Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

# DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Petr Zadražil**

Study programme: Open Informatics

Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Using Markov models and textual news data in financial series prediction**

**Guidelines**:

1. Design an approach allowing to exploit information stored in textual data (mainly news) for learning time series predictors. The method will define a transformation from the textual representation to one suitable for machine learning algorithms.
2. Perform a multi-factor evaluation of the background knowledge impact on the performance of the prediction (a factor correponds e.g. to the choice of the machine learning algorithm, the input data category, or the validation scenario).
3. Explore and/or improve several variations of the Markov Model paradigm (such as hierarchical, layered, second-order) for financial data prediction or for decision-making on such data, and establish their performance ranking experimentally.

**Bibliography/Sources**:

Brockwell P. J., Davis R. A., Introduction to Time Series and Forecasting, New York, 2002 (Springer)
Schelter B., Winterhalder M., Timmer J., Handbook of Time Series Analysis, Weiheim, 2006 (Wiley)
Tsay R. S., Analysis of Financial Time Series (Third), 2010 (Wiley)
Gupta N., Hauser R., Forecasting Financial Time-Series using Artificicial Market Models, 2005
Brockwell P.J., Davis A.D., Time_Series - Theory And Methods SE, SPR, 2006
Holger K., Schreiber T., Nonlinear Time Series Analysis, Dresden, 2002 (CUP)

**Diploma Thesis Supervisor**: Doc.Ing. Filip Železný, Ph.D.

Valid until to the end of the summer semester 2011/2012.

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická LS

prof. Ing. Boris Šimák, CSc.
Dean

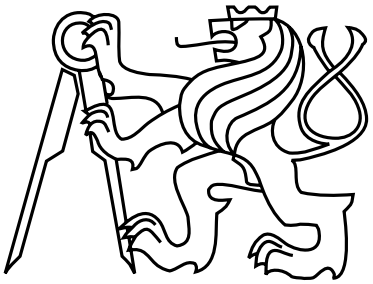
Prague, March 25, 2011

# Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on January 3, 2012 .....

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

# Using Markov Models and Textual News Data in Financial Series Prediction

*Bc. Petr Zadražil*

Supervisor: doc. Ing. Filip Železný, Ph.D

Study Programme: Artificial Intelligence

Field of Study: Open Informatics

January 3, 2012

# Aknowledgements

I would like to thank the following people:

- My supervisor doc. Ing. Filip Železný, Ph.D for all the inspiration and patience.
- Doc. Dr. Boris Flach for providing a great insight into the world of Markov Models.
- Ing. Libor Grafnetr for valuable remarks and the thesis idea.
- All my friends and family for support and encouragement.

# Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on January 3, 2012 .....

# Abstract

This thesis covers analysis of financial markets and experimental evaluation of a possibility of automated trading. The major part of the thesis describes design, implementation and testing of two machine learning tools. The first one predicts market movements using newspaper articles, whereas the second one uses *Hidden Markov Models* for strictly technical analysis. A brief overview and comparison of other frequently used approaches is also included.

# Abstrakt

Práce se zabývá analýzou finačních trhů a experimentálním zhodnocením možností automatizovaného obchodování. Stěžejní část spočívá v návrhu, implementaci a testování dvou systémů strojového učení. První z nich slouží k predikci pohybu trhů s pomocí novinových článků, zatímco druhý využívá *Skryté Markovovy Modely* k čistě technické analýze. Práce rovněž zahrnuje stručný přehled a porovnání dalších často používaných technik.

# Contents


**1 Introduction** **1**
- 1.1 Motivation . . . 2
- 1.2 Objectives . . . 2
- 1.3 Overview . . . 2

**2 Background** **4**
- 2.1 Markets . . . 5
- 2.2 Systems for Algorithmic Trading . . . 7
- 2.3 Text Mining Integration . . . 14
- 2.4 Other Used Algorithms . . . 18

**3 Design** **20**
- 3.1 Simplifying Assumptions . . . 21
- 3.2 Text Integration . . . 21
- 3.3 Time-Series Analysis . . . 24

**4 Implementation** **27**
- 4.1 Implementation Environment . . . 28
- 4.2 System Design . . . 29
- 4.3 Article Retrieval . . . 31
- 4.4 Text Processing . . . 32
- 4.5 Article Preprocessing . . . 32
- 4.6 Data Series Processing . . . 33
- 4.7 Other Software Tools . . . 39

**5 Testing and Results** **41**
- 5.1 Data Selection . . . 42
- 5.2 Text Analysis . . . 42
- 5.3 Markov Models . . . 49

**6 Future Work** **59**

**7 Conclusion** **61**

**A List of Abbreviations** **68**

**B NY Downloader User Guide** **70**
B.1 Installation . . . . . 71
B.2 Usage . . . . . 71

**C Validation Framework User Guide** **72**
C.1 Installation . . . . . 73
C.2 Usage . . . . . 73

**D Additional Plots and Tables** **74**

**E DVD Content** **87**

# List of Figures


2.1 Examples of simple strategies based on patterns or indicators . . . . . 8
(a) A Head & Shoulders pattern example . . . . . 8
(b) A Moving Average example . . . . . 8

3.1 Textual Data Analysis Flow . . . . . 21
3.2 Time-Series Analysis Flow . . . . . 24

4.1 Examples of application candidates . . . . . 29
(a) A Matlab screen example . . . . . 29
(b) Example usage of the RapidMiner . . . . . 29
4.2 Screenshot of the *NY Downloader* . . . . . 40

5.1 The standard deviation of a relative difference . . . . . 43
5.2 Simple HMM Models Logarithmic Likelihood . . . . . 50
(a) Relative Difference . . . . . 50
(b) Logarithmic Difference . . . . . 50
5.3 Simple HMM models direction prediction . . . . . 50
(a) Relative Difference . . . . . 50
(b) Logarithmic Difference . . . . . 50
5.4 The first iteration of the two best trading HMMs . . . . . 53
(a) 8 State Relative Difference HMM . . . . . 53
(b) 3 State Relative Difference HMM . . . . . 53
5.5 Learning Together Performance . . . . . 53
(a) Direction Accuracy . . . . . 53
(b) 3 States Simple Threshold Trading . . . . . 53
5.6 Examples of application candidates . . . . . 57
5.7 Trading Performance of Models During Validation . . . . . 57
5.8 Value Distributions for Individual *HMM* States . . . . . 58

B.1 Screenshot of the *NY Downloader* . . . . . 71

# List of Tables


2.1 ILP Comparison: Percent of correct "up" and "down" estimates. . . . . 10
2.2 ILP Comparison: Gain per year in simulated trading. . . . . 10
2.3 Sentiment analysis results . . . . . 16
2.4 Performance of the system proposed by Fung, Yu and Lam . . . . . 17
2.5 AZFinText: Closeness Results . . . . . 17
2.6 AZFinText: Directional Accuracy Results . . . . . 18
2.7 AZFinText: Simulated Trading Engine Results . . . . . 18

5.1 Correctly Classified Yahoo Articles Using a Simple SVM Model . . . . . 45
(a) Relative Difference 1 . . . . . 45
(b) Relative Difference 5 . . . . . 45
5.2 Correctly Classified Yahoo Articles After Filtering . . . . . 46
(a) Technology Business Filtering . . . . . 46
(b) Unique Filtering . . . . . 46
5.3 Correctly Classified Yahoo Articles Using Naive Bayes . . . . . 46
5.4 Correctly Classified Yahoo Articles After Filtering . . . . . 47
(a) Technology Business Filtering . . . . . 47
(b) Unique Filtering . . . . . 47
5.5 Correctly Classified Yahoo Articles Using Naive Bayes . . . . . 47
5.6 Correctly Classified Yahoo Articles After Term Reduction . . . . . 48
(a) Term Reduced to 10% . . . . . 48
(b) Term Reduced to 20% . . . . . 48
5.7 Validation of Article Classification . . . . . 48
5.8 Average Daily Profit of the Confidence Strategy . . . . . 51
(a) Relative Difference . . . . . 51
(b) Logarithmic Difference . . . . . 51
5.9 Average Daily Profit of the Threshold Strategy . . . . . 52
(a) Relative Difference . . . . . 52
(b) Logarithmic Difference . . . . . 52
5.10 Parameters of the Final Models . . . . . 56
5.11 Parameters of the Final Models . . . . . 56

D.1 Number of articles related to given subject . . . . . 75
D.2 Ratio of Positive/Negative articles by Category . . . . . 76
D.3 Average Daily Profit of Confidence Strategy if learnt together . . . . . 77

(a) Relative Difference . . . 77
(b) Logarithmic Difference . . . 77
D.4 Average Daily Profit of Threshold Strategy if learnt together . . . 78
(a) Relative Difference . . . 78
(b) Logarithmic Difference . . . 78
D.5 Average Daily Profit of Relative Difference with Confidence based Strategy for Different Window Sizes . . . 79
(a) 4 Chunks in a Window . . . 79
(b) 2 Chunks in a Window . . . 79
(c) 1 Chunk in a Window . . . 79
D.6 Average Daily Profit of Threshold Strategy for Different Window Sizes Lernt Together . . . 80
(a) 4 Chunks in a Window . . . 80
(b) 2 Chunks in a Window . . . 80
(c) 1 Chunk in a Window . . . 80
D.7 Average Daily Profit of Stationary HMM with Confidence Strategy . . . 81
D.8 Average Daily Profit of Confidence Strategy for Different Smoothing Values . . . 82
(a) Smoothing 10 . . . 82
(b) Smoothing 1 . . . 82
(c) Smoothing 0.1 . . . 82
D.9 Average Daily Profit of Confidence Strategy for Cropping Values . . . 83
(a) Cropping 0.4 . . . 83
(b) Cropping 0.2 . . . 83
(c) Cropping 0.1 . . . 83
D.10 Average Daily Profit of 2nd Order HMMs . . . 84
(a) Relative Difference - Simple Threshold . . . 84
(b) Relative Difference - Confidence Threshold . . . 84
(c) Logarithmic Difference - Simple Threshold . . . 84
(d) Logarithmic Difference - Confidence Threshold . . . 84
D.11 Average Daily Profit of 3rd Order HMMs . . . 85
(a) Relative Difference - Simple Threshold . . . 85
(b) Relative Difference - Confidence Threshold . . . 85
(c) Logarithmic Difference - Simple Threshold . . . 85
(d) Logarithmic Difference - Confidence Threshold . . . 85
D.12 Combined HMMs . . . 86
(a) Stationary Logarithmic Difference based HMM learnt together . . . 86
(b) Stationary Logarithmic Difference based 2nd Order HMM learnt together 86

# Chapter 1

# Introduction

> Any sufficiently advanced technology is indistinguishable from magic.
>
> Arthur C. Clarke

## 1.1 Motivation

Computers are providing valuable assistance or completely replacing humans in many different fields and world's financial markets are not left behind. It all started few decades ago with simple data accumulation, storage and visualisation. Now there are sophisticated and complex systems capable of adept autonomous trading. But those producing profit are kept in secret by their owners.

According to rough estimates these computer algorithms are responsible for 70% of *Wall Street* trade volume [Salmon and Stokes, 2010]. The number is a little overrated, because most of the algorithms are *High-Frequency Trading* ones, which produce large volumes of trades even with comparatively smaller amounts of money in each trade. Nevertheless, the amount of money under the control of machines is definitely rising.

Creating such trading algorithm is highly challenging in many ways. One has to predict future price behaviour from huge amount of data from different sources. It is not easy to reveal hidden dependencies and incorporate them. Furthermore, there is a large number of other competing algorithms, individuals or whole companies.

It is not likely that there will be a complete functional trading system at the end of this work. The amount of problems to solve is large and we even might be overlooking a substantial part of them. But it is not possible to complete something without even starting and eventual success would bring pleasant payback in form of both psychical and materialistic satisfaction. Furthermore, we believe that knowledge and abilities obtained by analysing financial time series can be easily transformed and applied in many other problem areas.

## 1.2 Objectives

As we are new to the field, the main objective is to obtain a better insight into the area. Even though it is much more rewarding to do everything one's way, the knowledge of other people's work can keep your effort on the right track and prevent you from visiting already discovered dead ends.

Our first area of interest is the usage of background knowledge in financial time series prediction. After the initial research is over, a model will be designed, implemented and tested on historical, but not overly outdated data.

The second major goal is to perform evaluation of a few *Hidden Markov Models* variations and experimentally establish their ranking. Again all the design, implementation and testing steps will he performed.

We will also try to locate both - the benefits and the weak points of used techniques and come up with possible future improvements.

## 1.3 Overview

The rest of the document is organized as follows. The chapter 2 is dedicated to accumulation of the available knowledge. The design is proposed in the chapter 3 and the final product is

described in the chapter 4. Results of our tests are summarized in the following chapter 5. The chapter 6 describes possible fixes, improvements and future goals. And finally, the chapter 7 summarizes our effort.

# Chapter 2

# Background

> A business that makes nothing but money is a poor business.
>
> Henry Ford

## 2.1 Markets

The term market refers to the same concept as hundreds years ago, however the form and goods have changed. The target markets of this research are modern open financial markets allowing computer trading.

### 2.1.1 Market Types

This section summarises common publicly accessible markets. There are also differences between markets around the world even though they have the same type. The most important thing is that modern markets are accessible virtually using electronic trading systems.

**Currency Market** The currency market also known as *forex* serves as an exchange place for world's currencies. It is popular, traded continuously without breaks, but is also considered as a risky one.

**Stock Markets** These markets trade shares in companies. Their advantage is that in addition to the value of the share you can get dividends in case the company is profitable.

**Futures Markets** The items traded here are *futures* of specified commodities. The futures are contracts on future delivery of a commodity with fixed term and strict specification. Common traded subjects include coal, corn, gold, coffee, various financial instruments and many more. One advantage is that you do not to have to pay full price while buying something. You can also virtually sell items without having them, buy them later and pair the contracts. As a result even negative price movement can be highly profitable.

### 2.1.2 Theories

There is a number of theories how the markets work. Although most of them do not exactly reflect the reality, they can help to reveal important facts about the market. The following part of the chapter will review the most notable ones.

#### 2.1.2.1 Efficient Market Hypothesis

Efficient Market Hypothesis *EHM* comes in three different forms. The week form claims, that future price cannot be predicted by analysing past prices, because the future price movement depends entirely on information not contained in the price alone. According to this theory all pattern matching algorithms or trading strategies trying to take advance of market inefficiencies cannot generate systematic profit.

The semi-strong version further assumes, that prices adapt public information so fast, that no excess returns can be generated using that information. If this theory is true then neither *Technical* nor *Fundamental* analysis should be able to generate a long term profit.

The strong version further expects, that even private information instantly changes the price and therefore absolutely nobody and nothing is able to generate a long term profit

without a luck. As a results the trading success of all the subjects on the market should form something close to a Gaussian distribution.

There is a strong community disagreeing with this hypothesis trying to disprove it using both empirical and theoretical arguments.

#### 2.1.2.2 Random Walk Hypothesis

A random walk hypothesis leaves even less space for prediction. It claims that the markets as predictable as coin flip. Therefore no matter how much information is available the future direction remains a secret until it happens.

#### 2.1.2.3 Adaptive Market Hypothesis

The Adaptive Market Hypothesis states, that amount of information price reflects depend on the environmental conditions and nature of trading subjects. Therefore highly traded markets with large number of interested subject will reflect the actual price precisely than some abandoned market. This theory is based on *EHM*, but it is combined with behavioral economics and applies principles of evolution and financial interaction. And what is most important, it does not deny a possibility to make a profit.

### 2.1.3 Types of Analysis

There is a lot of analysis methods, but just two basic analysis direction will be mentioned.

#### 2.1.3.1 Technical

Technical analysis presumes, that all the required information is already present in the price. It sounds a bit crazy, but actually it is known that large quantities of people behave a bit predictively and as they react on market changes, they follow some patterns. According to this approach the patterns could be discovered and used. The technical analysis is compatible with *EHM*, because the *EHM* claims that discovering such patterns would affect the price and destroy the pattern.

#### 2.1.3.2 Fundamental

The fundamental analysis is a different approach. It suppose, that the price reflects the value of the asset. If the value changes, the asset becomes mispriced and trading these mispriced assets can produce gain. The evaluation tries to use all the available information including newspaper articles, situation of competing assets or global economical mood.

The fundamental analysis has two main problems. The first one is how to obtain all the relevant information as soon as possible. And the second problem is how to predict the impact of that information.

## 2.2 Systems for Algorithmic Trading

A system for algorithmic trading is a machine taking all available relevant information, processing it and producing trading operations for the market. Common input data contains market data like price, volume or even individual bids. This data can be further accompanied by news feeds, weather forecasts, advises issued by specialised companies or anything other which might have influence on the price and can be processed by a computer. The output operations are usually *Buy/Sell* commands frequently extended by *Stop-Loss* values or price restrictions.

There is a large number of methods how to get *Buy/Sell* commands using the historical input data. Some of them do not produce these commands directly, but they are trying to estimate the market's future value or direction and then apply some *strategy* to extract the commands. This can be useful for the learning process. The rest of the section contains the most widely used methods for single time-series analysis. The analysis of multiple time-series is covered only briefly in selected parts. A short overview of algorithms created using different classification [Zhang and Zhou, 2004] is available. There is also a more detailed work concentrating on the data-mining approach [Zemke, 2003].

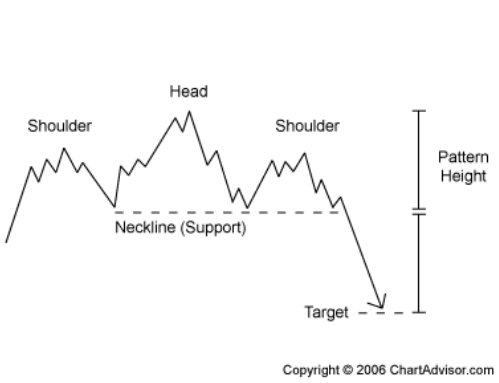
### 2.2.1 Historical Methods

Few decades ago, when the algorithmic trading was a new idea, the situation on world's markets looked differently. The traders used the computer mostly for displaying time series together with *indicators*. The *indicators* are mathematical transformations of one or more time series to one or more different time series. The average short-term trader had one or more *strategies* which were tested on the past data and which were producing *Buy/Sell* commands later executed by a human or a machine.

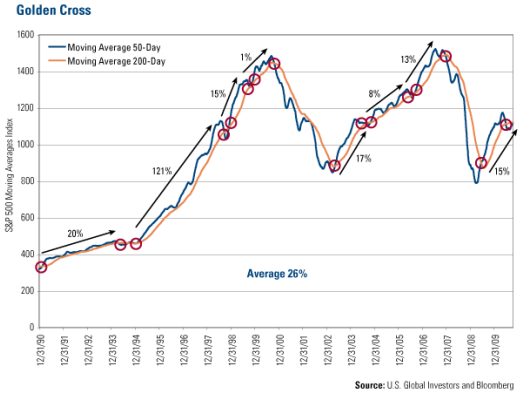
Usual strategies back then were based on searching simple patterns or watching for specific *indicator* values [Elder, 1992]. An example of a pattern can be the one called *Head & Shoulders* 2.1a. It can predict the market's changing direction [Langager and Murphy, 2010]. An example of indicator-based system can be the *2 Moving Average Cross* system 2.1b. It is based on two moving averages with different lengths. When the shorter moving average is above the longer moving average, the value is about to increase and vice versa [Derrick, 2010]. Some of these simple methods are used by traders and analysts even these days.

### 2.2.2 Econometric Models

This group includes all the mathematical models like *ARMA*, *ARIMA*, *GARCH* and many others. The name of a model is derived from its building blocks. For example the *ARMA* model is build from two parts. The *AR* stands for *Auto Regressive* and the *MA* part stands for *Moving Average*. Generally, these models are trying to disassemble the complex market movements to noise and several simple mathematically definable movements. All these models allow to estimate the future price based on previously learnt parameters. According to the amount of published books and articles on the topic, these models are quite popular and produce great results so we cannot just pretend they do not exists [Brockwell and Davis, 1996][Palma, 2006]. But they are so heavily used and researched that we decided to leave these models to more economically-based investigators and concentrate on different ones.




(a) A Head & Shoulders pattern example




(b) A Moving Average example

Figure 2.1: Examples of simple strategies based on patterns or indicators

### 2.2.3 Neural Networks, Genetic Algorithms, Clustering, Support Vector Machines

*Neural networks* together with *Genetic Algorithms* are powerful weapons capable of solving many different problems in many different areas. They are especially useful when solving complex recognition or relational problems or when there is no better idea how to approach a problem. In case of the financial time-series they are used for both modelling the data or joining several simple models to a more advanced and powerful one.

Both *Clustering* and *Support Vector Machines* can be used to separate input data to different groups. These methods are heavily dependent on good data preprocessing. Clustering has generally a little lower performance on financial time-series because it is more susceptible to noise.

All the methods in this subsection are the subject of a different thesis [Grafnetr, 2011] and therefore they will not be further investigated here.

### 2.2.4 Relational Data Mining

Relational data mining is a branch of data mining searching for dependencies between data and revealing hidden rules and structure. It is a little younger method compared to instance-based learning and currently is being dynamically evolved. The cornerstone of this method is a rule which is generally an implication in first order logic. The *FOL* gives this method more expression power than the standard decision trees have. The rules can reveal a new fact about instances:

$$isFather(x, y) \wedge isWoman(y) \Rightarrow isDaughter(y, x)$$

or categorize an instance:

$$hasFourLegs(x) \wedge \neg isRobot(x) \Rightarrow isAnimal(y, x)$$

or make assumptions for the future:

$$price(t) > price(t-1) \wedge volume(t) > volume(t-1) \Rightarrow price(t+1) > price(t)$$

or be any other type of a meaningful implication. Similar sign of all the algorithms is that they are trying to discover the simplest possible meaningful rules. However, the measures of simplicity and meaningfulness may vary between different algorithms.

Relational data mining wasn't very popular in the past, because of its complexity. Historical approaches were based on exhaustive search, therefore dependent on the size of the possible implications space, which is exponential to the number of predicates and logical operations allowed in one rule.

Nowadays, relational data mining is used for different purposes all across data mining area. Best results were achieved on biological datasets. However, relational-based methods are becoming more popular in time series analysis. They have several advantages against traditional methods. The result is human-friendly, because all the rules can be directly analysed and interpreted. This can be useful when adjusting the algorithm to fit the time series or when extracting deeper knowledge. The opposite example are neural networks, which are acting as a black-box and they are hard to understand without additional analysing method.

#### 2.2.4.1 FOIL and MMDR

One of the first well documented *RDM* experiments on financial time series were performed using *FOIL and MMDR* algorithms [Kovalerchuk and Vityaev, 2000].

The *FOIL* is based on generating clauses in *DNF* containing only positive examples. The algorithm uses heuristic depending on information gain of a predicate together with the branch and bound method. It also contains a stop criterion, which helps to avoid creating complicated rules containing small amount of examples. This leads to higher resistance to noise.

The *MMDR* algorithm incorporates additional improvements over the *FOIL* such as typed predicates, additional user-defined constraints or initial hypothesis (partial, possibly incorrect rule given by the user). In addition, the rules in *MMDR* are probabilistic. That means not all the rules have to be completely true, but they have to stay within some statistical significance limits. All these improvements are further reducing the search space and reducing the noise impact (especially the probabilistic part).

Tests in the above work were realized using price prediction and simulated trading on *S&P 500*. The learning part was 1985-1994 and validation (trading) was 1995-1998. Both algorithms were learnt to predict the next day price direction. The algorithm was given more information in addition to the *S&P 500*'s price:

- weekday
- first and second differences between prices

There are two tables (2.1 and 2.2) comparing *FOIL* and *MMDR* with different methods.

