time series preprocessing and SVM [39] for impact classification. [59] uses a hand-made thesaurus to forecast intraday stock price trends from information contained in press releases and report a very nice result.

3.1.2. Bursty Features

As discussed, we consider the news articles as an important data source that have a significant impact on the investors' behaviors, which in turn will affect the stock price changes. And we observe that the emergence of an important event is caused by a burst of features (keywords): some features "suddenly appear frequently" when the event emerges, and their frequencies drop when the event fade away. By monitoring the distribution changes of the features in the news articles, we can identify whether there is any new event occurred. Specifically, if the feature, $f \in F$, suddenly appears frequently in some time windows (e.g. a day), w, we consider that an event emerges. Below, we discuss how to handle the issue of "suddenly appearing frequently" as the problem of bursty feature identification.

A bursty feature is a keyword that appears at an abnormally high rate in a bounded time interval (bursty period). It is important to note that identifying the bursty periods is important because we need to know which periods the bursty features will have impact on the stock price changes.

The identification process of the bursty feature is similar to that of [25]. It presents a probability model to capture the probabilistic distribution of burst events. Here, the probability of bursty of a feature, $P(w, f; p_e)$, is computed by:

$$P(w, f; p_e) = \sum_{k=1}^{n_{f_w}} p(k; N_w, p_e) \tag{3.1}$$

$$P(k; N_w, p_e) = \binom{N_w}{k} p_e^k (1 - p_e)^{N_w - k} \tag{3.2}$$

where p_e is the probability that f appears in a time window given that it is not bursty, N_w is the total count of the features that appear in w, and n_{f_w} is frequency of f appears in w. $P(w, f; p_e)$ is the cumulative distribution function of the binomial distribution, and $P(k; N_w, p_e)$ is the corresponding probability mass function.

3.1.3. Markov Model

There are also a lot of work [26, 43] which use hidden markov model to model interest time-series. [26] explores the segmental hidden markov model to model the pattern of time series and each state is responsible for the generation of a segment of overall time series. In [26], the Viterbi-like algorithm is used to recognize a specific pattern from a new time series. It also extends the model as an online algorithm for detecting the ending point. [26] focuses on applying the segmental HMM model for pattern matching and does not consider the influence from information of environment.

[36] proposes the Non-homogeneous hidden markov model for relating precipitation occurrence at multiple rain-gauge stations to broad scale atmospheric circulation patterns. [44] describes how to capture temporal and multivariate dependencies in the multivariate time-series data, with a review and comparison on HMM and NHMM. These works just present a theoretical framework for statistical modeling, but do not address how to construct and control the model on complex information environments, like the stock markets.

3.1.4. Volatility

In this thesis, we not only look at the rise/fall of time series, but also we study the change of volatility of time series [67]. Volatility is the standard deviation of the continuously compounded returns of a stock within a specific time horizon and is used to measure how widely prices are dispersed from the average as follows:

$$\sigma = \sqrt{\sum_{i=1}^{n} [R_i - E(R_i)]^2 P_i} \tag{3.3}$$

where R_i is the possible rate of return, $E(R_i)$ is the expected rate of return, and P_i is the probability of R_i.

The volatility of time series index is influenced by the news event. For example, [67] discovered a relationship between the news and abnormal stock prices behavior. But it focused more on how to detect these influential news using text categorization.

3.1.5. Event Detection

The problem of Topic Detection and Tracking (TDT) [2] is a classical research problem for many years. The first stream of work used graphical probabilistic models to capture the generation process of document stream [6, 29]. [76] extended the LDA model and incorporated location information. [75] analyzed multiple coordinated text streams and detected correlated bursty topics. [57] added the social network information as a regulation into the topic detection framework. On the other hand, there are another stream of work which detected topic based on the bursty features [47]. [25] detected bursty features and clustered them into bursty events. [23] further built an event hierarchy based on the bursty features. [32] analyzed the characteristics of bursty features (power and periodicity) and detected various types of events based on it. [52] proposed an algorithm to track short phrased and organized them into different news threads. Although these work can detect topics and track event efficiently. They can not tell the users which topics/events are changing the real world's interested time series, President Approval Rating, Stock Market Index.

3.2. Integrating Multiple Data Sources for Stock Prediction

In our work, instead of using news articles directly for prediction, we identify the associated events by selecting a representative set of bursty features (keywords) that have impacts on a single stock. We present a non-homogeneous Markov model (NHMM) that can identify the hidden market states from the sequences of observed stock price segments. One of the distinctive features of this model is that the observed stock price segments are influenced by a stream of associated event states that are computed from the burst of selected keywords in news articles.

The main contributions of this work are summarized below. First, we propose a modeling approach based on non-homogeneous Markov model (NHMM) by taking multiple data sources into consideration for price prediction in stock market. Here, the multiple data sources are stock prices (time-series data) and news articles (textual data). These two data sources are different in nature. Second, we study how to deal with the influences of news articles on the stock price changes in NHMM. For a specific stock, we identify the associate events in the news articles that will trigger different kinds of market states. We will discuss how we determine the associate events by using bursty features, as well as how the market states could be identified based on the associated events. Third, we conduct preliminary experiments using real datasets, which confirms the effectiveness of our proposed approach.

In the following sections, we discuss a non-homogeneous Markov model and use it to model stock market in Section 3.2.1. In Section 3.2.2, we discuss our event-driven approach with the focus on how to use the bursty features obtained from the news articles together with the stock prices in NHMM. Section 3.2.3 evaluates our model. Section 3.2.4 concludes this work.

3.2.1. Modeling Stock Market

The overview of our approach is shown in Figure 3.1. On the left hand side, it illustrates that stock prices are associated with a set of relevant news articles where an event (represented by a set of bursty features(keywords) extracted from news articles) may trigger a change of the stock price immediately. Note: each news article is indexed by a timestamp. On the right hand side, it outlines a model which considers the relationships between stock price changes and events occur. In brief, we identify the hidden market states, S, that will generate a sequence of observed stock segments (short-term trends), $\mathcal{O}$, meanwhile it (S) will be affected by a sequence of associated event states, X_o. X_o consists of

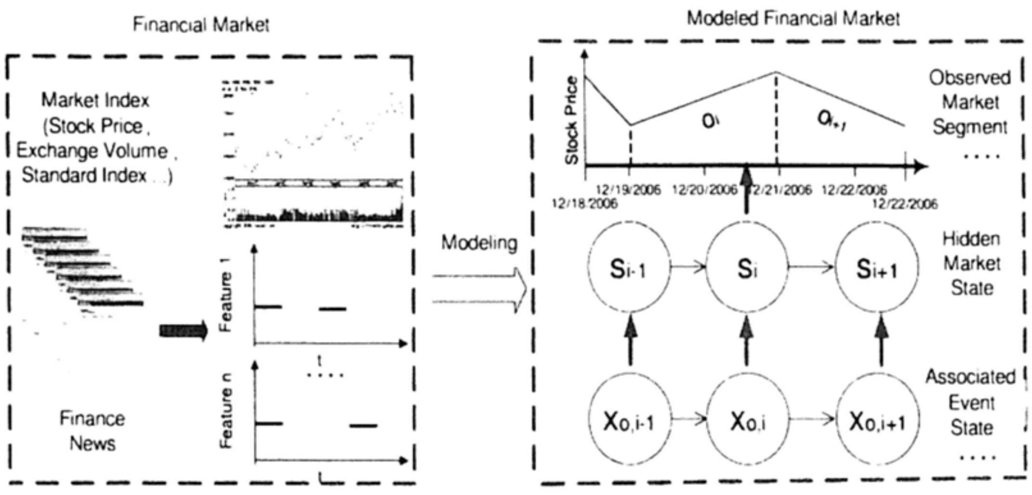


Figure 3.1: Modeling Financial Market

the bursty features in the finance news corpus that are significant related to the stock prices change. On the top of the right hand side of Figure 3.1, it shows two observed trends, o_i and o_{i+1}. They are generated by the hidden market states , s_i and s_{i+1}, respectively (in the middle). The transition from s_i to s_{i+1}, which is represented by the solid arrow, is influenced by the associated event state, $x_{o,i+1}$, which is shown at the bottom of the figure.

The reason why we adopt the Non-Homogeneous Hidden Markov model (NHMM) [36] in this work is given below. It is widely agreed that market state sequence, $S = \{s_0, s_1, \ldots, s_n\}$, is a kind of stochastic process, where it can be represented by a discrete Markov chain. Each state, s_i, in S denotes the situation of the market at a certain period of time.In S, the state s_{i+1} depends only on the state of s_i and the corresponding associated event state, $x_{o,i+1}$, but does not depend on the previous status, s_j for $j < i$. This memoryless characteristic of S is based on the Efficient Market Hypothesis (EMH) [8]. EMH states that the current market state must reflect all of the current market information and the change of state must satisfy the properties of random walk. Under EMH, that state s_i will reflect all of the information in the previous state but the transition behavior of s_i to s_{i+1} will not be influenced by any of the previous market state.

We will present the mathematical details of the modeling process in next part.

Probabilistic Modeling

NHMM defines a joint distribution of the observed states and the hidden states, where the observed states, o_i, are also rely on the influence variables, x_i. The following assumptions hold for NHMM:

$$P(o_i|s_{1:i}, o_{i-1}, x_{o,1:i}) = P(o_i|s_i), \tag{3.4}$$

$$P(s_i|s_{1:i-1}, x_{o,1:i}) = \begin{cases} P(s_i|s_{i-1}, x_{o,i}) & i \geq 2, \\ P(s_1|x_{o,1}) & i = 1. \end{cases} \tag{3.5}$$

where $s_{1:i}$ and $x_{o,1:i}$ denote the sequence of hidden states and the sequence of the corresponding associated event states, respectively. Eq. (3.4) formalizes the generation process that observed state, o_i, will only depend on the current hidden state, s_i, and is independent with the previous observed states, o_{i-1}, and also the associated event states, $x_{o,1:i}$. Eq. (3.5) addresses the memoryless characteristic of the discrete state chain, S. The current state, s_i, is independent of the hidden state sequence that is the previous of s_{i-1}, as well as the corresponding previous associated event states, $x_{1:i-1}$. Based on these two equations, we will specify the model by parameterizing $P(o_i|s_i)$ and $P(s_i|s_{i-1}, x_{o,i})$ in the following sections.

The observed variable in our scenario is the trends of stock prices. Each trend is characterized by two elements: (1) time (duration of its occurrence); and (2) slope. In order to model these information, we represent each segment $o_i \in \mathcal{O}$ as a vector (θ_i, d_i) where θ_i is the slope of the trend that is generated by the regression line against the original data points in that specific time period on the time series, and d_i is its duration. Then, we combine the semi-Markov models [26, 35] into our model to allow arbitrary state durations. In this approach, the slope distributions and the state duration distributions are both set to be a Gaussian distribution. Since the slope θ_i and d_i are independent, we have:

$$P(o_i|s_i) = P(\theta_i|s_i)P(d_i|s_i) \tag{3.6}$$

We employ the multinomial logistic regression to parameterize the hidden state transition $P(s_i|s_{i-1}, x_{o,i})$. Multinomial logistic regression is a widely used framework in the speech and language processing area. It is based on probabilistic machine learning, also known as Maximum Entropy, that usually used to tackle the classification problems which involved multiple classes [40]. The parameterized state transition probability is described as follows:

$$P(s_i = n|s_{i-1} = m, x_{o,i}) = \frac{\exp(\sigma_{mn} + \rho_n x_{o,i})}{\sum_{k=1}^{K} \exp(\sigma_{mk} + \rho_k x_{o,i})} \tag{3.7}$$

$$P(s_1 = n|x_{o,1}) = \frac{\exp(\lambda_n + \rho_n x_{o,1})}{\sum_{k=1}^{K} \exp(\lambda_k + \rho_k x_{o,1})} \tag{3.8}$$

Here, K is the number of hidden Markov states, σ is the baseline Markov state transition matrix, ρ is the corresponding weight vector for each $x_{o,i}$, and λ is the constant first-state probability vector. The denominator in the summation form is the normalization factor, so as to ensure that the value of these two equations is from 0 to 1.

With the parameterized equations for the hidden Markov state transition probability, we can estimate the parameters λ_i, σ_{ji} and ρ_i by using the classical EM (expectation-maximization) algorithm, which is described in [44].

Classification and Prediction

Online stock trend classification and prediction is given below based on NHMM, where the stream data of stock price, and news articles are archived continuously.

Let $y = \langle y_1, y_2, \cdots, y_t \rangle$ be a sequence of stock prices (a single time series), where y_i, for $1 \leq i \leq t$, is a stock price (a point on the time series), with y_1 as the first point of current market state and y_t as the newest archived point. The prediction is conducted only when y_{t-1} is identified as the end point of the current market state (This approach will be discussed in more details in Section 3.2.2.), and y_t is the starting point of next market state. The process is given in Algorithm 5. The function *create_segment()* approximates $y_1, y_2 \ldots y_{t-1}$ as a new segment o_i (line 2). It computes the associated event state, $x_{o,i+1}$, from the news articles (line 3). We will discuss this issue in Section 3.2.2 (Eq. (3.19)).

Algorithm 5 Stock Trend Classification and Prediction

```
INPUT: coming Stream $\langle y_1, y_2, \cdots, y_t \rangle$
1: if $y_{t-1}$ is detected as the end point of current segment then
2:     $o_i = create\_segment(y_1 y_2 ... y_{t-1})$
3:     Compute $x_{o,i+1}$ as the current associated event state (Eq. 3.19)
4:     $s_i = MLS(o_i)$
5:     for each state $k \in K$ do
6:         compute $p(s_{i+1} = k | s_i, x_{o,i+1})$
7:     end for
8:     $s_{i+1} = \arg\max_k p(s_{i+1} = k | s_i, x_{o,i+1})$
9:     return $s_{i+1}$
10: end if
```

In line 4, for s_i, it conducts classification by calling $MLS()$ to obtain the Most Likely State (MLS), which is given in Algorithm 6. Let us discuss Algorithm 6 below.

After the classification phase of Algorithm 5, in line 5-9, the state transition probability for each possible state will be calculated, and the one with the maximum transition probability will be returned as the predicted state. Users can therefore issue buy/sell decision according to the parameters of the predicted state, s_{i+1}, which is the mean values of the slope and the duration of the corresponding trend.

In Algorithm 6, α_k represents the probability that the NHMM model, which has generated the first i observed segments of O, is in hidden state k at step i. And each computation of $\alpha_k(i)$ will use the value of $\alpha_r(i-1), \forall r \in [1 : K]$. $\alpha_k(1)$ is computed by the following equation:

$$\alpha_k(1) = p(o_1 | s_1 = k) p(s_1 = k | x_{o,1}) \tag{3.9}$$

Algorithm 6 takes the state with the highest value of $\alpha_k(i)$ as the most likely state of s_i.

Algorithm 6 MLS

```
INPUT: Detected segment $o_i$
1: for each state $k \in K$ do
2:     $\alpha_k(i) = p(o_i|s_i = k) \sum_{r=1}^{K} \alpha_r(i-1) p(s_i = k|s_{i-1} = r, x_i)$
3: end for
4: $s_i = \arg\max_k \alpha_k(i)$
5: return $s_i$
```

3.2.2. An Event-Driven Approach

The NHMM was discussed in the previous section. However, there are two problems left unsolved: (I) the computation of the associated event state $x_{o,i}$ and (II) the detection of the end point of current market state. In this section, we will concentrate on identifying associated events to solve these two problems. There are several issues. First, after detecting bursty features according to Section 3.1.2, we discuss how to evaluate the influences of the feature on a specific time t. Second, we discuss how to identify the associated events and compute their influence on stock price changes. Finally, we discuss how to detect NHMM states, which is strongly related to the data segmentation, or in other words, the early determination of the end point of the segments.

Feature Influence Identification Given a stream of news articles, we first identify bursty feature rate according to Section 3.1.2. In order to discretize the bursty rate time-series into M-State transition sequence, so as to be used in our NHMM, we discretize the probability, $P(w, f; p_e)$, into M states, where $M = \{0, 1, \ldots, M-1\}$. Hence, we need $M-1$ thresholds between 0 and 1. Each state indicates its burst intensity. The higher the value, the more strong is the bursty intensity. In the following discussions, we explain our approach using $M = 2$ which identifies two states for the burst time-series, the bursty state and non-bursty state.

Given all the bursty state sequences of features $f \in F$ (F is the whole feature

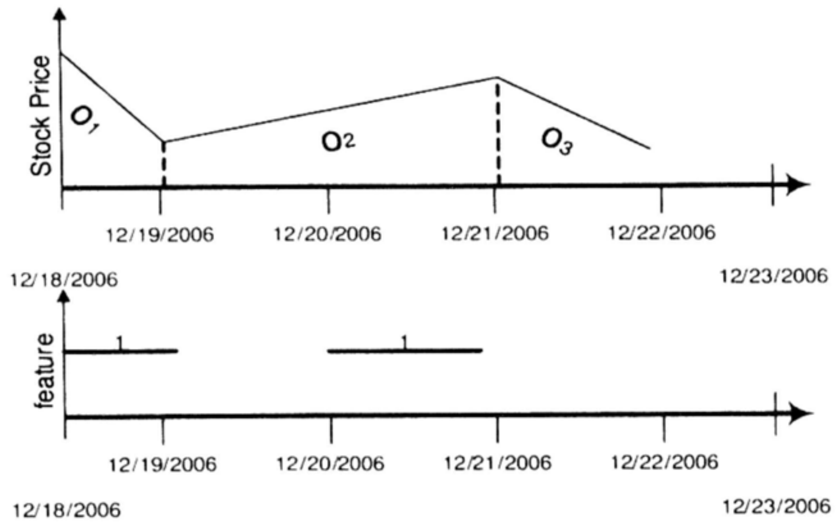


Figure 3.2: Feature State Influence

set) that are obtained as discussed in previous section, we need to identify the associated event influence vector x_t at time t. Here, we use "influence rate", $C(f_j, t)$, to measure the influence of a feature state at time t, and $C(f, t)$ to measure the total influence rate of f for this stock at time t.

