4 Methodology

4.1 Overview

NELION is an Internet based personal investment tool and portfolio management software. It retrieves stock data from the Internet, manages it and generates investment suggestions and portfolio updates via e-Mail and alerts via short messaging system, SMS. Additionally, an investor can view his portfolio via a web page and can manipulate his preferences and execute purchases or sales.

The system is separated into a task agent, an administration tool and an HTML interface, each of which attaches to the common NELION database.

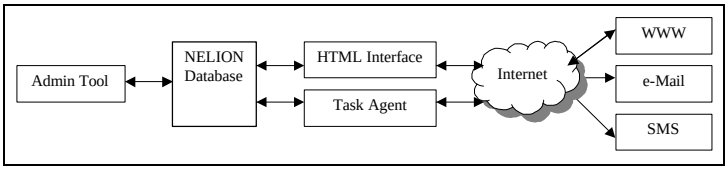

Figure 4.1.1: Block Diagram NELION

4.2 The HTML Interface

The HTML Interface provides an investor using NELION with a means to manage his portfolio and edit his preferences. After entering his e-mail address as a user name and the

associated password under www.nelion.net the investor gets an overview of his porfolio including the current value, stocks owned and current recommendations. A graph compares the return on investment of his portfolio compared to the Dow Jones Industrial Index as well as the Nasdaq.

From this main page, the investor can select hyperlinks that allow him to maintain his investment preferences and account parameters, buy or sell or research specific stocks.

4.3 The Administration Tool

The Administration Tool is designed for the administration of the data, parameters and tasks on the database. As such, it is only used by the NELION system manager and does not require any investment experience, since it only provides a means to maintain data but does not contain any logic for stock prediction or portfolio distribution.

The data entry screens mirror the structure of the database and consequently include a data entry screen for all the major tables. For the investor data, the Administration Tool provides one page for the data kept in the investor table. Additionally, there are pages to view the current portfolio, its development in a graphical format, the purchasing and sales history, as well as the return on investment.

The stock interface includes pages with lists of the mathematical models and current predictions with different horizons, in addition to the standard information, such as the company name and ticker, current price and volume. A

parameter dialog allows the administrator to maintain all the adjustable settings of the system, while the task list shows the jobs that the task agents are currently working on, or that is waiting for execution by one of them.

4.4 The Database

The database is the central store of information and has to scale to several gigabytes in size in order to be able accommodate historic data, models, recommendations and portfolio histories for thousands of stocks and investors. The diagram below shows a simplified conceptual data model. The additional tables required for the model storage have not been included for simplicity. The complete data model is shown in Appendix D. The primary keys for each table is included and underlined in the figure.

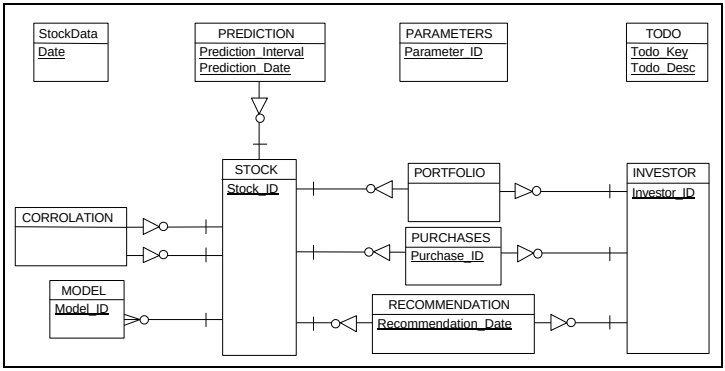

Figure 4.4.1: Simplified Conceptual Data Model

The historic price and volume data for each stock that is tracked is stored in a separate table, which is created when the stock is entered. The figure above only shows one representative table, StockData.