It can be seen that *MMDR* algorithm outperformed all other algorithms in both comparisons. Also it is obvious that the few differences between *MMDR* and *FOCL* have a great

| Method | 1995-1996 | 1997-1998 | 1995-1998 |
|---|---|---|---|
| Neural Network | 68% | 57% | 62% |
| Decision Tree | 67% | 60% | 64% |
| MMDR | 78% | 85% | 82% |
| FOIL | 51% | 45% | 48% |

Table 2.1: ILP Comparison: Percent of correct "up" and "down" estimates.

| Method | 1995-1996 | 1997-1998 | 1995-1998 |
|---|---|---|---|
| Adaptive Linear | 21.9% | 18.3% | 20.1% |
| MMDR | 26.7% | 43.9% | 35.3% |
| Buy and Hold | 30.4% | 20.6% | 25.5% |
| Neural Network | 18.9% | 16.1% | 17.5% |

Table 2.2: ILP Comparison: Gain per year in simulated trading.

impact on the performance. It is slightly discomforting that *MMDR* was the only active algorithm which was better than the passive *Buy and Hold* strategy (buy stock first day, hold it the whole time, sell last day). However, the comparison itself depends on many factors including the chosen time series and methods variants and parameters. It is quite possible that a group of *neural networks* learned by an advanced algorithm can outperform *MMDR* on a different time series. But *MMDR* definitely seems to be a great tool for financial time series analysis.

#### 2.2.4.2 ILP

Another good example of usage is training an *ILP* strategy for trading USD-DEM daily closing value [Badea, 2000]. At first, each instance of training data was classified using a threshold method and marked with none, one or two of the following labels: *buy*, *sell*, *not buy*, *not sell*. This is a slightly different approach than in the previous example, because the strategy is learned directly (there is no *up and down* prediction). It is also important to mention that not only the turnaround points were marked as *buy* and *sell*, but the turnaround neighbourhood was marked as well. This is because the turnaround point itself is difficult to spot without the surrounding part of the data series.

After this the *ILP* rules were learned using *Prolog*. Besides the data series itself, other computed series were fed to the learning algorithm:

- moving average crossover system
- *RSI* oscillator system
- *ADX* system triggered by crossing a threshold after two successive up movements
- stochastic *SlowK-SlowD* crossover system
- stochastic oscillator system

All these measures were computed for 5, 10, 15, 25, 40 and 65 days intervals.

The resulting strategy was able to perform simulated trading with 80% profitability compared to an over-optimized moving average crossover system. It also discovers 25-30% of trading opportunities. Author himself felt optimistic about the result of his initial experiment, but no further work was found.

#### 2.2.4.3 Conclusion

We were unable to find any more up-to-date information about financial time series and relational data mining or inductive logic programming. This is slightly puzzling, as they seem to be used and evolved in almost every other area. This can be caused by two extremes. Either they are inapplicable and no author wants to publish an unsuccessful work or they perform very well and the successful research is kept private.

### 2.2.5 Markov models

*Markov Models* are being used in a wide range of applications because of their simplicity and robustness. There are many attempts to analyse financial time series data, however each of the authors performs test on a different time series and compares with different algorithms with varying parameters. This makes it hard to compare the new test results with the old ones. Because of the autonomous trading systems, the market responsiveness is much higher nowadays and algorithms working in previous decades might be useless today. To make the situation even worse, often the parameters of algorithms used for comparison are completely missing.

#### 2.2.5.1 Basic Markov Model

The *Markov Model* (*MM*) is a structure, where observations are converted into discrete states. The probability that some state will be the next state depends only on the current state (in case of prediction to the future). If we want to predict data using this model, we have to make the values discrete.

Test using this particular model was performed e.g. by Constantine P. Papageorgiou [Papageorgiou, 1997]. He used *MM* to predict the direction of the *CHF/USD* exchange rate. The data were preprocessed to log ratios between consecutive prices

$$r_t = ln(\frac{p_t}{p_t - 1})$$

and afterwards mapped to 9 discrete values using a logarithm and a histogram.

To incorporate history into the model, each state is represented by 3 consecutive values. Otherwise, the model would take into account only a previous difference while predicting. The test data were from 1985 to 1991 and prediction was successful in 60-65% of the test samples. However, trading agent based on this model would not generate profit if transaction fees are taken into account. It was also not so accurate in detecting *reversals* (the points, where the trend changes). On the other hand the model is easy to learn and easy to predict, with both operations incredibly fast compared to other, more complicated models.

#### 2.2.5.2 Hidden Markov Model

Another kind of a *MM* tested on financial time series analysis is a *Hidden Markov Model* (*HMM*). It consist of discrete hidden states and discrete values. There are transition probabilities between the hidden states and emission probabilities from a hidden state to a value. The main difference against the *MM* are the hidden states, which are a kind of the model's memory.

This approach was recently tested by Bicego, Grosso and Otranto [Bigeco et al., 2008], who were trying to predict market direction in the next step. They chose to transform financial time series into sequences of *up* and *downs* and train two *HMM's*, one per each direction. When predicting, the model which better fit the actual data was the one determining the direction estimate. Uncertain sequences were filtered out by the following formula:

$$|log(p_+) - log(p_-)| < \theta$$

,where $\theta$ is a threshold, $p_+$ is the probability that a sequence was generated by *up* model and $p_-$ is the probability that a sequence was generated by *down* model.

The performance of this approach was tested on 5 indexes with different risks from time interval (30.11.1995-5.2.2001). The part before June 1998 was used to train models, the rest was used for testing. The method showed almost optimal performance and without difficulties outperformed both - a single *HMM* and *Polynomial Local Trends* used for comparison purposes.

We ought investigate this model more closely, because the demonstrated prediction power seems to be astonishing. The 'but' we are likely to be missing is perhaps hidden in the mechanism that rejects the uncertain sequences and therefore increases the performance by virtually reducing the test data.

The very same approach was examined later and compared to the *C-Support Vector Classification* with *RBF* kernel function [Park et al., 2009] on *SAMSUNG*, *POSCO* and *Hyundai Heavy Industry* stock in *KOSPI 100*. The obtained results again spoke in favor of the *HMM* model. However, the prediction accuracy was 55% $\pm$ 5, which is much less than in the previous test.

On the other hand, there are recent sources claiming that a *SVM* is able to outperform the *HMM* [Rao and Hong, 2010]. But in the study, several *SVM kernels* were tested, additional input from news sources was used and its parameters were carefully learned. We can only surmise what would have happened if a more advanced form of the *HMM* was used or additional news data were given as input to the *HMM*.

#### 2.2.5.3 Hidden Semi-Markov Model

An extension to the standard *HMM* tested in finance is a *Hidden semi-Markov Models* (*HSMM*) [Bulla and Bulla, 2006]. These models have zero transition probabilities between the same states. Instead, they contain a *survivor function* defining the probability of the model remaining in the current state in future steps. This particular extension breaks the *Markovian Property* of the model, because the future state becomes dependent on more than one past state. However, it is easier to model a different sojourn function than a standard geometrical one. *HSMMs* are widely used on biological-based data or in speech recognition.

There were three different sojourn functions tested: (*negative binomial sojourn time distribution*, *normal conditional distribution* and conditional $t$ distribution). The evaluation data were 1987-2005 sector indices from STOXX Ltd. preprocessed as *log ratios* between consequent values. According to various statistical parameters the winner of the test was the *normal conditional distribution* model. However, we cannot compare the result with others, because no value was given for *successful direction prediction ratio*, which all the other tests so far were using.

#### 2.2.5.4 Hybrid Approaches

Certain researchers find the *Hidden Markov Model*, even with extensions far too limiting and are expanding the field by combining the *HMM* with other algorithms. A good example of this approach is the *ANN-GA-HMM-WA* model [Hassan et al., 2007]. Briefly summarized, it is a model, where the daily price statistical values (open, high, low and close) serve as an input. They are fed to a *Neural Network* which is an input for a *Hidden Markov Model*. The *HMM* is initialized by a *Genetic Algorithm* for better convergence and then learned as usual. The output of the *HMM* is used to measure similarity between instances. The most similar instances are then used to estimate the future value using time-based weighting (older values are considered to be less important). The tests were performed on 2003-2005 stocks of *Apple Computer Inc.*, *IBM Corporation* and *Dell Inc.* The results have shown that the *ANN-GA-HMM-WA* is better than a previously tested Hidden Markov Model with Interpolation and that the results are comparable to the *ARIMA* method. This may not seem to be a great achievement, but the authors have pointed out that their method does not require analysis (regime analysis, seasonality test) before forecasting.

#### 2.2.5.5 Mixture Hidden Markov Model

The *Mixture Hidden Markov Model* (*MHMM*) will be mentioned briefly, as we have not found a comparison to another model. However, the idea seems to be worth mentioning. The difference between *MHMM* and *HMM* is an additional time-constant parameter affecting both the transition and the emission probabilities. The model is then learned using multiple time series, however each of the series can have different time-constant parameter distribution.

This particular approach was tested on daily Asian stock market values [Dias et al., 2010]. The individual series were at first separated into two different clusters based on statistical parameters (eg. curtosis) of each of the time-series. Then the parameters *MHMM* were learned assuming the series in each cluster have the same distribution of time-constant parameter. According to the results, the model was predicting values better than a model having a different time-constant parameter distribution for each time series. The only thing missing here is the direct comparison of results to another approach.

#### 2.2.5.6 Hierarchical Markov Models

Both *Hierarchical Markov Models* and *Layered Markov Models* are in fact restricted *Hidden Markov Models* (it is possible to create *HMM* with identical characteristics from both of them). The restriction itself allows to learn a special case of the more complicated model

using a smaller amount of data. They also suffer less from over-fitting. The main area of usage used to be speech recognition, where the input signal hierarchy has a logical explanation. The low part of the model represents phonemes, higher part represents words and even higher part represents sequences of words. Similar representation can be found even in finance, where the hierarchical levels can represent behaviour in horizons of different lengths.

Recent and well-documented tests of *Hierarchical Markov Models* can be found in *Journal of Applied Econometrics* [Geweke and Amisano, 2011]. The models were compared on the daily closing price of *S&P 500* stock and *one-year bonds*. According to this study, the two level *Hierarchical Markov Normal Model* performs similarly to *t-GARCH* model, which performs better than *Markov Normal Models* (*Hidden Markov Models* using *normal* distribution). There were also other models, such as *GARCH*, *EGARCH* or *Stochastic Vector*, but their results were far behind the *HMNM*. The authors also claim reasonable learning times even for large models. They were able to learn the $m_1 = m_2 = 6$ *HMNM* model on 10 years of data in 40 minutes, where the $m_x$ is the number of hidden states in $x^{th}$ layer.

#### 2.2.5.7 Conclusion

It can be seen that the family of *Markov Models* is quite large, widely used and easy to combine with other methods. We definitely did not include all possible combinations. The models can be also combined e.g. with *wavelets* [Ferrer and Brun, 2000], *dynamic time warping* [Oates et al., 2001] and many other algorithms. The obtained results are comparable to conventional *autoregressive models*. Thanks to the fact that these models are easily extensible, it is possible to add additional knowledge to the model and significantly improve the results. We think this group of models is worth further investigation as it is widely used in other areas and its popularity among researchers in the time series analysis is rising.

## 2.3 Text Mining Integration

Textual information is one of the commonly used background knowledge types for the financial time-series. There is a huge number of publicly available text data sources ranging from news and annual reports of companies to Twitter statuses and blog articles. Each source of textual data has its own specific pros and cons. For example, the news can be slower than internet social networks, whereas social networks incline to provide subjective or wrong information. The unofficial sources also produce more sarcasm, which can be misinterpreted by machines.

The goal of text mining is to extract relevant information from the text data and transform it into a form suitable for machine processing. The machine friendly form is then usually further processed and advises about the future direction of time-series are produced. The following part of this section covers several basic methods to analyse text and incorporate results to the prediction algorithm. The subsections are organized from the simple methods to the more advanced ones and definitely do not constitute a complete list. There is another publication containing a list of prediction systems based on text analysis [Mittermayer and Knolmayer, 2007]. Their study contains a well arranged comparison chart. However, some approaches are missing completely.

### 2.3.1 Article Frequency

The easiest way to incorporate news into a time-series predictor is to use the number of relevant articles as a feature. The idea behind this is that if there is something about to happen (either good or bad), there will be an increased number of articles. This may seem too simple to work, but according to the tests, adding the number of articles from Google News containing predefined keywords increased the prediction accuracy of 11 stock's directions by 0.03 in average [Rao and Hong, 2010]. And the prediction rate of the index *S&P 500* was improved from 0.53 to 0.7. It is necessary to point out that for some time series this enrichment of the input data was counter-productive. But this might be caused by a wrong set of keywords or a small testing sample.

Even thought this model is rather simple, it is clearly capable of improving the prediction power. It can be easily extended to accept different kinds of text-data sources. Another strong side is its immunity to the article duplicity, because it is likely that the same amount of duplicated articles will be present in both the learning and the validating phase. The disadvantage is that the model does not predict the direction of a movement. It only says that something might go on in the near future.

### 2.3.2 Sentiment Analysis

Another easy to integrate method is to make use of sentiment analysis libraries. These libraries use complex classifiers which were learnt on large databases of textual data classified by people. Depending on the type of library, the usual output is a measure of *objectivity*, *positivity* or different mood factors. Some libraries however provide only an interface and the learning data has to be provided by the user. This allows everybody to learn his own classifiers, however sometimes additional algorithm parameters are needed and their estimation can take a significant amount of time.

The following article analysing influence of public mood on *Dow Jones Industrial Average* can serve as an example application [Bollen et al., 2010]. The public mood was extracted from twitter feeds assuming the majority of the tweets was written by *US* citizens. There were two different systems used for the extraction. The *Opinion Finder*, which measures only the positivity/negativity of the mood. Second one, the *Google-Profile of Mood States* outputs six different values for each text document. Their names are *Calm*, *Alert*, *Sure*, *Vital*, *Kind* and *Happy*.

The final experiment was performed on the *DJIA* from February 28, 2008 to Dec 19, 2008 using a five neuron *Self-organizing Fuzzy Neural Network*. The inputs were the *DJIA* values three days back accompanied with different combinations of sentiment indicators, also with a three-day history. As can be seen from the results 2.3, the network itself ($I$) performs the same as a network enriched by the *Opinion Finder* values($I_{OF}$). So in this case the *Opinion Finder* is useless. However, adding the *Calm* value alone ($I_C$) leads to a significant improvement in the up-down direction prediction and combination of *Calm* and *Happy* ($I_{C,H}$) has the smallest prediction error. It is worth mentioning that according to additional tests the influence of mood was classified as non-linear.

| Evaluation | $I$ | $I_{OF}$ | $I_C$ | $I_{C,A}$ | $I_{C,S}$ | $I_{C,V}$ | $I_{C,K}$ | $I_{C,H}$ |
|---|---|---|---|---|---|---|---|---|
| MAPE (%) | 1.94 | 1.95 | 1.83 | 2.03 | 2.13 | 2.05 | 1.85 | **1.79** |
| Direction (%) | 73.3 | 73.3 | **86.7** | 60.0 | 46.7 | 60.0 | 73.3 | 80.0 |

Table 2.3: *MAPE* and direction accuracy of DJIA prediction using Twitter, Sentiment Analysis and SOFNN.

### 2.3.3 Word Frequency

Word frequency analysis uses counts of different words in each text document as a feature. The most common approach is to create a term-document table, divide all the documents to different categories according to their believed impact on the time-series and then learn some classifier. Later, the classifier is used to predict the impact of newly arrived text data. The used classifier and term weighting function may differ, but combination of *SVM* and *TF-IDF* weighting scheme is quite common.

It is possible to significantly improve the results using several preprocessing techniques. Good examples of these techniques are: removing punctuation, converting to a lower case form, removing numbers, removing email addresses, removing stop-words or stemming. The stop-words are short words without meaning like "the", "are" or "of". Stemming is a process of converting words to the corresponding base form (eg. transformation from "beautifully" to "beautiful") and unifying the synonyms (e.g. changing "lift" to "elevator" or "color" to "colour").

An example application of the word frequency approach is a system proposed in the article *The Predicting Power of Textual Information on Financial Markets* [Fung et al., 2005]. News articles were classified to the classes "up" and "down" according to the actual direction of the market. In order to eliminate noise, the whole time-series was approximated by linear segments using an iterative *Split & Merge* algorithm. Sometimes, a document is not aligned to the actual trend, but to a window of trends [Lavrenko et al., 2000] and then the document can belong to more than one group. After the classification was completed, the term-document table was constructed and rare terms appearing in less than 10% of documents were dropped. The resulting table was weighted using the *TF-IDF* and then a hyperplane maximizing the margin between the positive and the negative examples was found (it is basically what *SVM* does even though it was solved using a different method).

The results of the system are summarized in the table 2.4. Sadly, it is not clear what exactly has been traded and how long the simulation takes, so the return values are almost useless. It can be seen from the hit ratio that the model predicts values 3 to 5 days ahead the best. It is understandable why the 7 days prediction based on articles does not work well. However, it is not clear, why the prediction 1 day ahead performs so poorly. It might be caused by the time-series approximation algorithm (in case the approximated series is used for validation too) or by the high level of noise (in case the original series is used).

### 2.3.4 AZFinText System

The *AZFinText* is an experimental system evaluating the performance of three different text preprocessing methods [Schumaker and Chen, 2009]. The first preprocessing method

| | 1 day | 3 days | 5 days | 7 days |
|---|---|---|---|---|
| up/down | 51.8% | 61.6% | **65.4%** | 55.7% |
| return | 6.58 | 15.06 | **21.49** | 7.22 |
| deviation | **1.47** | 3.400 | 4.135 | 3.791 |

Table 2.4: The up/down hit percentage, accumulated return and standard deviation of the return for different delay periods of the system proposed by Fung, Yu and Lam.

consists of typical steps like stop-words removal and stemming. The second approach filters out everything but *Noun Phrases*, i.e. only nouns, pronouns and related words (usually adjectives) are taken into account. The third method is based on the second one, but each word is classified into one of predefined categories (eg. location, organization ...).

The prediction time is intra-day, namely the stock value after 20 minutes is estimated. There were four different prediction models tested. The first one was named *Regression* and used extrapolated linear regression of the 60 minutes period before an article's publication. The second one was named *M1* and used nothing but the text data. It was present solely for comparison to previous works. The third model was named *M2* and in addition to the text data it used stock value at the time an article arrives. The last one was called *M3* and in addition to the test data it also used the value of the linear regression. For learning all these models the *SVM* algorithm was used.

The results were evaluated on the *S&P500* from Oct. 26 to Nov. 28, 2005. Three different criteria were used: the mean squared error (*MSE*) (table: 2.5), direction accuracy (table: 2.6) and simulated trading (table: 2.7). It can be seen that the model *M2* outperforms all other models on average. From the text-processing point of view, the winner is the *Noun Phrases* model. Combined with the model *M2* it dominates all other combinations with a single exception in simulated trading. It is possible that further reducing the feature set brings some advantages whereas complicating the models with additional classifications increases complexity and prevents the model to be fully learned.

| MSE | Regress | M1 | M2 | M3 |
|---|---|---|---|---|
| Bag of Words | 0.07279 | 930.87 | 0.04422 | 0.12605 |
| Noun Phrases | 0.07279 | 863.50 | 0.04887 | 0.17944 |
| Named Entities | 0.07065 | 741.83 | 0.03407 | 0.07711 |
| Average | 0.07212 | 848.15 | 0.04261 | 0.12893 |

Table 2.5: AZFinText: Closeness Results

### 2.3.5 NewsCat

The *NewsCat* is a prototype using a hand-crafted list of features in addition to the traditional bag of words approach [Mittermayer and Knolmayer, 2006]. The list of features contains selected words ("up", "down"), phrases("formal investigation", "sales climb") and tuples ("reduction" NEAR "financial guidance", "approve" NEAR "share buyback") which are

| Directional Accuracy | Regress | M1 | M2 | M3 |
|---|---|---|---|---|
| Bag of Words | 54.8% | 52.4% | 57.0% | 57.0% |
| Noun Phrases | 54.8% | 56.4% | 58.0% | 56.9% |
| Named Entities | 54.2% | 55.0% | 56.4% | 56.7% |
| Average | 54.6% | 54.6% | 57.1% | 56.9% |

Table 2.6: AZFinText: Directional Accuracy Results

| Simulated Trading | Regress | M1 | M2 | M3 |
|---|---|---|---|---|
| Bag of Words | -1.81% | -0.34% | 1.59% | 0.98% |
| Noun Phrases | -1.81% | 0.62% | 2.57% | 1.17% |
| Named Entities | -2.26% | -0.47% | 2.02% | 2.97% |
| Average | -1.95% | -0.05% | 2.06% | 1.67% |

Table 2.7: AZFinText: Simulated Trading Engine Results

believed to have an impact on the market. After the preprocessing steps, the features from the list are merged with those from the bag of words and a *Naive Bayes classifier* is learnt. The classifier distinguishes between four groups named quite descriptively: "GOOD", "BAD", "NEUTRAL" and "UNCLEAR". The additional group "UNCLEAR" was defined for articles which cause a significant move of the price - but which returns in less than 15 minutes to the original value.

Data were evaluated on 15-second intervals of the S&P 500 stocks from April 1 to December 31. News were obtained from the *PRNewswire* archives and contained additional metadata like the companies mentioned in each article or the category of the article. During the testing, distinct feature filtering algorithms, document representations and learning algorithms were evaluated. The best tested configuration achieved 83% accuracy in article classification. The average profit of a simulated transaction was 0.21 which gives a total profit of 70 for all 335 transaction performed. After changing the exit strategy from 1% profit to a more conservative value of 0.5% and addition of -2% stop-loss, the average profit per transaction increased to 0.28. Even though the simulated trading ignores transaction fees, slippage and other volatility issues, the results obtained are quite encouraging.

## 2.4 Other Used Algorithms

### 2.4.1 SVM Classifier

The main idea of a *Support Vector Machine* classifier is to fit a maximum margin hyperplane through the feature space separating the instances of one class from the instances of another class [Karatzoglou et al., 2006]. A new instance can be classified using the model just by computing the side of the hyperplane the instance is on. The algorithm is known to work well even for problems with large dimensionality and it was successfully used for news articles classification before [Mittermayer and Knolmayer, 2007].

Because not all the data can be separated by a hyperplane directly, the *SVM* implementations are often featuring multiple *kernels*. The *kernel* transforms the input data to a different space and then the transformed data is separated by a hyperplane. Common *kernel* functions are a *Radial Basis Function* or a polynomial function.

Another limitation of the *SVM* is that one model can separate data into only two categories. The workaround is to create multiple models and decide the category from individual results. The first option is to create a separate model for each pair and the category with most positive results is the winning one. The second possibility is to a create model for each category versus rest of the categories, taking the best score as the winner. The second approach can be faster for a larger number of categories while the first mentioned method is considered to be more robust.

### 2.4.2 Naive Bayesian Classifier

The Naive Bayesian classifier is one of the simplest classifiers known. Despite it's simplicity, it is known to work well on problems with a large multidimensional space [Russell and Norvig, 2003] and it is successfully used as a cornerstone of many e-mail *spam* filters.

The classifier is based on the following expression:

$$class(f_1, \ldots, f_n) = \arg\max_c P(c) * \prod_{i=1}^{n} P(f_i|c)$$

where $c$ is a class and $f_i$ is an observed feature. The beauty of the classifier lies in the simplicity of both the learning and the validation. The learning process is basically counting of examples, while the validation is selecting a largest product.

The only drawback is the independence assumption. The classifier is the optimal one only if the features are independent of each other. This can be barely expected in any real world problem. Luckily, the classifier often works even for problems with dependent features, but it just might not be the optimal one.

# Chapter 3

# Design

> Thirty to forty years ago, most financial decisions were fairly simple.
>
> Scott Cook

## 3.1 Simplifying Assumptions

Because the market itself is a very complex system, we would like to make a couple of simplifying assumptions. There is actually no price on the market, but a large amount of bids, which are the individual buy/sell offers. The bid can be defined by different ways, but the amount and the price are present almost always. The buying and selling bids are processed by the market's algorithm, paired and executed. There is more than one way how the pairing algorithm can be defined, but we will ignore that fact.

As we will focus on trading in intervals of days, we will make quite usual assumptions that there is only one price and that there is always enough liquidity. The price gap can be simulated later by using the *slippage* during trading performance evaluation. The liquidity itself will be completely ignored as it is very unlikely that it will be an issue during usual market operation, if dealing with daily trades of common stocks.