Figure 3.2 gives an example of how a feature influences segments of stock price, where the stock price is split into three segments o_1, o_2 and o_3. As shown in Figure 3.2, o_1 starts from 12/18/2006(MM/DD/YYYY) and ends at 12/19/2006, o_2 starts from 12/19/2006 and ends at 12/21/2006, o_3 starts from 12/21/2006 and ends at 12/22/2006. The feature state sequence f exhibits two bursts in the time window, [12/18/2006, 12/19/2006] and [12/20/2006, 12/21/2006], respectively.

Let f_j denote a bursty state of feature f and $[b_{f_j}, e_{f_j}]$ denote the time window of f_j. We define $C(f_j, t)$ as follows:

$$C(f_j, t) = \begin{cases} 1 & \text{if } b_{f_j} \le t < e_{f_j} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

It is based on the Efficient Market Hypothesis (EMH), and many existing works ([48], etc) built their prediction systems based on EMH. In Figure 3.2, with Eq. (3.10), for the two bursty states of feature f, only the state for the time window [12/18/2006, 12/19/2006] is correlated with o_1 and the influence rate is

1.

Weighted Feature Influence Identification: Unfortunately, Eq. (3.10) is often violated in real market situations. In Figure 3.2, the bursty state in the time window [12/20/2006, 12/21/2006] may have influence on o_3. This time lag is due to the fact that the real world market may not be so efficient as described in the Efficient Market Hypothesis. The influence rate of f_j may fade as the time distance between e_{f_j} and t becomes larger. Based on this observation, Eq. (3.11) formalizes the weighted influence rate.

$$C(f_j, t) = \begin{cases} \exp^{-\sigma(t-e_{f_j})} & \text{if } e_{f_j} \leq t \\ 1 & \text{if } b_{f_j} \leq t < e_{f_j} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

Here, the σ indicates the fading intensity of influence rate of the bursty state. The total influence rate of feature f at time t can be computed as below:

$$C(f, t) = \sum_{j=1}^{|f|} C(f_j, t) \tag{3.12}$$

where $|f|$ equals to the number of states in the state sequence of feature f.

Associated Events Identification

Let us assume that the influence rate of features $f \in F$ at time t is identified. Let O be a sequence of stock price segments generated by any segmentation algorithms [42]. In this section, we describe how we identify the events and features that are related to stock price segments O. For the events that are highly related to O, we call them associated events X_o. Similarly, for the features that are highly related to O, we call them associated features f_o.

As stated in above, the emergence of events contains a set of bursty features, therefore, in order to identify X_o, we need to identify a set of $f_o \in F$ which is formally defined as below:

Definition 1. X_o are associated events to O if they contains a set of associated features f_o which are significantly related to O.

In definition 1, one problem is that we have to carefully define the phrase "significantly related to". Given a feature $f \in F$ and a sequence of stock price segments O, we determine whether f is significantly related to O by two measures: *inter-correlation*(C_R) and *intra-correlation*(C_A). We first present how they are computed, then explain why they are computed in this way. For the *inter-correlation*(C_R):

$$C_R(f, O) = \frac{1}{|O|} \sum_{i=1}^{|O|} C(f, b_{o_i}) \tag{3.13}$$

where $C(f, b_{o_i})$, as defined in Eq.3.12, is the influence rate of f on the beginning of the segment o_i, or in other words, the change point between o_{i-1} and o_i. Eq.3.13 computes the average influence of feature f on the change of stock price. This is why C_R is termed as *inter-correlation*. For the *intra-correlation*(C_A), with the similar concept to information gain [70], we divide the stock price segments O into three classes (up, down, steady) using the trend labelling approach we proposed in [22]. We also identify two classes of feature states (burst or not), then $C_A(f, O)$ can be computed by:

$$C_A(f, O) = E(O) - [p(f)E(O, f) + p(\bar{f})E(O, \bar{f})] \tag{3.14}$$

where $p(f)$, $p(\bar{f})$ are the probability of the feature burst or not and $E(O)$ is the entropy [72] of the segment set that can be computed as below:

$$E(O) = -\sum_{i=1}^{3} p(O^i) \log p(O^i) \tag{3.15}$$

where O^i contains all the segments belonging to one of the three classes(up, down and steady) and $p(O^i)$ is its proportion. And $E(O, f)$ and $E(O, \bar{f})$ are the entropies of stock segments set classified with f and with $\bar{f}$ respectively and $E(O, f)$ is computed as below

$$E(O, f) = -\sum_{i=1}^{3} p(O^i, f) \log p(O^i, f) \tag{3.16}$$

$$p(O^{\imath}, f) = \frac{\sum_{o_j \in O^{\imath}} C(f, b_{o_j})}{\sum_{o_j \in O} C(f, b_{o_j})} \tag{3.17}$$

and $p(O^{\imath}, \bar{f})$ can be computed likewise. Eq.3.14 computes the information gain from f which measures the significance of the feature f with respect to classifying different classes of stock price segments O. This is why C_A is termed as *intra-correlation.*

Intuitively, the feature f that is significant related to O should have both high *inter-correlation*, $C_R(O, f)$ and *intra-correlation*, $C_A(O, f)$. However, simply adding or multiplying them together is not appropriate because the two measures evaluate the correlation between the feature and the change of stock price from different aspects. To capture these ideas, we first identify two subsets of F, $F_{R,O}$ and $F_{A,O}$, which contains the top M and N features ranked by $C_R(O, f)$ and $C_A(O, f)$ respectively. Then the associated events X_o are formed by a set of associated features f_o as:

$$X_o = F_{R,O} \cap F_{A,O} \tag{3.18}$$

where the intersection part of $F_{R,O}$ and $F_{A,O}$ contains the features that have both relatively high *inter-correlation* and *intra-correlation.*

Therefore, the associated event influence vector $x_{o,t}$ at time t can be formed by the influence rate of all $f \in X_o$ at time t as

$$x_{o,t} = \{C(f_o^1, t), C(f_o^2, t), ..., C(f_o^{|X_o|}, t)\} \tag{3.19}$$

where f_o^i is the i-th feature in X_o. Finally, the associated event state $x_{o,i}$ for o_i equals to $x_{o,b_{o_i}}$, where b_{o_i} is the beginning time of segment o_i. As mentioned in Section 3.2.1, associated event state $x_{o,i}$ would be computed for prediction and in next section, we would explore associated event influence vector $x_{o,t}$ to identify the state of stock market.

Market State Identification

Recall, given a sequence of stock price points, $\langle y_1, y_2, \ldots, y_t \rangle$, the prediction is conducted when y_{t-1} is identified as the end point of the current market state and y_t is the starting point of the next market state. Below, we discuss how to to detect the end point of each state when constructing a NHMM. This issue is related to data segmentation. We first examine a traditional segmentation approach which only considers information from single data source (the stock prices only). Then, we propose a new segmentation approach that considers information from multiple sources.

The single-source segmentation (SS) is based on the sliding window segmentation algorithm [42]. Given a stream of stock price data $< y_1, y_2, ... >$, the algorithm try to extend current market state by approximating the data from y_1 to the right. At some point y_t, if the approximation error ϵ exceeds the max error threshold, then an end point y_{t-1} is detected as the end point of current market state and the subsequence from the start point y_1 to y_{t-1} is transformed into a segment, which is the linear interpolation of $\langle y_1, y_2, \cdots, y_{t-1}\rangle$. Then the algorithm continues to search for a new potential segment starting from y_t.

SS is able to identify the end points of the hidden market states by detecting the changes of the shapes of stock price streams. However, it does not consider the information from other sources, like the bursty features obtained from news articles. Recall: in our NHMM, the bursty features influence will lead to the transition of market state by using influence event vectors. In order to detect the end points of hidden states by monitoring the change of associated event influence vectors, we propose a multiple-source segmentation approach.

The new multiple-source segmentation (MS) works similar as SS, but it will declare an end point of segment when either the following two condition happens.

1. The approximation error of $\langle y_1, y_2, \cdots, y_{t-1}\rangle$ is larger than the max error threshold.

2. The similarity of associated event influence vector between the events at the time of y_1 and at the time of y_t is smaller than a minimum similarity threshold.

Let x_{o,t_1} and x_{o,t_2} be two events. We measure the event similarity by the cosine

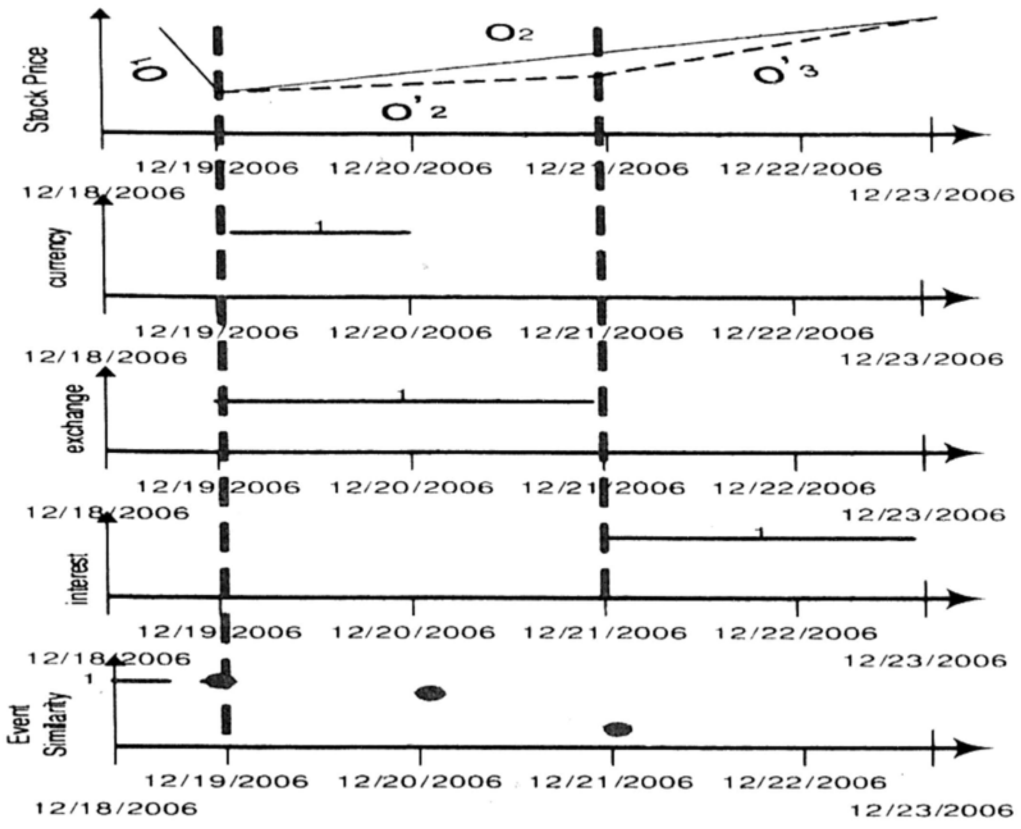


Figure 3.3: State Changes

similarity. The idea behind is that an influence event will trigger the burst of a set of features and two different events should lead to burst of different features. If the bursty features between two time points are significantly different, it implies that there is emergence of two different events.

An example shown the difference between these two approaches SS and MS is in Figure 3.3. In Figure 3.3, the top figure show observed stock-segments, namely, o_1 and o_2, where o_1 is a down-trend (from 12/18/2006 to 12/19/2006) and o_2 is an up-trend (from 12/19/2006 to 12/21/2006). The SS approach will model it using two states s_1 and s_2 to generate the two observed stock-segments. Note: in Figure 3.3, the next three figures show three bursty features: `currency`, `exchange`, and `interest`. The bursty feature `currentcy` is from 12/19/2006 to 12/20/2006, the bursty feature `exchange` is from 12/19/2006 to 12/21/2006, and the bursty feature `interest` is from 12/21/2006 to 12/22/2006. The MS also considers

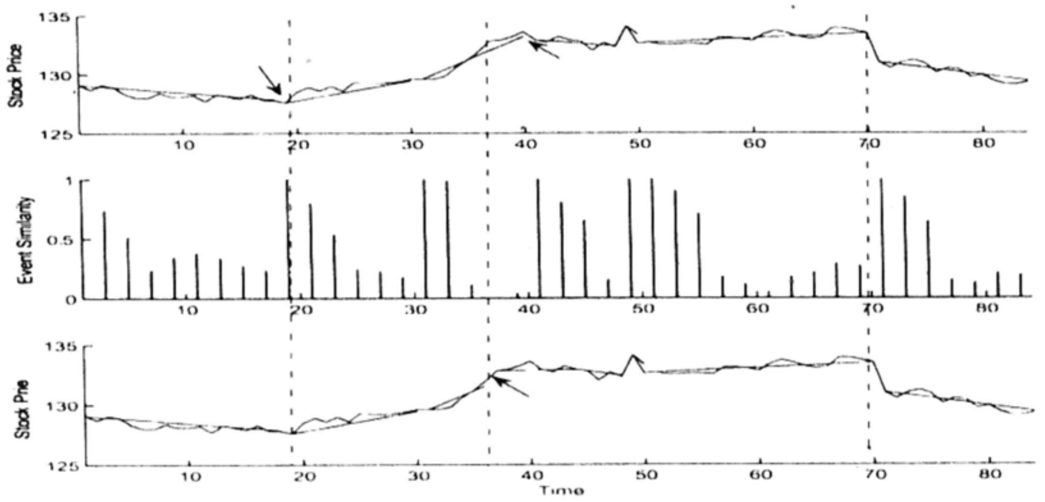


Figure 3.4: SS v.s. MS

the impacts of bursty features, and detects that there is a change among the bursty features from **currency** and **exchange** to **interest**. The change occurs at 12/21/2006. Based on both the observed stock-segments and bursty features, MS models it using three states for the stock-segments o_1, o'_2 and o'_3, which are shown as dotted lines in the top figure of Figure 3.3. Intuitively, it suggests that the observed stock-segment o_2 is composed of two different factors (o'_2 and o'_3). The bursty feature **interest** contributes significantly to the continuous observed stock-segment and should be considered as a new state in the modeling. In the last axis of Figure 3.3, we show the event similarity at the beginning of the segment o_2 is 1 and then decline sharply on 12/21/2006. Since the event similarity is smaller than the minimum event similarity threshold, MS declare that day is the end point of o'_2.

One advantage of MS is that it can detect changes in an early stage. Figure 3.4 illustrates an example. The top panel shows the stock price of the stock HSBC from 2/1/2006 to 3/30/2006, where SS identifies seven segments by monitoring the changes of the stock price. In the middle panel, we track the associated event similarity between the event at the start of the current segment and that at the current time. It is worth noting that, in the beginning of the segment, the similarity is rather high and then drops steadily over the time. The bottom panel shows how MS works. At most of time, it works exactly as SS does, and identifies the change of stock trend. It also utilizes the information provided by event similarity. It is noted that from time tick 30, the stock price starts a significant rise trend, and SS identifies it until the time 42, when a drop trend begins. On the other hand, as we monitor the event similarity, starting from time 30, it drops dramatically at time 34 and even reaches 0 at time 36, which indicates the emergence of a new event. MS notes this change and declares a new market state at time 36, in an much earlier stage than SS does. With information from multiple sources, MS is more sensitive to the changes in a stock market. We will illustrate its benefit in our experiment studies.

3.2.3. Evaluation

We evaluate our proposed approach by conducting a market simulation based on two different measurements: the accuracy of the predictions and the accumulated profits/loss according to a buy-and-hold test.

Experiment Setup

We conducted the following experiments to compare with our proposed approach:

1. **SS-based Prediction**: We detect the end of point of the segment by SS and use the mean slope of the predicted hidden state s_{i+1} as our predicted trend slope.

2. **MS-based Prediction**: We detect the end point of the segment by MS and use the mean slope of the most likely state s_{i+1} as the prediction trend slope.

3. **Baseline Prediction**: We detect the end point of the segment by SS, but the trend prediction is determined solely on the slope formed by the last point of the previous segment and the first point of upcoming segment.

Sectors	Stock (Highest rise)	Stock (Highest drop)	Stock (Largest fluctuation)	Stock (Largest volume)
Properties	0083	0016	0083	0001
Utilities	0002	0006	0002	0006
Commerce	0941	0008	0883	0941
Finance	0388	0011	2388	0005

Table 3.1: Selected Stocks(Code)

4. **HMM Prediction**: We detect the end point of the segment by SS, but the trend prediction is based on HMM only, without using any of the news articles. This approach is used to illustrate how much the context-sensitivity(news article) improve the result per se.

The stock prices (opening and closing) and the news articles, used in the market simulation, are archived from the Hong Kong Exchange Market and The Standard[1] during 1/1/2005 and 12/31/2006, respectively. There are thousands of stocks in the market. We selected the stocks as follows. (1) They are Hang Seng Index (HSI) stocks, because HSI is a market indicator. (2) The stocks are selected from the four main sectors: Properties, Utilities, Commerce & Industry, and Finance. (3) For each of the sectors, we chose the stocks with the largest fluctuation, the largest exchange volume, the highest percentage of rise, and the highest percentage of drop. The 12 stocks selected are summarized in Table 3.1.