The system hinges on the two main entities, stocks and investors. For each stock, the system stores the company name, stock exchange abbreviation called the 'ticker" as well as the volatility. The internal Stock_ID number is the primary key of the table and used as a foreign key in all related tables. The correlation between all stocks, for example, uses the two Stock_IDs as its primary key and merely stores the correlation as an attribute. The prediction table needs two additional fields, the prediction interval and the prediction date, as a primary key and stores the percentage prediction increase or decrease as a positive or negative float as its only attribute. The stock model component is only represented by a single table, which defines its own internal model number Model_ID as a primary key. The additional tables that are required to store the assorted models are not included in the diagram.

The investor entity requires less support tables but contains more fields within the table itself but also uses an Investor_ID as a primary key. Besides the investor name and his e-mail addresses for update and notification purposes as well as SMS updates, the system stores the investment preferences in the form of risk adversity parameters, expected minimum annual return as well as the minimum transaction volume. Additionally, it tracks parameters that control how frequently the investor receives e-mails with an update of the current

portfolio and purchase and sale recommendations. In order to be able to show the change in portfolio value from one e-mail to the next, it also stores the account value from the last update. Furthermore, it maintains a field for the cost of each transaction.

Lastly, each investor has an investor type, where three different options are possible: A “Test Investor” is used to simulate trading behavior and the resultant portfolio for different configurations in a past period, so that a new potential user of the system can select a configuration with the desired characteristics. For these investors, the system maintains a start and end date for this test. An “Auto-Trader” is an automatic investor that autonomously acts on the investment recommendations in a “live” simulation on current data. This function allows the analysis of the system as a proof of concept. Finally, there is the “Normal Investor”, who receives regular updates, but that has to update his purchases and sales on the system, whether they were recommended or not.

The portfolio, purchases and recommendation tables provide the link between investors and stocks, since each investor has a portfolio containing zero or more stocks. Each inherits the respective internal identifiers as foreign keys.

The purchase table shows when these stocks were purchased and sold and at what price these transactions were executed. The current and historic recommendations are kept in the corresponding table along with their recommendation dates.

The parameters are not connected to the remaining tables, since they only store system values, which will be read by the task agent or Administration Tool for specific functions. The table describes the six parameters that are stored in this table.

Parameter	Description
TimerInterval	Specifies the frequency with which the Administration Tool updates the task list on the screen
BankRate	Guaranteed Interest Rate from the broker or bank
Diff2NY	Time difference to the New York Stock Exchange. This is used to schedule the Internet download task
Mutation	The likelihood of mutation in the genetic algorithm
SMS Threshold	If the value of a stock changes more than this threshold, an SMS message is sent to all investors who own the stock, as well as the System Administrator
SMS for System Administrator	The SMS e-mail address of the system administrator

Table 4.4.1: System Parameters

The task table has an implicit link to the stocks and investors, since all tasks relate to one of these two entities. Due to this dual connection, there cannot be an explicit database constraint and the referential integrity of the link has to be verified by the task agent.

4.5 The Task Agent

The task agent supports the stock prediction and portfolio management calculations, as well as the Internet interface. It retrieves individual tasks from the task list, marks them as “taken” until they have been executed and then deletes them from the database.

The task agent can perform nine different tasks. The Internet Load function to retrieve data from the World Wide Web is

always the first step. Given this data, the next tasks, calculating the volatility, mathematical models and the correlation between stocks can be executed. These tasks can be grouped together into a single task for a new time series. Further tasks include sending a portfolio update, possibly including transaction recommendations, to the investor. Lastly, the task agent executes the test investor function and a background thread that performs model optimization with a genetic algorithm. All of these are explained in detail in the following sections.

4.5.1 Internet Load

The Internet Load function connects to the Internet and downloads historic price and volume data for each stock tracked by NELION. If a stock has just been added to the system so that the data table is empty, it will attempt to load data starting from January 1, 1980. In case this function was not invoked for several days, it retrieves missing data in one download, bringing the stock data up-to-date.

The system stores the closing price for every day since the stock has started trading. On weekends, public holidays and days where the trading volume was nil, it assumes that the price has not changed but still inserts a record into the corresponding table. This facilitates the monthly model calculation, which uses the last day of each month as a basis. Additionally, it allows for consistent correlation calculations of

stocks that are traded in different markets and with different public holidays.