We are also lucky to be designing just a prototype so we don't have to care as much about the efficiency of our computations as the real-word trading applications do. Also, the daily trading has to cope with a much lower volume of information than the high-frequency trading. We will further decrease the amount of information by focusing solely on a narrow sample of stocks. So even a system with minor inefficiencies will remain computationally feasible.

## 3.2 Text Integration

The text integration and analysis process is following the usual pattern (figure: 3.1). The input in form of textual data is preprocessed and transformed to the feature set. These features are further classified using a classifier and corresponding time series data.

The features are then split into train and test data. The train data is used to learn a model. The learnt model tries to classify the test data and then the result is compared.

The only drawback of this approach is that the initial data classification works with all the data. So if it uses, for example, the standard deviation, then the older data are in fact classified with the knowledge of the new data. Therefore, a future-independent classifier has to be used, otherwise the results are biased and have to be interpreted carefully.

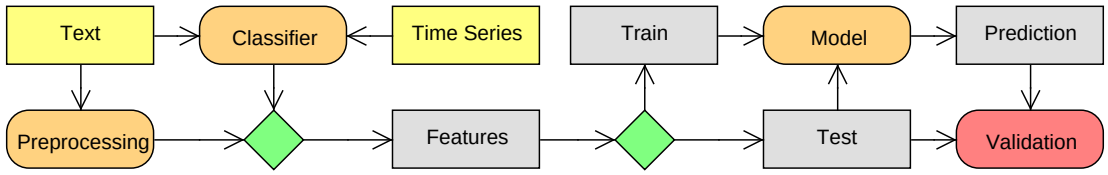



Figure 3.1: Textual Data Analysis Flow

The diagram displays individual testing iterations. There will be another set of data used solely for final validation purposes.

### 3.2.1 Text Sources

There are many relevant publicly available data sources. They can be categorized into two main groups. The first group is made of official sources and mostly includes news articles and reports released by the companies themselves. The news are supposed to be the most objective source of information, however there can be a problem with release times. Even though there are specialized news feeds aggregating information about traded companies, there has to be someone to actually write the article and release it. Likewise, the reports released by every company tend to present a better state than the company is actually in. However, commonly included numbers about historical sales and profits are hard to manipulate and constitute relevant information.

The second group consists of all the rest of the textual information on the internet including blogs, discussion boards and even social networks statuses. These informations can be personally biased and contain computer-unfriendly language constructions like sarcasm or slang words. However, when averaged, they can provide "public mood" about a given topic [Bollen et al., 2010] and sometimes they can be faster than the conventional news.

However, for our research we have chosen the more traditional source of information, the news. One of the reasons was the availability of the historical data. It is hard to search for old blog posts and download them automatically, whereas we know there are some publicly accessible news sources containing the data we need.

#### 3.2.1.1 News Sources

The following possibilities were found while searching for free news sources. There are many paid services providing data in computer-friendly format. Sometimes they even provide estimated impact values on time series, but we want to stay free of charges.

**Reuters** [Reuters, 2011] The *Reuters* claims to have exactly what we are looking for. They provide a searchable news *API* with tagged entities. Therefore, it would be easy to find relevant articles. However, they were not accepting new accounts for registration.

**Wikinews** [Wikinews, 2011] The *Wikinews* is a wiki news server, that means everyone can edit or add a historical event. However it is more focused on global news and therefore not completely targeting our needs.

**iHackerNews** [iHackerNews.com, 2011] *iHackerNews* even provides option for downloading the whole database of news. But the database contains mostly technology news, which is only a subset of the news we are interested in.

**USA Today** [USA Today, 2011] The *USA Today* is almost fulfilling our needs. It is free, capable of search and returns the set of organizations mentioned in an article. The only drawback it has is its restricted capacity. The *API* is limited to 1000 calls per day (one call returns 10 articles), so downloading several years worth of news would take a long time. Also, the response contains only a link to a web page with the full article.

**NY Times** [NY Times, 2011] The *NY Times API* features similar functionality as the *USA Today API*, but they impose only 10 000 queries-per-day limitation their *API*. The rest is the same.

Eventually we decided to use the *NY Times API*, even though they do not provide the article content directly. We assumed it won't be so hard to download the text from their web servers and they have a good range of *API* features including search, summaries and tagged article organizations. The *API* produces the output serialized as a *JSON* object, which is good, because most programming languages have *JSON* support included or at least available as a library.

### 3.2.2 Article Preprocessing

The very first step of the article preprocessing was duplicates removal. Some of the articles have been downloaded multiple times because they were related to multiple subjects we were interested in. The duplicates often lead to presence of an identical example in both the learning and the testing set, which makes results useless.

Also, we would like to allow filtering of the articles based on various features like category, length or date. Removing irrelevant articles from the learning set can significantly improve the model quality.

### 3.2.3 Feature Extraction

As mentioned before, we have decided to use the *Bag of Words* representation in our model because more sophisticated models often need large amount of data to train.

The designed system is required to allow traditional text feature extraction concepts such as stemming, transforming to lower case and removing punctuation, stopwords or previously specified words.

We would also like our bag of words to allow removing of infrequent words or words with no effect on the result (equally distributed in all the categories). The removal of records leads to smaller dimensionality, which often improves the performance.

The last requirement is to allow *tf-idf* weighting as it seems to work out for most of our predecessors.

### 3.2.4 Classification

The classification is one of the hardest parts. It usually incorporates matching the article release date against the stock price trend. However, the trend detection is a tricky part, because the prices are known to contain a significant amount of noise. An article might even be related to multiple subjects and it is not easy to decide which one, if any, is the relevant one. And, of course, there is the danger that the article is commenting on some past event, having no impact at all.

We will simplify the problem by partially ignoring the dangers mentioned above and use a simple threshold category detection based on the relative price change at a given day. Hopefully, the noise will be filtered by the model and the result will be satisfying. Otherwise, we might change the strategy and try to find another classification algorithm.

### 3.2.5 Models

Because our features are in fact counts of individual words in the articles, we will deal with a large feature space. Therefore, our model has to withstand that dimensionality problem.

We would like to allow testing using either *SVM* or *Naive Bayes* classifiers as they both meet our requirements.

### 3.2.6 Validation

The validation process has to be capable to use both the *Window Validation* and the *Cross Validation*. The *Window Validation* reflects the reality more closely, since during newer data validation just the older data are used for learning. On the other hand, the Cross Validation is more likely to provide better results with a smaller amount of data.

The performance will be measured by calculating the amount of misclassified items from the validation set.

## 3.3 Time-Series Analysis

The time-series analysis sticks to a widely used data analysis pattern (figure: 3.2). The data is preprocessed and transformed into a set of features. These features are split into two independent parts forming the training and the testing set. The training set is used as a learning data for a *HMM* model. The test set is then analysed by the model and future values are predicted. The predicted values are compared with the real values and assumptions about the model quality are extracted.

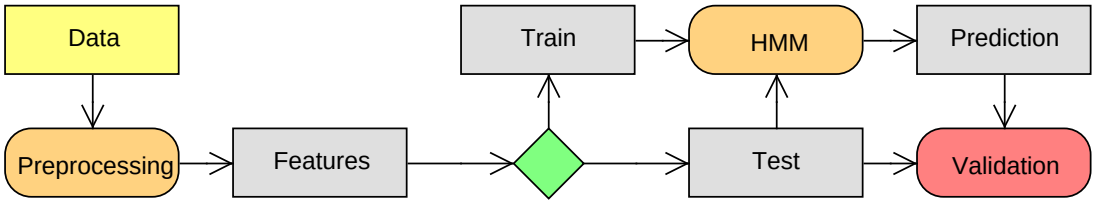



Figure 3.2: Time-Series Analysis Flow

In addition to the train and test data, there will be, of course, a validation set. The validation set will be never used before the final validation, which will confirm or disprove the performance estimates.

### 3.3.1 Time-Series sources

We tried to obtain free historical intra-day data. However, it turns out that generally only two of these properties can be picked. Free daily data was considered to be the best option, because it further reduced the amount of data to process and allows us to freely obtain a

sufficient amount of such data. Without historical data it would be impossible to learn and test our models.

The two most widely used services for free data are *Google Finance* [Google, 2011] and *Yahoo! Finance* [Yahoo!, 2011]. Both companies provide almost identical services. They both have free daily price history for all the common stocks from major *US* markets. In the end, the Yahoo was chosen because it has longer financial services tradition.

### 3.3.2 Features

The features will be created from the historical price values. However, the raw form of the data is not suitable for modelling and therefore preprocessing will be applied. The relative difference calculation will be the first one. Instead of the price itself, it reflects the change.

$$rd_i = \frac{x_i - x_{i-t}}{x_i}$$

Because the values have a wide spread we will try two approaches to reduce it. The first one is to cut off values beyond some threshold and the second one is to use logarithm of the relative difference.

### 3.3.3 Models

Our weapon of choice will be *Hidden Markov Models*. As we will be dealing with time series, we will use the *Homogeneous* variant of the model, which has all the transition probabilities constant during the time.

The individual states will be represented using a Gaussian distribution and the whole model will be learned using an unsupervised learning method. Our model has to be capable of random initialization and prediction of future values.

We would also like to support *Higher Order* models and *Stationary* models.

### 3.3.4 Validation

During the validation process of a *HMM*, the *Window Validation* will be used to simulate the real world scenario. The amount of training data should be sufficient and hopefully the same chunking algorithm will be used for both the text and the time series data.

The comparison will be performed using three different indicators. The first one will be the *Logarithmic Likelihood*, which is used as a measure of how much a model reflects the data. We will further divide the value by the length of the data to allow better comparison of models representing different time intervals. The advantage is that this measure can be easily computed. On the other hand, it does not exactly measure the quality of the model for trading purposes. The model might be providing a wrong value, but as long as the direction is right we do not care. Being aware of the previously mentioned drawback, the measure will be used primarily for estimating the right number of hidden states rather than for comparing different models to each other.

The second value for comparison will be the future direction estimate. The direction estimate is more likely to reflect the real world usage, because the direction of a future

movement is more important for us than the movement itself. However, the direction estimate also does not measure exactly what we need. It assigns the same weight to errors of all sizes. In case of trading, a small error (incorrectly predicted direction of a small change) is barely noticeable whereas a large one can be expensive.

The last performance indicator will be the simulated trading profit. The simulated trading is the closest measure to the real world application. However, it has some drawbacks. It might be more vulnerable to the noise. Also the performance depends on both the model and the trading strategy. Therefore a good model can be degraded by a poor strategy and a great strategy can have positive impact on otherwise a lousy model.

The dependence on the strategy will be partially eliminated by using exactly the same strategies for all the models. We would like to have two strategies. One dependent on the future value and another dependent on the probability of the future value direction.

# Chapter 4

# Implementation

> Even a mistake may turn out to be the one thing necessary to a worthwhile achievement.
>
> Henry Ford

## 4.1 Implementation Environment

There is always the possibility to implement everything by yourself, but using existing applications or libraries can speed up the development and it can be desirable in the prototyping phase. It is possible that the result will have higher computational demands than a one-purpose optimized code and that the libraries become a little restricting on functionality. However, the initial boost in the development speed can be priceless.

There are plenty of data-mining applications and scripting languages for prototyping. Among them we choose the following three.

### 4.1.1 Matlab

The *Matlab* [MATLAB, 2009] is a commercial software developed by the *MathWorks* company. It was designed for scientific computation and it can be equipped with additional packages further extending the possible usages. It is capable of all the necessary mathematical operations and possibilities of data visualization are quite extensive.

Although the graphical interface itself and a lot of its functionality is written in Java, the core computation functions are written in *C*, *C++* or *Fortran*, which has a positive impact on the computational performance.

In addition, the *Matlab* is heavily used in computer vision courses, so we are familiar with it.

### 4.1.2 R

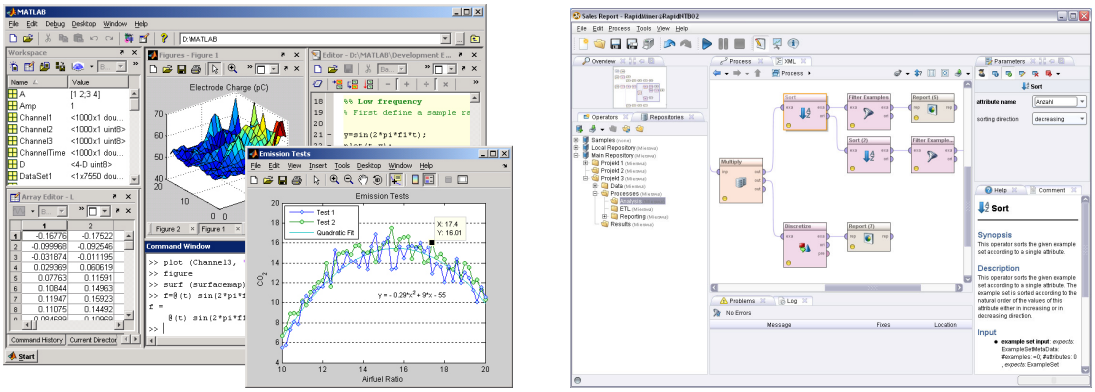
The *R* [R Development Core Team, 2011] programming language is an open source implementation of the *S* programming language designed especially for data processing and scientific tasks. The core functionality includes various classification, regression and visualisation methods and can be further extended using packages. Most of the publicly available packages are stored in the *CRAN* [CRAN, 2011] repository and can be easily downloaded and used in a couple of seconds.

Most of the *R* functionality is written in the *R* programming language itself, with the exception of the computationally-intensive parts. Those are written in *C* or *Fortran*, similarly to the *Matlab* language. Some of the extensions are designed to communicate with other languages, so it is not a problem to call a *Java* or a *C#* function from the *R* code.

### 4.1.3 RapidMiner

The *RapidMiner* [Mierswa et al., 2006] is a *Java* application designed for data analysis. The system contains a lot of common classifiers, validation methods and visualisation tools. Also, it can be easily extended using pieces of *Java* code or modules connecting to third party languages.

A nice feature of the environment is that the whole process can be easily created using a graphical interface. It is sufficient to connect boxes representing operations by links representing the data flow 4.1b. The creation of the system is so fast that it is possible to learn a neural network model on your dataset using cross-validation in less than ten minutes after the software is installed.



(a) A Matlab screen example

(b) Example usage of the RapidMiner

Figure 4.1: Examples of application candidates

### 4.1.4 Conclusion

Even though the usage of the *RapidMiner* was very tempting, in the end the*R* language was chosen as the main development platform. The decision was made for two reasons:

The *R* is widely used for statistical purposes in both the academic and the commercial sphere. As a result, the additional packages seem to contain all the necessary analysis tools required to complete this work. This does not seem to be true for the *RapidMiner*, which does not contain a suitable extension for *Markov Chains*.

The second reason was curiosity. The research sources are full of articles about *R* and its successful applications, while our experience with it is virtually non-existent. Taking a deep breath and diving in to the world of *R* is the fastest way how to get some experience and form a professional opinion.

## 4.2 System Design

The optimal way would be to write a custom data-mining package for *R* and then use its capabilities in separate testing scripts. However, due to missing experience with creation of such packages, a less sophisticated way was chosen. Instead of packages, a set of files containing individual methods was created.

### 4.2.1 Directory Structure

To bring an order to the number of *R* scripts a directory structure was designed. The root framework directory contains the following subdirectories:

**output** tables and graphs created by the framework are stored here

**packages** additional packages not present on the *CRAN*

**data_cache** a cache storing precomputed models for validation scripts

**modules** holds *R* scripts containing the framework functions (subsubsection 4.2.1.1)

**chunks** holds *R* scripts with individual tests and visualisations (subsubsection 4.2.1.2)

**saves** manually saved workspaces with intermediate results

In addition to these directories, the root directory contains two scripts. The *init.R* script loads all the required libraries to the memory and initializes all the functions and variables used internally in the framework. And the *install.R* installs all the required libraries from the *CRAN* and the local storage.

#### 4.2.1.1 Modules

The modules are separate files, each containing functions dedicated to a specific area.

**init_data.R** loads information about series and articles

**init_fun.R** common methods not belonging to a separate file

**init_hmm.R** additional methods compatible with *depmixS4* package

**init_chunks.R** data splitting methods for the window and the cross- validations

**init_series.R** loading and downloading of series

**init_text.R** loads all the text related files

**init_text_bayes.R** custom implementation of the *Naive Bayes* compatible with *tm* package

**init_text_categorizing.R** text categorizing methods

**init_text_corpus.R** corpus preparation and loading methods

**init_text_features.R** text tf-idf and weighting

#### 4.2.1.2 Chunks

The chunks are scripts dedicated to specific tasks. Most of them are evaluating some test scenario and producing graphs and tables. Following chunks are available for text mining:

**dataseries_category_item_counts.R** produces text categorization statistics

**dataseries_time_alignment.R** compares price movement near articles

**text_base_process.R** initializes the testing corpus

**text_base_process_validation.R** initializes the validation corpus

**text_shift_one.R** tests text classification of single entity

**text_shift_one_validation.R** validates single entity classification

**text_shift_separate.R** test classification of multiple entities

**text_shift_together.R** tests classification of multiple entities together

And following chunks are present for *HMM* evaluation:

**hmm_regression_separate.R** learns a *HMM* with specified parameters for each of the series and outputs a summary

**hmm_regression_together.R** learns one *HMM* from all the series and outputs a summary

**hmm_regression_validation.R** performs a validation of the most successful models using a validation data

**series_graph.R** produces a simple graph of the data series

## 4.3 Article Retrieval

As there is no direct way to obtain an article from the *NY Times API* using *R*, a custom application for the news retrieval was written (There is a single package capable of receiving news from the *NY Times Newswire API*, but it allows only one month of historical data). The application was named *NY Downloader* and it was completely written in the *C#* programming language using *Visual Studio 2010*. It allows a user to search articles by different parameters and download them using a given number of download workers. The application itself also takes care of the daily and secondly limits imposed by the *New York Times* on free users.

### 4.3.1 Output Data

The data storage format is an array of *JSON* objects similar to those returned by the *NY Times API*, but with two additional text properties. The first one is called "full_html" and contains the *HTML* obtained from the *URI* defined in the *API*. The second one is called "full_text" and holds the relevant part of the text data obtained from the *HTML*.

The following fields and facets are requested from the *NY Times API*: "abstract", "author", "body", "byline", "classifiers_facet", "column_facet", "date", "des_facet", "desk_facet", "geo_facet", "material_type_facet", "nytd_byline", "nytd_des_facet", "nytd_geo_facet", "nytd_org_facet", "nytd_per_facet", "nytd_section_facet", "nytd_title", "org_facet", "per_facet", "source_facet", "title", "url" and "word_count". They contain the *URI* of the article and a lot of useful information about the article itself. Although not all are important, the "abstract" can be used in the future for extracting the most important information and "org_facet" contains the names of organizations mentioned by the article. However, not all articles contain all the fields and it is up to further processing to check their presence.

The *Json.NET* library was used for all the data manipulation [Newton-King, 2011]. It allows to bind the obtained *JSON* to a *LINQ* object, modify and write it. The writing itself is done continuously during the download, so there is no memory consumption overhead.

### 4.3.2 Text Extraction

The response of the *NY Times API* contains only a link to a web page where the text of the article can be seen. The "pagewanted=all" parameter was appended to the *URI* before downloading, because it eliminates problems with paging. Using the "for print" version of an article was also considered, but it leads to problems with a different style and to occasionally receiving a log-in page.

After the download, the relevant part of the article was found and searched. At first, both the *SgmlReader*[mindtouch, 2010] and the *Html Agility Pack* [Mourier et al., 2011] libraries were used to transform the *HTML* document into the *XML* one and then the text part was extracted using *XPath*. However, the transformation does not work correctly for neither of the libraries and the output was often damaged. This behaviour was the reason to put the "full_html" property into the stored *JSON* objects, because it makes repeated improving of the text extracting algorithm possible without constantly querying the *NY Times* servers.

The final prototype uses the *Html Agility Pack*'s inner representation together with a *XPath* query for text extraction directly from the *HTML*. The *XPath* itself is a bit complicated, because the *NY Times* are using different templates for distinct time ranges and news categories:

```
"(//div[contains(@class,'text') or contains(@class,'caption')
    or contains(@class,'body') or contains(@class,'article') or
     contains(@id,'article')]//p) | (//nyt\_text//p) | (//div[
    contains(@id,'caption')]) | (//div[contains(@class,'legend')
    ]/div)"
```

Listing 4.1: XPath expression for the text extraction

## 4.4 Text Processing

## 4.5 Article Preprocessing

Because the articles were downloaded separately for each company, it occurred many times that the same article was present in files belonging to multiple companies. The first step of preprocessing was the removal of duplicates. They were identified by the combination of the date and the title of the article.

The text preprocessing is implemented using filter functions. Each of the functions modifies an article in some way and it might or might not be applied depending on a specific scenario.

**Entities** : Removes all the *HTML* entities.

**Lines** : Removes all the line endings.

**Punctuation** : Removes all the punctuation marks.

**Numbers** : Removes all the number characters.

**Lower** : Makes the text lower case.

**Stopwords** : Removes all the stopwords like "the", "about" or "be".

**Whitespace** : All the whitespace characters are replaced by a single space.

**Tag** : Appends the word category at the end of each word.

Majority of these functions is a part of the *tm R* package [Feinerer, 2011], which has been used for the text data analysis. The exceptions are the *Entities*, which has been implemented by us and the *Tag* function, which is a part of the *openNLP* [Feinerer and Hornik, 2010] package.

The transformation from articles to the bag of words representation is also a part of the *tm* package. The bag of words is represented as a sparse matrix.

## 4.6 Data Series Processing

A code capable of learning and validating a *Markov Model* was required for our analysis. An option was modifying a source enclosed in a publication called *Hidden Markov Models for Time Series* [Zucchini and MacDonald, 2009]. The code snippets from the book are well documented and easily extensible. But using a custom written code slightly increases a chance of an error, so we decided to look for a library leaving the custom code as a back-up option.

### 4.6.1 Libraries Available

The search revealed that there are five *HMM* focused libraries on the *CRAN*. An article comparing all or a subset of these libraries was not found, therefore we are forced to create a short summary ourselves.

#### 4.6.1.1 depmixS4: Dependent Mixture Models

The package seems to be easy to use and is capable of multivariate time-series approximation [Visser and Speekenbrink, 2010]. Furthermore, the constraints for transition probabilities can be set, which allows to simulate more complicated Markov Models. It is also easily extensible, so usage of a custom distribution is possible. According to the documentation, the fit function is capable of optimizing a model for a set of series. This package seems to meet our requirements.

#### 4.6.1.2 HMM: Hidden Markov Models

This package can handle basic *HMM* models with discrete time and space [Himmelmann, 2010]. The advantage of this package seems to be the simplicity of usage, because it contains only a few methods for model creation, parameters estimation and probability computation. The disadvantage is again the simplicity, because the amount of features is significantly smaller than in the previous library.

#### 4.6.1.3 msm: Multi-state Markov and hidden Markov models in continuous time

The package is capable of handling both the discrete and continuous time-series [Jackson, 2011]. Unlike the previous packages, this one also allows to specify series-specific and time-specific covariates and much more. In fact, the complexity of the package is so enormous that it may be hard to use.

#### 4.6.1.4 HiddenMarkov: Hidden Markov Models

Another library capable of creating and fitting Markov models [Harte, 2011]. It allows to use different distributions for representing continuous-time observations and features the usual set of learning and validation functions. This package is currently under development, so methods described in its documentation may change and further functionality is promised to be available in the future.

#### 4.6.1.5 rEMM: Extensible Markov Model

This library actually does something a little different than what was expected after reading the name [Hahsler and Dunham, 2010]. However, it is possible to build a *Markov model*-based trading system using this library, therefore it is mentioned here. Basically, it feeds values from the *threshold nearest neighbour* algorithm to a *first order Markov model*, which uses them as states. After the learning, the model is capable of predicting future states based on the transition probabilities.

#### 4.6.1.6 Conclusion

Finally, we decided to go with the *depmixS4* package, which provided most of the required functionality and was claimed to be easily extensible. In addition, it featured an optimized *C*-based likelihood computation, which can save up precious computing time.