For the data preprocessing of the news articles, all features are stemmed using the Porter stemmer. Features are words that appear in the news articles, with the exception of digits, web page address, email address and stopwords. Features that appear more than 80% of the total news articles in a day are categorized as stopwords. Meanwhile, features that appear less than 5% of the total news articles in a day are categorized as noisy features. Both the stopwords and noisy features are removed. All features are weighted using the $tf \cdot idf$[70]

[1]`http://www.thestandard.com.hk`

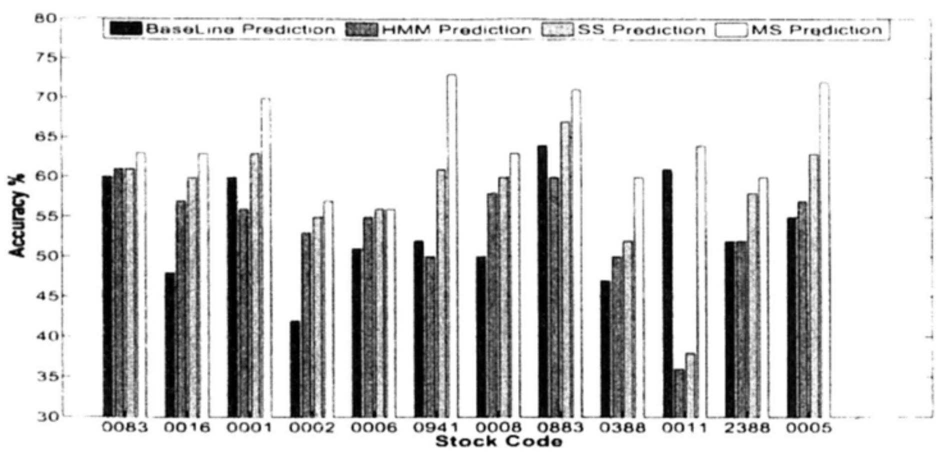


Figure 3.5: Prediction Accuracy

schema whenever necessary. After the data preprocessing, we identify a total number of 928 features. Then the technology as we discussed in section 3.2.2 is explored to identify the associated events for each specific stock. All data from 1/1/2005 to 5/31/2006 are used for training, and the data from 6/1/2006 to 12/29/2006 are used for evaluation.

The non-homogeneous hidden Markov model (NHMM) is trained using the NHMM toolbox[2]. We conducted a series of experiments to obtain the best setting of the NHMM model. Heretoforth, unless otherwise specified, we use the following optimal parameters: Number of features selected using *inter-correlation* and *intra-correlation*: M=N=100; and Number of hidden states: $K = 6$; Minimum similarity among events: $\delta = 0.01$; Influence fading rate: $\sigma = 1$. The regression maximum error is set to maintain the duration of each segment has the mean around 3. All other parameters vary and depend on the specific stock.

Prediction Accuracy

The accuracy of the prediction is obtained by checking whether the direction of the predicted trend is the same as the actual trend. For instance, if the prediction for the upcoming trend is "up" and the upcoming trend is really

[2]Sergey provides a nice toolbox containing algorithms for modeling multivariate time series with hidden Markov models (`http://www.cs.ualberta.ca/~sergey/MVNHMM/`)

rising, then we say that the prediction is correct. Otherwise, if the prediction is "down", but the upcoming trend is rising, then we say that the prediction is wrong.

Figure 3.5 summarizes the results of the accuracy of the predictions using four different approaches against the 12 selected stocks. In all the 12 selected stocks, except for 0011.HK, the SS-based Prediction approach performs much better than the baseline approach, while the MS-based Prediction approach always outperforms the SS-based Prediction approach. However, we note that in some of the stocks (0083, 0016, 0002) their differences are insignificant, which means the improvement from SS-based Prediction to MS-based Prediction approach is marginal. One may criticize the importance of including the market information (news articles) during the segmentation in MS. We will explain it in details in the next section – profits generated by the market simulation.

Consider Figure 3.5. It is obvious that the MS-based Prediction approach excels all other approaches. By using the MS-based Prediction approach, four stocks (0001, 0941, 0883 and 0005) obtained the accuracy higher than 70%, in which three of these stocks (0001, 0941 and 0005) came from the largest exchange volume from their sector. This implies that our approach is most effective when it applies on the stocks which have the largest amount of transactions.

The prediction accuracy of 0006 using the SS-based Prediction approach is the same as the MS-based Prediction approach. The reason is that there are extremely few news articles that are related to 0006 directly. As a result, the prediction made by both of the approaches are the same, as we do not have any news articles to assist/change our predictions.

The result from HMM prediction further supports the effect of using news articles. Although it sometimes performs better than the baseline approach (0083, 0016, etc), it performs worse than both SS-based Prediction and MS-based Prediction approach in all the stocks.

To summarize, our MS Prediction approach is most effective when there are many news articles related to the stock (such as the sector of Commerce &

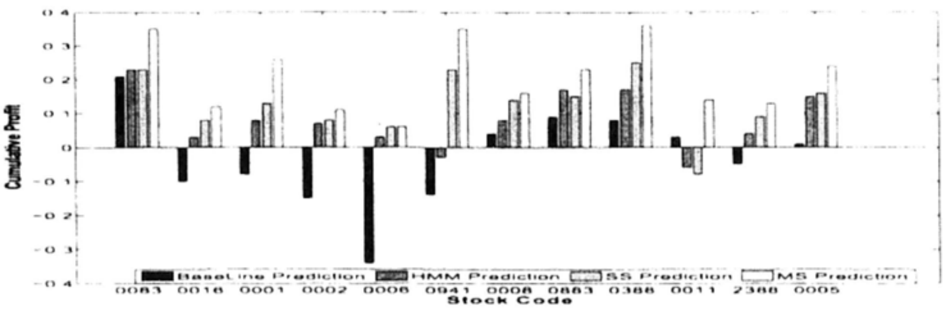


Figure 3.6: The rate of return (12 stocks)

Finance) and may not perform significantly better for the stocks with few news articles (such as the sector of Utilities).

Profits Generated

We evaluate the profits generated by different approaches using a Buy-and-Hold test [34]. The profit is measured by the rate of return, r which is computed as below:

$$r = \sum_{i=0}^{t} \frac{y_{i+1} - y_i}{y_i} \tag{3.20}$$

where y_i is the price of stock at time i. In this test, profit (loss) is made when shares are sold (short are bought). The assumption of zero transaction cost is taken. Two strategies that are used to determine the decisions of buy, sell and hold are given below.

1. If the prediction of the upcoming trend is positive, then shares of that stock are purchased. If a profit of 1% or more could be made within this detected segment, then all shares are sold immediately; otherwise they are sold at the beginning of the next segment.

2. If the prediction of the upcoming trend is negative, then shares of that stock are sold for short. If the trading price is dropped 1% or more than the shorted price within this detected segment, then shares of that stock are purchased immediately; otherwise they are purchased at the beginning of next detected segment.

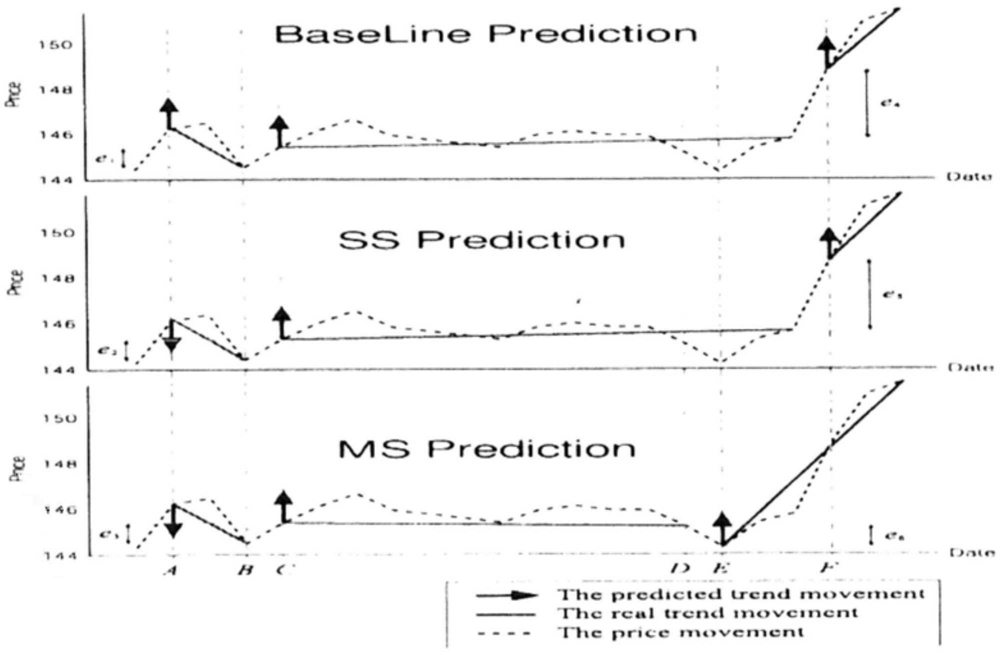


Figure 3.7: Predictions on 0005 (HSBC) from 10/12/2006 to 11/20/2006

We are concerned with the rate of return, how much shares are bought in each transaction is ignored. Figure 3.6 shows the cumulative profit of the 12 selected stocks using different prediction approaches. Our MS-based Prediction approach does not make any loss. All other approaches may incur loss in some stocks.

In the previous section, we showed that the improvement, for some of stocks in terms of the prediction accuracy, between SS-based Prediction and MS-based Prediction approach is insignificant. However, in terms of the profit generated, the improvement is noticeable. For instance, for the stock 0001, the prediction accuracy is only improved by 2% (Figure 3.5), but the profit generated is improved by almost 20%. We explain it below. Take the stock 0005 (HSBC) as an example. Figure 3.7 shows the prediction made by different approaches from 10/12/2006 to 11/20/2006 against 0005 (HSBC). The prediction approaches are: Baseline Prediction approach (top), SS-based Prediction approach (middle) and MS-based Prediction approach (bottom). At point A, a prediction is made by the three different prediction approaches simultaneously. Since the baseline approach relies on the slope of the last segment and the first point of the upcoming segment, this is why an "up" prediction is made. On the other hand, the SS-based Prediction approach and the MS-based Prediction approach both make use of the contents of the news articles for prediction, that is why both of them made a correct "down" prediction at point A. Another prediction is made at point B because the approximation errors (e_1, e_2 and e_3) all exceed the predefined threshold. At that point all approaches made a correct prediction. Now for the MS-based Prediction approach, it made the third prediction at point E, whereas both of the other approaches made the third prediction at point F. MS-based Prediction approach can make prediction at point E, not because the error, e_6 exceeds the predefined threshold. The reason that MS-based Prediction approach made a prediction is because at that moment the contents of the news articles are very different from the contents of the news articles that we have archived previously. As a result, the prediction is made *early* than the other two approaches. Although all of the three approaches made correct predictions, MS-based Prediction approach made an early prediction, which implies that it can generate a larger profit. This is why even though in some cases the prediction accuracy of SS-based Prediction approach and MS-based Prediction approach are similar, the resulting profits generated can be noticeably different.

Bursty Features VS. News Sensitive

Another interesting problem for our approach is whether the bursty features from news articles would help to improve the prediction performance compared to other news based prediction approaches and in what extent it can help. We compare MS-based Prediction with news sensitive prediction, NS-based Prediction [22], where the training is conducted in three phases: first the news articles relevant to a specific stock are aligned to the stock trend according to their timestamp, then the approach would cluster the news articles into positive and negative sample set, finally, the classification model is built to account for the prediction task. And in the testing phase, the arrival of a relevant news article would trigger the prediction system and an alarm will be generated if the news

Stock	MS	NS	Stock	MS	NS
0083	0.35	0.01	0016	0.12	0.10
0001	**0.26**	-0.12	0002	0.11	-0.03
0006	0.06	0.03	**0941**	**0.35**	0.10
0008	0.16	-0.21	0883	0.23	0.09
0388	0.36	0.19	0011	0.14	-0.04
2388	0.13	-0.07	**0005**	**0.24**	0.08

Table 3.2: Comparison between MS and NS

article is classified as a positive or negative news. The difference in performance of profit generation is highlighted in Table 3.2.

From Table 3.2, the cumulative profit of MS-based Prediction outperforms NS-based Prediction in the selected stocks, where in most cases, the margin is larger than 20%. Even for stocks which have large exchange volume and a lot of relevant news articles (0001, 0941 and 0005), the performance of NS-based Prediction is not comparable to MS-based Prediction. The reason is that NS-based Prediction makes a strong requirement on the dataset that the news articles for both training and prediction should be relevant to the specific stock. But in our experiment setup, the news articles are from a general finance news corpus. Under this settings, the approach in [22] is unable to distinguish the news articles with impact on the price change of a specific stock. The noisy training set eventually leads to the unsatisfying performance of the classification model. On the other hand, MS-based Prediction looks deeper into the text corpus by identifying the relevant events as a set of bursty features significantly related to stock price changes and achieves much better results. Apart from the reason of dataset, instead conducting prediction purely based on the news articles as NS-based Prediction, MS-based Prediction also considers the information from the time series of stock price, which gives the prediction a basis as well as additional confidence.

3.2.4. Summary

In this work, we proposed an event-driven approach based on a non-homogeneous hidden Markov model (NHMM). With NHMM, we consider the stochastic process of market state sequence S as a discrete Markov chain, where s_{i+1} depends only on the state s_i and the corresponding event state x_{i+1}. A key issue addressed in this work is how to identify the associated event states for a specific stock and how to make use of them as a part of NHMM for stock price prediction. We gave our solutions using bursty features that are the keywords suddenly appearing frequently in bursty periods. We considered the feature influence on stock prices, and proposed a new segmentation approach to segment stock prices with the consideration of the impacts of bursty features. We studied our approach for a stock market, but our approach can be applied to other financial markets and applications when it is best to utilize information from multiple data sources. We conducted our experimental studies using real datasets. The experimental results confirmed the effectiveness of our approach.

3.3. Stock Risk Mining by News

In this section, we present our technique to predict stock volatility by analyzing news content. Although there have been many existing studies [83, 49, 67] which can predict the up/down trend of stock prices, volatility prediction and ranking from news is a new and challenging problem. We highlight the unique issues of volatility prediction, and discuss why the existing work for stock trend prediction can not be directly applied.

First, volatility carries different information from a trend. Figure 3.8(a) and (b) show the stock volatility and stock prices during 37 trading days (from Sept. 01, 2008 to Nov. 09, 2008), respectively. We can see that there is no obvious correlation between these two time series. Some volatility bursts occur at the turning points of stock price trends (e.g., point 13 and point 27) while others

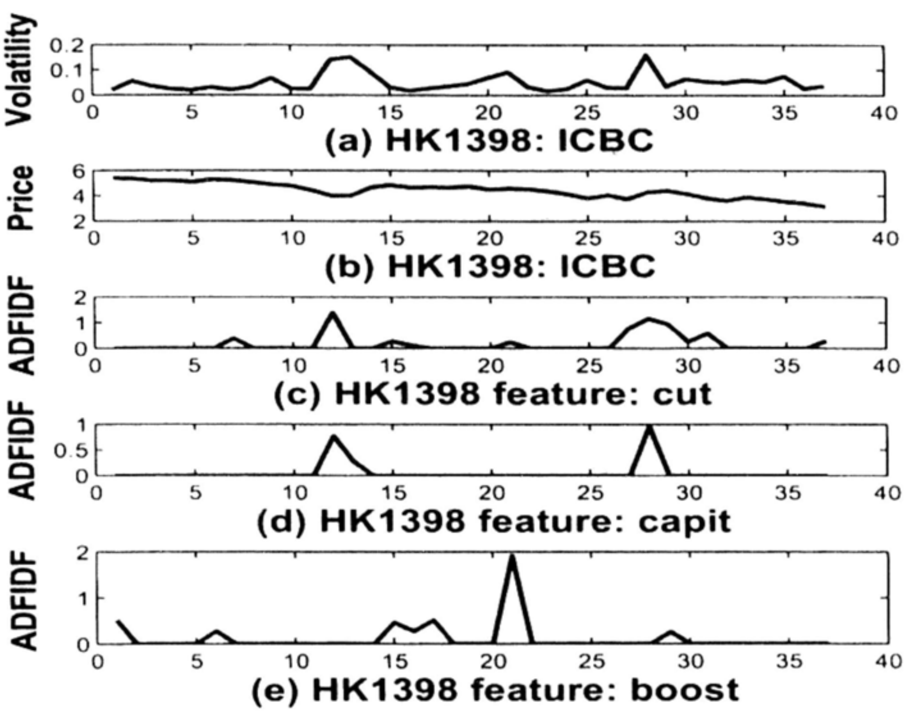


Figure 3.8: Volatility, Prices, and Features

appear when there is no obvious change of stock price trend (e.g., point 21). This is because that volatility is computed as the standard deviation of stock price and therefore reflects market activities from a microscope perspective. As shown in the example, dramatic changes of daily stock prices can cause volatility bursts, but stable daily stock prices do not necessarily imply a stable volatility. A stock which has very stable daily prices may have a big fluctuation in intra-day prices, thus may produce a large volatility value.

Second, the class distribution of training text samples is very skewed if we use a text categorization approach based on news articles to predict stock volatility. Consider the daily ICBC stock prices in 37 days in Figure 3.8. There are 12 up trends and 25 down trends, with a ratio of 1:2 between up and down trends. On the other hand, there are only 4 volatility bursts out of 37, with a ratio of 1:8 between bursty and non-bursty volatility. Furthermore, there are 185 news articles associated with the up trend, and 363 news articles associated with the down trend, with a ratio of 1:2. On the contrary, there are only 51 news articles as positive samples, associated with bursty volatility, and 497 news articles as negative samples, associated with non-bursty volatility, with a ratio of about 1:10. We have observed similar skewed distributions in our large-scale experiments as well. The small number of positive samples related to the rare volatility bursts makes the problem of predicting volatility bursts challenging.

Third, volatility burst prediction shares some similarity with the prediction of the slopes of stock trends, as both problems focus on the magnitude of changes. Existing studies [83, 49, 67] can predict the up/down trend, but may not predict the slope of a trend accurately, because the available information is not sufficient. This evidence also suggests that the existing methods on trend prediction may not work on volatility prediction.

In this thesis, we concentrate on volatility prediction by utilizing both time series data (stock prices) and textual information (news articles). First, the textual information is transformed into time series by using the measure ADFIDF (Adjusted Document Frequency Inverse Document Frequency). Second, representative bursty volatility features are selected based on the co-occurrences of historical stock price volatility and news articles. Third, the feature weights, which measure the degree of importance of those features for each stock are learned. Then, based on the feature weights and the incoming news, the volatility of the corresponding stock is predicted. To improve the prediction accuracy on stocks which have very limited news reports, a random walk model is used to propagate the impacts from news among stocks based on their correlation. Finally, a volatility index is constructed as a time series of predicted volatility. Stocks can be further ranked based on the predicted volatility values.

Main Contributions

The main contributions of the work are summarized as follows.

- We study a new problem of predicting stock risks based on the predicted volatility by utilizing both time series information (stock price) and textual information (news articles).

