Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master's Thesis Nr. 3

Systems Group, Department of Computer Science, ETH Zurich

Forecasting Stock Market Volatility with Search Engine Query Statistics

by

Romain Beker

Supervised by

Prof. Donald Kossmann

October 2010–March 2011

inf | Informatik
Computer Science

Abstract

We provide a framework to design, create, test and evaluate linear forecasting models incorporating query statistics. As part of the framework we provide algorithms for identifying search terms that have forecasting power. We test the algorithms performances empirically with Google statistics on three applications: stock market volatility, arrivals in the Republic of Seychelles and initial unemployment claims in the USA. We also showcase the framework by improving stock market volatility forecasts. Finally we compare Google and Twitter query statistics.

Acknowledgments

I'm grateful to my supervisor Prof. D. Kossmann for his support, his guidance, his patience and his trust. I thank Dr. M. Grineva for her help. I also thank the Chair of Entrepreneurial Risks at the ETH Zurich and in particular Prof. D. Sornette and Dr. V. Filimonov for the fruitful discussions.

Last but not least I thank C. Papazian, my family and my friends for their support.

Contents

1 Introduction **3**

2 Google Insight Data Source **5**

3 Twitter Data Source **7**

4 Stock Market Volatility **8**
4.1 Actual Volatility 8
4.2 Implied Volatility 8
4.2.1 Financial Options 9
4.2.2 The VIX 9

5 Defining Google Insight Models **11**
5.1 Traditional Forecasting Models 11
5.2 Enhancing Forecasting Models 12
5.3 Training Forecasting Models 13
5.4 Evaluating Forecasting Models 13
5.5 Time Serie Correlation 14

6 Model Creation: The Process **15**
6.1 Model Structure 16
6.2 Incorporating Keywords 16
6.2.1 The Naive Method 18
6.2.2 The Incremental Increase Method 18
6.2.3 The Incremental Increase Top N Variant 20
6.2.4 The Incremental Decrease Method 21
6.2.5 The Correlation Method 22
6.3 Weighting Keywords 23
6.4 Choosing Keyword Offsets 24

7 Model Creation: Experiment Part I **26**
7.1 Data . . . 26
7.1.1 Input Data . . . 26
7.1.2 Keyword Data . . . 28
7.1.3 In-Sample Parition . . . 29
7.1.4 Benchmark Structure . . . 30
7.2 Keyword Selection . . . 30
7.2.1 Initial Unemployment Claims . . . 30
7.2.2 Arrivals in the Republic of Seychelles . . . 35
7.2.3 Implied Volatility . . . 39

8 Model Creation: Experiment Part II **45**
8.1 Data . . . 45
8.1.1 Input Data . . . 45
8.1.2 Keyword Data . . . 45
8.1.3 In-Sample Partition . . . 45
8.2 Benchmark . . . 45
8.3 Results: Benchmark . . . 46
8.4 Results: Machine-Created Model with Automatic Keyword and Offset Selection . . . 47
8.5 Result: Model with Manual Keyword and Offset Selection . . . 49
8.6 Results: Rolling-Window Model . . . 51

9 Google Insight vs Twitter **52**
9.1 Results for All Keywords . . . 53
9.2 Results for the specific Keywords Subset . . . 54

10 Conclusion & Future Work **56**

A Keyword Universe **59**

Chapter 1

Introduction

In recent years Internet search engines have become important in the way people inform themselves. When people type in queries on search engines they implicitly express their interest in the object of the query. The aggregated interests from users all over the world provide an extremely rich source of information about social behaviors, opinions and sentiments. Search engine statistics provide a proxy for that information and contrasts with classical means of obtaining the information such as phone polls. Indeed among others search engine statistics are cheaper and incorporate much larger sample sizes than traditional techniques. In the case of the market leader Google, one can expect more than 10 billion search queries per month [1].

This master's thesis seeks to provide a framework for incorporating search engine statistics in forecasting models. It also aims at demonstrating the framework by improving stock market volatility forecasting models by using search statistics. In the financial world there is a strong motivation behind predicting volatility. Trading strategies with high profit potential can be derived from the predictions.