### 4.6.2 Usage and Modifications

The basic usage of the library is simple [Visser and Speekenbrink, 2010]. For example, the following piece of code creates a *HMM* model with states based on a *Gaussian* distribution and then estimates its parameters:

```
R> mod <- depmix(rt ~ 1, data = speed, nstates = 2,
+   family = gaussian())
R> fm <- fit(mod)
```

Listing 4.2: Example of HMM fitting using depmixS4

However, the library itself does not support higher order or stationary *HMMs*, neither some kind of smoothing. As a response to this insufficiency, we have created the wrapper function *hmm_fit_custom* and the original *EM* algorithm was improved to support all the requested features.

#### 4.6.2.1 Value Prediction and Confidence

So far, the usage of distribution families is limited to the *Gaussian()* and *Multinomial("identity")*. The limitation itself is not imposed by the *depmixS4* library or the improved learning algorithm, but by the validation itself. Because the library does not provide a function for predicting the future we were forced to write it ourselves. Two separate functions were created to address this task.

The first one is called *hmm_predict* and makes it possible to predict future observations. The function actually returns the most probable future observations $X^*$ such that:

$$X_i^* = \arg\max_x P(O_i = x | O_{1:i-t} = X_{1:i-t}, M)$$

for a given model $M$, observations $X$ and a time shift $t$. To accomplish this, the hidden state probability distributions for each state are computed using the *Forward-Backward* algorithm. Then the time shift is performed by multiplying the state distribution probabilities by the transition matrix. Finally, an assumption about a future observation is made. For the *Gaussian* distributions, the weighted mean is returned to minimize the square error. For the multinomial distribution, the most probable discrete state is returned.

Technically, using the weighted mean for the Gaussian distribution is not correct. Because in the simulated trading we do not care about the square of the error, but rather about the error itself. Sadly, minimizing the plain error can result in multiple equally good solutions.

This is the reason a second function called *hmm_confidence* was introduced. It is capable of processing queries about the future value. The probability that a future observation will be in a given interval is returned for a Gaussian distribution. The multinomial distribution can be questioned about the probability of a state combination. More precisely, for a vector of weights $w$ corresponding to discrete observations, a time shift $t$, a model $M$ and observations $X$ it returns probabilities $C$ such that:

$$C_i = \sum_{j=1}^{|S|} w_j * P(O_i = S_j | X_{1:i-t}, M)$$

where $S$ is a set of states. The method uses the identical computation trick as the previous one. The only difference is the last step where the probabilities of the query are computed from the most probable future state distribution.

#### 4.6.2.2 Scaling

The values used for learning the models were so small that the floating point operation precision was a problem during model fitting. Therefore, the input values are transformed so that they have a mean value of 0 and a variance 1.

After the learning is completed, the Gaussian distributions corresponding to hidden states are transformed using an inverse transformation to match the original data. The second option would be to leave the model as it was learnt and transform the validation data. But this approach might lead to confusion in case of validation and would be less straightforward for trading strategies, because the code removing the normalisation would have to be present on multiple different places.

#### 4.6.2.3 Higher Order Models

Higher order model *HMM*s are transformed to a corresponding first order *HMM*. Thanks to this transformation the evaluation algorithm can remain exactly the same as for the first order models and the learning has been just slightly modified. It is obvious that this simplification leads to a significant loss in both the *CPU* and the memory consumptions. However, it is possible to optimize the corresponding parts of the code later.

The easiest way to explain the transformation is to use an example. Suppose we would like to have a two state second order *HMM* with states $a$ and $b$. The model can be parametrized as follows:

$$\begin{aligned}
initialstates :& P(S_1 = a) \\
secondstates :& P(S_2 = a|S_1 = a) \\
& P(S_2 = a|S_1 = b) \\
transitions :& P(S_i = a|S_{i-1} = a, S_{i-2} = a) \\
& P(S_i = a|S_{i-1} = a, S_{i-2} = b) \\
& P(S_i = a|S_{i-1} = b, S_{i-2} = a) \\
& P(S_i = a|S_{i-1} = b, S_{i-2} = b) \\
emissions :& P(X_i = x|S_i = a) \\
& P(X_i = x|S_i = b)
\end{aligned}$$

The first step in a transformation to a first order *HMM* is to replace the states with a crossproduct of the states ($aa$, $ab$, $ba$, $bb$). And then following restrictions are imposed on the model:

$$\begin{aligned}
transitions :& P(S_i = ba|S_{i-1} = aa) = 0 \\
& P(S_i = bb|S_{i-1} = aa) = 0 \\
& P(S_i = aa|S_{i-1} = ab) = 0 \\
& P(S_i = ab|S_{i-1} = ab) = 0 \\
& P(S_i = ba|S_{i-1} = ba) = 0 \\
& P(S_i = bb|S_{i-1} = ba) = 0 \\
& P(S_i = aa|S_{i-1} = bb) = 0 \\
& P(S_i = ab|S_{i-1} = bb) = 0 \\
emissions :& P(X_i = x|S_i = aa) = P(X_i = x|S_i = ba) \\
& P(X_i = x|S_i = ab) = P(X_i = x|S_i = bb)
\end{aligned}$$

The transformation to an even higher order is similar. The disadvantage is a high sparsity of the matrix.

The learning itself was intended to be performed using parameter restriction. However, it was discovered that restricting the parameters makes the learning process incredibly slow

as the direct optimization is used instead of the *Baum-Welch* algorithm. In the end, the learning method of the library was appropriately modified to support these simulated higher order *HMMs*.

Restricting the transition parameters to zero was not a problem. If the zeros are appropriately placed during the initial parameter estimation, they will never disappear during the learning process.

The emission probabilities are estimated together for each group of states and then assigned to each member. This does not produce computation overhead during the learning process, but the validation could be further improved. This is because during validation, the response of each state is computed separately even though they will be the same for states belonging to one group.

#### 4.6.2.4 Stationary Models

Stationary models have initial probabilities equal to the stationary distribution of the transition probabilities. To achieve this a slightly dirty approach was chosen. The model is learned using the usual *EM* algorithm (*Baum-Welch*), but after each iteration the initial state probabilities distribution is replaced by a stationary distribution of the transition matrix. We are aware that this technique is not perfect, but it is simple and the difference should not be large especially if dealing with rather a long time series.

#### 4.6.2.5 Smoothing

The smoothing is implemented in a similar way as the stationariness mentioned before. A specified amount of Laplacian smoothing is added to the initial and the transition probabilities after each iteration. This makes all these distributions more uniform, which decreases the likelihood of the training data. But it also eliminates the *no transition at all* problem during validation and it might improve the likelihood of the validation data.

#### 4.6.2.6 Weights

The original learning algorithm did not support weighting inputs, so the appropriate modifications have been added to our improved version. As the ability of weighting the inputs is quite common in case of the *Baum-Welch* algorithm, it would be a waste of time to further comment it [Moss, 2008].

### 4.6.3 Naive Bayes

A custom implementation of the *Naive Bayes* classification is used. Our version is compatible with the representation of the *tm* package and as the input a corpus or a *term-document* matrix can be used for both learning and validation.

The inner mechanism exploits the following fact that

$$log(\prod_i x_i) = \sum_i log(x_i)$$

.

Therefore, logarithm values for the model can be precomputed and the whole process of computing classification probabilities can be expressed as a matrix multiplication. The classification itself is just a selection of the most probable classification.

### 4.6.4 SVM

Again, after a short research it was obvious that there are a couple of libraries capable of solving various *SVM* tasks. The selection itself is based on the *Journal of Statistical Software* [Karatzoglou et al., 2006], which compares four libraries:

- kernlab: ksvn [Karatzoglou et al., 2004]
- e1071: libsvm [Dimitriadou et al., 2011]
- klaR: lightSVM [Weihs et al., 2005]
- svmpath [Hastie, 2009]

The *svmpath* library was rejected, because it is not suitable for large datasets. The *ksvn* also has speed issues and in addition requires a separate installation of the library (free for non-commercial usage), so it was not chosen either.

The two remaining packages have the core parts written in *C++*. Both the libraries have a similar set of features allowing to perform the classification and regression tasks using multiple kernels. Even though the *ksvm* library has more built-in kernel functions, the *libsvm* was used in our implementation. The reasons were the computation performance and the ability to work with sparse matrices.

There are also two more *SVM* implementations for *R* not covered by the article. The first is called *shogun*, but it is no longer present on *CRAN*. However, it can be downloaded and compiled manually. The second is called *penalizedSVM* [Becker et al., 2010] and allows penalisation of attribute weights. As this feature is not required by our design, we will stick to the *libsvm* library.

### 4.6.5 Simulated Trading

There are several *R* packages dealing with simulated trading. However, they are hard to use as they require specification of the trading rules in some abstract way, creation of a portfolio object and the evaluation is done by calling a specified sequence of functions. These functions maintain some internal states hidden from the user in their environment, which makes the calls sequence-dependent and a lot of mistakes can be introduced by not cleaning that environment. Moreover, the evaluation is really slow and it took a lot of time to get compatible versions together.

That why we decided to stick to our custom evaluation function, which is really simple. Our strategy provides a vector containing "0" for no trade, "1" for buy and "-1" for sell short. During the evaluation, we just multiply the vector with relative difference changes and sum it up. It is equivalent to spending exactly the same amount of money every evening and selling the stock on the evening after. This is not realistic as the market fees are completely ignored

and holding stock for several days has been decomposed to multiple buy/sell operations. However, the evaluation is fast and precise enough for our needs.

Our strategies are pretty simple functions. We just call the *hmm_predict* or *hmm_confidence* (subsubsection 4.6.2.1) function and compare the results with some threshold. If the result is larger than the threshold, then the stock is bought. If short trading is allowed, another lookup is made and if the prediction is smaller than negation of the threshold, the stock is bought short.

## 4.7 Other Software Tools

Following software tools were used during the thesis creation:

- RStudio: *R* editor [RStudio, 2011]
- Texmaker: LaTeX editor [Brachet, 2011]
- UMLet: *UML* drawing tool [UMLet, 2011]

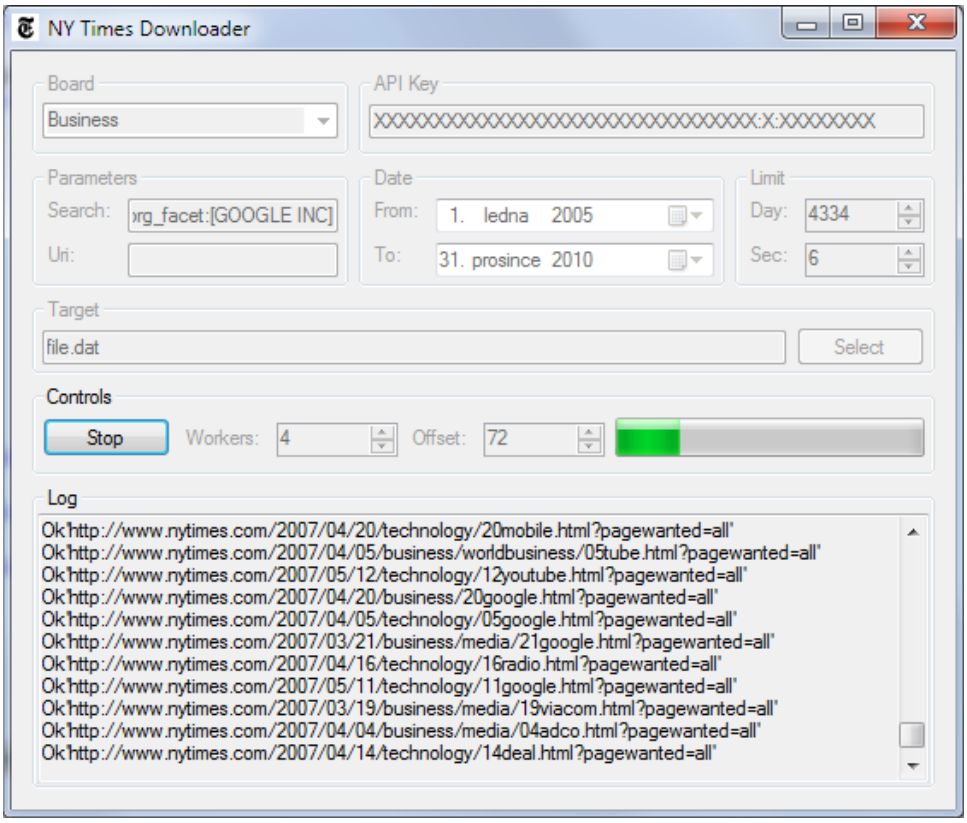



Figure 4.2: Screenshot of the *NY Downloader's* user interface. The *GUI* contains several controls for restricting the retrieved articles. More advanced searching can be achieved using the 'Search' and 'Uri' fields.

# Chapter 5

# Testing and Results

> One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man.
>
> Elbert Hubbard

## 5.1 Data Selection

The number of time-series we can choose from is huge, but we are constrained by the number of news articles available. Therefore, currently the best choice we can make is to leave futures or currency trading behind as there is a large number of relevant articles and focus on stock exchanges.

Many of the previous researches focused on the index *S&P 500*, taking into account all the available news for all the companies present in this index. However, using the *S&P 500* to measure impact on a particular company does not seem to be the right way, because the companies are grouped and weighted according to the market capitalization. Therefore, the index will hold still if the price of some less important company changes and the index will most probably change if some generally good news article appears, slightly changing the values of all the components. As a result, the articles categorized according to the *S&P 500* will measure the general economical mood rather than impact on an individual company.

In order to select companies with a sufficient amount of news information, all the *Business* and *Technology* articles between years 2000 and 2010 were downloaded and organizations present in their *org_facet* were analysed. As we plan to investigate whether the article classification of one company can improve the prediction performance for another company, we want the companies to be from the same area of business. According to our results, the companies producing the largest amount of news were software companies (table: D.1). After this, the software companies not present on the market were filtered out and we ended up with the following list:

- Amazon.com Inc.
- AOL Inc.
- Apple Inc.
- eBay Inc.
- Electronic Arts Inc.
- Google Inc.
- Microsoft Corp.
- Yahoo! Inc.

## 5.2 Text Analysis

Tuning of model parameters was performed on news articles from years 2000 to 2010. The articles between January and November 2011 were left for validation.

### 5.2.1 News Articles Characteristics

Before creating and testing the language models, two simple tests were performed on the data. We hoped they would help us to get some additional insight into the data itself.

#### 5.2.1.1 Time Alignment

The first of the tests was performed on all the text data available. The relative change $r(t)$ of closing values $x$ was computed for two different day spans $t = \{1; 3\}$.

$$r_i(t) = \frac{x_i - x_{i-t}}{x_i}$$

After this, the news articles were aligned to the corresponding time series relative changes in such a way that the article release date was corresponding to the second day of a time span. In other words, the $t = 1$ relative change was the same, while the $t = 3$ relative change was shifted by 2 days. If the article was released during a weekend, then the article was aligned to the next Monday.

$$r'_i(t) = r_{i+t-1}(t)$$

The last step was calculation of the *standard deviation* of the data. As the relative change is distributed symmetrically (the mean of relative change is almost equal to 0), the *standard deviation* can be considered a measure of how much is the market likely to change after an article is released.

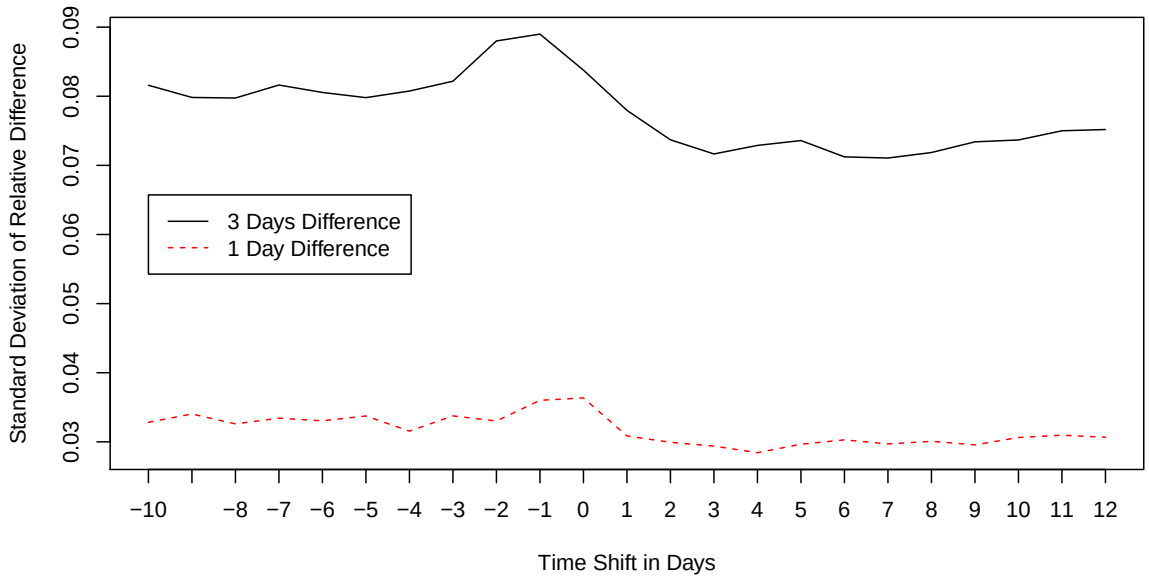



Figure 5.1: The standard deviation of a relative difference for days relative to the article's release date

It can be seen from the figure 5.1 that the average article impact is the greatest at the day of the release and the day before. The day after the release, the market is becoming more stable and after approximately four days it starts recovering to its previous fluctuation. The *Efficient Market Hypothesis*, which states that all the known information is being reflected by the market immediately, could serve as an explanation of this behavior. It is also very likely that some news articles are reflected by the market prior to its release by a news portal.

#### 5.2.1.2 Article Distributions

The second simple test of the text-data focused on the distribution of up-down direction among the article's organizations and categories. A threshold value of relative change 0.05 of a 3 day difference was chosen to classify an article into one of the influence categories: *"positive"*, *"negative"*, *"none"* or *"unknown"* if required data was missing. The choice of the threshold value was based on the relative change distribution of all the time-series, therefore more stable stocks can be a bit misclassified and have more weight in the *"none"* category. The 3 days difference was chosen over 1 day difference with the intention of partially eliminating the noise.

Partial results from the analysis are in the table (table: D.2). It was hoped that the values of the *"none"* column could be used to decide which categories are relevant (the more *"none"* values, the less importance) and which time shift is the best. The *"none +1"* is the desired one (because it predicts the future behaviour), whereas the *"none -1"* was expected to give the most relevant results (high *"none"* value in less relevant columns).

The expectations were not completely fulfilled. The level of noise has to be high, since the values, especially the ones related to groups with a small article count can be surprising. For instance, the *AOL* highest value in the *"none"* column is rather suspicious as well as the low *"none"* value of *"Week in Review"*. However, the high *"none"* values of *"Obituaries"* and *"Home and Garden"* are quite a pleasant surprise.

As for the time shift, the most reasonable column seems to be the *"none"*. Both the *"none +1"* and *"none -1"* have rows with suspicious values like *"Theater"* or *"Education"*. Also unanticipated was the measure of impact of the *"New York and Region"*, *"Corrections"* and *"Arts"* categories, because our expectation was much lower.

### 5.2.2 Parameters Tuning

After the initial data check, it was time to try out our model. The data was separated into the training part dedicated for parameter tuning and the validation part for the final model validation. The training part was from year 2000 to year 2009 while the range from 2010 to 2011 was left for the validation purposes. The parameters were tuned using the eight fold *Window Validation* method.

#### 5.2.2.1 Base Model

At first, we decided to try a model containing all the common preprocessing steps. The term count was reduced to 15% and only the most frequent terms were kept. A *SVM* with a linear kernel was used as the classification algorithm. The tests were performed only for *Microsoft*, *Google*, *Yahoo* and *Apple* data as these companies had the largest number of articles. All the companies were analysed separately.

The features were extracted using a simple threshold algorithm into four categories: "positive examples", "negative examples", "neutral examples" and "unknown examples". Unknown examples were those with a date that could not be matched with a relative change of the price. Positive were articles with the corresponding relative change greater than 1/2

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.306 | 0.323 | 0.337 | 0.310 |
| Learn -1 | 0.403 | 0.403 | 0.399 | 0.374 |
| Learn 0 | 0.306 | 0.357 | 0.336 | 0.350 |
| Learn 1 | 0.291 | 0.293 | 0.344 | 0.327 |

(a) Relative Difference 1

| Timeshift RelDiff5 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.289 | 0.328 | 0.328 | 0.293 |
| Learn -1 | 0.310 | 0.361 | 0.337 | 0.306 |
| Learn 0 | 0.322 | 0.339 | 0.370 | 0.341 |
| Learn 1 | 0.352 | 0.352 | 0.339 | 0.346 |

(b) Relative Difference 5

Table 5.1: Correctly classified Yahoo articles for relative differences 1 and 5 days using a *SVM* model.

of the average relative change. Negative articles were determined similarly and the rest were the neutral examples.

We also decided to introduce a *time-shifting* parameter allowing slight time delays between articles and the related time-series. The underlying idea was that the articles might be already known to traders a day or two in advance.

In addition, a relative difference of 1, 3 and 5 days was tested. They represented a different trade-offs between the value precision and noise.

The most successful model was a *Yahoo!* model with 1 day relative difference. It scored 40% probability of correct classifications (table: 5.1a). The second best model was again a Yahoo model. It was the one with relative difference of 5 days (table: 5.1b). One can notice that short horizon tends to work mostly for previous day and long horizon mostly for diagonal.

#### 5.2.2.2 Data Filtering

The following test was devised to decide if it is better to keep all the articles or filter them in some way. We kept just the articles labelled as *Business* or *Technology*. Even thought there was a slight performance improvement in some cases, the filtering was mostly harmful (table: 5.2a). So even articles not related to business or technology might have an impact on the classification quality.

We further tried to filter out all the articles containing more than one company. It is quite difficult to decide the major company an article is actually about, so it seemed like a good idea to remove these problematic articles. However, a dramatic decrease in classification quality was experienced (table: 5.2b), so we abandoned the idea of filtering for now.

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.3851 | 0.3346 | 0.3741 | 0.3806 |
| Learn -1 | 0.3891 | 0.3955 | 0.3848 | 0.3588 |
| Learn 0 | 0.3676 | 0.3130 | 0.3239 | 0.3134 |
| Learn 1 | 0.3417 | 0.2869 | 0.3111 | 0.3024 |

(a) Technology Business Filtering

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.2862 | 0.3534 | 0.3035 | 0.3044 |
| Learn -1 | 0.3105 | 0.3707 | 0.2951 | 0.4164 |
| Learn 0 | 0.3259 | 0.4258 | 0.3749 | 0.3436 |
| Learn 1 | 0.2866 | 0.3357 | 0.3357 | 0.3025 |

(b) Unique Filtering

Table 5.2: Correctly classified Yahoo articles using a *SVM* model with different filtering.

| Timeshift RelDiff5 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.370 | 0.357 | 0.3547 | 0.329 |
| Learn -1 | 0.360 | 0.316 | 0.331 | 0.304 |
| Learn 0 | 0.365 | 0.324 | 0.375 | 0.334 |
| Learn 1 | 0.352 | 0.311 | 0.347 | 0.298 |

Table 5.3: Correctly classified Yahoo articles using the *Naive Bayes* model and relative difference 5.

#### 5.2.2.3 Naive Bayes

The following test was designed to compare the *Naive Bayesian* model to the previous models. The test was performed with the smoothing parameter equal to 1.

The obtained results were consistent with the results of tests performed by others. The *Naive Bayes* was outperformed by *SVM*. As usually, counter examples can be found, but in most cases the *SVM* provided better and more stable results (table: 5.3).

#### 5.2.2.4 Term Weighting

Commonly used approach is the *Term Frequency - Inverse Document Frequency* weighting. We have decided to investigate how it would work in our case.

The resulting performance again did not beat our base model. It looks like every modification to our base model makes things only worse.