- We propose a new feature selection algorithm to select bursty volatility features which have co-occurring bursty patterns with the volatility bursts of stocks. A set of such selected bursty volatility features can accurately represent the stock volatility. Feature weights are learned from historical stock prices and news articles to measure the impact of bursty keywords on stock volatility.

- We further use random walk to propagate the impacts of news among stocks based on their correlation. The random walk approach can greatly improve the volatility prediction performance for those stocks with very limited news reports. The volatility prediction and ranking methods are built on top of the random walk model.

- We conducted extensive experimental studies using real datasets and demonstrated the superiority of our approach in comparison with existing approaches.

In the rest sections, the definition of stock volatility and the problem formulation are introduced in Section 3.3.1. We study bursty volatility feature selection in Section 3.3.1, and stock volatility prediction in Section 3.3.2. Section 3.3.3 presents the experimental study. Finally, Section 3.3.4 concludes this work.

3.3.1. Problem Statement

Problem Statement: Given a set of stocks $S = \{S_1, S_2, ...\}$ where S_i is a time series, and a set of documents $T = \{T_1, T_2, ...\}$ available before or at time t, we focus on predicting stock volatility and ranking stocks based on the predicted volatility at the next time unit $t + 1$, based on the available textual information.

Bursty Volatility Features

To predict stock volatility, we could detect breaking events from available news articles that are indicators of volatility bursts. Here, the breaking event is defined as the event which suddenly appears and trigger the burst of stock volatility. So the problem is how to select a small set of features which can represent all breaking events.

As we discussed, the number of volatility bursts in a stock S_k is considerably small in comparison with the total number of up/down trends occurring in the same stock. Even though the number of news articles that are related to the volatility bursts in the stock S_k, is also observed to be small, we believe that the features in those documents can potentially predict/rank volatility bursts effectively. Here we define the features which trigger the burst of time series volatility as bursty volatility features. The desirable properties of a feature are discussed below.

Bursty Occurrences: An effective feature needs to be a bursty feature rather than a stable feature over a time interval. It is most likely that such bursty features can effectively represent volatility bursts.

High Indicative Ability: An effective feature needs to have high ability to indicate volatility bursts, i.e., the bursts of a feature need to be a good indicator of the volatility bursts of the corresponding stock. Features whose high occurrences are always accompanied with volatility bursts are more preferable than those features whose high occurrences only cause volatility bursts occasionally.

High Coverage and Low Redundancy: A minimal set of selected effective features needs to cover the volatility bursts as much as possible. By coverage we mean that the set of selected effective features, as a whole, captures all volatility bursts. By redundancy we mean that some selected features may give similar information.

In the following, we discuss bursty feature measurement and how to select bursty volatility features.

ADFIDF Measure

As each stock is representing a company, if there are some important things related to the company, the news appears immediately. Generally, the wider the news is reported, the more important the news is. If there is no bursty news, the value which measures the feature burstness should be around average. In the following, we discuss how to capture the wideness of a text feature.

Given a set of stocks $S = \{S_1, S_2, \ldots\}$, where a stock $S_k = [s_{k1}, s_{k2}, \ldots, s_{k\mathcal{I}}]$ is a sequence of stock prices in the time interval $\mathcal{I}$. In the same time interval $\mathcal{I}$, there exists a set of news/documents, $T = \{T_1, T_2, \ldots\}$, where a document $T_i \in T$ contains a set of features $\{f_{ij}\}_{j=1}^{m}$. We assume that it is known which stock S_k a document T_i is related to. The assumption is reasonable since most financial news providers do provide such information when distributing financial news articles. Then the features in the document T_i can also be identified to which stock they are related. We represent a feature f related to a stock S_k in the time interval $\mathcal{I}$ as a time series, $f(k) = [f^k(1), f^k(2), \ldots, f^k(\mathcal{I})]$, where $f^k(t)$ is defined as follows.

$$f^k(t) = \frac{DF_{k,f}(t)}{N_k(t)} \times \log\left(\frac{N_k(T)}{DF_{k,f}(T)}\right) \tag{3.21}$$

where $DF_{k,f}(t)$ is the number of related documents in T containing the feature f for the stock S_k at time t, and $N_k(t)$ is the total number of documents in T related to the stock S_k at time t. Here t is a time unit defined by user, such as one month, one day, or one hour. In this work, t is defined as one day in our evaluation. Therefore, $f^k(t)$ reflects the wideness of the feature f for the stock S_k at time t. In the following, we call $f^k(t)$ the ADFIDF (Adjusted Document Frequency Inverse Document Frequency) value of a feature f related to stock S_k at time t.

Algorithm 7 FeatureRank(S_k, F_k, γ)

```
INPUT: stock $S_k$, bursty feature set $F_k$, decay factor $\gamma$
OUTPUT: a list of pairs $(f_j, E(S_k, f_j))$ for $f_j \in F_k$
in descending order
 1: compute $TB_{f_i}$ for $f_i \in F_k$;
 2: for all $f_i \in F_k$ do
 3:     compute $E(S_k, f_i)$ using Eq. (3.22);
 4: end for
 5: $\mathcal{E} \leftarrow \emptyset$;
 6: while $F_k \neq \emptyset$ do
 7:     sort $F_k$ in decreasing order based on $E(S_k, f_i)$;
 8:     let $f$ be the first feature in the sorted $F_k$;
 9:     remove $f$ from $F_k$;
10:     append the pair $(f, E(S_k, f))$ into $\mathcal{E}$;
11:     for all $f_j \in F_k$ do
12:         $B = TB_{f_j} \cap TB_f$;
13:         if $B \neq \emptyset$ then
14:             $V(S_k, t) \leftarrow \gamma \cdot V(S_k, t)$ for $t \in B$;
15:             update $E(S_k, f_j)$ based on Eq. (3.22);
16:         end if
17:     end for
18: end while
19: return $\mathcal{E}$;
```

Given all $f^k(t)$ values, $\forall t \in \mathcal{I}$, we identify the bursty time interval using the technique discussed in Section 3.1.2. We denoted it as TB_f, for feature f, representing a set of time intervals where f appears to be a bursty feature.

It is worth noting that the commonly-used measure, TFIDF (term frequency inverse document frequency) [70], cannot be used since we need the features that witness a stock in a time interval rather than the importance of the features for a document. Therefore, we use a new measure ADFIDF. The idea of DFIDF is brought from [33], but ADFIDF is different from it. The original DFIDF is computed for the whole document set, so the DF and IDF values are global. However, ADFIDF is computed for a subset of documents containing all news related to a specific stock S_k. Moreover, the ADFIDF value is computed for a specific time unit t as in $f^k(t)$.

Bursty Volatility Features

We identify all bursty features based on ADFIDF. Then we introduce a *co-occurrence rate*, denoted as $E(S_k, f)$, to measure how a bursty feature f and the volatility bursts of a stock S_k occur at the same time. The larger $E(S_k, f)$ is, the more important the feature f to the stock S_k. The idea of co-occurrence is, if a feature always bursts together with the stock volatility bursts in the same time interval, the feature is valuable for identifying volatility bursts. $E(S_k, f)$ is defined below.

$$E(S_k, f) = \frac{V(S_k, TB_f)}{|TB_f|} \bigg/ \frac{V(S_k, \mathcal{I})}{|\mathcal{I}|} \tag{3.22}$$

where $V(S_k, \mathcal{I})$ is the sum of the bursty volatility values regarding stock S_k in the time interval $\mathcal{I}$.

$$V(S_k, \mathcal{I}) = \sum_{t \in \mathcal{I}} V(S_k, t) \tag{3.23}$$

where $V(S_k, t)$ refers to volatility at time t computed using Eq.(3.3) and then transformed using the approach we deal with bursty features. Recall that TB_f is the set of bursty time intervals of the feature f, and $\mathcal{I}$ is the entire time interval. In Eq.(3.22), the numerator computes the average volatility over the co-occurrence time intervals of the bursty feature f and volatility bursts. The normalization by the denominator makes the co-occurrence rates of features with respect to different stocks comparable. So the rational behind the equation is that we give higher co-occurrence rate to the features in whose bursty periods the stock index volatility is also very high.

Bursty Volatility Features Selection

In the previous subsections, we have discussed ADFIDF for feature burstness measure and the co-occurrence rate $E(S_k, f)$ for indicative ability measure for a feature f. We will discuss how to select a compact set of bursty volatility features to ensure high coverage and low redundancy.

Consider the example in Figure 3.8. There are three volatility bursts of the ICBC stock, denoted as S_k, shown in Figure 3.8(a). In addition there are three ADFIDF sequences of bursty features "cut" (denoted as f_x), "capit" (f_y) and "boost" (f_z), shown in Figure 3.8(c)-(e), respectively. Here, the two ADFIDF sequences f_x^k and f_y^k have 2 similar bursts corresponding to 2 out of 3 volatility bursts of S_k, and the only burst in f_z^k corresponds to the remaining volatility burst in Figure 3.8(a). The three bursty volatility features, f_x, f_y, and f_z, together cover the three volatility bursts in S_k. By "cover", we mean that the features jointly represent the volatility information about S_k. Assume $E(S_k, f_x) = E(S_k, f_y) \geq E(S_k, f_z)$ and the goal is to select top-2 bursty volatility features. If we select both f_x and f_y, then one of them is considered as redundant and the third volatility burst cannot be captured.

In order to select a set of representative bursty volatility features, we design an algorithm to rank all bursty volatility features such that the top-k features, to be selected from the ranking list, will be more likely to accurately capture the corresponding volatility bursts of a stock. The algorithm FeatureRank is outlined in Algorithm 7. The main idea is to reduce $E(S_k, f_y)$ if its burst time interval TB_{f_y} is overlapped with another TB_{f_x} for a higher ranked feature f_x, using a feature decay factor γ. As shown in Algorithm 7, it takes a stock S_k, a set of bursty features F_k related to S_k, and a feature decay factor γ. It computes the bursty time interval TB_{f_i} for every feature $f_i \in F_k$ (line 1), and then computes $E(S_k, f_i)$ using Eq.(3.22) (lines 2-4). Let $\mathcal{E}$ keep a list of pairs $(f_j, E(S_k, f_j))$ in descending order of $E(S_k, f_j)$. In a while loop (lines 6-18), in every iteration, it selects the top bursty volatility feature f from F_k and appends the pair $(f, E(S_k, f))$ to $\mathcal{E}$. It then recomputes $E(S_k, f_j)$ for all remaining $f_j \in F_k$ using the decay factor γ, if there is an overlap between the burst time interval TB_f of the selected feature f and TB_{f_j} of the feature f_j.

According to Algorithm 7, the top-2 bursty volatility features in Figure 3.8 would be either f_x ("cut") or f_y ("capit") plus f_z ("boost").

3.3.2. Volatility Prediction

We have discussed how to select bursty volatility features in last section. Such bursty volatility features are selected based on how the burst features in documents co-occur with the volatility bursts in stocks. In this section, we discuss how such selected bursty volatility features are used to predict the stock volatility. The bursty volatility features can have both direct impacts on stock volatility and propagated impacts on stock volatility through stock-stock correlation, as volatility of a stock may affect and be affected by others.

To predict the stock volatility at time t, we use news articles which arrive before t. For example, to predict stock volatility on a particular day, we collect news articles which appear before 10:00AM on that day (market opening time) for prediction. The news articles which appear after 10:00AM will be used for next day prediction.

Graph Construction

We construct an edge-weighted node-labeled graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \mathcal{V}_F \cup \mathcal{V}_S$ is a set of nodes, $\mathcal{V}_F$ represents the set of bursty volatility features, and $\mathcal{V}_S$ represents the set of stocks. $\mathcal{E} = \mathcal{E}_{FS} \cup \mathcal{E}_{SS}$ is a set of edges, where $\mathcal{E}_{FS}$ represents a set of edges from a node in V_F to a node in V_S, and $\mathcal{E}_{SS}$ represents a set of edges from a node in V_S to another node in V_S. A node in $\mathcal{V}$ is associated with a unique label, so we treat labels as node identifiers. The edge weight on an edge $(v_f, v_s) \in E_{FS}$ represents the impact of a bursty volatility feature $v_f \in \mathcal{V}_F$ to a stock $v_s \in \mathcal{V}_S$. The higher the weight, the larger impact of the feature on the stock. The edge weight on an edge $(v_{s_1}, v_{s_2}) \in E_{SS}$ represents the degree of co-occurrences of volatility bursts between two stocks v_{s_1} and v_{s_2} in $\mathcal{V}_S$. The higher the weight, the more co-occurrences of the volatility bursts of two stocks.

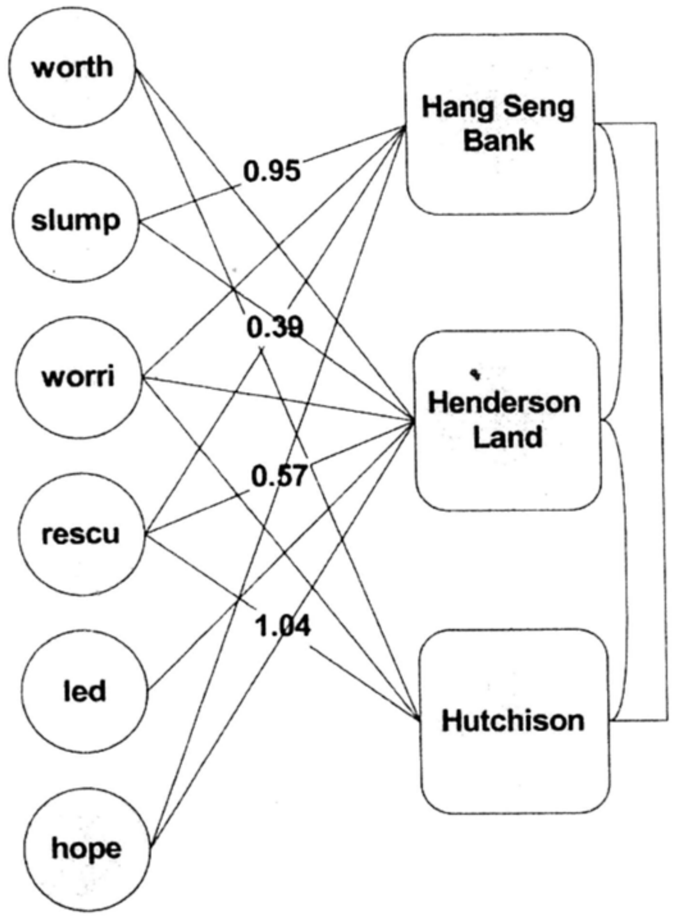


Figure 3.9: Volatility Prediction Based on Random Walk

Figure 3.9 shows a simple graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}_F$ contains 6 bursty volatility features, "worth", "slump", "warri", "rescu", "led", and "hope". $\mathcal{V}_S$ contains 3 stocks, "Hang Seng Bank", "Henderson Land", and "Hutchison". Table 3.3 shows the edge weights from a feature (a node in $\mathcal{V}_F$) to a stock (a node in $\mathcal{V}_S$) for Figure 3.9. The feature "rescu" is linked to all three stocks with different weights, which means that all these stocks are influenced by the feature "rescu". Some features may only have impacts on a subset of stocks. For example, the feature "led" does not have any impacts on "Hang Seng Bank" or "Hutchison", so there is no edge from "led" to "Hang Seng Bank" or "Hutchison".

Based on the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we perform random walk and calculate the volatility of a stock at time $t+1$ based on the available bursty feature information

Feature	Hang Seng Bank	Henderson Land	Hutchison
worth	0	0.88213	1.2434
slump	0.94723	0.61459	0
worri	0.98096	0.72786	0.33304
rescu	0.38712	0.56985	1.0362
led	0	0.61459	0
hope	0.63762	0.68553	0

Table 3.3: Impacts of Bursty Volatility Features

at time t, as well as predicted volatility of correlated stocks, as in Eq.(3.24).

$$\mathbf{V}(S_k, t+1) = \alpha \sum_{(f_i, S_k) \in \mathcal{E}_{FS}} f_i^k(t) \cdot E(S_k, f_i) + (1-\alpha) \sum_{j \neq k} \rho(S_j, S_k) \cdot \mathbf{V}(S_j, t+1) \tag{3.24}$$

Here, the first part measures the accumulated direct impacts from bursty volatility features to a stock. This is the information we captured from news to stocks. Recall that $f_i^k(t)$ is the ADFIDF value to indicate how the feature f_i is related to stock S_k at time t (Eq.(3.21)), and $E(S_k, f_i)$ is the co-occurrence rate to measure how feature bursts of f_i and volatility bursts of S_k occur at the same time. The second part captures the propagated volatility bursts from correlated stocks S_j based on random walk, as stocks may affect each other in the stock market. This part can also improve volatility prediction for stocks which have very little related news. The correlation factor $\rho(S_j, S_k)$ is computed as follows.

$$\rho(S_j, S_k) = \frac{\sum_{\tau=1}^{t} (V(S_j, \tau) - \overline{V(S_j)})(V(S_k, \tau) - \overline{V(S_k)})}{\sigma_{V(S_j)} \sigma_{V(S_k)}} \tag{3.25}$$

Here, $V(S_k, \tau)$ is the bursty volatility at time τ computed using Eq.(3.3). $\overline{V(S_j)}$ and $\overline{V(S_k)}$ are the mean volatility values of the two stocks S_j and S_k, respectively. $\sigma_{V(S_j)}$ and $\sigma_{V(S_k)}$ are the standard deviation of volatility for the two stocks in the time interval $[1, t]$.

Volatility Prediction

Based on the graph and random walk model, we discuss the procedure of volatility prediction. Volatility prediction involves two phases, namely a training phase and a testing phase. The training phase is done based on a set of documents (news articles) T, and a set of stocks S, obtained in the time interval $[1, \mathcal{I}]$. The testing phase is, given a set of new documents T', on a time step t, to predict stocks volatility on the next time unit $t+1$.