Significant changes in the stock price tend to signify dramatic occurrences either for the stock itself or for the market as a whole. Since this may require the attention of the investor, the system notifies the administrator and all investors who own a stock via a mobile phone SMS message. The sensitivity of this threshold is controlled by the SMS threshold parameter, which defaults to 20% so that stock price changes that exceed that value will result in a message.

4.5.2 Calculate Volatility

The volatility of each stock is calculated using the following equation:

$$s(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|x_i - x_{i-1}\right|}{x_i} \qquad \text{Equation 4.5.1}$$

This value measures the mean absolute percentage change in price over the entire interval for which the system has data. Though it treats public holidays like regular trading days, it ignores weekends.

4.5.3 Calculate Models

For each stock, NELION calculates prediction models of four different types: Autoregressive models of degree n (ARN), artificial neural networks (ANN), k-nearest neighbor models (KNN) and Markov models (MM). Since the system permits prediction horizons of one day, seven days (one week) and 30 days (one month), it calculates models for each of these. As input, it correspondingly uses the closing stock prices of the last n days, weeks or months to predict the closing stock price one prediction interval into the future.

This function serves as a bootstrap for the genetic algorithm described below, which uses the available models to further search the parameter space for better predictors.

The data was divided into a test and a training set but in order to capture trends throughout the available data, each input tuple had a 50% chance of being assigned to one of the two.

The quality of a model was measured by calculating the normalized mean squared error (NMSE) of the predictions in the test set.

$$\mathrm{NMSE} = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n} \left(\mathrm{prediction}_i - \mathrm{target}_i\right)^2}{\left(\max\left(\mathrm{target}_i\right) - \min\left(\mathrm{target}_i\right)\right)^2} \qquad \text{Equation 4.5.2}$$

Since each stock price time series has a different dynamic, NELION calculates models of each type and prediction

horizon with various parameter combinations and stores the best two configurations of each type on the database. The details of each model type are described in the following paragraphs.

The autoregressive models (ARN) use the Durbin-Levinson algorithm described in Chapter 2 to calculate a linear prediction model. The only parameters that could be adjusted for the model type were the number of input values, which ranged between two and 14 values. It stored the best two models for each prediction horizon on the database.

The artificial neural network (ANN) models in NELION use a single hidden layer with a single output unit, which represents the prediction of the model. The units in the input layer are mapped to the input tuple of the network. Each unit in the hidden and output layers is fully connected to the previous layer and has an additional link to a threshold input, which has a constant input of one.

Since the model is only defined for an input range of between zero and one, all input data is normalized to a range between zero and 0.5. This ensures that all values remain within unity, since the stock prices in our experiments never doubled their value within one unit of the investment horizon.

As suggested by Weigend and Nix, the hidden units have a sigmoidal transfer function while the output unit uses a linear transfer function [Weigend, Nix, 1994]. The artificial neural network was trained using the back propagation network as described in Chapter 2. The learn rate and momentum

parameters were set to 0.1 for all units, but in an effort to speed up convergence, the learn rate was left dynamic and increased or decreased by a factor two if consecutive updates were in the same direction.

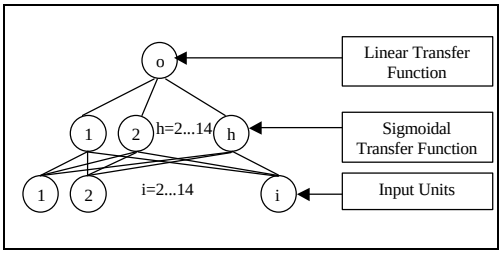

Figure 4.5.1: Artificial Neural Network

The system used batch learning so that weight updates on each link were performed after every epoch since this proved to be more reliable than on-line learning. Since the test error initially tends to exhibit rather erratic behavior, NELION imposes a minimum number of 500 epochs. Learning was stopped after three consecutive epochs increased the test error or until it reached a maximum number of learning epochs. This value was set at 1000 for an investment horizon of one day, 2000 for one week and 3000 for one month. These values were identified through experimentation and helped some configurations, which remained near the minimum error but never achieved three consecutive increases.