#### 5.2.2.5 Together / Alone

The 'together' test tried to learn one model for categorizing all the examples. However, this approach didn't work as expected (table: 5.5). The performance was even lower. The

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.303 | 0.318 | 0.291 | 0.340 |
| Learn -1 | 0.365 | 0.345 | 0.342 | 0.338 |
| Learn 0 | 0.310 | 0.348 | 0.306 | 0.303 |
| Learn 1 | 0.350 | 0.318 | 0.298 | 0.328 |

(a) Technology Business Filtering

| Timeshift RelDiff5 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.337 | 0.378 | 0.319 | 0.301 |
| Learn -1 | 0.357 | 0.347 | 0.334 | 0.326 |
| Learn 0 | 0.296 | 0.352 | 0.357 | 0.293 |
| Learn 1 | 0.339 | 0.367 | 0.296 | 0.332 |

(b) Unique Filtering

Table 5.4: Correctly classified Yahoo articles using *SVM* model with different filtering.

| Together RelDiff5 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.341 | 0.348 | 0.343 | 0.341 |
| Learn -1 | 0.326 | 0.331 | 0.328 | 0.330 |
| Learn 0 | 0.329 | 0.329 | 0.323 | 0.337 |
| Learn 1 | 0.326 | 0.342 | 0.342 | 0.335 |

Table 5.5: Correctly classified Yahoo articles using *Naive Bayes* model and relative difference 5.

possible cause might be a different set of significant words for each company and/or wrong initial article classification.

#### 5.2.2.6 Term Reduction

The last model alteration was tuning of the term reduction parameter. Lower values of this parameter mean lower total number of terms and therefore lower dimensionality. As usual, excessively low number would lead to oversimplification and therefore poor results and an overly high number would include even insignificant terms, which lowers the performance too.

Values ranging from 5% to 30% with were tried. The results however were deeply volatile and even the closest members (10%, 20%) had produced completely different results (table: 5.6). Therefore, it is likely that the performance of the best model so far is caused mostly by coincidence.

### 5.2.3 Validation

The validation was performed using data from years 2010 and 2011. It was an interesting coincidence that we ended up with a model identical to the initial estimate. All the

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.2871 | 0.3020 | 0.3294 | 0.3336 |
| Learn -1 | 0.3401 | 0.3420 | 0.3800 | 0.3693 |
| Learn 0 | 0.3568 | 0.3249 | 0.3589 | 0.3126 |
| Learn 1 | 0.2936 | 0.2995 | 0.3333 | 0.3565 |

(a) Term Reduced to 10%

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.2870 | 0.3210 | 0.3062 | 0.3272 |
| Learn -1 | 0.2975 | 0.2869 | 0.2786 | 0.3377 |
| Learn 0 | 0.2998 | 0.2660 | 0.2976 | 0.3251 |
| Learn 1 | 0.2703 | 0.2875 | 0.2978 | 0.2956 |

(b) Term Reduced to 20%

Table 5.6: Correctly classified Yahoo articles after a term reduction. The *SVM* model was used.

| Timeshift RelDiff1 | Test -2 | Test -1 | Test 0 | Test 1 |
|---|---|---|---|---|
| Learn -2 | 0.2500 | 0.0625 | 0.1875 | 0.6875 |
| Learn -1 | 0.1250 | 0.2500 | 0.2500 | 0.5000 |
| Learn 0 | 0.3750 | 0.3125 | 0.5000 | 0.3750 |
| Learn 1 | 0.4375 | 0.5000 | 0.5000 | 0.3125 |

Table 5.7: Validation of article classification using the best model found. It is a *SVM* based model with the term count reduced to 15% and without any additional weighting or filtering.

improvements of the carefully chosen initial model were in fact harmful.

The final classification performance was 50% of correct answers for a nearly homogeneous distribution of three categories (table: 5.7). The only problem is that even though two years of data were left for validation purposes, the amount of articles was rather small. Therefore, additional tests have to be repeated in the future to confirm or reject this result.

We will also try to further investigate the problem. Our weak point is likely to be in the article selection, classification or both of them. We will try to identify a better way of filtering out irrelevant articles. Also switching the classification problem for a regression one might bring a new light into the problem and help us to better understand it.

The good news is that the capabilities of our framework were not entirely examined and we can continue in testing.

The missing test results are stored on enclosed *DVD* (Appendix E).

## 5.3 Markov Models

Similarly to the text analysis, the data was also separated into the learning and the validation set, but because of a small mistake the learning set was enlarged to an interval December 1999 to January 2011. The mistake was discovered after all testing was finished and repeating the whole process with the previously intended interval would be impossible before the deadline. Therefore, only February 2011 to November 2011 was left as the validation data.

All the tests were performed using the *sliding window validation*. This simulated the fact that we are not aware of the future. More precisely, 8 chunks of the same length were created from the data. The very first 3 chunks were used to learn the model and 4th used for evaluation in the first iteration. Then the window moved, 4 chunks were used for learning and 5th one for evaluation. As a result, there were a total of 5 evaluations, which were later averaged to create the final value.

### 5.3.1 Parameters Tuning

All these models have one Gaussian distribution representing a hidden state. The learning process has a randomized initialization and 50 iterations of the *EM* algorithm. The number of hidden states is up to 11, because larger models are likely to be overfitted and the learning takes too long.

#### 5.3.1.1 Relative/Logarithmic Difference

This test is supposed to determine, whether a relative difference is preferable in raw form or its logarithm. It is believed that the conversion to logarithmic scale can have a positive impact on the performance, because of lower sensitivity to extreme jumps.

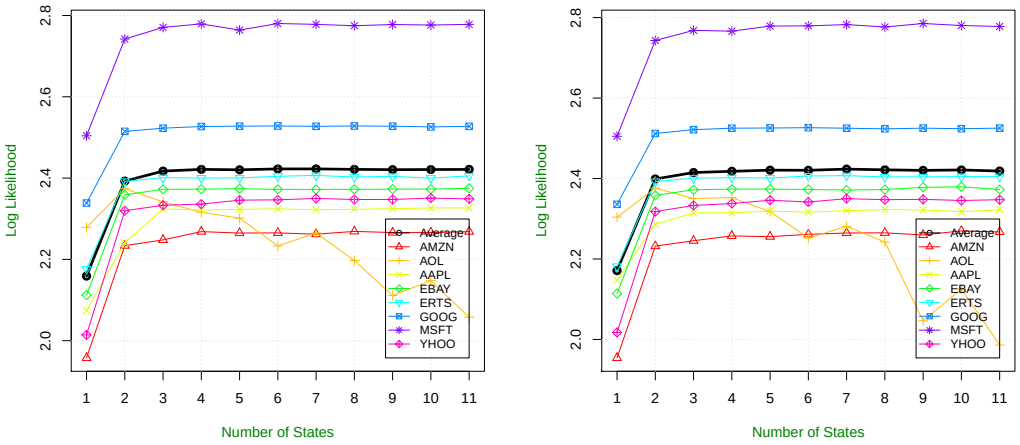
The likelihood test (table: 5.2) has no clear winner, because the curves are rather similar. It can only be said that the model with the logarithmic data input has a bit smoother curves. As for the number of hidden states, both the curves start with a steep improvement and then they remain around the same value with the maximum near 7 states. However, it is better to keep things simple, so the best results are expected for 3 state models. They have a similar likelihood as the 7 state ones and a significantly smaller amount of parameters to estimate.

It can be also seen that the real maximum might be a bit further, if the *AOL* data is omitted. The *AOL* likelihood curve has a lot of noise in it, because it is short (just a year of data).

The direction test does not have a clear winner either (figure: 5.3). It can be seen that *Microsoft* and *Apple* are the only companies where prediction works. An interesting fact is that taking the exact opposite value for *AOL* would result in almost 60% of correct answers.

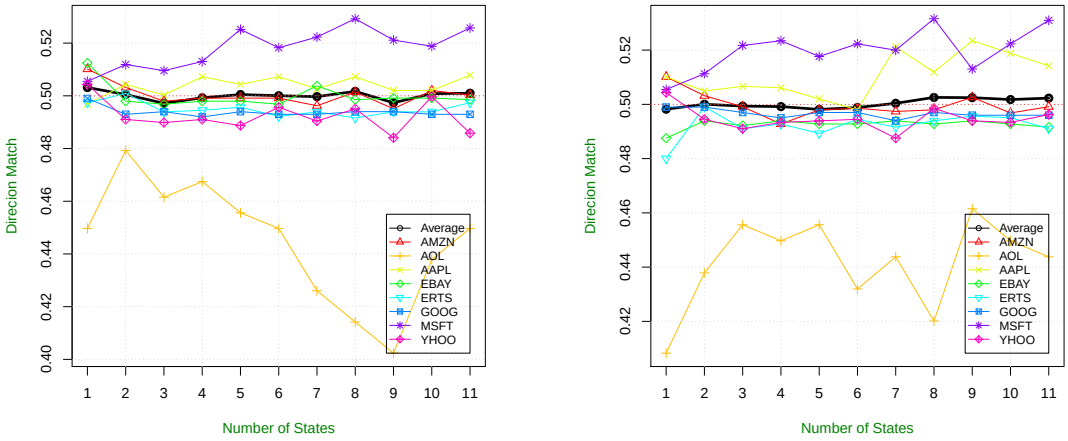
For the relative difference the best estimating model is the one featuring 1 hidden state. This is equivalent to one Gaussian distribution and therefore the value predicted is in fact the mean of the learning data.

According to the results of the trading simulations the preferred trading method is the *Confidence Threshold* (table: 5.8). The *Simple Threshold* (table: 5.9) method has just one comparable income. It is a *Logarithmic Difference* based model with 11 states.




(a) Relative Difference (b) Logarithmic Difference

Figure 5.2: Simple HMM models logarithmic likelihood. The image shows the average logarithmic likelihood of the testing data. The values are plotted individually for each series and also together as a weighted average.




(a) Relative Difference (b) Logarithmic Difference

Figure 5.3: The figure shows simple HMM models direction prediction. The image displays the probability that the model correctly determines the future direction of the time series.

There are two configurations with a daily average profit exceeding 0.05%, which is more than 12% a year. They are both a *Relative Difference* HMM equipped with the *Confidence Threshold* trading strategy. Even though the 8 state model has better performance, the 3 state model (figure: 5.4) seems to be the preferable one as it has smoother lines. Also the

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.273 | 0.267 | 0.263 | 0.289 | 0.403 | 0.329 | 0.012 | 0.000 | 0.000 |
| 3 | 0.513 | 0.513 | 0.507 | 0.452 | 0.453 | 0.436 | 0.036 | −0.007 | 0.000 |
| 4 | 0.485 | 0.487 | 0.504 | 0.489 | 0.495 | 0.335 | 0.012 | 0.000 | −0.004 |
| 5 | 0.495 | 0.496 | 0.505 | 0.415 | 0.457 | 0.368 | 0.074 | 0.000 | −0.004 |
| 6 | 0.346 | 0.358 | 0.392 | 0.437 | 0.497 | 0.323 | 0.005 | −0.009 | −0.004 |
| 7 | 0.288 | 0.309 | 0.349 | 0.417 | 0.416 | 0.433 | 0.093 | −0.005 | −0.004 |
| 8 | 0.512 | 0.494 | 0.512 | 0.513 | 0.532 | 0.348 | 0.017 | −0.013 | −0.004 |
| 9 | 0.276 | 0.281 | 0.249 | 0.261 | 0.326 | 0.413 | 0.010 | −0.007 | −0.004 |
| 10 | 0.298 | 0.290 | 0.294 | 0.288 | 0.343 | 0.314 | 0.060 | −0.010 | −0.006 |
| 11 | 0.386 | 0.366 | 0.360 | 0.368 | 0.394 | 0.230 | 0.025 | 0.007 | −0.004 |

(a) Relative Difference

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.134 | −0.075 | −0.007 | 0.000 | 0.000 |
| 2 | 0.461 | 0.451 | 0.443 | 0.246 | 0.208 | 0.100 | −0.018 | 0.000 | 0.000 |
| 3 | 0.457 | 0.469 | 0.464 | 0.478 | 0.383 | 0.271 | −0.014 | −0.007 | 0.000 |
| 4 | 0.402 | 0.409 | 0.377 | 0.397 | 0.438 | 0.242 | −0.010 | −0.007 | 0.000 |
| 5 | 0.438 | 0.438 | 0.443 | 0.335 | 0.377 | 0.210 | 0.047 | −0.003 | −0.004 |
| 6 | 0.299 | 0.313 | 0.301 | 0.420 | 0.328 | 0.169 | −0.006 | −0.007 | −0.004 |
| 7 | 0.378 | 0.377 | 0.398 | 0.410 | 0.353 | 0.253 | −0.048 | −0.002 | −0.004 |
| 8 | 0.398 | 0.390 | 0.396 | 0.431 | 0.401 | 0.380 | 0.041 | −0.010 | −0.004 |
| 9 | 0.401 | 0.374 | 0.347 | 0.343 | 0.309 | 0.309 | −0.001 | 0.007 | −0.006 |
| 10 | 0.329 | 0.364 | 0.278 | 0.300 | 0.237 | 0.158 | −0.045 | −0.005 | 0.000 |
| 11 | 0.337 | 0.322 | 0.336 | 0.319 | 0.352 | 0.283 | 0.038 | 0.002 | −0.006 |

(b) Logarithmic Difference

Table 5.8: The table contains average daily profit for the confidence-based strategy. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing the number of hidden states of the model.

success of the 8 state model is mostly caused by a great improvement in *AOL* time series, which is short and noisy.

#### 5.3.1.2 Learning Together

This model tries to learn from all the time series together and then evaluate on all the time series together. The idea is that more data series equals more information. And more information usually results in a better model. The only problem might be the difference of time series. If the series contains differently behaving groups, the future model might be useless.

The *Logarithmic Likelihood* test does not reveal any new knowledge and therefore the results are omitted. Inquiring readers might find directories containing all the graphs and tables on the enclosed DVD (Appendix E).

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.045 | 0.025 | 0.060 | −0.079 | −0.040 | 0.000 | 0.000 | 0.000 |
| 2 | 0.168 | 0.162 | 0.154 | 0.071 | 0.101 | 0.258 | 0.073 | 0.001 | 0.000 |
| 3 | 0.204 | 0.215 | 0.219 | 0.223 | 0.336 | 0.392 | 0.081 | 0.000 | 0.000 |
| 4 | 0.178 | 0.172 | 0.199 | 0.223 | 0.264 | 0.283 | 0.071 | −0.004 | 0.000 |
| 5 | 0.276 | 0.295 | 0.265 | 0.326 | 0.318 | 0.302 | 0.100 | −0.004 | 0.000 |
| 6 | 0.332 | 0.364 | 0.371 | 0.354 | 0.377 | 0.292 | 0.114 | −0.003 | 0.000 |
| 7 | 0.258 | 0.247 | 0.219 | 0.269 | 0.246 | 0.254 | 0.178 | −0.004 | 0.000 |
| 8 | 0.316 | 0.304 | 0.293 | 0.282 | 0.338 | 0.259 | 0.118 | −0.003 | −0.002 |
| 9 | 0.157 | 0.173 | 0.155 | 0.111 | 0.164 | 0.243 | 0.125 | −0.005 | 0.000 |
| 10 | 0.310 | 0.314 | 0.306 | 0.321 | 0.308 | 0.280 | 0.088 | −0.001 | 0.000 |
| 11 | 0.294 | 0.287 | 0.261 | 0.290 | 0.259 | 0.326 | 0.157 | −0.004 | 0.000 |

(a) Relative Difference

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.140 | −0.064 | 0.000 | 0.000 | 0.000 |
| 2 | 0.298 | 0.312 | 0.284 | 0.289 | 0.177 | 0.182 | −0.016 | 0.000 | 0.000 |
| 3 | 0.277 | 0.277 | 0.271 | 0.260 | 0.271 | 0.330 | −0.010 | 0.000 | 0.000 |
| 4 | 0.146 | 0.147 | 0.153 | 0.134 | 0.136 | 0.305 | −0.016 | 0.000 | 0.000 |
| 5 | 0.110 | 0.101 | 0.117 | 0.170 | 0.154 | 0.218 | 0.064 | −0.004 | 0.000 |
| 6 | 0.102 | 0.092 | 0.093 | 0.106 | 0.239 | 0.281 | 0.006 | 0.000 | 0.000 |
| 7 | 0.284 | 0.295 | 0.308 | 0.309 | 0.197 | 0.188 | −0.019 | −0.002 | 0.000 |
| 8 | 0.368 | 0.361 | 0.360 | 0.341 | 0.287 | 0.264 | −0.008 | −0.002 | −0.002 |
| 9 | 0.385 | 0.376 | 0.383 | 0.356 | 0.204 | 0.110 | 0.012 | −0.008 | 0.000 |
| 10 | 0.395 | 0.401 | 0.389 | 0.373 | 0.231 | 0.127 | −0.054 | −0.005 | 0.000 |
| 11 | 0.444 | 0.433 | 0.455 | 0.444 | 0.328 | 0.215 | 0.078 | −0.008 | 0.000 |

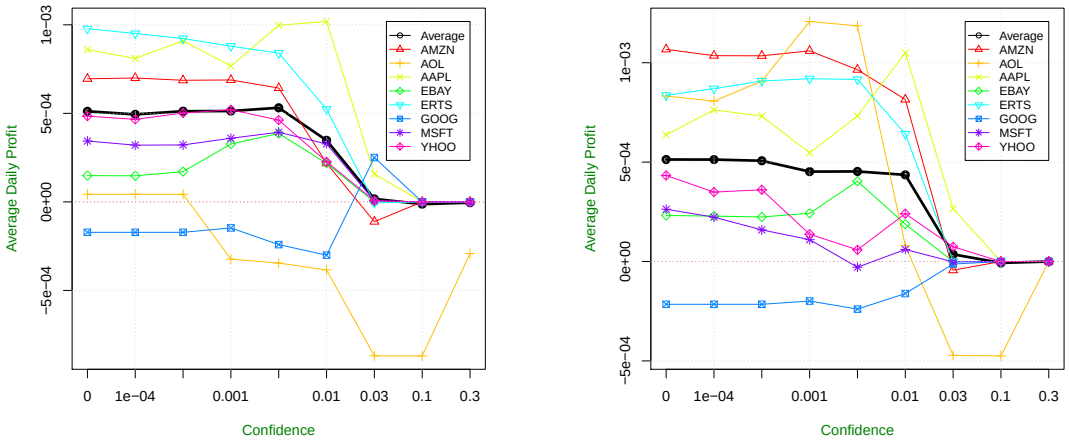
(b) Logarithmic Difference

Table 5.9: The table contains average daily profit for the threshold-based strategy. The values are increased 1000 times to avoid leading zeros. The columns are representing predicted value thresholds while the rows are representing the number of hidden states of the model.

The performance of the direction test seems to be improved. The *Relative Difference* based HMM learnt all together is the first discovered model with average prediction near 51% (figure: 5.5a). It is not much, but it is better than previous models.

The trading is also slightly improved (tables: D.4, D.3). The average income of *Relative Difference* based 3 states *HMM* with *Simple Threshold* strategy is 0.00068, which is approximately 17% per year. It is also the first model which does not produce loss on any individual series (figure: 5.5b).

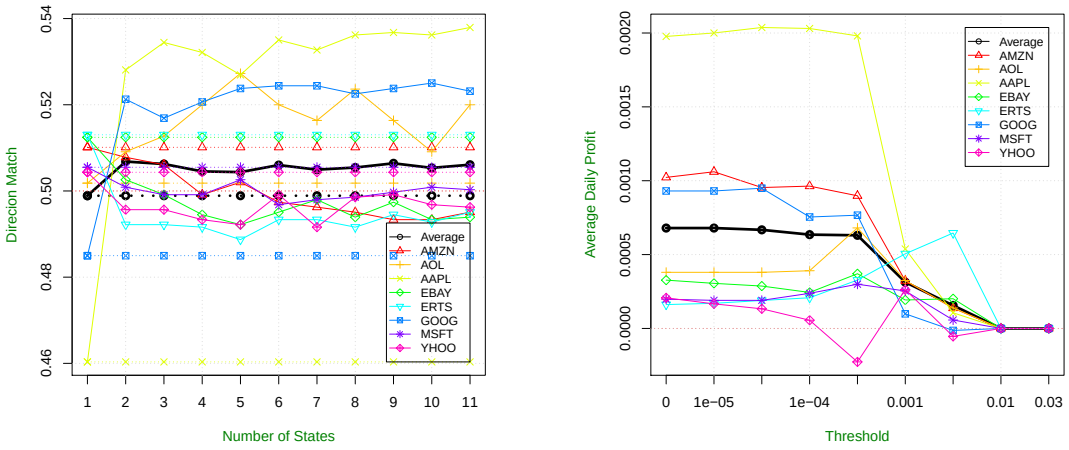
Generally, the learning together seems to have more advantages than disadvantages, but it surely depends on the selection of the individual series to be joined.




(a) 8 State Relative Difference HMM

(b) 3 State Relative Difference HMM

Figure 5.4: The performance of the best trading models so far. They are both HMMs based on a *Relative Difference* with the *Confidence* based trading strategy. The vertical axis represents the average daily income and the horizontal one represents the threshold of the trading strategy.




(a) Direction Accuracy

(b) 3 States Simple Threshold Trading

Figure 5.5: The figures are showing performance of *Relative Difference* based HMM which is learnt from all the series together.

#### 5.3.1.3 Window Length

The window length test is supposed to determine the impact of shortening the window on model qualities. It can be seen that shortening the learning window actually degrades the

performance of the model (tables: D.5, D.6). The assumed cause is the loss of some learning information.

The only exception is a window with length 1, which makes a slight improvement in at least the *Separate* learning method (table: D.5c). The improvement is mostly noticeable for a higher number of states, so it is either a noise or a complicated behaviour. Maybe it might be a good idea to introduce history weighting to the models or combine a short term separately learnt model with a long term together learnt model.

#### 5.3.1.4 Stationariness

The stationary *HMMs* are those, which initial state distribution is the same as the stationary distribution of the transition probabilities. These models are widely used in infinite time series analysis as they are eliminating the initial noise created by choosing a starting point. In our case it might be better to take as the initial state distribution for testing the state distribution at the end of the learning data.

Because of the length of the testing windows the parameter is expected to affect just a few state distributions from the very beginning of the test data. So the total impact should be small.

The results of the stationary tests confirm the hypothesis about a small impact. It is slightly surprising that almost all the average incomes are actually a bit lower (table: D.7). It might be caused by the learning method or noise in the data.

#### 5.3.1.5 Smoothing

Smoothing might be helpful in the fight against noise or low amount of data. But excessive amount of smoothing can hide relevant information from the model. The smoothing test is supposed to determine what is the right amount of smoothing for our purposes.

The smoothing affects only initial and transition probabilities and leaves the Gaussian distributions unaffected. It is expected that models with a higher number of states will be affected more than the ones with a lower number of states, because the transition probabilities of the smaller models are already pretty close to the uniform distribution.

The smoothing test was performed for values 0.1, 1, 10 and 100. According to the results the smoothing is not such a good idea. Even the lowest value lowers the performance a bit, while values of 10 and more lead to really poor results (table: D.8). It is possible that there is some magical tiny value increasing the performance, but it is unlikely to improve the results significantly.

#### 5.3.1.6 Cropping

The cropping of large values can be useful in case some values in a time series are almost alone in completely different scale. Those values might be caused by an unpredictable external influence or noise and their reflection in a model might be contrapositive. On the other hand too much cropping might cause a loss of relevant information.

The test was performed with values 0.1, 0.2 and 0.4 which were chosen based on the *Relative Difference* values distribution. The results have confirmed that it is a good idea

to crop at least the most extreme values (table: D.9). But too much cropping significantly reduced the performance.

#### 5.3.1.7 Higher Order HMMs

Higher order models are supposed to be a specialized tool for particular cases. They allow to learn a more complicated model on a smaller amount of data by imposing restrictions. All the success depends on how much the data behaves according to the model.

Two kinds of these models were tried, the second order *HMM* and the third order *HMM*. Both models in fact do not fulfil the *Markovian Property* as the future state depends on the previous two or three, but they can be transformed in a way that the property is preserved.