The training phase is done as follows. First, for each stock $S_k \in S$, we compute the volatility over the time interval $[1, \mathcal{I}]$, denoted as $V(S_k) = [\sigma_1, \sigma_2, \ldots, \sigma_{\mathcal{I}}]$, where σ_i is computed using Eq.(3.3). Then we determine a set of bursty features $F_k = \{f_1, f_2, \ldots\}$, where $f_i \in F_k$ corresponds to a time series of ADFIDF $f_i^k = [f_i^k(1), f_i^k(2), \ldots, f_i^k(\mathcal{I})]$ in the time interval $[1, \mathcal{I}]$. $f_i^k(t)$, $t \in [1, \mathcal{I}]$, is computed using Eq.(3.21). Second, we compute the co-occurrence rate $E(S_k, f_i)$ for every $f_i \in F_k$ using Eq.(3.22). Third, we obtain a list of pairs $(f_i, E(S_k, f_i))$ using Algorithm 7 to rank the features with a decay factor γ. Finally, we compute the correlation $\rho(S_j, S_k)$ between two stocks S_j and S_k using Eq. (3.25).

The testing phase is done as follows. Suppose that we obtain a set of new documents T', at time t. First, we compute $f_i^k(t)$ for every f_i in a document in T', that is related to S_k. Second, we construct an edge-weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. $\mathcal{V} = \mathcal{V}_F \cup \mathcal{V}_S$, where $\mathcal{V}_F$ is the set of features that both appear in T' and are burst volatility features obtained in the training phase. The edge weight for an edge (f_j, S_k), from a bursty volatility feature f_i to a stock S_k is assigned as $E(S_k, f_i)$ which is computed in the training phase. The edge weight for an edge (S_j, S_k), between two stock nodes, is assigned as $\rho(S_j, S_k)$ computed in the training phase. An example is illustrated in Figure 3.9. Third, we compute $\mathbf{V}(S_k, t+1)$, for every $S_k \in S$, using Eq.(3.24) iteratively, until it converges based on random walk.

Volatility Index and Volatility Ranking

Based on the predicted stock volatility, we could perform two analytical tasks: volatility index construction and stock volatility ranking.

A volatility index for stock S_k in the time interval $\mathcal{I}$ is a time series of

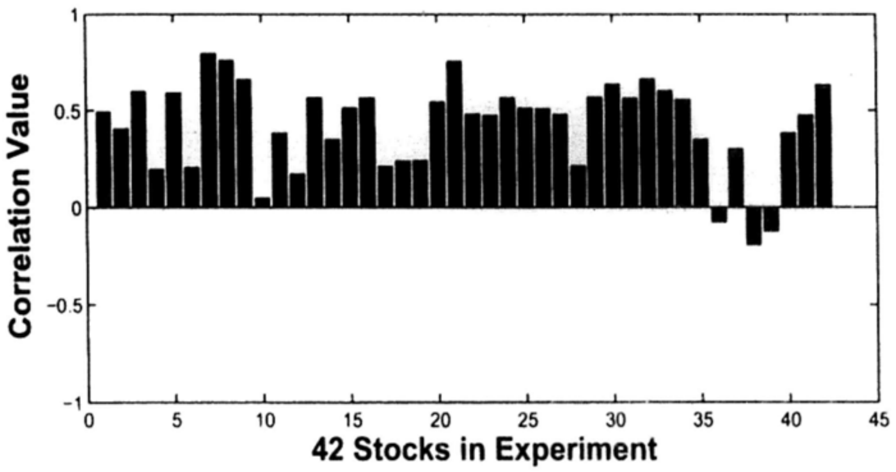


Figure 3.10: Prediction based on Bursty Features

predicted volatility values: $NI(S_k) = [\mathbf{V}(S_k, 1), \mathbf{V}(S_k, 2), \ldots, \mathbf{V}(S_k, \mathcal{I})]$. We call it VN-index since it is a volatility index constructed from news. If the predicted volatility is accurate, the correlation between VN-index and the real volatility sequence in the testing period should be large. The correlation is quantitatively measured using the Pearson correlation coefficient.

$$\rho(NI(S_k), V(S_k)) = \frac{\text{cov}(NI(S_k), V(S_k))}{\sigma_{NI(S_k)}\sigma_{V(S_k)}} \tag{3.26}$$

where $NI(S_k)$ is the volatility index sequence, $V(S_k)$ is the real volatility sequence, and cov means covariance. We will evaluate the quality of the constructed VN-index in the experiment part.

Besides the volatility index construction, we can further rank stocks based on their predicted volatility values $\mathbf{V}(S_k, \mathcal{I}+1)$ for each stock $S_k \in S$. The ranking quality will also be evaluated below.

3.3.3. Evaluation

In this section, we evaluate our proposed volatility prediction approach through two groups of experiments: volatility index construction and volatility ranking.

We archive the minute-level intra-day stock prices and the news articles

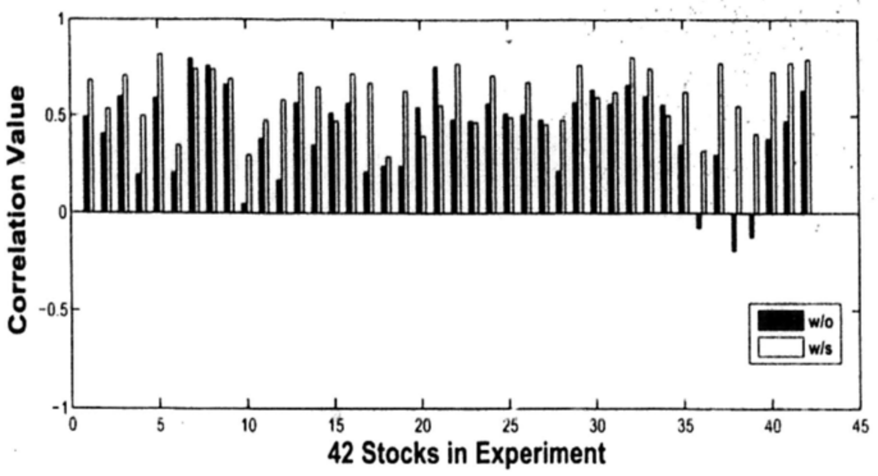


Figure 3.11: Prediction based on Random Walk

from the Hong Kong Exchange Market and Don Jones Factiva database[3] from Jan. 1, 2008 to Dec. 31, 2008, respectively. All 42 component stocks for Hang Seng Index (HSI) are selected, which are the most influential and most widely held public stocks in Hong Kong. At each day t, the daily realized volatility is computed by applying Eq.(3.3) on the 1-minute time series.

In total, over 150,000 news articles are collected. Each news article is related to a specific stock according to Factiva's classification system. Besides, we only tag news articles which appear before 10:00AM as the news article at that day for prediction, since most newspapers will release their news story before the market opens. The news articles which appear after 10:00AM will be labeled as the news articles of next day for prediction, e.g., the news release at 7:00PM will be used for next day prediction.

For the preprocessing of these news articles, all features are stemmed using the Porter stemmer. Features are words that appear in the news articles, with the exception of digits, web page address, email address and stop words. Features that appear more than 80% of the total news articles in a day are categorized as stop words. Features that appear less than 5% of the total news articles in a day are categorized as noisy features. Both the stop words and noisy features

[3]`http://www.factiva.com/`

are removed. All data from Jan. 1, 2008 to Nov. 30, 2008 are used for training, and the data from Dec. 1, 2008 to Dec. 31, 2008, are used for evaluation.

We perform the experiments on a PC with a Pentium IV 3.4GHz CPU and 2GB RAM.

Volatility Index Construction

In this part, we construct the VN-index based on predicted volatility values and evaluate the quality. We focus on the following questions:

(1) What are the effects of the proposed techniques (e.g., direct impacts from bursty volatility features versus propagated impacts based on random walk) in our algorithm? How much improvement can each of them contribute respectively?

(2) What is the overall quality of the VN-index compared with the ground truth?

Prediction based on Bursty Features

First, the VN-index is constructed purely based on the news information without taking the stock-stock correlation into consideration. That means the predicted volatility is computed by setting $\alpha = 1$ in Eq.(3.24).

The result is show in Figure 3.10. Each column in the figure represents a correlation value between the real stock volatility and the VN-index for a stock. The average correlation value is 0.4252, the maximum one is 0.7951, and the minimum one is -0.1944. Although the overall performance looks good (note that the average value of correlation between stocks is only 0.4094), the performance varies dramatically for different stocks.

We further analyze the result for those stocks whose correlation is very low, i.e., the predicted volatility is inaccurate. We find that for those stocks, their related features are much less than the average number. When a stock's related features are not sufficient to describe the stock price changes, the volatility prediction based on news is inaccurate.

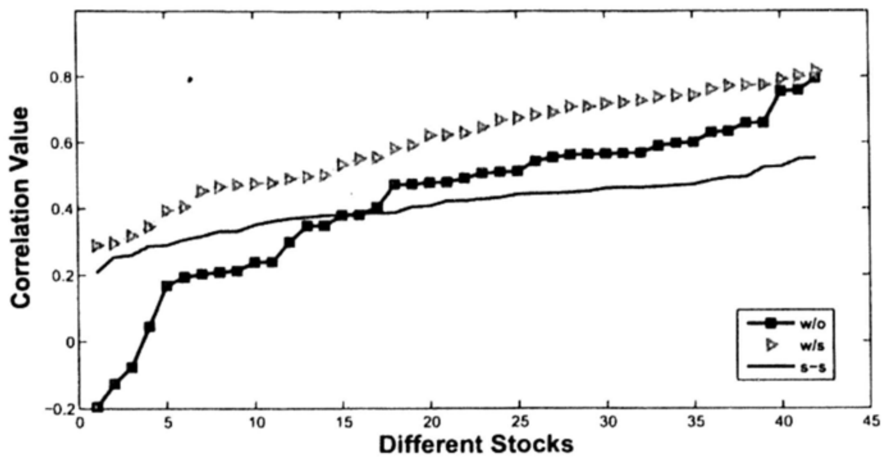


Figure 3.12: Comparison with Correlation between Stocks

Prediction based on Random Walk

To improve the prediction for stocks which have very insufficient news reports, we exploit the stock-stock correlation through random walk to propagate the news impacts. In this experiment, the VN-index is constructed based on Eq.(3.24).

As shown in Figure 3.11, the left columns are the correlation results based on volatility prediction from news only, while the right columns are the correlation results based on both news direct impacts and propagated impacts from random walk. When random walk is added to the prediction model, the average correlation is 0.6021, which improves a lot from 0.4252. In addition, the correlation value for every stock is positive.

Comparison with Stock-Stock Correlation

We further compare the correlation of the predicted volatility and the true volatility and the correlation between stocks. As shown in Figure 3.12, 's-s' means the correlation value between one stock and the other 41 stocks in the whole year of 2008. 'w/s' and 'w/o' represent the results with random walk and without random walk, respectively.

The mean value of correlation between stocks is 0.4094. For our approach without random walk, the average correlation between stock volatility and VN-

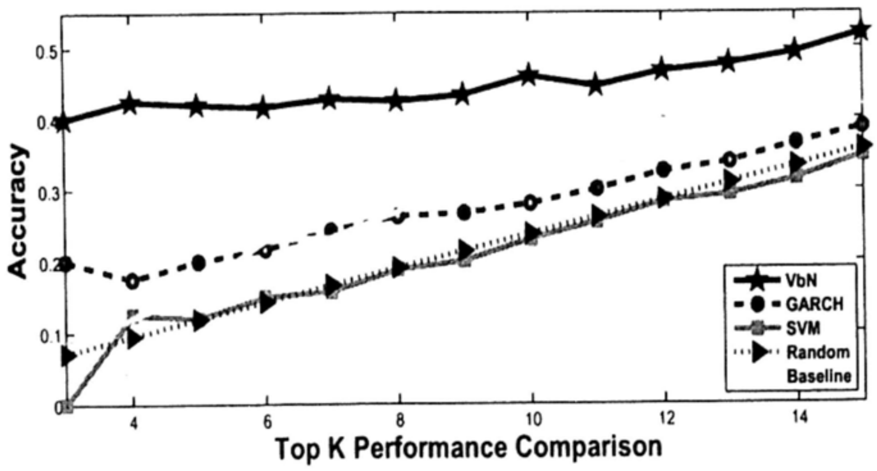


Figure 3.13: Volatility Ranking Comparison

index is 0.4252. When random walk is applied, the performance is even better. That means the co-movement of VN-index and stock volatility is better than the co-movement of volatility of different stocks in the stock market.

Volatility Ranking

In this section, we evaluate the quality of ranking stocks based on their predicted volatility values. We focus on the following questions:

(1) How does our approach compare with other approaches in volatility ranking?

(2) What are the effects of the proposed techniques (e.g., bursty feature, random walk) in our algorithm? How much improvement can each of them contribute respectively?

(3) If we combine our approach with traditional approach purely based on historical stock price data (e.g., GARCH), can we achieve any further improvement?

In this experiment, we use a much smaller training set for evaluation. Specifically, the data from Sept. 01, 2008 to Oct. 24, 2008 are used for training, and the data from Oct. 25 to Nov. 09, 2008 are used for evaluation.

Normalized Real Volatility	VbN	GARCH	SVM
CHALCO	Li & Fung	Esprit Hldgs	Esprit Hldgs
Li & Fung	CHALCO	Li & Fung	FIH
New World Dev	Esprit Hldgs	HK & China Gas	CITIC Pacific
Esprit Hldgs	FIH	New World Dev	CHALCO
COSCO Pacific	Henderson Land	Sino Land	Sinopec Corp
MTR Corporation	COSCO Pacific	Hutchison	CNOOC
Sino Land	Hutchison	China Shenhua	CLP Hldgs
China Mer Hldgs	HK & China Gas	Henderson Land	Hutchison
Cathay Pac Air	China Mer Hldgs	FIH	Hang Seng Bank
Hang Lung Prop	Sino Land	Cheung Kong	Li & Fung
China Unicom	New World Dev	HKEx	BOC Hong Kong
CITIC Pacific	Cathay Pac Air	Hang Seng Bank	Bank of E Asia
China Resources	Yue Yuen Ind	China Mer Hldgs	China Resources
Yue Yuen Ind	Hang Seng Bank	Cathay Pac Air	SHK Prop
Henderson Land	Cheung Kong	CHALCO	ICBC

Table 3.4: Ranking Result Comparison

Ranking Quality Comparison

We compare our proposed volatility ranking approach, denoted as VbN, for Volatility-by-News, with the following approaches.

- **Random Selection**: The volatility rank list is formed based on random selection. The accuracy is the statistical mean accuracy value for ranking.
- **Baseline Model**: The volatility rank list is formed based on average volatility on the training set.
- **GARCH**: We apply GARCH model to predict the volatility of next day and rank the stocks based on the predicted volatility. We use a five year daily stock data for training GARCH model, because if the time series is not long enough, the performance will be bad. The UCSD Garch toolbox is used in the experiments.
- **SVM**: we label news articles as positive and negative based on whether the volatility bursts occur after the news release, using a similar approach as

in [67]. We use the most promising text classification model support vector machine (SVM) [39], to train the text classifier. Based on the classifier and the news features in the testing phase, the volatility of stocks is ranked.

In this experiment, since stocks have volatility in different scales, all the predicted and real volatility values are normalized by their mean value and are transform into relative volatility. The accuracy is measured by overlap-similarity [72], $OS(\tau_1, \tau_2)$, which indicates the degree of overlap between the top n volatility stocks of the two rankings τ_1 and τ_2, where τ_1 is the ranking computed by a prediction model (i.e., sort the stocks by the predicted volatility), and τ_2 is the actual ranking from ground truth (i.e., sort the stocks by the realized stock volatility). The overlap of two stock sets A and B (each of size k) is defined as $\frac{|A \cap B|}{k}$. As a case study, Table 3.4 shows a comparison among VbN, GARCH, and SVM against the ground truth, using top-15 stocks that have high volatility bursts in the next time unit. Here the stock ranking list for the three models are based on their predicted volatility values. In Table 3.4, stocks are ranked in descending order of the corresponding volatility value. For results in Table 3.4, the overlap-similarity between VbN and the ground truth (normalized real volatility) is 0.67, larger than that between GARCH/SVM and the ground truth, which are 0.53 and 0.33, respectively.

We further test VbN in comparison with the other four methods by varying k= 3–15 in the top-k ranking list. Figure 3.13 shows the mean value of accuracy comparison between different methods over the entire testing period. From Figure 3.13, when k is small ($k = 3$), the accuracy of VbN is 40% higher than SVM, 25% higher than Random Selection, and 20% higher than GARCH. When k increases, the accuracy of all methods increase, but VbN outperforms the other methods in all cases. When $k = 15$, VbN achieves an accuracy of 50% which is around 15% higher than other approaches.

Volatility Ranking based on Bursty Features

In this experiment, we evaluate the effectiveness of bursty features and the

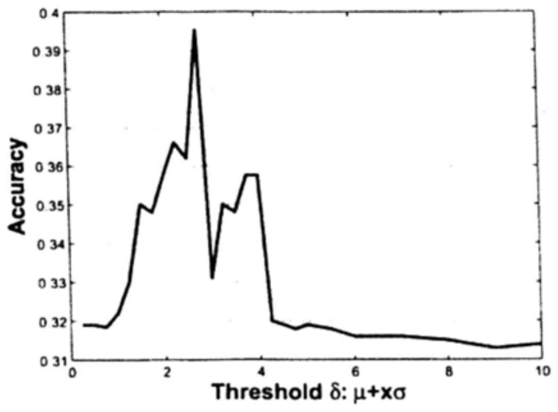


Figure 3.14: Volatility Ranking with Different δ

impacts of the threshold δ on volatility ranking. If $\delta = 0$, all related features are included in the bursty feature set. On the other hand, if δ is set to a large value, there may not be any bursty feature being selected. Figure 3.14 shows the results. The x-axis is in a range of $\mu + x\sigma$, where μ is the mean of ADFIDF, σ is its deviation, and x is an integer in the range of $[0, 10]$. y-axis is the average accuracy of top-k results for k= 1–15. As shown in Figure 3.14, capturing bursty features is a very important factor for ranking stocks based on volatility. When $\delta = \mu + 2.75\sigma$, the ranking accuracy is the highest (39.5%), which is 7.7% higher than using all the bursty features (31.8%) and 8% higher than using no news information (31.5%). As the purpose of this experiment is to measure the effect of bursty features, we set $\alpha = 1$ in Eq.(3.24).