NELION tested all combinations of artificial neural network models with between two and 14 input units and between two

and 14 hidden units and stored the best two for each prediction interval.

The k-nearest neighbors (KNN) models algorithm retrieves each tuple in the test set and searches the training set for the constellations, which resemble the given pattern most closely. NELION calculates all models with between two and 14 input values and identifies between k=2 and k=14 "nearest neighbors." The distance from the input tuple to the tuples in the training set is calculated using the Euclidean metric, though the genetic algorithms described below can select between this and a Gaussian or constant metric.

The prediction for each tuple is the weighted average of the k nearest neighbors, where the weight of each neighbor is inversely proportional to the distance as shown in the equation below.

$$w_i = \frac{d_i}{\sum_{n=1}^{k} d_n}$$ Equation 4.5.3

The Markov models (MM) use between four and 14 input values and select between four and 20 random states from the training set. The system then assigns each tuple in the training set to one of these states and then counts the number of transitions from one state to another. Given these numbers, it is possible to calculate the probability of each transition. The prediction was the weighted outcome of between one and all

states used in the model. The weighting algorithm is the same as for k-nearest neighbors algorithm.

4.5.4 Calculate Correlations

The correlation between stock x and stock y measures how closely the two time series are related and is calculated as follows:

$$r(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$ Equation 4.5.4

In the formula above, x_i represents the price of time series x at a specific time i and $\bar{x}$ is the mean price for all i. The task calculates the correlation between the given time series and all other time series tracked in the system. If time series x equals time series y, the correlation is unity by definition.

4.5.5 Send E-Mail Update

In order to provide the investor with an update on his portfolio, its total value and the loss or gain for each stock currently held, the system periodically sends an e-mail to the specified address. The frequency of these messages is set for each investor, though the task can be created manually at any time for one specific or all investors.

4.5.6 Calculate Recommendation

Given models for all stocks tracked in NELION, the system calculates predictions for investment horizons on a daily, weekly and monthly basis. It is assumed that the transactions executed by the investor have a negligible influence on the stock market as a whole. The system uses the model with the lowest NMSE to predict the future stock price. It is worth noting, that the MM and KNN models use the entire historic data to predict the future stock prices and not only the test set, as was done during the model calculation.

The predictions for each stock are stored on the database as the percentage change from the current stock price and form the basis of a recommendation for each investor. Additionally, however, the recommendations take the current portfolio as well as risk adversity parameters of the investor into account by calculating the relative risk of all favorable future portfolios. The parameters pertain to stock correlation, volatility, model error and trading volume as well as a minimum transaction amount.

The correlation between two stocks measures the likelihood of congruent movement in response to external market forces, like interest rate changes, new laws, political conflict or acts of nature. The volatility measure described above measures the variability of the stock price. Stocks with a high volatility tend to show erratic price movements, making them a riskier investment than those with a low volatility. NELION is able to forecast some stocks with greater precision than others,

resulting in a lower NMSE for these time series. An investor can specify that the recommendations should favor these stocks, since this would decrease the risk of the resulting portfolio. A stock with a high turnover volume tends to show greater price stability. Additionally, the transactions by the investor affect the market price of the stock to a lesser degree for stocks with a high volume. Consequently, this reduces the risk associated with these kinds of stock.

In order to identify the portfolio with the lowest relative risk given the investor parameters, NELION calculates the risk of the current portfolio using the following equation.

$$Risk = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{C_{i,j} \cdot L_{i,j} \cdot E_{i,j}}{O_{i,j}} \qquad \text{Equation 4.5.5}$$

In this equation, the factors in the numerator ($C_{i,j}$ for the correlation, $L_{i,j}$ for the volatility and $E_{i,j}$ for the measure of error) increase the risk of the portfolio and the volume factor $O_{i,j}$ in the denominator decrease it.