Recent studies have illustrated in a variety of domains how Google statistics can help in the process of forecasting. In particular it has been shown to improve the forecast of seasonal influenza propagation [2] as well as initial unemployment claims [3]. Google data has also been used in the context of modelling car, retail, home and travel sales [1]. In addition, it can be shown that Google Insight statistics for specific search terms also correlate with variables such as cinema admissions [5] and the success of movies, music songs or video games [8]. Other studies have also highlighted excellent performance when predicting stock-market direction [9] and the movies success [10] using Twitter.

The master's thesis is organized as follows. In chapter 2 we present our pri-

[1] Source: ComScore,September 2010, October 2010, November 2010, December 2010

mary source of search engine statistics provided by Google through the Google Insight tool. We then present in chapter 3 our secondary data source for query statistics: Twitter. We continue in chapter 4 with a brief introduction to stock market volatility.

In chapters 5 and 6 we provide a framework to design, create, test and evaluate forecasting models incorporating query statistics. The framework also features algorithms to automatically identify search terms whose statistics are relevant for the forecasting problem at hand. As part of this master's thesis we have developed an accompanying software tool that implements the framework. In the chapters 7 and 8 we present experiments on empirical data to first test the algorithms for identifying search terms and then show how we can improve volatility forecasts. We compare results for the search statistics aware models with results for the AR(1) benchmark model.

In chapter 9 we investigate the relation between query data from our two sources (Google and Twitter) in order to understand whether the sources could be used alternatively. We finally conclude with possible future work.

Chapter 2

Google Insight Data Source

Google Insight [1] (previously known as Google Trends) is a tool by Google that allows to display and download search statistics for search terms given as input. In the context of this paper we refer to search terms as ”keywords”.

Figure 2.1 shows as example the search statistics for the keyword ”visit Switzerland”. The y-axis displays the relative number of searches for the keyword and the x-axis gives the date.

Figure 2.1: Search statistics for ”visit Switzerland”

For each keyword Google Insight provides us with a time serie indicating the relative number of queries for the keyword.

[1] Google Insight

Timeframe

Google Insight publishes weekly aggregated search statistics. The data gets published with a delay of one week.

Universe

Google Insight computes the search statistics for an estimated 50 million keywords [2]. Some keywords contain only monthly data due to lower number of queries.

Classification

Google uses a classification system for the search statistics. The classifier features two dimensions. The first is the search category (ex. travel, sports, society...) and the second is the regional interest (ex. USA, Europe, Zurich,...) indicating the origin of the queries. One can obtain the global search statistics or the statistics for a subset of categories and regions.

In this paper we only explore global search statistics without investigating categories and regional interests.

Data Normalization & Scaling

Google normalizes its data by using a proprietary process for which almost no information is publicly available. It seems however that for a large part the process takes into account relative number of searches across regions [3]. For instance searches for a given keyword occurring from New York, USA will be weighted less than searches occurring from Zug, CH for the same keyword because much more queries occur from New York overall.

The search statistics are scaled to fall between 0 and 100 [4]. For a keyword time serie the normalized numbers of queries are divided by the biggest number and multiplied by 100. When not enough data is available the relative number of queries is zero.

[2] See [2]
[3] Google Insight normalization
[4] Google Insight scaling

Chapter 3

Twitter Data Source

While technically not a search engine, Twitter can also be exploited to extract keyword statistics. Each tweet can be considered as a complex search query that contains information.

In order to extract keyword statistics from Twitter tweets we start by recording tweets in real-time using the Twitter API[1]. It allows to collect about one million public tweets per day (24h).

For each tweet we record its timestamp, the username of its author and the message itself.

To compute the keyword statistics for a specific timeframe we count, for each keyword, its occurrences over the timeframe. For instance, if we are interested in the daily time frame we count for every day the occurrences of each keyword. We then normalize the number of occurrences to fall between 0 and 100.

More sophisticated strategies for computing the keyword statistics could be the object of future work.