We evaluate the effectiveness of VbN based on random walk with $\alpha = 0.95$ in Eq.(3.24). We observe that the ranking of all component stocks in Hang Seng Index is not noticeably affected by varying α. It is because that the component stocks of Hang Seng Index are reported intensively. Therefore, the improvement based on propagated impacts by random walk is not obvious. In this experiment, we test another set of 42 stocks including 11 HSI-component stocks and 31 non HSI-component stocks that only receive few news articles from time to time. Figure 3.15 shows the mean value accuracy comparison for the bottom-k out of the 42 stocks, when $\alpha = 1$ (without random walk) and $\alpha = 0.95$ (with random

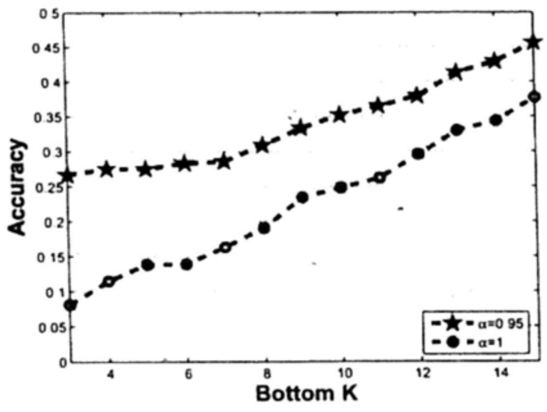


Figure 3.15: Volatility Ranking by Random Walk

walk) over the entire testing period. As seen in Figure 3.15, when $\alpha = 0.95$ (with random walk), its accuracy becomes 27.5% which is 20% higher than the accuracy when $\alpha = 1$ (without random walk). When we increase the k value of the bottom-k stocks from 1 to 15, the smallest accuracy margin is still as large as 10%, which indicates random walk is effective to improve the accuracy for ranking stocks that do not frequently receive news articles.

3.3.4. Summary

In this section, we studied a new research problem of predicting and ranking stock volatility based on news, where volatility is an important stock risk measure. We discussed the unique challenges of volatility prediction/ranking, and showed that the existing approaches on stock trend prediction cannot effectively solve our problem. We defined the bursty volatility features and proposed an algorithm to select a set of highly indicative bursty volatility features to represent volatility bursts. The main idea is to utilize features in news articles to strengthen the prediction and ranking of volatility. In addition, we proposed a random walk based approach that propagates the news impacts through correlated stocks. We conducted extensive performance study on volatility index construction and stock volatility ranking using real datasets and demonstrated the effectiveness of our proposed approach.

3.4. Detecting Priming News Events

In the last two sections, we have studies the effect of news event on time series movement. An question rises naturally, what are these event and how they are organized. In this section, we try to answer this question and study the problem of detecting priming events from a time series index and an evolving document stream. Particularly, we use the news event related to approval index of United States President to demonstrate our technique. And the technique can be applied to financial data to detect priming events related to a financial time series directly.

In Figure 3.16, we take the weekly approval index of US President Obama from Jan 20, 2009 to Feb 28, 2010 as an example to illustrate the difficulty of this problem. In Figure 3.16, the approval index (blue line) evolves and drops from 67% to 45% in the last 56 weeks. In Particular, there are two periods when the index drops dramatically. One is from February to March with a drop from 60% to 55%. The other is in July with a drop from 54% to 48%. For a user who is interested in politics and wants to know what events trigger these significant changes, he/she may issue a query "President Obama" to the search engine. But the result will only be a list of news articles indicating the events that President Obama participates in during these periods. In Figure 3.16, we tag a small part of them on the index. As we can see, in the first period, there are 7 events including his meeting with Kevin, announcement of tax cut, signed an executive order, proposed a bill for financial reform, etc. Only with this information, we cannot fulfill the user's need since we cannot differentiate the role that each event plays to drag down the approval index in that time period. This urges us to think about the following questions. What makes an event priming? Does it contain some elements which will attract public eyes and change their mind? Besides, if such elements do exist, could we find their existence evidence from other time period and use them to justify the importance of the local event containing them?

In this thesis, we call such elements influential topics and use them as basic

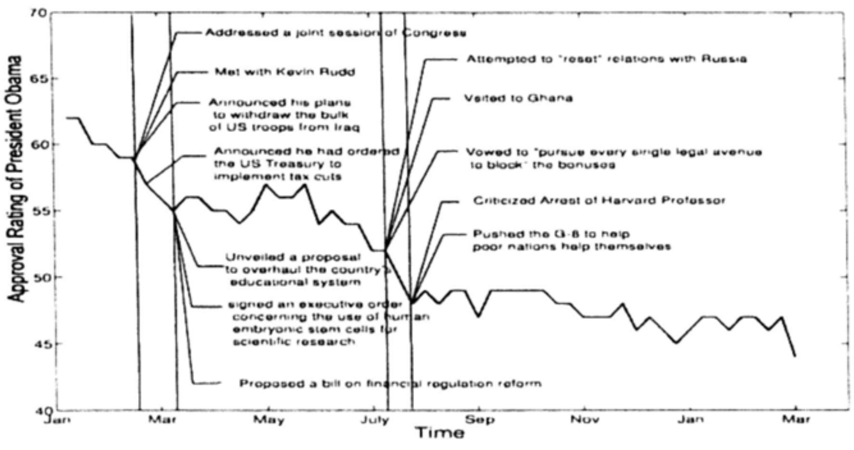


Figure 3.16: Approval Index of President Obama

units to form a priming event. Specifically, we identify the influential topics at a global level by integrating information from both a text stream and an index stream. Then at a micro level, we detect such evidences and organize them into several topic clusters to represent different events going on at each period. After that, we further connect similar topic clusters in consecutive time periods to form the priming events. Finally, we rank these priming events to identify their influence to the index.

The contributions of this work are highlighted as follows.

- To the best of our knowledge, we are the first to formulate the problem of detecting priming events from a text stream and a time series index. A priming event is an object which consists of three components: (1) Two timestamps to denote the beginning and ending of the event; (2) A sequence of local influential topic groups identified from the text stream; (3) a score representing its influence on the index.

- We design an algorithm that first discovers the influential topics at a global level and then drills down to local time periods to detect and organize the priming events based on the influential topics.

- We evaluate the algorithm on a real world dataset and the result shows

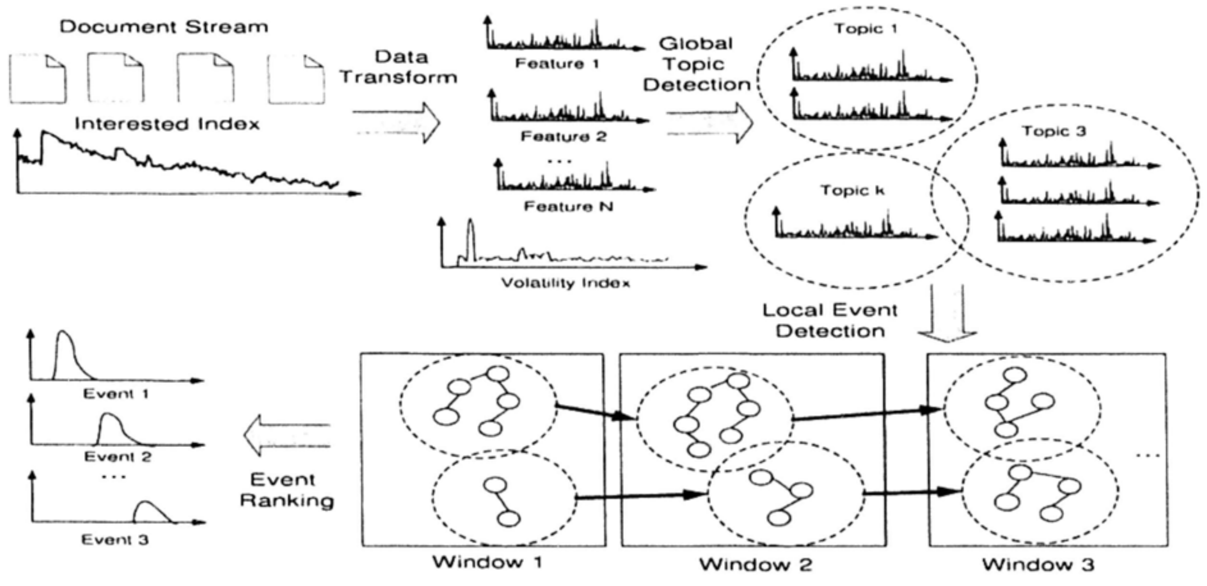


Figure 3.17: Framework

that our method can discover the priming events effectively.

In the rest of sections, Section 3.4.1 formulates the problem and gives an overview to our approach. Section 3.4.2 discusses how we detect bursty text features and measure the change of time series index. Section 3.4.3 describes the influential topic detection algorithm and Section 3.4.4 discusses how we use the influential topics to detect and rank priming events. Section 3.4.5 presents our experimental result. Section 3.4.6 concludes this work.

3.4.1. Problem Formulation

Let $\mathcal{D} = \{d_1, d_2, ...\}$ be a text corpus, where each document d_i is associated with a timestamp. Let I be the interested index, which consists of $|W|$ consecutive and non-overlapping time windows W. $\mathcal{D}$ is then partitioned into $|W|$ sets according to the timestamp of the documents. Let F be a set containing all different features in $\mathcal{D}$, where a feature $f \in F$ is a word in the text corpus.

Given the interested index I and the text corpus $\mathcal{D}$, our target is to detect the priming events that trigger the movement of the index. As discussed above, the first step is to discover the influential topics. A possible approach [25, 32] is to first retrieve all the topics from the documents using the traditional approach in topic detection and tracking (TDT). Then we detect the influential topics by comparing the strength of the topic with the change of the index over time. We argue that this approach is inappropriate because the topic detection is purely based on the occurrence of the words and ignores the behaviors of the index. Consider the feature *worker*. Typical TDT approach would consider its co-occurrences with other features when deciding to form a topic. By enumerating all the possibilities, it will form a topic with features such as *union* because *worker union* frequently appears in news documents. However, if we take the presidential approval index into consideration, the most important event about *worker* related to the index would be that President Bush increased the Federal Minimum Wage rate for workers ever since 1997. This event helped him to stop the continuous drop trend of the approval index and make it stay above 30%. Therefore, it is more favorable to group *worker* with *wage* rather than with *union*. This example urges us to consider how to leverage the information from the index to help us organize the features into influential topics. These influential topics take in not only the feature occurrence information but also the changing behavior of the index. We formally define such topics as follows.

Definition 1. ***(Influential Topics)*** *An influential topic θ_i is represented by a set of semantically coherent features $F_{\theta_i} \subseteq F$ with a score indicating its influence on the time series index I.*

Based on the definition of influential topics, the next step is to represent priming events using these topics. One simple and direct way is to take each occurrence of a topic θ_i as a priming event. However, this approach has one major problem. We observe multiple topics at a time window w and these topics are actually not independent but correlated and represent the same on-going event. Our topic detection algorithm may not merge them into a single topic because they only co-occur at that certain window but separate in other windows. For example, the topic of {strike target} would appear together with the topic {force troop afghanistan} in 2001. But in 2003, when the Iraq war starts, it co-occurs with the topic {gulf war} instead. Therefore, in order to capture such merge-and-separate behavior of topics, we define the local topic cluster as follows.

Definition 2. ***(Local Topic Cluster)*** *A local topic cluster $c_{w,i}$ in w consists of a set of topics which occur in highly overlapped documents to represent the same event.*

Based on the definition of local topic cluster, we further define a priming event as follows.

Definition 3. ***(Priming Event)*** *A priming event pe consists of three components: (1) Two timestamps to denote the beginning and the ending of the event in the window W_{pe}; (2) A sequence of local topic clusters $c_{w,i}, w \in W_{pe}$; (3) a score $Score(pe)$ representing its priming effect, i.e., how significant the event triggers the movement of the time series index.*

Our framework is outlined in Figure 3.17. There are three major steps: (1) Data Transformation, (2) Global Influential Topic Detection, (3) Local Priming Event Detection and Ranking. Details are given in the following sections.

3.4.2. Data Transformation

In this section, we present how we transform and normalize the features F and the index I.

We first determine the bursty probability of each feature using the technique introduced in Section 3.1.2. And let $P(f, w; p_e)$ represents the bursty rate of f in window w. With the transformation, we obtain the bursty probability time series for each feature $f \in F$ as $p(f) = \{p(f, 1), p(f, 2), \ldots, p(f, |W|)\}$ and the bursty windows of f are denoted as B_f.

We now discuss how to monitor the change of the index I to reflect the effect of priming events. According Section 3.1.4, we can transform the index time series I to the volatility of time series, $VI = \{VI_1, ..., VI_{|W|}\}$.

Given the volatility index VI, we observe that there are some abnormal behavior at certain time windows. For example, in the 911 event, there is a volatility burst for President Bush's approval index. Such phenomena will bring tremendous bias to the events happening in these volatility bursty windows. In order to avoid such bias, we further transform the volatility index VI to obtain a discrete representation with equal probability [54]. According to our experiment, the volatility index can fit into a logistic distribution. Therefore, we can produce a equal-sized area under the logistic curve [60] and transform the index volatility time series into a probabilistic time series $PVI = \{PVI_1, ..., PVI_{|W|}\}$.

3.4.3. Global Influential Topic Detection

Given the feature set F and the probability volatility index PVI, our task here is to identify a set of influential topics $\{\theta_1, ..., \theta_k\}$, where each topic θ_k is formed by a set of keywords $F_{\theta_k} = f_{\theta_k,1}, ..., f_{\theta_k,|\theta_k|}$. The problem can be solved by finding the optimal θ_k such that the probability of the influential bursty features grouped together is maximum for the text stream $\mathcal{D}$ and PVI. Below, we formally define the probability of θ_k:

Definition 4. ***(Influential Topic Probability)*** *The probability of an influential topic θ_k is given by*

$$P(\theta_k|\mathcal{D}, PVI) = \frac{P(PVI, \mathcal{D}|\theta_k)P(\theta_k)}{P(\mathcal{D}, PVI)}. \tag{3.27}$$

Since $P(\mathcal{D}, PVI)$ is independent of θ_k, we only consider the numerator of Eq. (3.27). We use the topic θ_k to account for the dependency between $\mathcal{D}$ and PVI. Therefore, given θ_k, $\mathcal{D}$ and PVI are independent. Our objective function then becomes:

$$\max P(PVI, \mathcal{D}|\theta_k)P(\theta_k) = \max P(PVI|\theta_k)P(\mathcal{D}|\theta_k)P(\theta_k), \tag{3.28}$$

Some observations can be made on Eq. (3.28). The second component $P(\mathcal{D}|\theta_k)$ represents the probability that the influential topic generates the documents. And intuitively, we expect the document overlap of the features from the same topic to be high. The third component of $P(\theta_k)$ represents the probability of the features to be grouped together. And two features should be grouped together if they usually co-occur temporally. Therefore, these two components basically require the features of θ_k to be coherent at both the document level and the temporal level. So generally, if more features are grouped together, the values of the second and the third components will decrease. And the first component $P(PVI|\theta_k)$ represents the probability that the influential topic triggers the volatility of the time series index. Obviously, if the features in the group cover more windows with high volatility probability, the value of the first component will be higher. This will make the algorithm look for the features with high potential impact on the index. Below, we show how we estimate these three components.

First, we define the document similarity of two features using Jaccard Coefficient [72].

$$sim(f_i, f_j) = \frac{|D_{f_i} \cap D_{f_j}|}{|D_{f_i} \cup D_{f_j}|}, \tag{3.29}$$

where D_{f_i} is the document set containing feature f_i. Then the $P(\mathcal{D}|\theta_k)$ can be estimated as below:

$$P(\mathcal{D}|\theta_k) = \frac{1}{|F_{\theta_k}|(|F_{\theta_k}| - 1)} \sum_{f_i, f_j \in F_{\theta_k}} sim(f_i, f_j). \tag{3.30}$$

Second, in order to compute $P(\theta_k)$, we estimate the temporal similarity of two features by comparing their co-occurrence over the whole set of windows W as below:

$$\rho(f_i, f_j) = \frac{|p(f_i) \cdot p(f_j)|}{|p(f_i)||p(f_j)|}, \tag{3.31}$$

where $p(f_i) = \{p(f_i, 1), p(f_i, 2), ..., p(f_i, |W|)\}$ is the bursty probability time series of f_i computed in Section 3.1.2. Then the probability of a set of features belonging to θ_k can be estimated by the average similarity for each pair of features:

$$P(\theta_k) = \frac{1}{|F_{\theta_k}|(|F_{\theta_k}| - 1)} \sum_{f_i, f_j \in F_{\theta_k}} \rho(f_i, f_j). \tag{3.32}$$

Finally, in order to estimate the $P(PVI|\theta_k)$, we define the influence probability for a feature $P(PVI|f)$ as:

$$P(PVI|f) = \frac{\sum_{w \in B_f} PVI_w}{\sum_{f \in F} \sum_{w \in B_f} PVI_w}. \tag{3.33}$$

Since the denominator of Eq.(3.33) holds the same for all the features, we just take the numerator for computation. And $P(PVI|\theta_k)$ can be estimated as

$$P(PVI|\theta_k) = \sum_{f \in F_{\theta_k}} P(PVI|f). \tag{3.34}$$

Finally, the topic can be extracted using a greedy algorithm by maximizing Eq. (3.28) for each topic in a similar way as in [25].