Sadly, *the Higher Order HMM*s failed our expectations and performed very poorly on the test data (tables: D.10, D.11). The best model is a third order 4 state *HMM* and has performance similar to the simple 3 state model. Taking into account the computational difficulties, it does not seem to be worthwhile using such complicated models.

#### 5.3.1.8 Combined Models

The combinations of previously mentioned modifications are likely to result in even better models. However, to try them all would require a lot of a processor time. Therefore, just a chosen subset of combinations is tested. The subset is made up mostly from combinations of previously successful models.

The results are not as promising as we had hoped. The combined models barely beat their single upgraded predecessors (table: D.12). The rest can be found on enclosed DVD (Appendix E).

### 5.3.2 Validation

As has been said before, validation is performed on data between February 2011 and November 2011. The validation dataset is small compared to the learning one. One of the reasons were the newspapers full of the worldwide crisis and it would have been misleading to learn on regular data and validate on the crisis data.

#### 5.3.2.1 Final Models

Final models were selected among the most successful tested models. It was one from each category which seems to do a good job and two from the final combined category (table: 5.10). Secondary selection criterion was the difference from previously selected models, as testing two exactly same models is not likely to reveal anything new.

#### 5.3.2.2 Results

The results were below our expectations. There was only one model which actually made a profit. It was the 11 state *HMM* learnt on a short (one year) period of data of all series

| Name | Order | States | Variable | Limit | Together | Stationary | Smoothing | Trade Type | Trade Threshold |
|---|---|---|---|---|---|---|---|---|---|
| Simple | 1 | 3 | RelDiff1 | 0 | False | False | 0 | conf | 0 |
| Together | 1 | 3 | RelDiff1 | 0 | True | False | 0 | thresh | 0 |
| Short | 1 | 11 | RelDiff1 | 260 | False | False | 0 | conf | 0 |
| Crop | 1 | 3 | RelDiff1_crop0.4 | 0 | False | False | 0 | conf | 0 |
| Ord 3 | 2 | 4 | DiffLog1 | 260 | True | False | 0 | thresh | 0 |
| Stationary | 1 | 3 | DiffLog1 | 0 | True | True | 0 | thresh | 0.001 |
| Ord 2 | 2 | 3 | DiffLog1 | 260 | True | True | 0 | conf | 0 |

Table 5.10: The parameters of the models for final validation. Limit is the number of days the model was learnt from.

together (table: 5.11). Even though it the had highest score during the testing process, it was still a surprise.

It is quite possible, that the winning model's performance is just a matter of luck. After a closer investigation (figure: 5.6), it was discovered that large part of the profit was caused by correctly predicting the fall of the *AOL*. Without the *AOL* the model barely outperforms the *Buy and Hold* strategy. In addition, it has to be taken into account that the simulation was performed without fees a slippage, so the real world performance would be even lower.

| Model | Direction | Trade Test | Trade Validate |
|---|---|---|---|
| Simple | 0.499 | 0.513 | −1.095 |
| Together | 0.488 | 0.680 | −1.208 |
| Short | 0.503 | 0.717 | 0.637 |
| Crop | 0.502 | 0.686 | −1.157 |
| Ord 3 | 0.480 | 0.587 | −0.434 |
| Stationary | 0.486 | 0.696 | −0.686 |
| Ord 2 | 0.477 | 0.511 | −1.479 |

Table 5.11: The parameters of the models for final validation.

The other models performed so badly that they were outperformed even by the *Buy and Hold* strategy (figure: 5.7). It is a open question if performing the validation several times using a smaller window would improve the performance.

#### 5.3.2.3 Distribution Fitness

In order to investigate the cause of the poor performance of *HMM*s, another test was performed. The goal of the test was to decide how much of the data are corresponding to the individual states of the *HMM*. Because a wrong distribution family could be one of the causes.

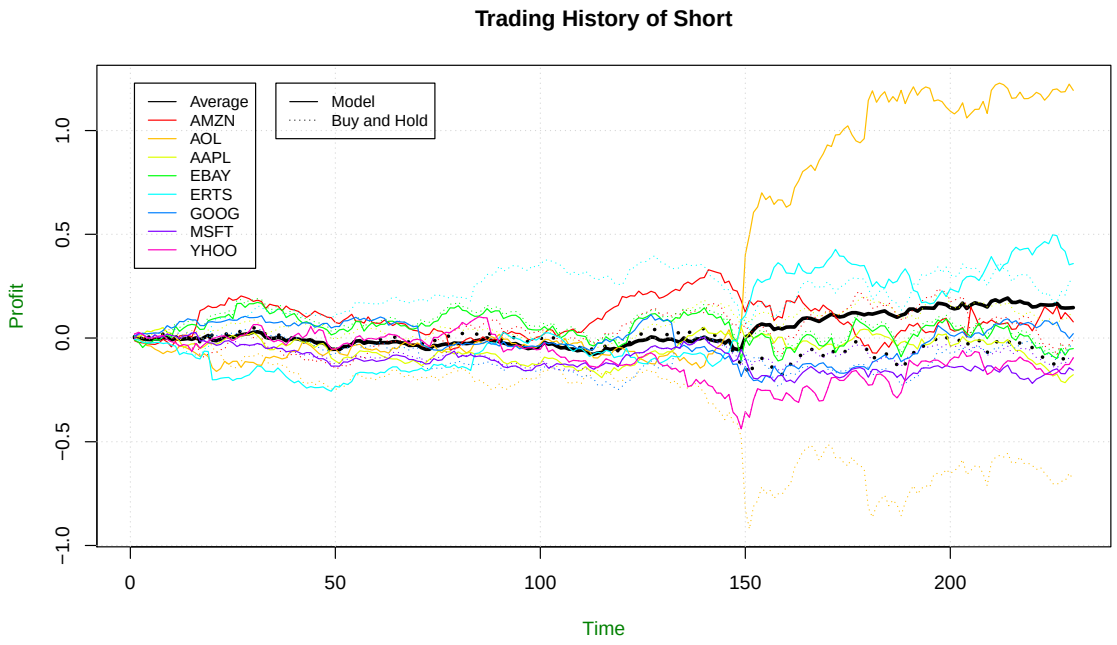



Figure 5.6: Examples of application candidates

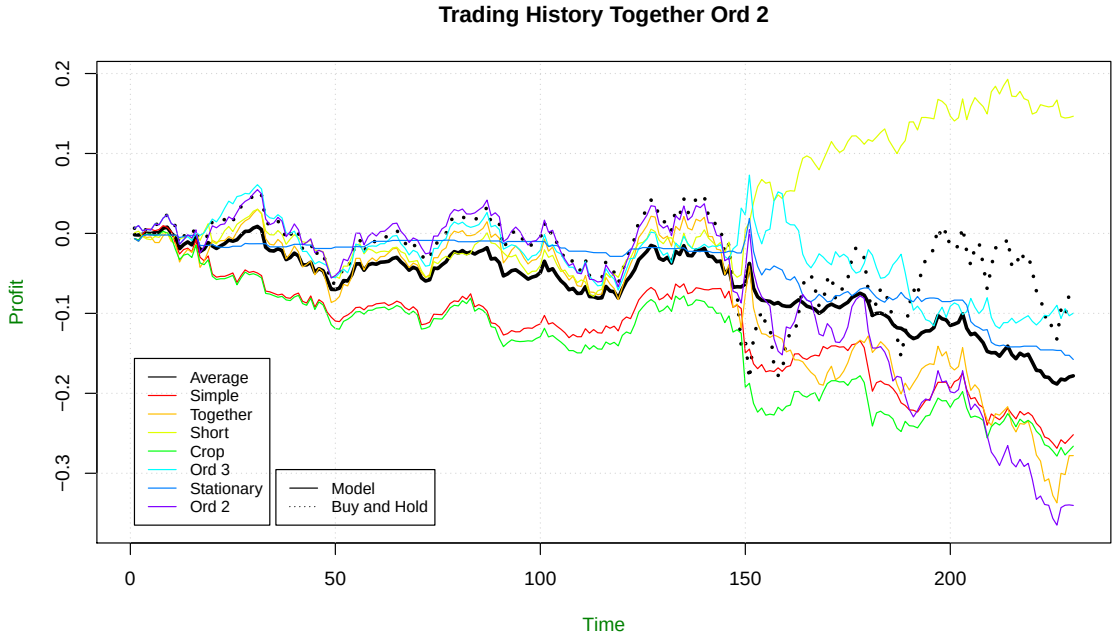



Figure 5.7: Trading Performance of Models During Validation

For each state of the *HMM* a histogram of data weighted by the probability that a par-

ticular record corresponds to the state was constructed. And then the Gaussian distribution corresponding to the state parameters was created for comparison.

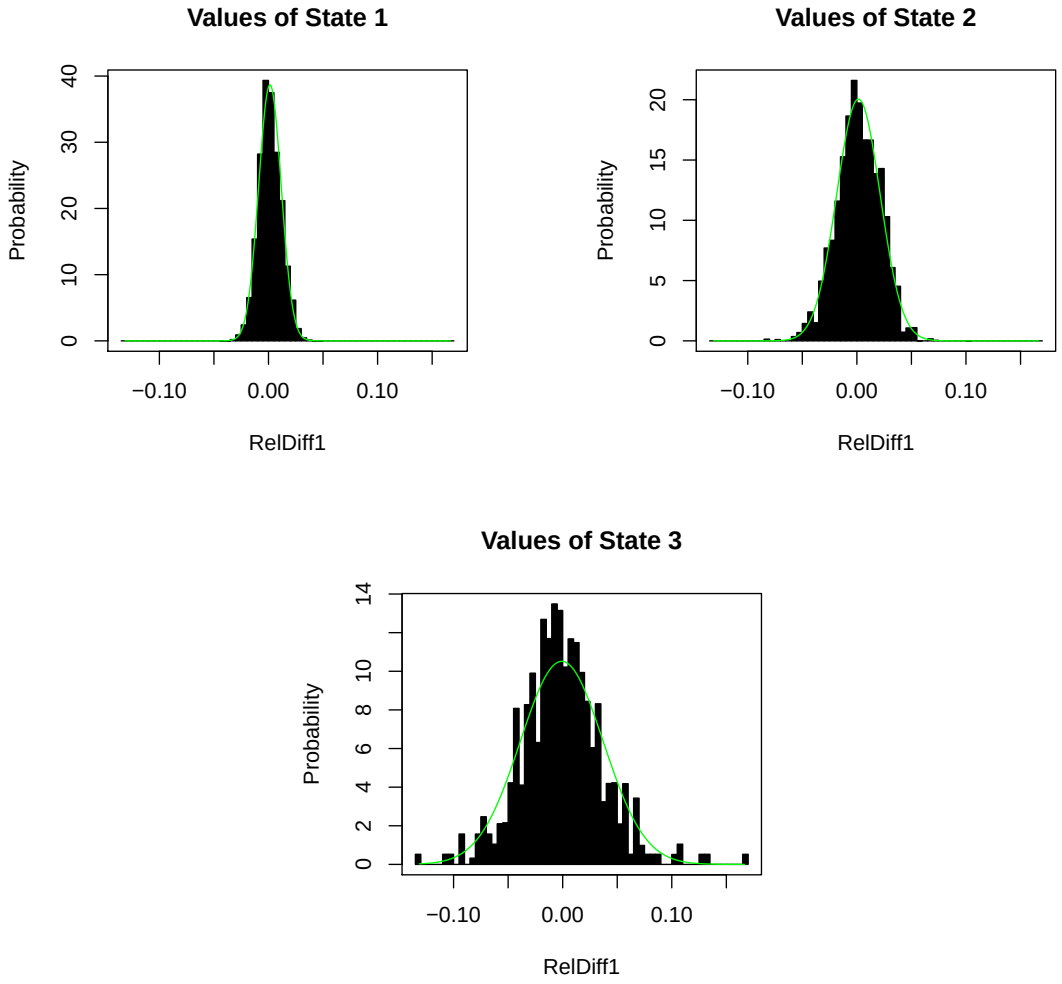



Figure 5.8: Figure displays Google values distributions for the individual states of the "Simple" *HMM*. The green line is a Gaussian approximation of that data used by the state.

All the results were mostly similar and in most cases the Gaussian distribution seemed to be an acceptable approximation (figure: 5.8). However, we have noticed that especially the models with a lower number of states did something different that what was expected. Instead of creating states according to the increasing and decreasing trends they converged to distributions with very similar mean and different deviation. Therefore a good direction advise cannot be expected using these models.

The cause of this behavior might be a random initialization process, which just assigns the data among the states using random weights from a uniform distribution. The result of this approach is in most cases a group of very similar distributions. Perhaps adjusting the random initialization process would lead to better results.

# Chapter 6

# Future Work

> Never put off till tomorrow, what you can do the day after tomorrow.
>
> Mark Twain

Despite an enormous effort, there are many unanswered questions left and many ideas waiting to be implemented and tested. The following part of the text will revise the most significant ones.

- Try out different time scales. We can only guess if machines will ever outsmart people, but they are definitely capable of fast and precise decisions. We are not bound to days and many computer based systems trade in fractions of a second. There has to be a reason for this.
- Improve the algorithm categorizing text articles. We believe that a better initial classification will lead to a more consistent model and accurate results. This point is partially connected with the previous one as data with a higher frequency are more likely to correctly reveal the impact of an article.
- We would also like to connect more models together. Generally, the most successful algorithms are created by combining a bunch of smaller ones [Netflix, 2011].
- Another goal is to closely interconnect the text analysis with the technical analysis. This could be achieved by combining the outputs, but sometimes providing an algorithm with additional input data improves the performance even more.
- Also, the trading evaluation will have to be improved. The used trading simulation system is great for initial estimations, yet the real result will be affected by the transaction price and slippage.
- An automatized parameter tuning solution will be also very handy. Designing and running all the tests by hand takes a lot of time, which could be invested in a much better way.
- The scalability has to be improved in the future as well. It is known that the amount of data can have considerably larger impact on performance than the model itself. For example, increasing the amount of learning data 100 times made a mediocre algorithm for word meaning classification outperform the best known algorithm with the original smaller dataset [Russell and Norvig, 2003].

The ideas mentioned above are definitely not a complete list of possibilities. It is very likely that everyone would like to expand the work in a different direction, as the possibilities are almost unbounded. And the hardest part is that in most cases answering one question reveals a couple of previously unthought-of ones.

# Chapter 7

# Conclusion

> It is human nature to think wisely and act in an absurd fashion.
>
> Anatole France

This thesis first provided a theoretical background for the topic of financial markets as a whole, together with a selective overview of existing research into various approaches. The area is under heavy development and the machine learning techniques used in different domains are being quickly adapted.

The rest of the work was dedicated to the design, implementation and testing of our own machine learning systems for prediction of market movements.

The design process has been successfully completed using the knowledge obtained during the research. Being new to the field we have mostly adhered to the traditional and well-documented approaches to obtain a reference point for our future research.

The implementation part has been completed using the *R* programming language and its wide collection of publicly available libraries. Nevertheless, some of the libraries were modified to fit our specific needs. Certain areas had to be implemented by us entirely and incorporated into the existing framework. This includes: custom input weighting, stationary and higher order models support, state transitions smoothing, text weighting and others. This has given us even deeper insight into functioning and behaviour of the individual methods.

The testing phase involved a implementation of a testing framework combined with several libraries. Testing confirmed the indispensableness of the validation data, which turned initial great results into ordinary ones. Such results cannot be considered on par with those from the state of the art solutions. However, all known comparable endeavours either used intraday or deeply historical data, which can have a significant positive influence on the results.

Finally, we would like to point out that the process of making this thesis was valuable in many directions. We have learned that a seemingly straightforward task such as designing, implementing and testing of two models can be extended into unanticipated size by incompatible libraries, insufficient documentation, unfamiliar programming languages and so forth.

However, we did achieve what we had set out for this thesis and we now have a certain level of knowledge of the area, several working models and an objective evaluation of their results.

We hope to take advantage of the obtained knowledge in the future and further improve the designed system. But it is clear that no further improvement can be achieved without dutifully repeating the "think, create, test, observe" cycle.

# Bibliography


L. Badea. Learning trading rules with inductive logic programming. In *Proceedings of the 11th European Conference on Machine Learning*, ECML '00, pages 39–46, London, UK, 2000. Springer-Verlag. ISBN 3-540-67602-3.

N. Becker, W. Werft, and A. Benner. *penalizedSVM: Feature Selection SVM using penalty functions*, 2010. URL <http://CRAN.R-project.org/package=penalizedSVM>. R package version 1.1.

M. Bigeco, E. Grosso, and E. Otranto. Recognizing and forecasting the sign of financial local trends using hidden markov models. Working Paper CRENoS 200803, Centre for North South Economic Research, University of Cagliari and Sassari, Sardinia, 03 2008.

J. Bollen, H. Mao, and X.-J. Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

P. Brachet. Textmaker, 2011. URL <http://www.xm1math.net/texmaker/>.

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, New York, 1996.

J. Bulla and I. Bulla. Stylized facts of financial time series and hidden semi-markov models. *Comput. Stat. Data Anal.*, 51:2192–2209, December 2006. ISSN 0167-9473. doi: 10.1016/j.csda.2006.07.021.

CRAN. The comprehensive r archive network, 2011. URL <http://http://cran.r-project.org/>.

J. Derrick. A golden cross for the s&p 500 index, 2010. URL <http://www.gold-speculator.com/us-global-investors/40936-golden-cross-s-p-500-index.html>.

J. G. Dias, J. K. Vermunt, and S. Ramos. *Mixture Hidden Markov Models in Finance Research*, pages 451–+. 2010. doi: 10.1007/978-3-642-01044-6_41.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, , and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011. URL <http://CRAN.R-project.org/package=e1071>. R package version 1.5-25.

A. Elder. *Trading for a Living: Psychology Trading Tactics Money Management*. John Wiley & Sons Inc., 1992. ISBN 0-471-59224-2.

I. Feinerer. *tm: Text Mining Package*, 2011. URL <http://www.jstatsoft.org/v25/i05/>. R package version 0.5-6.

I. Feinerer and K. Hornik. *openNLP: openNLP Interface*, 2010. URL <http://CRAN.R-project.org/package=openNLP>. R package version 0.0-8.

A. G. Ferrer and M. B. Brun. Forecasting OECD industrial turning points using unobserved components models with business survey data. *International Journal of Forecasting*, 16 (2):207–227, 2000.

G. P. C. Fung, J. X. Yu, and H. Lu. The predicting power of textual information on financial markets, 2005.

J. Geweke and G. Amisano. Hierarchical markov normal mixture models with applications to financial asset returns. *Journal of Applied Econometrics*, 26(1):1–29, 2011. ISSN 1099-1255.

Google. Google finance, 2011. URL <http://www.google.com/finance>.

L. Grafnetr. An environment for testing financial series predictors and learning such predictors with artificial neural networks. Master's thesis, Czech Technical University, 2011.

M. Hahsler and M. H. Dunham. remm: Extensible markov model for data stream clustering in R. *Journal of Statistical Software*, 35(5):1–31, 2010. URL <http://www.jstatsoft.org/v35/i05/>.

D. Harte. *HiddenMarkov: Hidden Markov Models*. Statistics Research Associates, Wellington, 2011. URL <http://cran.at.r-project.org/web/packages/HiddenMarkov>. R package version 1.4-4.

M. R. Hassan, B. Nath, and M. Kirley. A fusion model of hmm, ann and ga for stock market forecasting. *Expert Syst. Appl.*, 33:171–180, July 2007. ISSN 0957-4174. doi: 10.1016/j.eswa.2006.04.007.

T. Hastie. *svmpath: svmpath: the SVM Path algorithm*, 2009. URL <http://CRAN.R-project.org/package=svmpath>. R package version 0.93.

D. L. Himmelmann. *HMM: HMM - Hidden Markov Models*, 2010. URL <http://CRAN.R-project.org/package=HMM>. R package version 1.0 (Scientific Software Development www.linhi.com).

iHackerNews.com. iHackerNews.com api, 2011. URL <http://api.ihackernews.com/>.

C. H. Jackson. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 38(8):1–29, 2011. URL <http://www.jstatsoft.org/v38/i08/>.

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL <http://www.jstatsoft.org/v11/i09/>.

A. Karatzoglou, D. Meyer, and K. Hornik. Support vector machines in r. *Journal of Statistical Software*, 15(9):1–28, 4 2006. ISSN 1548-7660. URL <http://www.jstatsoft.org/v15/i09>.

B. Kovalerchuk and E. Vityaev. *Data mining in finance: advances in relational and hybrid methods*. Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN 0-7923-7804-0.

C. Langager and C. Murphy. Analyzing chart patterns: Head and shoulders, 2010. URL <http://www.investopedia.com/university/charts/charts2.asp>.

V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Language models for financial news recommendation. In *In Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 389–396. ACM Press, 2000.

MATLAB. *version 7.8.0 (R2009a)*. The MathWorks Inc., Natick, Massachusetts, 2009.

I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM. ISBN 1-59593-339-5. doi: http://doi.acm.org/10.1145/1150402.1150531. URL <http://rapid-i.com/component/option,com_docman/task,doc_download/gid,25/Itemid,62/>.

mindtouch. SqmlReader - C# library, 2010. URL <http://developer.mindtouch.com/en/docs/SgmlReader>.

M.-A. Mittermayer and G. F. Knolmayer. Newscats: A news categorization and trading system. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 1002–1007, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2701-9. doi: http://dx.doi.org/10.1109/ICDM.2006.115. URL <http://portal.acm.org/citation.cfm?id=1193291>.

M.-A. M. Mittermayer and G. F. Knolmayer. Text mining systems for predicting market response to news. *Proceedings of the IADIS European Conference Data Mining 2007*, 2007. URL <hhttp://www2.ie.iwi.unibe.ch/publikationen/berichte/resource/WP-184.pdf>.

L. Moss. Example of the baum-welch algorithm, 2008. URL <http://www.indiana.edu/~iulg/moss/hmmcalculations.pdf>.

S. Mourier, J. Klawiter, and Jesse. Html Agility Pack - C# library, 2011. URL <http://htmlagilitypack.codeplex.com/>.

Netflix. Netflix prize, 2011. URL <http://www.netflixprize.com/>.

J. Newton-King. Json.NET - C# library, 2011. URL <http://json.codeplex.com/>.

NY Times. NY Times api, 2011. URL <http://developer.nytimes.com/docs/article_search_api>.

T. Oates, L. Firoiu, and P. Cohen. Using dynamic time warping to bootstrap hmm-based clustering of time series. In R. Sun and C. Giles, editors, *Sequence Learning*, volume 1828 of *Lecture Notes in Computer Science*, pages 35–52. Springer Berlin Heidelberg, 2001.

W. Palma. *Long-Memory Time Series: Theory and Methods*. Wiley, 2006.

C. P. Papageorgiou. High frequency time series analysis and prediction using markov models. In *in Proceedings of the conference on Computational Intelligence for Financial Engineering*, pages 182–185, 1997.

S.-H. Park, J.-H. Lee, J.-W. Song, and T.-S. Park. Forecasting change directions for financial time series using hidden markov model. In *Proceedings of the 4th International Conference on Rough Sets and Knowledge Technology*, RSKT '09, pages 184–191, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02961-5.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL <http://www.R-project.org/>. ISBN 3-900051-07-0.

S. Rao and J. Hong. Analysis of hidden markov models and support vector machines in financial applications. Master's thesis, EECS Department, University of California, Berkeley, May 2010.

Reuters. Reuters Spotlight, 2011. URL <http://spotlight.reuters.com>.

RStudio. RStudio 0.94, 2011. URL <http://rstudio.org/>.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. ISBN 0137903952. URL <http://portal.acm.org/citation.cfm?id=773294>.

F. Salmon and J. Stokes. Algorithms take control of wall street, 2010. URL <http://www.wired.com/magazine/2010/12/ff_ai_flashtrading/all/1>.

R. P. Schumaker and H. Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27:12:1–12:19, March 2009. ISSN 1046-8188. doi: http://doi.acm.org/10.1145/1462198.1462204. URL <http://doi.acm.org/10.1145/1462198.1462204>.