The dependence on the correlation $C_{i,,j}$ is defined as follows, where C represents the investor specific correlation parameter, I_i is the portfolio value currently invested in stock i of and $r_{i,,j}$ is the correlation between stock i and j.

$$C_{i,j} = \lfloor C\left(\left(I_i I_j r_{i,j}\right) - 1\right)\rfloor + 1 \qquad \text{Equation 4.5.6}$$

Similarly, the volatility, error and volume components are defined as follows.

$$L_{i,j} = \lfloor L(l_i l_j - 1) \rfloor + 1 \qquad \text{Equation 4.5.7}$$

$$E_{i,j} = \lfloor E(e_i e_j - 1) \rfloor + 1 \qquad \text{Equation 4.5.8}$$

$$O_{i,j} = \lfloor O(o_i o_j - 1) \rfloor + 1 \qquad \text{Equation 4.5.9}$$

In these equations, L, E and O represent the investor parameters for volatility, model error adversity and volume preference respectively and must be in the interval [0,1]. The terms l_i, e_i, and o_i represent the volatility, model error and current trading volume for the stock i.

The relative weight of each factor is determined by the relationship between each parameter. A comparatively large value of L, for example, increases the weight of the volatility characteristics, which can be interpreted as a particular risk adversity as it pertains to volatility.

Each of the individual factors is reduced to unity in case one of the parameters C, L, E and O vanishes, making the risk calculation independent of that component. This is equivalent to the investor stating that the corresponding component should not be considered in his risk calculation.

Several boundary conditions were handled as exception cases in NELION. If the trading volume o_i of a particular stock was zero, the entire term in the double sum was disregarded. This is equivalent to disregarding this stock completely. If all investor parameters were set to zero, all terms in the double

sum would be unity and the calculated risk would be the same for all possible portfolios. This is equivalent to the investor not making any statement regarding his investment preferences and is disallowed by the system.

It is important to note that the equation above represents a *relative* risk calculation and that it does not map directly to a physical quantity. It does, however, permit NELION to compare the relative risk associated with different portfolios by initially calculating the relative risk value for the current portfolio and then searching for portfolios with a lower relative risk.

This is done using the gradient decent method over this n^2-dimensional parameter space. The standard algorithm is restricted to prevent the recommendation of negative ownership of specific stocks, called “short positions.” The system starts its search with a step size of one and iteratively calculates the resultant portfolio. In case the resultant portfolio does not have a lower relative risk, the step size is reduced by a factor two and the iteration is restarted. This process repeated until the step size has diminished to 10^{-8}.

Once the optimal portfolio is calculated, NELION filters it to ensure that the minimum transaction limit for the investor is not violated. This restriction prevents the system from recommending purchases or sales, where the cost of the transaction outweighs the benefit of it. This filter also validates that a sale suggestion of a particular stock does not result in a portfolio, where the sale of the remaining stocks of the same

company would force a transaction that would fall below the minimum transaction volume at the current price. In such a case, the system would recommend selling all of the stocks for this company.

For example, if NELION finds the optimum by selling 60 of the 100 stocks of company XYZ in the portfolio, and the future sale of the remaining 40 stocks would result in a transaction of less than the minimum transaction volume, it would recommend selling all 100 stocks.

In order to ensure that the expected portfolio return specified by the investor is met, the portfolio selection only includes stocks, for which the system has predicted a price increase greater than this threshold. This results in a customized portfolio recommendation for each investor, which is sent to his e-mail address for the next investment horizon. For the auto-investors, the recommendations are “executed” immediately, so that the purchases and the portfolio are updated automatically.

4.5.7 New Time Series

This task combines the tasks that are necessary for each new time series: Internet Load, Calculate Volatility, Calculate Models and Calculate Correlations.

4.5.8 Test Investor

In order to identify parameters that correspond to the risk and return expectations of an investor, NELION provides the Test Investor function. This task simulates the behavior of an investor for a specified period in the past so that the outcome of the resulting portfolio can be analyzed.

This function is designed with the assumption that test investors with all different parameter combinations are created on a specific database. The investors are then tested in a defined interval of sufficient length to be able to analyze their behavior.