However, Eq. (3.28) is different from the objective function defined in [25] since we extract topics with respect to an interested time series index I rather than purely based on text documents. Consider the *worker* example again. The document similarity and the temporal similarity of *worker* and *union* are 0.31 and 0.25, while those of *worker* and *wage* are 0.35 and 0.1. If we do not consider the index I by setting $P(PVI|\theta_k) = 1$, by Eq. (3.33), $P(worker, union|\mathcal{D}, PVI) = 0.31 \times 0.25 = 0.0775$ and $P(worker, wage|\mathcal{D}, PVI) = 0.35 \times 0.1 = 0.035$. As a result, the algorithm would combine *worker* and *union*. However, since $P(PVI|union) = 30$ and $P(PVI|wage) = 74$, by considering the feature influence to the index I, we have $P(worker, union|\mathcal{D}, PVI) = 0.0775 * 30 = 2.325$ and $P(worker, union|\mathcal{D}, PVI) = 0.035 * 74 = 2.59$. In this way, the algorithm will instead group *worker* and *wage* together to make an influential topic with respect to I.

As shown above, since influential topics carry the volatility index information, it brings benefits to the priming event detection. In the above example, if we detect a new event containing the common topic {*worker, union*}, the event may be trivial since the topic has a low influence probability in the history. But if we detect an event with {*worker, wage*} instead, this event has a higher probability to be a priming event. This is because the high influence probability of the topic indicates that the events with such topic attracted the public attention and changed people's mind in the past.

After extracting each θ_k, the bursty rate of θ_k in a window w can be computed as below:

$$P(\theta_k, w) = \frac{1}{|F_{\theta_k}|} \sum_{f \in F_{\theta_k}} p(f, w) \tag{3.35}$$

The bursty period of θ_k is determined in a similar way to detecting the bursty period of features and we use θ_w to denote all the bursty topics in window w.

3.4.4. Micro Priming Event Detection

In this section, we describe how to detect priming events. A priming event pe consists of three components: (1) Two timestamps to denote the beginning and the ending of the event in the window W_{pe}; (2) A sequence of local correlated topic clusters $C_w, w \in W_{pe}$; (3) A score representing its influence on the index.

Local Topic Cluster Detection

As discussed in Section 3.4.1, we usually observe multiple correlated bursty topics at a time window w representing the same event. Therefore, we first group the correlated topics into topic clusters at each time window w.

Intuitively, if two topics θ_i and θ_j belong to the same event, the reporter usually discusses them in the same news article and they would have a high degree of document overlap. We first define the document frequency vector for a topic θ_k at a window w.

Let θ_w be a set of bursty topics in window w and D_w be the set of documents in window w. We define $D_{f,w} = \{d_{f,w,1}, d_{f,w,2}, ..., d_{f,w,|D_w|}\}$ be the term frequency vector for feature f in the documents in window w.

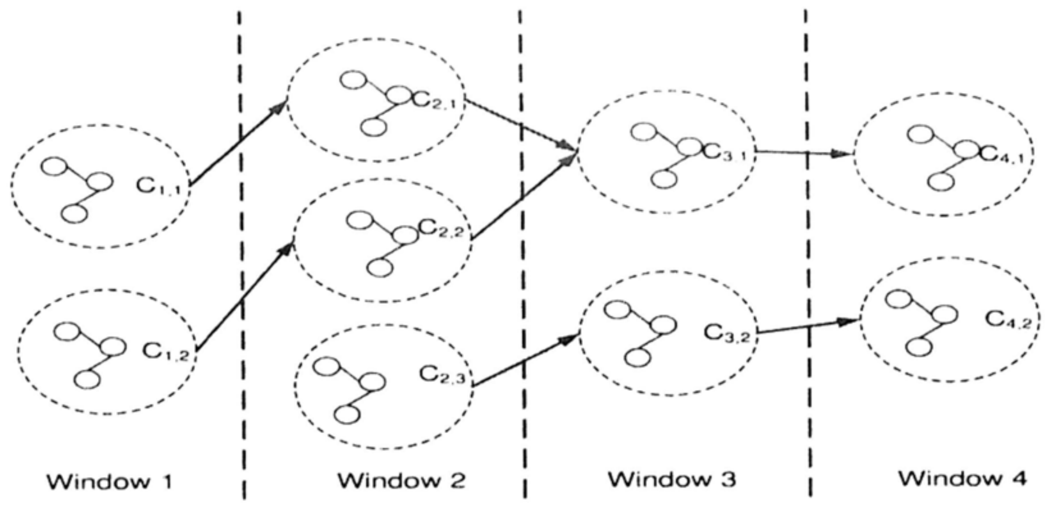


Figure 3.18: Evolving Topic Clusters

Then, the document frequency vector for a topic θ_k at a window w is computed by the average of the term frequency vector as below:

$$D_{\theta_k,w} = \frac{1}{|F_{\theta_k}|} \sum_{f \in F_{\theta_k}} D_{f,w}. \tag{3.36}$$

Then the similarity between two topics θ_i and θ_j at window w can be estimated by the cosine similarity:

$$sim(\theta_i, \theta_j, w) = \cos(D_{\theta_i,w}, D_{\theta_j,w}). \tag{3.37}$$

In order to cluster the set of topics θ_w into several topic clusters, we use the K-Means clustering algorithm [72]. We determine the optimal cluster number by examining the quality of the clustering result under different cluster number k. And the quality of the clustering is measured based on the ratio of the weighted average inter-cluster to the weighted average intra-cluster similarity [62]. After clustering, for a window w, we obtain a set of topic clusters $C_w = \{c_{w,1}, c_{w,2}, \ldots\}$.

Composite Topic Cluster Path Detection

A detected topic cluster $c_{w-1,i}$ represents the progress of a priming event at window $w-1$, and the following question is whether this event will keep developing in the next window w. Intuitively, if we find another topic cluster

$c_{w,j}$ which is similar to $c_{w-1,i}$, then we can claim that they belong to the same event and can associate them together.

We measure the similarity between two clusters in two consecutive windows, $c_{w,i}$ and $c_{w-1,j}$ by looking at their intersection of influential topics. More specifically, the similarity of two topic clusters can be computed by adding the topic probability $P(\theta_k|\mathcal{D}, PVI)$ as a weight to the Jaccard Coefficient.

$$sim(c_{w,i}, c_{w-1,j}) = \frac{\sum_{\theta_k \in c_{w,i} \cap c_{w-1,j}} P(\theta_k|\mathcal{D}, PVI)}{\sum_{\theta_k \in c_{w,i} \cup c_{w-1,j}} P(\theta_k|\mathcal{D}, PVI)}. \tag{3.38}$$

Eq. (3.38) assigns a higher similarity score to two topic clusters whose overlapping topics have higher topic probability, i.e., the topics are more coherent and influential.

Then we link two clusters together if their similarity score is higher than a threshold σ and form a directed acyclic graph (DAG) between windows. Figure 3.18 shows a topic cluster graph with 9 clusters linked by 7 edges lying on 4 consecutive windows. And we formally define a Path P in this graph as follows.

Definition 5. ***(Topic Cluster Path)*** *A topic cluster path P of length l in a topic cluster graph is a sequence of l clusters: $c_{w_1,i_1} \rightarrow c_{w_2,i_2} \rightarrow \ldots \rightarrow c_{w_l,i_l}$, such that $\{w_1, \ldots, w_l\}$ are l consecutive windows and there is an edge between two consecutive clusters in the graph.*

We can easily identify three cluster paths from the graph in Figure 3.18. Two paths starting from window 1, P_1: $c_{1,1} \rightarrow c_{2,1} \rightarrow c_{3,1} \rightarrow c_{4,1}$, P_2: $c_{1,2} \rightarrow c_{2,2} \rightarrow c_{3,1} \rightarrow c_{4,1}$ and another path starting from window 2, P_3: $c_{2,3} \rightarrow c_{3,2} \rightarrow c_{4,2}$. We can see that P_1 and P_2 have two overlapping clusters $c_{3,1}$ and $c_{4,1}$. This indicates that although these two cluster paths initial from different topic clusters, they may express different aspects of the same priming event. For example, in the gulf war event, one topic cluster path may show the progress of the battle in Iraq, while another path may record the actions from US's allay. Therefore, we measure the similarity between two overlapping paths as follows:

$$sim(P_i, P_j) = \frac{|P_i \cap P_j|}{\min(|P_i|, |P_j|)}. \tag{3.39}$$

If the similarity between two paths $sim(P_i, P_j)$ is higher than a threshold γ, then we group these two paths and form a priming event.

Algorithm 8 DiscoverPrimingEvents

INPUT: News document stream D and index I

OUTPUT: priming events PE_w

```
1: loop
2:   Move $w$ and retrieve the burst topics at $w$, $\theta_w$
3:   $C_w \leftarrow KMeans(\theta_w)$
4:   for each $c_{w,i} \in C_w$ do
5:     $newPath = true$
6:     for every $c_{w-1,j} \in C_{w-1}$ do
7:       Compute topic similarity $cSim(c_{w,i}, c_{w-1,j})$ according to Eq. 3.38
8:       if $sim(c_{w,i}, c_{w-1,j}) > \sigma$ then
9:         for each path $P_k$ in the inverted path list $IP_{c_{w-1,j}}$ do
10:          Add $c_{w,i}$ to $P_k$
11:          Add $P_k$ to inverted path list of $c_{w,i}$, $IP_{c_{w,i}}$
12:          $newPath = false$
13:        end for
14:      end if
15:    end for
16:    if $newPath == false$ then
17:      for every path pair $P_m$,$P_n$ in $IP_{c_{w,i}}$ do
18:        add $c_{w,i}$ to $InterSect(P_m, P_n)$
19:      end for
20:    else
21:      generate a new Path $P_k$
22:      add $P_k$ to inverted path list $IP_{c_{w,i}}$
23:    end if
24:  end for
25:  Output $PE_w = GenerateEvent(P, intersect)$
26: end loop
```

Algorithm 9 outputPrimingEvents

```
INPUT: Topic Cluster Path $P$ and path intersection record $Intersect$
OUTPUT: priming events $PE$
1: compute $sim(P_i, P_j)$ for every $P_i, P_j \in P$ according to Eq.3.39
2: repeat
3:    merge $P_n$ into $P_m$
4:    for every other path $P_r \neq P_m$ do
5:       $intersect(P_m, P_r) = intersect(P_m, P_r) \cup intersect(P_n, P_r)$
6:       update $sim(P_m, P_r)$
7:    end for
8: until No path pair with $sim(P_m, P_n) > \gamma$
9: return $PE = P$
```

The process is described in Algorithm 8. Let w be the newly arrival window and θ_w be the influential topics bursty in this window. Line 3 groups the topic into topic clusters C_w. From line 6, for each cluster in C_w, $c_{w,i}$, we compare $c_{w,i}$ with all the topic cluster in the last time window by computing cluster similarity according to Eq. (3.38) (Note if w is the first window, we would skip this step.) In lines 8-14, we link two clusters $c_{w,i}$ and $c_{w-1,j}$ if their similarity is higher than σ and extend all the paths in the path inverted list of $c_{w-1,j}$ to $c_{w,i}$. Besides, we also maintain the inverted path list of $c_{w,i}$, $IP_{c_{w,i}}$. In lines 16-20, if $c_{w,i}$ belongs to one of paths in $w-1$, we maintain the path node intersection in lines 19-23 for further processing. But if it is not an extension from any path in $w-1$, we generate a new path for it. Finally, the *outputPrimingEvents* algorithm identifies and ranks the priming events in line 25.

Algorithm 9 shows how to process the path information and eventually output the priming events. In line 1, we compute the path similarity using the path node intersection information. Then in lines 2-8, we merge two paths if their similarity is higher than γ. After merging, we update the path intersection information for the new list, as well as the path similarity. The algorithm terminates when there is no path that can be merged. The final path list is the priming events.

Priming Event Ranking

Let $\theta_{pe,w}$ be the set of influential topics contained in a priming event *pe* at window w. We describe how to rank the events according to their influence on the index. If the priming event is important, it must satisfy the following conditions:

1. High Event Intensity $B_{pe,w}$ for $w \in W_{pe}$: The event must contain influential topics that have high bursty effect. The intensity is estimated as follows:

$$B_{pe,w} = \sum_{\theta_k \in \theta_{pe,w}} P(\theta_k, w)P(\theta_k|\mathcal{D}, PVI). \tag{3.40}$$

2. High Index Volatility PVI_w: The index must have high volatility during the bursty period of this event.

We combine these two conditions and define the priming rate of the priming event as below:

$$Score_(pe) = \sum_{w \in W_{pe}} B_{pe,w} \cdot PVI_w. \tag{3.41}$$

A ranking list of the priming events PE can then be obtained according to the score of each event.

3.4.5. Evaluation

We archive the President Approval rating index and the news articles from the Gallup Poll[4] and ProQuest database[5] from Jan. 1, 2001 to Feb. 28, 2010, respectively.

In ProQuest, we take "President Bush" and "President Obama" as the query keywords and extract 15,542 and 1,643 news articles, respectively. For the preprocessing of these news articles, all features are stemmed using the Porter stemmer. Features are words that appear in the news articles, with the exception of

[4]`http://www.gallup.com`

[5]`http://www.proquest.com/`

Topics	Bush	Obama
θ_1	bin laden	file bankruptcy
θ_2	north Korean	Israel peace
θ_3	Israel palestinian	finance regulation
θ_4	immigration illegal	mortgage loan
θ_5	destruct mass	small business lend

Table 3.5: Influential Topics

digits, web page address, email address and stop words. Features that appear more than 80% of the total news articles in a day are categorized as stop words. Features that appear less than 5% of the total news articles in a day are categorized as noisy features. Both the stop words and noisy features are removed.

For President Bush's Approval Rating, the poll is taken every 10 days approximately. And for President Obama, we take a weekly average rating based on the Gallup daily tracking. We further identify the bursty probability time series and volatility time series according to the methods introduced in Section 3.1.2 and Section 3.1.4.

After these preprocessing, we have two real datasets.

- **Bush**. It contains 1186 bursty feature probability streams and 1 volatility time series with equal length of 281 in the 8 years of Bush's administration.
- **Obama**. It contains 1197 bursty feature probability streams and 1 volatility time series with equal length of 56 in the 13 months of Obama's administration.

We implemented our framework using C# and performed the experiments on a PC with a Pentium IV 3.4GHz CPU and 3GB RAM.

Identifying Influential Topics

With the algorithm in Section 3.4.3, we identify 385 and 207 influential topics from Bush and Obama, respectively. Table 3.5 gives the top 5 influential

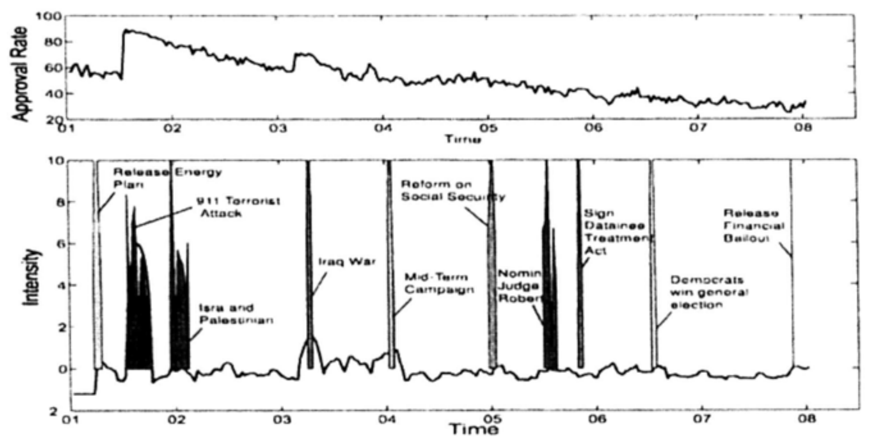


Figure 3.19: Top 10 Priming Events of President Bush

topics and the rank is based on the influential topic probability in Eq. (3.27). As shown in the second column of the table, {bin laden}, {north Korea} are 2-gram names which are regarded as the largest threaten for Bush's administration from 2001-2008. Each of the other three topics consists of a set of features which are not 2-gram names but coherent and influential keywords in the document stream. The third column of the table shows the influential topics from Obama. Compared with Bush's where 4 out of 5 topics are about international affairs, there are significant evolution of the influential topics of Obama's topics. In particular, only the second topic of Obama is about the international issue, i.e., peace progress of Israel. Others are all about how obama deals with financial tsunami in domestic: The first topic {file bankrupty} and the fourth topic {mortgage loan} are about the crisis itself, i.e., Obama supported the bankruptcy of Chrysler and General Motors in April and June, 2009 and the mortgage loan issue; while the third topic {finance regulation} and the fifth topic {small business lend} are about his policy on solving the crisis, i.e., regulating the finance industry and lending to small business. As discussed before, these influential topics detected at a global level give us evidences in detecting priming events at a micro level.