[1] We use the Twitter4J implementation for the Twitter API

Chapter 4

Stock Market Volatility

We introduce two different types of stock market volatility, the actual volatility and the implied volatility.

4.1 Actual Volatility

Actual volatility for a given financial instrument (stock, bond, currency,..) is a statistical measure of the dispersion of its returns within a specific time horizon. The value of the actual volatility is obtained by computing the standard deviation of the returns over the specific time horizon and is usually expressed in annualized terms. Actual volatility expresses the amount of uncertainty and risk in the fluctuations of a financial instrument. Commonly, the higher the volatility, the larger the changes in price (in both directions) and the riskier the instrument. Actual volatility could also be interpreted as a measure of activity on the financial market. The higher the volatility the higher the activity on a financial market. We argue that rising activity on a market could lead to (or be preceded by) increasing number of Google searches for some keywords such as ”buy stocks” or ”sell stocks”. And vice-versa for decreasing activity. We therefore believe that search engine statistics can potentially contain information relevant for modeling volatility.

4.2 Implied Volatility

The second type of volatility called implied volatility refers to the market expectation of future (actual) volatility over a given time horizon. At this point in order to explain how implied volatility is derived from market prices we briefly and superficially introduce financial options and the Black&Scholes model for pricing them. For more information we suggest consulting [18] and [19].

4.2.1 Financial Options

Options are financial instruments that come in two types: the call option and the put option. The call option is characterized by the underlying asset, the strike price and the expiration date. When one buys a call option on a given asset one obtains the right (but not the obligation) to buy at the expiration date a predertimned amount of the asset at the strike price. Conversely a put option on a given asset gives the right to sell at the expiration date a predetermined amount of the asset at the strike price.

A prevalent way of valuing a call or put option is given by the Black&Scholes model. The model prices options as follows:

$$C = SN(d_1) - Xe^{-rT}N(d_2) \tag{4.1}$$

$$P = 1 - C \tag{4.2}$$

with

$$d_1 = \frac{ln(S/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \tag{4.3}$$

$$d_2 = d_1 - \sigma\sqrt{T} \tag{4.4}$$

C and P denote the prices of the call and put option respectively. S designates the market price of the underlying asset. X is the strike price, r is the risk-free interest rate and T the time (in days) until the expiration date. N denotes the cumulative normal distribution and σ is the actual volatility of the underlying asset during the lifetime of the option. Thus the exact value of σ is unknown until the date of expiration. In order to price options using the equations 4.1 and 4.2 one needs to provide an estimate for σ.

Implied volatility is the volatility (σ) derived from an option pricing model such as Black&Scholes when knowing the prevailing price of an option. Option prices can be observed on the financial markets.

For the purposes of this paper we are interested in implied volatility. Estimating implied volatility better than the market consensus allows to identify options that are over -or under-priced and develop a profitable strategy trading them.

4.2.2 The VIX

The VIX is the market volatility index calculated by the Chicago Board Options Exchange (CBOE)[1]. The index introduced in 1993 has become a popular measure

[1]CBOE VIX

of S&P 500's [2] implied volatility for the next 30-day period. The index not only serves for indicating the market's expectation of future volatility, it also represents a "fear gauge" reflecting investor's risk aversion.

As of September 22 2003 the VIX is calculated from prices of options from the S&P 500 index. For the detailed VIX calculation please see [12].

In the context of this paper the VIX serves as our volatility data.

[2]The S&P 500 or the Standard & Poor 500 Index reflects prices of 500 large-cap american stocks. S&P 500

Chapter 5

Defining Google Insight Models

In this section we define Google Insight forecasting models. They are forecasting models improved with Google Insight statistics. We also introduce performance measures for evaluating the models.

5.1 Traditional Forecasting Models

The goal of a model is to approximate the value of an input time serie at a specific time. When the time is in the future, we call the model a forecasting model. In the scope of this paper we consider linear forecasting models of the form:

$$input_t = a_0 + a_1 * input_{t-1} + a_2 * input_{t-2} + ... + a_n * input_{t-n} \quad (5.1)$$

$$0 < n <= t \quad (5.2)$$

The $a's$ are constants whose values are to be determined when training the model. $input_x$ denotes the value of the input time serie at time x.