N. Times. New york times: The article search api documentation, 2011. URL <http://developer.nytimes.com/docs/read/article_search_api>.

UMLet. UMLet 11.3, 2011. URL <http://www.umlet.com/>.

USA Today. USA Today developers guide, 2011. URL <http://developer.usatoday.com/docs/read/articles>.

I. Visser and M. Speekenbrink. depmixs4: An r package for hidden markov models. *Journal of Statistical Software*, 36(7):1–21, 8 2010. ISSN 1548-7660. URL <http://www.jstatsoft.org/v36/i07>.

C. Weihs, U. Ligges, K. Luebke, and N. Raabe. klar analyzing german business cycles. In D. Baier, R. Decker, and L. Schmidt-Thieme, editors, *Data Analysis and Decision Support*, pages 335–343, Berlin, 2005. Springer-Verlag.

Wikinews. Wikinews, 2011. URL <http://en.wikinews.org/>.

Yahoo! Yahoo! finance, 2011. URL <http://finance.yahoo.com/>.

S. Zemke. Data mining for prediction. financial series case, doctoral thesis, the royal, 2003.

D. Zhang and L. Zhou. Discovering golden nuggets: data mining in financial application. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(4):513–522, 2004. doi: 10.1109/TSMCC.2004.829279. URL <http://dx.doi.org/10.1109/TSMCC.2004.829279>.

W. Zucchini and I. L. MacDonald. *Hidden Markov Models for Time Series - An Introduction Using R*. Chapman & Hall /CRC, 2009. ISBN 978-1-58488-573-3.

# Appendix A

# List of Abbreviations

**ANN** artificial neural network (the same meaning as *NN*)

**API** application programming interface

**CRAN** comprehensive R archive network

**DNF** disjunctive normal form

**DJIA** Dow Jones Industrial Average

**GA** genetic algorithm

**GUI** graphical user interface

**HMM** hidden Markov model

**HTML** hypertext markup language

**HMNM** hierarchical Markov normal model

**HSMM** hidden semi-Markov model

**IDF** inverse document frequency

**ILP** inductive logic programming

**MAPE** mean absolute percentage error

**MM** Markov model

**MNM** Markov normal model

**MHMM** mixture hidden Markov model

**MSE** mean square error

**NN** neural network

**RBF** radial basis function

**SOFNN** self-organizing fuzzy neural network

**SVM** support vector machine

**TF** term frequency

**UI** user interface

**URI** uniform resource identifier

**USA** United States of America

**WA** weighted average

**XML** extensible markup language

# Appendix B

# NY Downloader User Guide

## B.1 Installation

The application does no require installation at all. It can be directly run from arbitrary directory supposing that the current user has reading permission there. The only requirement is the presence of the *.NET 4* framework runtime.

## B.2 Usage

As can be seen the user interface B.1 is simple. The only thing the user has to actually fill in is the personal *API Key* which can be obtained from *NY Times* website [NY Times, 2011] free of charge. Without it the application won't work correctly.

### B.2.1 Basic Download

To obtain a news articles just select a date range, target file and hit the download button. For advanced usage visit the *NY Times* article search *API* website [Times, 2011] and discover the supported range queries.

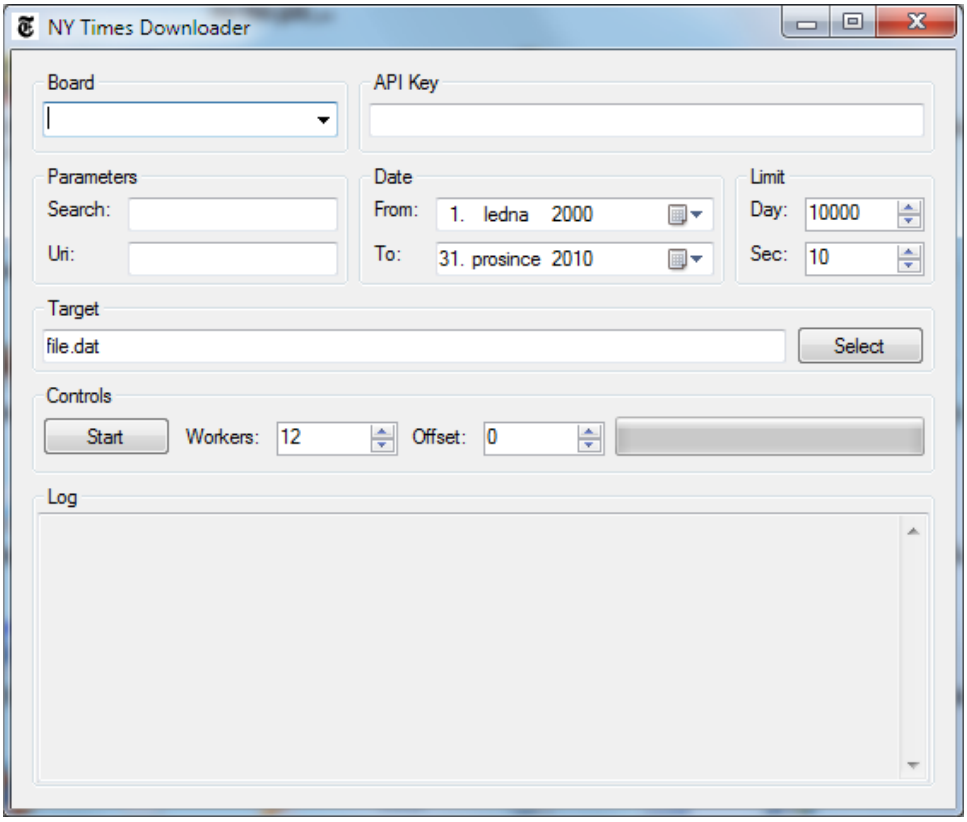



Figure B.1: Screenshot of the *NY Downloader's* user interface after startup.

# Appendix C

# Validation Framework User Guide

## C.1 Installation

To install the framework follow these instructions:

1. Download and install the *R*. [R Development Core Team, 2011]
2. Copy a content of the "/src/R/" directory to a writeable location.
3. Make the target location your *R* working directory.
4. Run "install.R" script to install required libraries.
5. Edit the paths in "init.R" file to reflect the data locations.
6. Run "init.R" script to load the framework.
7. The framework is up and ready.

## C.2 Usage

To see how it works just select a file from "/src/R/chunks/" subdirectory and have a fun.

You can of course make use of the framework functions and make your own validation script. The already present ones might serve as a good starting point.

# Appendix D

# Additional Plots and Tables

| Company Name | Number of Articles |
|---|---|
| GOOGLE INC | 1782 |
| MICROSOFT CORP | 1426 |
| FEDERAL RESERVE SYSTEM | 1092 |
| EUROPEAN UNION | 1054 |
| SECURITIES AND EXCHANGE COMMISSION | 1026 |
| APPLE INC | 904 |
| GENERAL MOTORS CORP | 694 |
| YAHOO INC | 659 |
| YOUTUBE.COM | 474 |
| SONY CORP | 444 |
| FORD MOTOR CO | 443 |
| TREASURY DEPARTMENT | 436 |
| HEWLETT-PACKARD CO | 434 |
| FACEBOOK.COM | 422 |
| FOOD AND DRUG ADMINISTRATION | 398 |
| CITIGROUP INC | 393 |
| JUSTICE DEPARTMENT | 390 |
| WAL-MART STORES INC | 372 |
| TOYOTA MOTOR CORP | 358 |
| INTEL CORP | 355 |
| VERIZON COMMUNICATIONS | 336 |
| MYSPACE.COM | 327 |
| YOUTUBE | 316 |
| GOLDMAN SACHS GROUP | 315 |
| NEW YORK TIMES | 310 |
| DISNEY, WALT, CO | 309 |
| AMERICAN INTERNATIONAL GROUP | 307 |
| AT&T CORP | 305 |
| NBC UNIVERSAL | 305 |
| AMAZON.COM INC | 301 |
| HOUSE OF REPRESENTATIVES | 295 |
| COMMERCE DEPARTMENT | 294 |
| CHRYSLER LLC | 294 |
| CBS CORP | 292 |
| SENATE | 289 |
| BOEING CO | 282 |
| LABOR DEPARTMENT | 282 |
| FEDERAL COMMUNICATIONS COMMISSION | 281 |
| APPLE COMPUTER INC | 280 |
| DELL INC | 273 |
| MICROSOFT CORPORATION | 270 |
| TIME WARNER INC | 266 |
| EUROPEAN COMMISSION | 258 |
| NEWS CORP | 257 |
| NATIONAL BROADCASTING CO | 250 |
| MORGAN STANLEY | 242 |
| BANK OF AMERICA CORP | 241 |
| AIRBUS INDUSTRIE | 240 |
| INTERNATIONAL BUSINESS MACHINES CORP | 234 |
| UBS AG | 230 |

Table D.1: Number of articles related to given subject from year 2000 to 2010.

| Category | Unknown | Known | Pos | None -1 | None | None +1 | Neg |
|---|---|---|---|---|---|---|---|
| MSFT | 65 | 2285 | 0.31 | 0.40 | 0.40 | 0.40 | 0.29 |
| YHOO | 34 | 1220 | 0.32 | 0.31 | 0.33 | 0.32 | 0.35 |
| GOOG | 136 | 1196 | 0.28 | 0.44 | 0.44 | 0.47 | 0.28 |
| AAPL | 23 | 572 | 0.35 | 0.35 | 0.35 | 0.36 | 0.29 |
| AMZN | 14 | 451 | 0.40 | 0.26 | 0.23 | 0.26 | 0.37 |
| EBAY | 2 | 381 | 0.33 | 0.30 | 0.30 | 0.29 | 0.37 |
| ERTS | 6 | 274 | 0.31 | 0.29 | 0.31 | 0.35 | 0.38 |
| AOL | 94 | 25 | 0.28 | 0.24 | 0.40 | 0.36 | 0.32 |
| Business | 304 | 5155 | 0.32 | 0.37 | 0.37 | 0.38 | 0.31 |
| Technology | 290 | 4833 | 0.32 | 0.36 | 0.37 | 0.38 | 0.31 |
| Books | 7 | 248 | 0.36 | 0.34 | 0.33 | 0.37 | 0.31 |
| Opinion | 13 | 234 | 0.31 | 0.33 | 0.34 | 0.33 | 0.35 |
| Front Page | 15 | 227 | 0.26 | 0.39 | 0.38 | 0.40 | 0.36 |
| Arts | 2 | 207 | 0.38 | 0.34 | 0.34 | 0.41 | 0.28 |
| U.S. | 5 | 144 | 0.31 | 0.41 | 0.44 | 0.40 | 0.25 |
| New York and Region | 10 | 134 | 0.36 | 0.26 | 0.27 | 0.26 | 0.37 |
| Movies | 6 | 103 | 0.22 | 0.43 | 0.47 | 0.47 | 0.31 |
| Corrections | 11 | 87 | 0.36 | 0.23 | 0.25 | 0.24 | 0.39 |
| Health | 3 | 70 | 0.27 | 0.35 | 0.39 | 0.28 | 0.34 |
| Education | 2 | 60 | 0.23 | 0.36 | 0.42 | 0.36 | 0.35 |
| Magazine | 1 | 58 | 0.36 | 0.36 | 0.41 | 0.36 | 0.22 |
| World | 4 | 54 | 0.28 | 0.40 | 0.43 | 0.46 | 0.30 |
| Week in Review | 1 | 43 | 0.40 | 0.39 | 0.23 | 0.30 | 0.37 |
| Washington | 3 | 41 | 0.32 | 0.48 | 0.46 | 0.49 | 0.22 |
| Science | 1 | 38 | 0.34 | 0.41 | 0.55 | 0.41 | 0.11 |
| Style | 5 | 29 | 0.28 | 0.41 | 0.45 | 0.41 | 0.28 |
| Unknown | 1 | 29 | 0.17 | 0.40 | 0.52 | 0.37 | 0.31 |
| Sports | 2 | 25 | 0.28 | 0.33 | 0.36 | 0.44 | 0.36 |
| Travel | 0 | 20 | 0.00 | 0.25 | 0.35 | 0.25 | 0.65 |
| Theater | 0 | 7 | 0.29 | 0.29 | 0.43 | 0.29 | 0.29 |
| Obituaries | 0 | 5 | 0.00 | 0.80 | 0.80 | 0.80 | 0.20 |
| Home and Garden | 1 | 4 | 0.25 | 0.40 | 0.75 | 0.60 | 0.00 |
| Real Estate | 0 | 3 | 0.00 | 0.33 | 1.00 | 0.67 | 0.00 |
| Dining and Wine | 0 | 2 | 0.00 | 1.00 | 0.50 | 0.00 | 0.50 |
| Automobiles | 0 | 2 | 0.50 | 0.50 | 0.50 | 0.50 | 0.00 |
| Editors' Notes | 1 | 2 | 0.50 | 0.50 | 0.50 | 0.50 | 0.00 |
| Paid Death Notices | 0 | 2 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Job Market | 0 | 1 | 0.00 | 0.00 | 0.00 | −− | 1.00 |

Table D.2: The first two numerical columns are showing the number of Known and Unknown (unclassified) articles. The rest of the table shows ratio of Positive/Neutral/Negative articles in each group. The columns "None -1" and "None +1" are Neutral ratios for previous or consequent day. The upper part of the table contains organizations while the lower holds article categories.

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.095 | −0.095 | −0.095 | −0.095 | −0.095 | −0.150 | 0.000 | 0.000 | 0.000 |
| 2 | 0.588 | 0.584 | 0.598 | 0.582 | 0.573 | 0.574 | 0.000 | 0.000 | 0.000 |
| 3 | 0.635 | 0.649 | 0.659 | 0.666 | 0.623 | 0.548 | 0.089 | 0.000 | 0.000 |
| 4 | 0.546 | 0.543 | 0.545 | 0.529 | 0.488 | 0.448 | 0.074 | 0.000 | 0.000 |
| 5 | 0.419 | 0.421 | 0.399 | 0.400 | 0.514 | 0.241 | 0.081 | 0.000 | 0.000 |
| 6 | 0.482 | 0.470 | 0.467 | 0.491 | 0.537 | 0.468 | 0.078 | 0.000 | 0.000 |
| 7 | 0.472 | 0.473 | 0.462 | 0.461 | 0.457 | 0.372 | 0.090 | 0.000 | 0.000 |
| 8 | 0.454 | 0.467 | 0.460 | 0.490 | 0.417 | 0.370 | 0.051 | 0.000 | 0.000 |
| 9 | 0.462 | 0.458 | 0.462 | 0.465 | 0.427 | 0.302 | 0.050 | 0.000 | 0.000 |
| 10 | 0.454 | 0.453 | 0.447 | 0.453 | 0.387 | 0.489 | 0.056 | 0.000 | 0.000 |
| 11 | 0.409 | 0.402 | 0.418 | 0.417 | 0.394 | 0.287 | 0.028 | 0.000 | 0.000 |

(a) Relative Difference

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.349 | −0.349 | −0.277 | −0.429 | −0.150 | 0.000 | 0.000 | 0.000 | 0.000 |
| 2 | 0.625 | 0.633 | 0.642 | 0.655 | 0.554 | 0.350 | 0.000 | 0.000 | 0.000 |
| 3 | 0.591 | 0.598 | 0.582 | 0.606 | 0.589 | 0.538 | 0.000 | 0.000 | 0.000 |
| 4 | 0.512 | 0.514 | 0.537 | 0.533 | 0.560 | 0.513 | 0.000 | 0.000 | 0.000 |
| 5 | 0.530 | 0.536 | 0.555 | 0.541 | 0.516 | 0.331 | 0.000 | 0.000 | 0.000 |
| 6 | 0.463 | 0.448 | 0.461 | 0.492 | 0.440 | 0.262 | 0.000 | 0.000 | 0.000 |
| 7 | 0.476 | 0.464 | 0.459 | 0.448 | 0.476 | 0.318 | 0.000 | 0.000 | 0.000 |
| 8 | 0.597 | 0.598 | 0.574 | 0.541 | 0.478 | 0.564 | 0.000 | 0.000 | 0.000 |
| 9 | 0.490 | 0.484 | 0.476 | 0.453 | 0.371 | 0.434 | −0.013 | 0.000 | 0.000 |
| 10 | 0.424 | 0.427 | 0.436 | 0.444 | 0.333 | 0.350 | 0.000 | 0.000 | 0.000 |
| 11 | 0.548 | 0.537 | 0.534 | 0.530 | 0.479 | 0.378 | −0.009 | 0.000 | 0.000 |

(b) Logarithmic Difference

Table D.3: The table contains average daily profit for confidence based strategy for model learnt on all the series together. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.095 | −0.095 | −0.095 | −0.095 | −0.095 | −0.150 | 0.000 | 0.000 | 0.000 |
| 2 | 0.584 | 0.572 | 0.590 | 0.589 | 0.504 | 0.264 | 0.109 | 0.000 | 0.000 |
| 3 | 0.680 | 0.680 | 0.668 | 0.635 | 0.630 | 0.313 | 0.156 | 0.000 | 0.000 |
| 4 | 0.509 | 0.524 | 0.521 | 0.553 | 0.572 | 0.209 | 0.191 | 0.000 | 0.000 |
| 5 | 0.603 | 0.597 | 0.575 | 0.528 | 0.508 | 0.230 | 0.152 | 0.000 | 0.000 |
| 6 | 0.531 | 0.533 | 0.523 | 0.511 | 0.546 | 0.196 | 0.192 | 0.000 | 0.000 |
| 7 | 0.441 | 0.452 | 0.454 | 0.455 | 0.492 | 0.180 | 0.211 | 0.000 | 0.000 |
| 8 | 0.489 | 0.484 | 0.494 | 0.466 | 0.494 | 0.182 | 0.251 | 0.000 | 0.000 |
| 9 | 0.544 | 0.549 | 0.543 | 0.529 | 0.477 | 0.183 | 0.255 | 0.000 | 0.000 |
| 10 | 0.474 | 0.470 | 0.466 | 0.450 | 0.429 | 0.200 | 0.243 | −0.003 | 0.000 |
| 11 | 0.521 | 0.516 | 0.533 | 0.520 | 0.410 | 0.161 | 0.200 | 0.000 | 0.000 |

(a) Relative Difference

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.349 | −0.349 | −0.277 | −0.429 | −0.150 | 0.000 | 0.000 | 0.000 | 0.000 |
| 2 | 0.579 | 0.571 | 0.582 | 0.626 | 0.570 | 0.126 | 0.000 | 0.000 | 0.000 |
| 3 | 0.622 | 0.622 | 0.632 | 0.668 | 0.622 | 0.420 | −0.005 | 0.000 | 0.000 |
| 4 | 0.506 | 0.495 | 0.498 | 0.515 | 0.533 | 0.327 | −0.010 | 0.000 | 0.000 |
| 5 | 0.511 | 0.513 | 0.527 | 0.527 | 0.541 | 0.460 | −0.012 | 0.000 | 0.000 |
| 6 | 0.486 | 0.482 | 0.471 | 0.476 | 0.501 | 0.356 | −0.010 | 0.000 | 0.000 |
| 7 | 0.521 | 0.522 | 0.514 | 0.490 | 0.494 | 0.358 | 0.001 | 0.000 | 0.000 |
| 8 | 0.580 | 0.576 | 0.572 | 0.565 | 0.573 | 0.394 | −0.004 | 0.000 | 0.000 |
| 9 | 0.451 | 0.459 | 0.456 | 0.458 | 0.498 | 0.317 | −0.001 | 0.000 | 0.000 |
| 10 | 0.442 | 0.436 | 0.436 | 0.426 | 0.483 | 0.313 | −0.002 | 0.000 | 0.000 |
| 11 | 0.615 | 0.632 | 0.637 | 0.632 | 0.583 | 0.316 | −0.001 | 0.000 | 0.000 |

(b) Logarithmic Difference

Table D.4: The table contains average daily profit for threshold based strategy for model learnt on all the series together. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.512 | −0.507 | −0.507 | −0.445 | −0.313 | −0.117 | −0.014 | 0.000 | 0.000 |
| 2 | −0.293 | −0.294 | −0.283 | −0.228 | −0.147 | −0.075 | 0.020 | 0.000 | 0.000 |
| 3 | 0.103 | 0.074 | 0.035 | 0.057 | 0.116 | 0.072 | 0.016 | −0.004 | 0.000 |
| 4 | 0.176 | 0.172 | 0.196 | 0.220 | 0.202 | 0.179 | −0.051 | −0.002 | −0.004 |
| 5 | 0.358 | 0.387 | 0.392 | 0.363 | 0.284 | 0.173 | −0.042 | −0.005 | −0.004 |
| 6 | 0.161 | 0.149 | 0.154 | 0.186 | 0.144 | 0.108 | −0.018 | −0.004 | 0.000 |
| 7 | 0.093 | 0.074 | 0.104 | 0.119 | 0.146 | 0.137 | −0.035 | −0.007 | −0.004 |
| 8 | 0.197 | 0.208 | 0.174 | 0.219 | 0.175 | 0.168 | −0.091 | 0.001 | −0.004 |
| 9 | 0.055 | 0.068 | 0.124 | 0.200 | 0.156 | 0.180 | −0.010 | 0.003 | −0.004 |
| 10 | −0.041 | −0.031 | −0.041 | 0.058 | 0.102 | 0.201 | −0.100 | −0.003 | −0.004 |
| 11 | 0.113 | 0.107 | 0.083 | 0.109 | 0.068 | 0.170 | −0.033 | 0.007 | −0.010 |

(a) 4 Chunks in a Window

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.170 | 0.170 | 0.172 | 0.172 | 0.164 | −0.063 | 0.011 | 0.000 | 0.000 |
| 2 | 0.310 | 0.319 | 0.340 | 0.288 | 0.343 | 0.261 | 0.251 | 0.004 | 0.000 |
| 3 | 0.094 | 0.093 | 0.092 | 0.062 | 0.135 | 0.303 | 0.281 | 0.008 | 0.006 |
| 4 | −0.121 | −0.094 | −0.088 | −0.061 | 0.062 | 0.203 | 0.265 | 0.004 | 0.000 |
| 5 | −0.083 | −0.078 | −0.078 | −0.026 | 0.100 | 0.141 | 0.217 | 0.008 | 0.002 |
| 6 | 0.033 | 0.048 | 0.013 | 0.024 | 0.143 | 0.287 | 0.236 | 0.006 | −0.005 |
| 7 | 0.043 | 0.033 | 0.043 | 0.037 | 0.090 | 0.281 | 0.286 | −0.007 | −0.009 |
| 8 | 0.180 | 0.170 | 0.187 | 0.218 | 0.259 | 0.415 | 0.292 | 0.000 | 0.004 |
| 9 | 0.258 | 0.237 | 0.241 | 0.291 | 0.371 | 0.411 | 0.232 | 0.005 | 0.008 |
| 10 | 0.207 | 0.192 | 0.194 | 0.227 | 0.303 | 0.328 | 0.363 | 0.009 | 0.004 |
| 11 | 0.300 | 0.291 | 0.275 | 0.308 | 0.362 | 0.400 | 0.390 | 0.031 | 0.002 |

(b) 2 Chunks in a Window

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.300 | 0.300 | 0.300 | 0.290 | 0.328 | 0.214 | 0.060 | 0.000 | 0.000 |
| 2 | 0.490 | 0.491 | 0.498 | 0.489 | 0.540 | 0.351 | 0.134 | 0.012 | 0.000 |
| 3 | 0.361 | 0.373 | 0.369 | 0.346 | 0.285 | 0.269 | 0.191 | −0.004 | 0.000 |
| 4 | 0.437 | 0.430 | 0.427 | 0.415 | 0.410 | 0.347 | 0.197 | 0.042 | −0.004 |
| 5 | 0.519 | 0.520 | 0.524 | 0.522 | 0.488 | 0.442 | 0.230 | 0.028 | −0.005 |
| 6 | 0.631 | 0.627 | 0.629 | 0.642 | 0.582 | 0.560 | 0.330 | 0.035 | −0.021 |
| 7 | 0.635 | 0.636 | 0.628 | 0.649 | 0.604 | 0.545 | 0.272 | 0.029 | −0.011 |
| 8 | 0.555 | 0.571 | 0.570 | 0.585 | 0.577 | 0.607 | 0.320 | −0.028 | −0.031 |
| 9 | 0.596 | 0.613 | 0.594 | 0.568 | 0.520 | 0.507 | 0.308 | 0.052 | −0.005 |
| 10 | 0.570 | 0.577 | 0.572 | 0.592 | 0.590 | 0.577 | 0.211 | 0.095 | 0.010 |
| 11 | 0.717 | 0.714 | 0.702 | 0.701 | 0.666 | 0.553 | 0.336 | 0.047 | −0.001 |