Assuming that the dynamics of the past hold in the future, one can then aggregate the results from many different parameter combinations. It is then possible to make statistical statements about investors with certain parameter combinations so that a potential user of NELION can select the risk and return structure suitable for his needs.

4.5.9 Parameter Selection with the Genetic Algorithm

Since the computer running the task agent only responds to requests entered into the task list on the database, it spends the majority of its time waiting for new jobs. This processing time is nevertheless available for productive tasks at no incremental costs. In order to take advantage of this power, NELION starts a background thread to search the parameter

space of the prediction models using a genetic algorithm if no other tasks need to be addressed immediately.

Genetic algorithms imitate the gene selection process from nature to mix different traits from two parents, in the hope of generating a child that can outperform either parent, as defined by some fitness function. Much like in its biological equivalent, where an animal, the genotype, is defined by its genetic makeup, its phenotype, a mathematical model can be specified by a series of parameters. These parameters are encoded in a string of bytes of a finite length.

Biological reproduction entails the selection of specific genes from the two parental phenotypes. Similarly, a mathematical genetic algorithm maps this crossover function to the random selection of bytes from the phenotypes of the two parents resulting in a child phenotype, which has inherited some features from either of its parents.

Biological mutation is a process by which a specific gene was not inherited from either parent but is randomly generated, frequently through some sort of defect or external influence. In the overwhelming majority of cases, this leads to children with undesirable characteristics. However, occasionally, this leads to a new trait that increases the likelihood of survival and begins to dominate the population thereafter. This dynamic can be imitated in genetic algorithms by selecting random bytes instead of inheriting them from one of the parents on occasion.

The child phenotype can be used to generate a new mathematical model, which can be trained and tested. If the model error is lower than either of its parents, it is apparently superior and can replace one of the two parents.

For each stock and model type (ANN, ARN, MM and KNN) the system stores two models with the lowest test error as parents. Using these two models, the algorithm uses crossover and mutation to generate new models, calculate the predictive quality of them and to replace the worse of the existing parent models if the child test error is lower than either of them. The likelihood of mutation is controlled through a system parameter, which can be anywhere between 0% (no mutation) and 50%, meaning that on average every second byte is randomly selected with no heritage from either parent phenotype.

The resulting phenotype is converted back to a genotype by interpreting the string and populating the parameters of a new model. These parameters are validated to ensure a valid and sensible configuration. The specific parameters and validation depends on the model type and are shown in the table below.

Model	Parameter	Validation
ARN	# input values	• Can not be more than twice the # of input values of either of the parent models • Can not be more than 32 • Must be at least 1
ANN	# input values (units)	• Can not be more than twice the # of input values of either of the parent models • Can not be more than 32 • Must be at least 1
	# hidden units	• Can not be more than twice the # of input values of either of the parent models • Can not be more than 32 • Must be at least 1
	Transfer Function	• Must be 1 or 2 representing the constant *a* in the transfer function
KNN	# input values	• Can not be more than twice the # of input values of either of the parent models • Can not be more than 32 • Must be at least 1
	# of nearest neighbors ("k")	• Can not be more than twice the # of nearest neighbors of either of the parent models • Can not be more than 32 • Must be at least 1
	Metric	• Must be 1, 2 or 3 representing Euclidean, Gaussian or Constant functions
	Weighting	• Must be 1, 2 or 3 representing Euclidean, Gaussian or Constant functions
MM	# input values	• Can not be more than twice the # of input values of either of the parent models • Can not be more than 32 • Must be at least 1
	# states	• Can not be more than twice the # of states of either of the parent models • Can not be more than 32 • Must be at least 1
	# states used to calculate prediction	• Can not be more than # states • Must be at least 1
	Weighting	• Must be 1, 2 or 3 representing Euclidean, Gaussian or Constant functions

Table 4.5.1: Genetic Algorithm Parameter Validation

Given the verified parameters, the system calculates the model and if its test error is lower than that of either of the other two models, the system replaces it on the database.