Identifying Priming Events

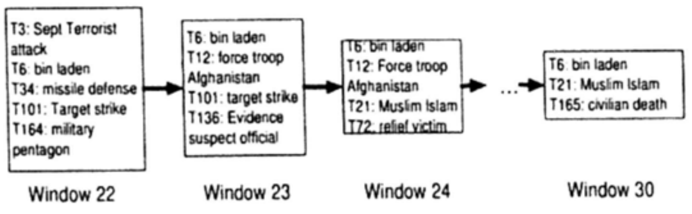


Figure 3.20: Priming Event: 911 Terrorist Attack

With the influential topics, we can identify the priming events with by the algorithm in Section 3.4.4. Figure 3.19 shows the approval index of President Bush and the top 10 priming events automatically detected over his eight years administration. As shown in the upper part of Figure 3.19, the approval index starts from 57% and has a big jump up to 90% in Sep 2001. Then it drops quickly until a rebound back to 70% in 2003. After that, it continues dropping with some small rebound in the middle and eventually reaches 34% in 2008. In the lower part, the blue line shows the volatility index according to Section 3.4.2 and the colored waves represent the ranked priming events (we normalize the value of these two time series and plot them in same graph). The rank is based on the score given by Eq. (3.41) and the wave with deeper color represents the priming event with a higher rank. The magnitude of the wave represents the intensity of the event according to Eq. (3.40). From the figure, we have two obvious observations: 1) the value of the volatility index increased when a significant trend change happened for the approval index. 2) During the periods when the volatility index increased significantly, we detect a burst of priming events. For example, after September 2001, we observe a significant increase of the volatility index reflecting the big jump of the approval index. At the same time, we detect the top 1 priming event about the 911 terrorist attack. Figure 3.20 further shows its structure which contains a composite topic cluster path with a length of 9 starting from window 22 to window 30. In each window, the path contains a topic cluster with a set of topics lying in a highly overlapped document set. For example, in window 22, the cluster consists of 5 bursty topics including the influential topics of {bin laden} and {Sept Terrorist Attack} that

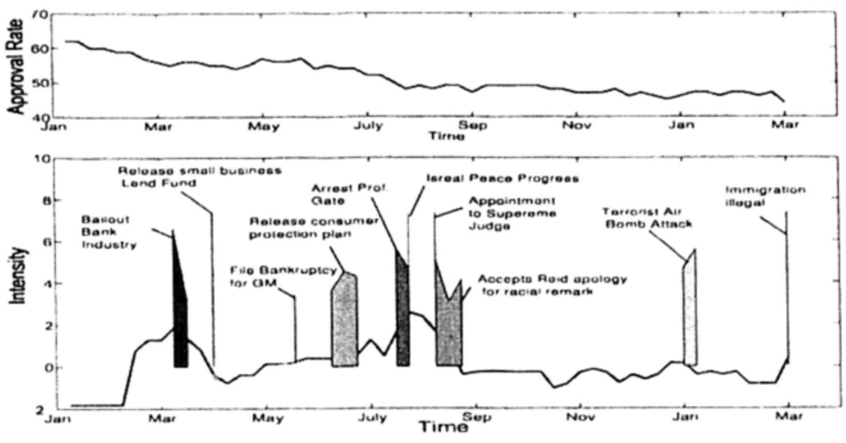


Figure 3.21: Top 10 Priming Events of President Obama

indicate the start of the 911 event. After that, in window 23, we observe another cluster containing {bin laden} and we connect it with the previous one since {bin laden} has a high influential topic probability and dominates the topic similarity measure according to Eq. (3.38). We can also see a certain degree of evolution between these two topic clusters since the second cluster contains a new topic {force troop Afghanistan} that is about U.S. sending troop to start the war. In the following windows, we can see the topic clusters evolute but all contain the influential topic {bin laden}. This event ends in window 30 with a topic {civilian death} indicating that the war results in the civilian death of the country. From this priming event, we can see that the attack makes the U.S. people united and support their President. Similarly, the volatility index also reflects the rebounding of the approval index in 2003. And the priming event covering that period is about the Iraq war with the ending of U.S. victory which increased the public support of President Bush. Other periods with significant increasing volatility such as those in Mar 2001 and Jul 2004 can also be explained by the detected priming events, i.e., President Bush released the energy plan and won the mid-term campaign over John Kerry.

Figure 3.21 shows the approval index of President Obama and the top 10 priming events detected over his 13 months administration. As discussed before,

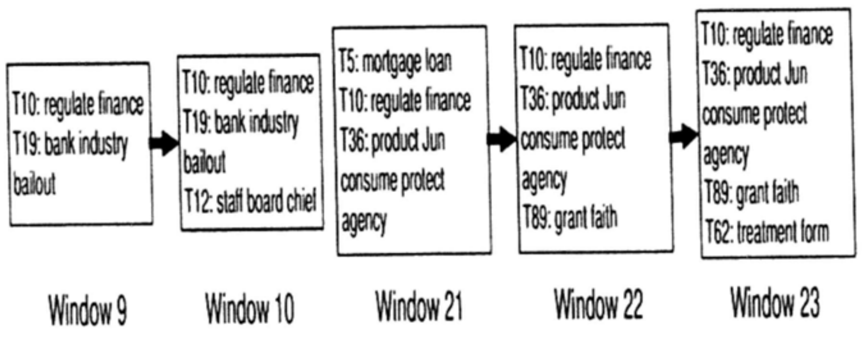


Figure 3.22: Two Priming Events of Finance Regulation

users may be curious about what happened in the two quick drop periods in March and July, 2009. In the lower part of Figure 3.21, we can see that the volatility corresponding to these two drops increased and we detect two priming events whose waves cover these two periods: President Obama's bailout policy on bank industry and his criticism of the police who arrested Prof. Gate. On the other hand, the algorithm also detects that his reform on lending to small business and the approval on bankruptcy of GM help his approval index to stabilize and even win back some points from the public in April and May, 2009, respectively.

In addition, among the top 10 priming events, we observe that two of them both contain the topic {regulate finance}. Figure 3.22 shows the event structure of them. As we can see, the first priming event is about the bank industry bailout program we just mentioned with a composite topic cluster path of length 2. In addition to the topic {regulate finance}, it also contains a more specific topic {bank industry bailout}, indicating the details of the finance regulation reform. The second event happened in June and has a composite topic cluster path of length 3. In addition to the topic {regulate finance}, it also contains a topic {product Jun consume protect angency}, indicating that it is about President Obama's release of the consumer protection plan, another form of the finance regulation. Therefore, we can conclude that, by detecting the highly influential topic such as {regulate finance} at a global level, we can form more specific priming events by integrating with other topics at a micro level.

Our experimental results justify that the priming events detected for President Bush and Obama are able to explain most of the index volatility bursty periods in their administrations.

3.4.6. Summary

In this work, we study the problem of detecting priming events based on a time series index and an evolving document stream. We measure the effect of the priming events by the volatility rate of the time series index. We propose a two-step framework to detect the priming events by first detecting influential topics at a global level and then forming priming events using detected influential topics at a micro level. The experimental result on the presidential approval rating shows that our algorithm is able to detect the priming events that trigger the movement of the rating effectively. For the future work, we plan to extend our algorithm to other application domains, such as detecting priming events for financial tsunami and sales volume of a company.

3.5. Chapter summary: time series stream and news stream

It is a very attractive and challenging problem to discover patterns from the co-evolving news stream and time series stream and apply the patterns to financial applications. We introduced three techniques which investigate these possibilities. First, we presented a NHMM model which integrate both news and price information into stock price trend prediction. Second, we presented a technique which predict the stock volatility introduced by news events. Third, we introduced an event extraction algorithm that detects and organizes priming event which pose great impact on time series movement. Through extensive experiment on real dataset, we conclude that it is possible to detect promising patterns from news stream and time series stream and support advanced financial applications.

CHAPTER 4

CONCLUSION

In this thesis, we focus on discovering patterns from two types of data streams, the time series index stream and news stream. We started from investigating the co-movement relationship of multiple time series. We introduced techniques to study two aspects of this problem. First, we proposed a co-movement model for constructing financial portfolio by analyzing and mining the co-movement patterns among two time series. Second, we presented an efficient streaming algorithm to discover leaders from multiple time series stream. Both of the algorithms are evaluated using real time series indices data and the result proves that co-movement patterns and detected leaders are promising and can support various applications including portfolio management, high frequency trading and risk management. Then, we studied the patterns between news stream and time series indices stream. We transformed the news stream into a set of bursty feature (keywords) time series streams and proposed three technique to study their relationship to time series index. First, we explored a Non-homogeneous Hidden Markov Model (NHMM) to predict the stock market process which takes both stock prices and news articles into consideration. Second, we proposed a risk analytical model to predict the volatility of price indices by integrating news information. Finally, we devised an algorithm to detect the priming event from text and a time series index. The evaluation on real world dataset suggests that the significant correlation exists between news stream and time series stream and our pattern discover algorithm can detect promising patterns from this relationship and support real world applications effectively.

BIBLIOGRAPHY

[1] Patricia A. Adler and Peter Adler. The market as collective behavior. In *The Social Dynamics of Financial Markets*, pages 85–105. 1984.

[2] James Allan. *Topic Detection and Tracking: Event-based Information Organization.* Kluwer, 2002.

[3] Carl D. Meyer Amy N. Langville. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[4] V. S. Bawa and E. B. Lingenberg. Capital market equilibrium in a mean-lower partial moment framework. *Journal of Financial Economics*, 5(2):189–200, 1977.

[5] Rafiqul Bhuyan. Information, alternative markets, and security price processes: A survey of literature. Finance 0211002, EconWPA, 2002.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] Herbert Blumer. Outline of collective behavior. In *Readings in Collective Behavior*, pages 22–45. Second edition, 1975.

[8] Zvi Bodie, Alex Kane, and Alan J. Marcus. *Investments.* Chicago: Irwin, third edition, 1996.

[9] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, April 1986.

[10] George Box, Gwilym M. Jenkins, and Gregory Reinsel. *Time Series Analysis: Forecasting and Control.* Prentice Hall, 1994.

[11] Michael J Brennan, Narasimhan Jegadeesh, and Bhaskaran Swaminathan. Investment analysis and the adjustment of stock prices to common information. *Review of Financial Studies*, 6(4):799–824, 1993.

[12] Richard P. Brent. *Algorithms for Minimization Without Derivatives.* Dover Publications, 2002.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Comput. Netw. ISDN Syst.*, 30(1–7):107–117, 1998.

[14] John Y Campbell, Sanford J Grossman, and Jiang Wang. Trading volume and serial correlation in stock returns. *The Quarterly Journal of Economics*, 108(4):905–939, November 1993.

[15] Kalok Chan. A further analysis of the lead-lag relationship between the cash market and stock index futures market. *Review of Financial Studies*, 5(1):123–152, 1992.

[16] Dietmar H. Dorr and Anne M. Denton. Establishing relationships among patterns in stock market data. *Data & Knowledge Engineering*, 2008.

[17] Louis H. Ederington and Jae Ha Lee. The Short-Run Dynamics of the Price Adjustment to New Information. *SSRN eLibrary*.

[18] Claude B. Erb, Campbell R. Harvey, and Tadas E. Viskanta. Forecasting international equity correlations. *Financial analysis Journal*, 50(5):32–45, 1994.

[19] Eugene F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105, 1965.

[20] Tom Fawcett and Foster J. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proc. of KDD'99*, pages 53–62, 1999.

[21] Leon Festinger. *A theory of cognitive dissonance.* Stanford, Calif.: Stanford University Press, Reprinted in 1968.

[22] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. News sensitive stock trend prediction. In *Proc. of PAKDD'02*, pages 481–493, 2002.

[23] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Huan Liu, and Philip S. Yu. Time-dependent event hierarchy construction. In *KDD*, pages 300–309, 2007.

[24] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu. The predicting power of textual information on financial markets. *IEEE Intelligent Informatics Bulletin*, 5(1):1–10, 2005.

[25] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. Parameter free bursty events detection in text streams. In *Proc. of VLDB'05*, pages 181–192, 2005.

[26] Xianping Ge and Padhraic Smyth. Deformable markov model templates for time-series pattern matching. In *KDD*, pages 81–90, 2000.

[27] Ramazan Gencay, Michel Dacorogna, Ulrich A. Muller, Olivier Pictet, and Richard Olsen. *An Introduction to High-Frequency Finance.* Academic Press, 2001.

[28] Greenspan. Excerpts from greenspan speech on global turmoil. In *The Markets, reprinted in The New York Times, November 6*, 1998.

[29] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *NIPS*, 2004.

[30] Daniel Gruhl, Ramanathan V. Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. The predictive power of online chatter. In *KDD*, pages 78–87, 2005.

[31] W.V. Harlow. Asset allocation in a downside-risk framework. *Financial analysis Journal*, 47(5):28–40, 1991.

[32] Qi He, Kuiyu Chang, and Ee-Peng Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214, 2007.

[33] Qi He, Kuiyu Chang, and Ee-Peng Lim. Analyzing feature trajectories for event detection. In *Proc. of SIGIR'07*, pages 207–214, 2007.

[34] T. Hellstrom and K. Holmstrom. Predicting the stock market, 1998.

[35] W. J. Holmes and M. J. Russell. Probabilistic-trajectory segmental hmms. *Computer Speech and Language*, 13:3–38, 1999.

[36] J. P. Hughes, P Guttorp, and S. P. Charles. A non-homogeneous hidden Markov model for precipitation occurrence. *Applied Statistics*, 48(1):15–30, 1999.

[37] Tsuyoshi Idé and Hisashi Kashima. Eigenspace-based anomaly detection in computer systems. In *KDD*, pages 440–449, 2004.

[38] Tsuyoshi Idé, Spiros Papadimitriou, and Michail Vlachos. Computing correlation anomaly scores using stochastic nearest neighbors. In *ICDM*, pages 523–528, 2007.

[39] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of 10th European Conference on Machine Learning (ECML'98)*, pages 137–142, Chemnitz, Germany, 1998.

[40] Daniel Jurafsky and James H. Martin. *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice-Hall, 2000.

[41] JJ Reuer KD Miller. Measuring organizational downside risk. *Strategic Management Journal*, 17(9):671–691, 1996.

[42] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

[43] Seyoung Kim, Padhraic Smyth, and Stefan Luther. Modeling waveform shapes with random effects segmental hidden markov models. In *Proc. of the 20th conference on Uncertainty in artificial intelligence*, pages 309–316, 2004.

[44] Sergey Kirshner. *Modeling of multivariate time series using hidden Markov models*. PhD thesis, Univedrsity of California, Irvine, 2005.

[45] Michael Klausner. Sociological theory and the bechavio of financial markets. In *The Social Dynamics of Financial Markets*, pages 57–81. 1984.

[46] F. Klein and John A. Prestbo. *News and the Market.* Chicago: Henry Regenry, 1974.

[47] Jon M. Kleinberg. Bursty and hierarchical structure in streams. In *KDD*, pages 91–101, 2002.

[48] Kazuhiro Kohara, Tsutomu Ishikawa, Yoshimi Fukuhara, and Yukihiro Nakamura. Stock price prediction using prior knowledge and neural networks. In *INTELLIGENT SYSTEMS IN ACCOUNTING, FINANCE AND MANAGEMENT VOL.*, 1997.

[49] Victor Lavrenko, Matthew D. Schmill, Dawn Lawire, Paul Ogivie, David Jensen, and James Allan. Mining of Concurrent Text and Time Series. In *Proc. of KDD'00 Workshop on Text Mining*, 2000.

[50] Henriksson R D. Leibowitz M L. Portfolio optimization with shortfall constraints:a confidence-limit approach to managing downside risk. *Financial analysis Journal*, 43(1):34–41, 1989.

[51] Kogelman S. Leibowitz M L. Asset allocation under shortfall constraints. *Journal of Portfolio Management*, 17(2):18–23, 1991.

[52] Jure Leskovec, Lars Backstrom, and Jon M. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, pages 497–506, 2009.

[53] A. Lewis. Semivariance and the performance of portfolios with options. *Financial analysis Journal*, 46.

[54] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, pages 2–11, 2003.

[55] David G. Luenberger. *Investment Science.* Prentice Hall, 1997.

[56] Harry Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[57] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *WWW*, pages 101–110, 2008.

[58] E. Meijering. Chronology of interpolation: From ancient astronomy to modern signal and image processing. In *Proc. of the IEEE*, pages 319–342, 2002.

[59] Marc-Andre Mittermayer and Gerhard F. Knolmayer. Newscats: A news categorization and trading system. In *ICDM '06*, pages 1002–1007, 2006.

[60] Douglas C. Montogomery and George C. Runger. *Applied Statistics and Probability for Engineers.* John Wiley & Sons, Inc., second edition, 1999.

[61] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximunm entropy for text classification. In *Proc. of the 16th International Joint Conference Workshop on Machine Learning for Information Filtering*, 1999.

[62] Tansel Özyer, Reda Alhajj, and Ken Barker. Clustering by integrating multi-objective optimization with weighted k-means and validity analysis. In *IDEAL*, pages 454–463, 2006.

[63] Qi Pan, Hong Cheng, Di Wu, Jeffrey Xu Yu, and Yiping Ke. Stock risk mining by news. In *ADC*, 2010.

[64] Spiros Papadimitriou, Jimeng Sun, and Philip S. Yu. Local correlation tracking in time series. In *ICDM*, pages 456–465, 2006.

[65] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR'98*, pages 275–281, 1998.

[66] Chotirat (Ann) Ratanamahatana, Eamonn J. Keogh, Anthony J. Bagnall, and Stefano Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In *PAKDD*, pages 771–777, 2005.

[67] Calum Robertson, Shlomo Geva, and Rodney C. Wolff. Can the content of public news be used to forecast abnormal stock market behaviour? In *ICDM*, pages 637–642, 2007.

[68] Patrik Säfvenblad. Lead-lag effects when prices reveal cross-security information. Working Paper Series in Economics and Finance 189, Stockholm School of Economics, September 1997.

[69] Yasushi Sakurai, Spiros Papadimitriou, and Christos Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[70] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., 1986.

[71] Alexander Sharpe and Bailey. *Investments.* Prentice Hall, 1999.

[72] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* Addison-Wesley, 2006.

[73] James D. Thomas and Katia Sycara. Integrating genetic algorithms and text learning for financial prediction. In *Proc. of the Genetic and Evolutionary Computing 2000 Conference Workshop on Data Mining with Evolutionary Algorithms*, 2000.

[74] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms.* MIT Press and McGraw-Hill, 2001.

[75] Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *KDD*, pages 784–793, 2007.

[76] Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD*, pages 424–433, 2006.

[77] Beat Wüthrich. Probabilistic knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):691–698, 1995.

[78] Di Wu, Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Zheng Liu. Integrating multiple data sources for stock prediction. In *WISE*, pages 77–89, 2008.

[79] Di Wu, Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Zheng Liu. Mining multiple time series co-movements. In *APWeb*, pages 572–583, 2008.

[80] Di Wu, Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Qi Pan. Stock prediction: an event-driven approach based on bursty keywords. *Frontiers of Computer Science in China*, 3(2):145–157, 2009.

[81] Di Wu, Yiping Ke, Jeffrey Xu Yu, Philip S. Yu, and Lei Chen 0002. Leadership discovery when data correlatively evolve. *World Wide Web, Accepted with minor revision.*

[82] Di Wu, Yiping Ke, Jeffrey Xu Yu, Philip S. Yu, and Lei Chen 0002. Detecting leaders from correlated time series. In *DASFAA*, pages 352–367, 2010.

[83] Beat Wuthrich, D Permunetilleke, S. Leung, Vincent Cho, J. Zhang, and Wai Lam. Daily prediction of major stock indices from textual www data. In *Proc. of KDD'98*, pages 364–368, 1998.

[84] Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.