(c) 1 Chunk in a Window

Table D.5: The table contains average daily profit of Relative Difference with confidence based strategy for different windows sizes. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.654 | −0.654 | −0.502 | −0.502 | −0.502 | −0.150 | 0.000 | 0.000 | 0.000 |
| 2 | 0.129 | 0.122 | 0.148 | 0.158 | 0.098 | −0.188 | 0.002 | 0.000 | 0.000 |
| 3 | 0.317 | 0.299 | 0.274 | 0.318 | 0.237 | 0.009 | 0.000 | 0.000 | 0.000 |
| 4 | 0.279 | 0.277 | 0.284 | 0.338 | 0.323 | 0.105 | −0.003 | 0.000 | 0.000 |
| 5 | 0.153 | 0.153 | 0.151 | 0.161 | 0.194 | 0.052 | 0.000 | 0.000 | 0.000 |
| 6 | 0.242 | 0.243 | 0.233 | 0.232 | 0.227 | 0.123 | 0.007 | 0.000 | 0.000 |
| 7 | 0.169 | 0.172 | 0.168 | 0.174 | 0.157 | 0.031 | 0.010 | 0.000 | 0.000 |
| 8 | 0.192 | 0.188 | 0.184 | 0.171 | 0.204 | 0.095 | −0.010 | 0.000 | 0.000 |
| 9 | 0.275 | 0.266 | 0.265 | 0.277 | 0.276 | 0.222 | 0.007 | −0.001 | 0.000 |
| 10 | 0.158 | 0.160 | 0.159 | 0.169 | 0.175 | 0.115 | −0.005 | −0.003 | 0.000 |
| 11 | 0.138 | 0.137 | 0.135 | 0.147 | 0.125 | 0.020 | 0.009 | 0.000 | 0.000 |

(a) 4 Chunks in a Window

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.210 | −0.210 | −0.210 | −0.210 | −0.079 | 0.072 | 0.000 | 0.000 | 0.000 |
| 2 | 0.062 | 0.068 | 0.076 | 0.092 | 0.133 | 0.013 | 0.000 | 0.000 | 0.000 |
| 3 | 0.338 | 0.314 | 0.328 | 0.343 | 0.285 | 0.089 | 0.001 | 0.000 | 0.000 |
| 4 | 0.162 | 0.159 | 0.142 | 0.137 | 0.118 | 0.093 | 0.007 | 0.000 | 0.000 |
| 5 | 0.115 | 0.120 | 0.126 | 0.153 | 0.163 | 0.028 | −0.001 | 0.000 | 0.000 |
| 6 | 0.125 | 0.104 | 0.083 | 0.197 | 0.160 | 0.005 | 0.004 | 0.000 | 0.000 |
| 7 | 0.153 | 0.154 | 0.155 | 0.156 | 0.228 | 0.025 | 0.002 | 0.000 | 0.000 |
| 8 | 0.157 | 0.166 | 0.163 | 0.144 | 0.144 | 0.027 | −0.003 | 0.000 | 0.000 |
| 9 | 0.199 | 0.190 | 0.186 | 0.171 | 0.291 | 0.009 | −0.007 | 0.000 | 0.000 |
| 10 | 0.145 | 0.144 | 0.144 | 0.161 | 0.205 | 0.013 | −0.002 | 0.000 | 0.000 |
| 11 | 0.344 | 0.349 | 0.340 | 0.289 | 0.461 | 0.029 | −0.020 | 0.000 | 0.000 |

(b) 2 Chunks in a Window

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.210 | −0.210 | 0.071 | 0.071 | 0.071 | −0.003 | 0.000 | 0.000 | 0.000 |
| 2 | 0.390 | 0.412 | 0.409 | 0.469 | 0.411 | 0.341 | 0.001 | 0.000 | 0.000 |
| 3 | 0.540 | 0.538 | 0.534 | 0.549 | 0.477 | 0.322 | 0.000 | 0.000 | 0.000 |
| 4 | 0.508 | 0.507 | 0.508 | 0.534 | 0.571 | 0.319 | −0.011 | 0.000 | 0.000 |
| 5 | 0.553 | 0.540 | 0.530 | 0.524 | 0.541 | 0.303 | −0.010 | 0.000 | 0.000 |
| 6 | 0.457 | 0.452 | 0.491 | 0.544 | 0.613 | 0.321 | −0.007 | 0.000 | 0.000 |
| 7 | 0.497 | 0.516 | 0.513 | 0.546 | 0.630 | 0.321 | −0.010 | 0.000 | 0.000 |
| 8 | 0.500 | 0.518 | 0.523 | 0.530 | 0.539 | 0.323 | −0.011 | 0.000 | 0.000 |
| 9 | 0.507 | 0.507 | 0.518 | 0.496 | 0.574 | 0.325 | −0.008 | 0.000 | 0.000 |
| 10 | 0.572 | 0.581 | 0.559 | 0.514 | 0.587 | 0.324 | −0.014 | 0.000 | 0.000 |
| 11 | 0.576 | 0.588 | 0.603 | 0.571 | 0.596 | 0.258 | 0.010 | 0.000 | 0.000 |

(c) 1 Chunk in a Window

Table D.6: The table contains average daily profit for threshold based strategy for different windows sizes learnt together. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.277 | 0.265 | 0.267 | 0.304 | 0.322 | 0.316 | −0.005 | 0.000 | 0.000 |
| 3 | 0.392 | 0.405 | 0.399 | 0.485 | 0.500 | 0.461 | 0.041 | −0.003 | 0.000 |
| 4 | 0.415 | 0.408 | 0.378 | 0.379 | 0.433 | 0.439 | 0.009 | 0.003 | 0.000 |
| 5 | 0.326 | 0.323 | 0.311 | 0.365 | 0.417 | 0.398 | 0.010 | −0.003 | 0.000 |
| 6 | 0.365 | 0.345 | 0.348 | 0.363 | 0.447 | 0.314 | 0.048 | −0.004 | 0.000 |
| 7 | 0.292 | 0.297 | 0.268 | 0.313 | 0.345 | 0.335 | 0.020 | −0.006 | 0.000 |
| 8 | 0.359 | 0.361 | 0.382 | 0.370 | 0.411 | 0.329 | 0.062 | −0.010 | −0.001 |
| 9 | 0.351 | 0.352 | 0.330 | 0.319 | 0.407 | 0.372 | 0.046 | −0.002 | −0.001 |
| 10 | 0.234 | 0.244 | 0.279 | 0.314 | 0.361 | 0.317 | 0.059 | −0.001 | 0.000 |
| 11 | 0.353 | 0.353 | 0.341 | 0.367 | 0.347 | 0.307 | 0.020 | −0.002 | −0.001 |

Table D.7: The table contains average daily profit of stationary *Relative Difference* based *HMM* with confidence based strategy. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.121 | 0.105 | 0.108 | 0.187 | 0.182 | 0.123 | −0.014 | 0.000 | 0.000 |
| 3 | 0.087 | 0.082 | 0.109 | 0.108 | 0.144 | 0.257 | −0.079 | −0.018 | 0.000 |
| 4 | 0.072 | 0.071 | 0.074 | 0.075 | 0.055 | 0.049 | 0.196 | −0.033 | −0.009 |
| 5 | 0.086 | 0.086 | 0.086 | 0.092 | 0.072 | 0.017 | 0.014 | −0.025 | −0.017 |
| 6 | −0.032 | −0.032 | −0.032 | −0.032 | −0.032 | −0.036 | 0.004 | 0.098 | −0.017 |
| 7 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | 0.009 | 0.195 | 0.004 |
| 8 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | 0.044 | −0.016 |
| 9 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.027 | −0.070 |
| 10 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.031 | 0.092 |
| 11 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | −0.030 | 0.088 |

(a) Smoothing 10

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.294 | 0.302 | 0.302 | 0.307 | 0.378 | 0.340 | 0.016 | 0.000 | 0.000 |
| 3 | 0.495 | 0.451 | 0.469 | 0.453 | 0.417 | 0.382 | 0.010 | 0.000 | 0.000 |
| 4 | 0.392 | 0.361 | 0.313 | 0.337 | 0.328 | 0.370 | −0.001 | 0.007 | 0.000 |
| 5 | 0.402 | 0.401 | 0.424 | 0.424 | 0.335 | 0.285 | 0.032 | −0.021 | 0.000 |
| 6 | 0.291 | 0.307 | 0.288 | 0.240 | 0.266 | 0.201 | 0.064 | −0.014 | 0.000 |
| 7 | 0.219 | 0.211 | 0.220 | 0.249 | 0.240 | 0.318 | 0.026 | −0.018 | 0.000 |
| 8 | 0.269 | 0.265 | 0.281 | 0.287 | 0.359 | 0.291 | 0.036 | −0.009 | 0.000 |
| 9 | 0.392 | 0.402 | 0.422 | 0.424 | 0.356 | 0.269 | −0.009 | 0.016 | −0.028 |
| 10 | 0.308 | 0.312 | 0.312 | 0.340 | 0.328 | 0.205 | 0.126 | −0.017 | −0.028 |
| 11 | 0.326 | 0.323 | 0.345 | 0.314 | 0.281 | 0.212 | 0.354 | −0.019 | −0.017 |

(b) Smoothing 1

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.278 | 0.280 | 0.259 | 0.298 | 0.393 | 0.331 | 0.044 | 0.000 | 0.000 |
| 3 | 0.432 | 0.429 | 0.400 | 0.429 | 0.468 | 0.343 | 0.060 | −0.007 | 0.000 |
| 4 | 0.412 | 0.416 | 0.402 | 0.384 | 0.428 | 0.392 | 0.047 | −0.007 | 0.000 |
| 5 | 0.442 | 0.443 | 0.440 | 0.428 | 0.499 | 0.383 | 0.062 | −0.010 | 0.000 |
| 6 | 0.276 | 0.302 | 0.297 | 0.334 | 0.385 | 0.278 | 0.025 | −0.009 | 0.000 |
| 7 | 0.336 | 0.327 | 0.320 | 0.328 | 0.330 | 0.353 | 0.008 | −0.008 | 0.000 |
| 8 | 0.509 | 0.477 | 0.511 | 0.499 | 0.383 | 0.298 | 0.095 | −0.009 | 0.000 |
| 9 | 0.328 | 0.335 | 0.339 | 0.325 | 0.309 | 0.291 | 0.047 | −0.008 | 0.000 |
| 10 | 0.312 | 0.328 | 0.328 | 0.347 | 0.358 | 0.247 | 0.005 | −0.008 | 0.000 |
| 11 | 0.441 | 0.436 | 0.423 | 0.367 | 0.310 | 0.253 | 0.013 | −0.010 | 0.000 |

(c) Smoothing 0.1

Table D.8: The table contains average daily profit for confidence based strategy for different smoothing values. The model is *Relative Difference* based *HMM*. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.119 | 0.124 | 0.095 | 0.095 | 0.069 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.467 | 0.455 | 0.453 | 0.452 | 0.470 | 0.359 | 0.058 | 0.000 | 0.000 |
| 3 | 0.686 | 0.685 | 0.674 | 0.617 | 0.517 | 0.475 | 0.058 | −0.007 | 0.000 |
| 4 | 0.367 | 0.365 | 0.356 | 0.403 | 0.420 | 0.393 | 0.056 | −0.004 | 0.000 |
| 5 | 0.251 | 0.261 | 0.303 | 0.341 | 0.330 | 0.440 | 0.033 | −0.005 | −0.004 |
| 6 | 0.321 | 0.320 | 0.314 | 0.295 | 0.331 | 0.351 | 0.021 | −0.008 | −0.004 |
| 7 | 0.441 | 0.445 | 0.469 | 0.511 | 0.520 | 0.431 | 0.043 | −0.002 | −0.004 |
| 8 | 0.362 | 0.370 | 0.371 | 0.374 | 0.351 | 0.376 | 0.055 | −0.008 | −0.004 |
| 9 | 0.420 | 0.420 | 0.396 | 0.350 | 0.439 | 0.349 | 0.069 | −0.005 | −0.004 |
| 10 | 0.271 | 0.268 | 0.276 | 0.337 | 0.350 | 0.283 | 0.053 | −0.011 | 0.000 |
| 11 | 0.395 | 0.386 | 0.406 | 0.413 | 0.422 | 0.396 | 0.046 | −0.008 | −0.004 |

(a) Cropping 0.4

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.243 | 0.243 | 0.219 | 0.157 | 0.069 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.413 | 0.407 | 0.399 | 0.457 | 0.409 | 0.324 | 0.075 | 0.000 | 0.000 |
| 3 | 0.684 | 0.689 | 0.674 | 0.666 | 0.597 | 0.464 | −0.008 | −0.007 | 0.000 |
| 4 | 0.408 | 0.400 | 0.408 | 0.431 | 0.384 | 0.346 | −0.044 | −0.004 | 0.000 |
| 5 | 0.557 | 0.548 | 0.560 | 0.547 | 0.460 | 0.367 | 0.004 | −0.005 | 0.000 |
| 6 | 0.449 | 0.415 | 0.409 | 0.376 | 0.422 | 0.312 | −0.013 | −0.009 | −0.004 |
| 7 | 0.491 | 0.503 | 0.448 | 0.468 | 0.472 | 0.345 | −0.037 | −0.009 | −0.004 |
| 8 | 0.401 | 0.378 | 0.391 | 0.453 | 0.455 | 0.390 | 0.014 | −0.012 | −0.006 |
| 9 | 0.349 | 0.344 | 0.394 | 0.381 | 0.392 | 0.339 | −0.055 | −0.007 | −0.002 |
| 10 | 0.324 | 0.339 | 0.344 | 0.381 | 0.454 | 0.254 | 0.006 | −0.008 | −0.004 |
| 11 | 0.489 | 0.492 | 0.498 | 0.482 | 0.463 | 0.394 | −0.049 | −0.007 | 0.000 |

(b) Cropping 0.2

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.186 | 0.186 | 0.186 | 0.210 | 0.032 | 0.013 | −0.007 | 0.000 | 0.000 |
| 2 | 0.389 | 0.389 | 0.412 | 0.372 | 0.364 | 0.386 | 0.005 | 0.000 | 0.000 |
| 3 | 0.262 | 0.257 | 0.259 | 0.280 | 0.402 | 0.233 | 0.030 | −0.007 | 0.000 |
| 4 | 0.262 | 0.261 | 0.258 | 0.302 | 0.339 | 0.304 | 0.011 | −0.008 | 0.000 |
| 5 | 0.291 | 0.279 | 0.272 | 0.318 | 0.409 | 0.297 | −0.016 | −0.005 | 0.000 |
| 6 | 0.205 | 0.207 | 0.239 | 0.266 | 0.316 | 0.137 | −0.058 | −0.006 | 0.000 |
| 7 | 0.150 | 0.165 | 0.192 | 0.245 | 0.322 | 0.296 | 0.015 | −0.002 | −0.006 |
| 8 | 0.196 | 0.181 | 0.196 | 0.180 | 0.094 | 0.204 | −0.011 | −0.009 | −0.004 |
| 9 | 0.350 | 0.332 | 0.341 | 0.299 | 0.304 | 0.380 | 0.045 | −0.015 | −0.006 |
| 10 | 0.299 | 0.298 | 0.279 | 0.281 | 0.245 | 0.240 | 0.020 | −0.006 | 0.000 |
| 11 | 0.227 | 0.223 | 0.239 | 0.241 | 0.237 | 0.181 | 0.022 | −0.014 | 0.000 |

(c) Cropping 0.1

Table D.9: The table contains average daily profit for confidence based strategy for different cropping values. The model is *Relative Difference* based *HMM*. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.045 | 0.025 | 0.060 | −0.079 | −0.040 | 0.000 | 0.000 | 0.000 |
| 2 | 0.179 | 0.165 | 0.151 | 0.158 | 0.190 | 0.098 | 0.080 | 0.009 | 0.000 |
| 3 | 0.086 | 0.075 | 0.088 | 0.006 | 0.044 | 0.055 | −0.059 | −0.004 | −0.005 |
| 4 | 0.056 | 0.049 | 0.058 | 0.086 | 0.179 | 0.228 | 0.022 | −0.005 | 0.000 |

(a) Relative Difference - Simple Threshold

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.107 | 0.118 | 0.154 | 0.153 | 0.305 | 0.307 | −0.029 | 0.000 | 0.000 |
| 3 | 0.355 | 0.338 | 0.326 | 0.305 | 0.256 | 0.053 | −0.076 | −0.008 | 0.004 |
| 4 | 0.404 | 0.402 | 0.375 | 0.327 | 0.312 | 0.164 | 0.002 | −0.018 | −0.001 |

(b) Relative Difference - Confidence Threshold

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.140 | −0.064 | 0.000 | 0.000 | 0.000 |
| 2 | 0.023 | 0.016 | 0.004 | 0.042 | 0.044 | 0.151 | 0.020 | 0.000 | 0.000 |
| 3 | 0.208 | 0.210 | 0.192 | 0.187 | 0.144 | 0.108 | −0.032 | −0.008 | 0.000 |
| 4 | 0.412 | 0.400 | 0.397 | 0.400 | 0.319 | 0.168 | −0.028 | −0.009 | 0.000 |

(c) Logarithmic Difference - Simple Threshold

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.134 | −0.075 | −0.007 | 0.000 | 0.000 |
| 2 | 0.200 | 0.192 | 0.164 | 0.061 | 0.094 | 0.223 | −0.026 | 0.000 | 0.000 |
| 3 | 0.408 | 0.409 | 0.406 | 0.395 | 0.332 | 0.066 | −0.086 | 0.000 | 0.000 |
| 4 | 0.410 | 0.428 | 0.438 | 0.418 | 0.413 | 0.229 | −0.053 | −0.016 | 0.001 |

(d) Logarithmic Difference - Confidence Threshold

Table D.10: The table contains average daily profit of 2nd order *HMM*s. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.045 | 0.025 | 0.060 | −0.079 | −0.040 | 0.000 | 0.000 | 0.000 |
| 2 | 0.117 | 0.120 | 0.120 | 0.134 | 0.237 | 0.082 | 0.086 | 0.029 | 0.000 |
| 3 | −0.057 | −0.106 | −0.091 | −0.060 | 0.000 | 0.114 | −0.025 | 0.008 | −0.002 |
| 4 | 0.080 | 0.091 | 0.093 | 0.139 | 0.133 | 0.143 | 0.024 | −0.002 | −0.002 |

(a) Relative Difference - Simple Threshold

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.049 | 0.054 | 0.025 | 0.060 | −0.079 | −0.070 | −0.007 | 0.000 | 0.000 |
| 2 | 0.206 | 0.205 | 0.196 | 0.165 | 0.324 | 0.190 | 0.020 | 0.000 | 0.000 |
| 3 | 0.159 | 0.142 | 0.126 | 0.065 | 0.106 | 0.037 | −0.013 | −0.020 | 0.007 |
| 4 | 0.359 | 0.360 | 0.389 | 0.339 | 0.343 | 0.137 | −0.062 | −0.020 | 0.005 |

(b) Relative Difference - Confidence Threshold

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.140 | −0.064 | 0.000 | 0.000 | 0.000 |
| 2 | −0.041 | −0.034 | −0.025 | 0.054 | −0.021 | 0.150 | −0.007 | 0.002 | 0.003 |
| 3 | 0.167 | 0.120 | 0.159 | 0.177 | 0.257 | 0.016 | 0.013 | 0.007 | 0.003 |
| 4 | 0.578 | 0.587 | 0.547 | 0.504 | 0.401 | 0.227 | −0.060 | 0.007 | 0.003 |

(c) Logarithmic Difference - Simple Threshold

| States | 0 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 | 0.1 | 0.3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.129 | −0.129 | −0.067 | −0.108 | −0.134 | −0.075 | −0.007 | 0.000 | 0.000 |
| 2 | 0.147 | 0.137 | 0.155 | 0.168 | 0.125 | 0.152 | 0.017 | 0.003 | 0.000 |
| 3 | 0.111 | 0.129 | 0.139 | 0.186 | 0.161 | 0.218 | −0.070 | −0.001 | 0.007 |
| 4 | 0.192 | 0.201 | 0.190 | 0.191 | 0.208 | 0.157 | 0.042 | 0.030 | 0.003 |

(d) Logarithmic Difference - Confidence Threshold

Table D.11: The table contains average daily profit of 3rd order *HMM*s. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.349 | −0.349 | −0.277 | −0.429 | −0.150 | 0.000 | 0.000 | 0.000 | 0.000 |
| 2 | 0.668 | 0.661 | 0.662 | 0.693 | 0.625 | 0.138 | 0.000 | 0.000 | 0.000 |
| 3 | 0.664 | 0.662 | 0.667 | 0.696 | 0.635 | 0.441 | 0.000 | 0.000 | 0.000 |
| 4 | 0.566 | 0.567 | 0.569 | 0.567 | 0.557 | 0.425 | 0.008 | 0.000 | 0.000 |
| 5 | 0.516 | 0.525 | 0.523 | 0.516 | 0.573 | 0.380 | −0.004 | 0.000 | 0.000 |
| 6 | 0.538 | 0.534 | 0.528 | 0.510 | 0.551 | 0.356 | −0.004 | 0.000 | 0.000 |
| 7 | 0.496 | 0.494 | 0.504 | 0.500 | 0.544 | 0.336 | −0.001 | 0.000 | 0.000 |
| 8 | 0.519 | 0.516 | 0.514 | 0.495 | 0.505 | 0.273 | 0.000 | 0.000 | 0.000 |
| 9 | 0.475 | 0.477 | 0.485 | 0.489 | 0.536 | 0.461 | 0.000 | 0.000 | 0.000 |
| 10 | 0.499 | 0.505 | 0.505 | 0.518 | 0.536 | 0.342 | 0.000 | 0.000 | 0.000 |
| 11 | 0.527 | 0.530 | 0.515 | 0.540 | 0.576 | 0.353 | 0.000 | 0.000 | 0.000 |

(a) Stationary Logarithmic Difference based HMM learnt together with simple threshold strategy.

| States | 0 | 1e-05 | 3e-05 | 1e-04 | 3e-04 | 0.001 | 0.003 | 0.01 | 0.03 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.210 | −0.210 | 0.071 | 0.071 | 0.071 | −0.003 | 0.000 | 0.000 | 0.000 |
| 2 | 0.507 | 0.489 | 0.489 | 0.484 | 0.430 | 0.280 | 0.001 | 0.000 | 0.000 |
| 3 | 0.506 | 0.511 | 0.493 | 0.460 | 0.405 | 0.322 | −0.003 | 0.000 | 0.000 |
| 4 | 0.466 | 0.454 | 0.460 | 0.457 | 0.425 | 0.303 | −0.003 | 0.000 | 0.000 |

(b) Stationary Logarithmic Difference based 2nd Order HMM learnt together using 1/8 window with simple threshold strategy.

Table D.12: The table contains average daily profit of *HMM*s with combined improvements. The values are increased 1000 times to avoid leading zeros. The columns are representing confidence thresholds while the rows are representing numbers of hidden states of the model.

# Appendix E

# DVD Content

An attached DVD has following structure:

```
|-- index.html          Project overview and links
|-- readme.txt          Short info about files
|-- bin                 Binaries of the NY Downloader
|-- data                Data used for analysis
|-- src                 Source codes
|   |-- NYDownloader          Downloading Utility
|   `-- R                 R scripts
`-- text                Documentation
    |-- zadrape1.pdf          PDF output
    `-- latex             LaTeX sources
        |-- figures           All the images including graphs
        `-- tables            All the tables
```