We define two types of basis elements that are combined to form a model. The first type of element,the constant, consists of a simple constant (ex. a_0). We call the second type of element the input element. The input element consists of a constant multiplied by the value of the input time serie at a specific time offset (ex. $a_1 * input_{t-1}$). Please note that offsets bigger than $t - 1$ induce a look-ahead bias and are therefore not permitted.

In order to build a traditional linear forecasting model one can form any combination of constants and input elements by combining them with the addition operation as shown in equation 5.1.

5.2 Enhancing Forecasting Models

In order to enhance the models with Google Insight keywords we introduce a new type of basis element called the keyword element. The keyword element complements the constant and the input element.

A keyword element has the following form:

$$a_0 * keyword_{t-n} \tag{5.3}$$

$$0 < n <= t \tag{5.4}$$

The keyword element consists of a constant multiplied by the value of the keyword time series at a specific time offset $t - n$.

We define K as the set containing all keywords. K therefore constitutes the keyword universe. A model doesn't necessarily use all keywords in K but can use subset $K' \subset K$ to improve its predictions. The item $keyword_{offset}$ as in equation 5.3 represents the weighted mean value of all keywords $K' \subset K$ used in the model:

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|}(keyword_{k,t} * weight_k) \tag{5.5}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{5.6}$$

Where $keyword_{k,t}$ denotes the value of the time serie for keyword k at time t and $weight_k$ the weight for keyword k.

Keyword elements can be combined with constants and input elements to create Google Insight models of the form:

$$input_t = a_0 + a_1 * input_{t-1} + ... + a_n * input_{t-n} + a_{n+1} * keyword_{t-1} + ... + a_{n+1+i} * keyword_{t-i} \tag{5.7}$$

$$0 < n, i <= t \tag{5.8}$$

$$keyword_t = \frac{1}{|\ K'\ |} * \sum_{k=1}^{|K'|}(keyword_{k,t} * weight_k) \tag{5.9}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{5.10}$$

5.3 Training Forecasting Models

The process of training a model aims at calibrating the model for a specific input serie. It involves estimating the model's parameters (its constants) so that the model fits the input serie as closely as possible. More specifically, we define the process of training a model as a linear least square problem:

$$Ax = b \tag{5.11}$$

$$\text{Minimize } \|r\| = \|Ax - b\| \tag{5.12}$$

Where the matrix A contains the structure of the model, the vector x contains the constants to estimate and the vector b the input serie.

In order to solve the least square we use the QR decomposition.

5.4 Evaluating Forecasting Models

There exists a multitude of measures to evaluate the performance of a model. Here we introduce one measure that serves the purposes of this paper. The interested reader can get a broader view on performance evaluation in [23].

In order to measure a model's performance we use the absolute mean relative error which is defined as follows:

$$MRE = \frac{1}{T} * \sum_{t=0}^{T} abs((model_t - input_t)/(model_t + input_t)) \tag{5.13}$$

Where T represents the number of samples, $model_t$ the model's value at time t and $input_t$ the actual input value at time t. The measure is a percentage and therefore falls between zero and one.

We complement the mean relative error by its standard deviation computed as:

$$std(MRE) = \frac{1}{T} * \sum_{t=0}^{T} (abs((model_t - input_t)/(model_t + input_t)) - MRE)^2 \tag{5.14}$$

And we define a model's accuracy as:

$$Accuracy = 1 - MRE \tag{5.15}$$

5.5 Time Serie Correlation

We use the following formula to compute the correlation ($lag = 0$) and cross correlation ($lag > 0$) of two time series.

Let x and y two time series of length n. For a given lag $l \leq n$ the (cross) correlation is given by:

$$corr(l) = \frac{\sum_{t=l}^{n}[(x(t) - m_x) * (y(t-l) - m_y)]}{\sqrt{\sum_{t=l}^{n}(x(t) - m_x)^2} * \sqrt{\sum_{t=l}^{n}(y(t-l) - m_y)^2}} \tag{5.16}$$

Where m_x and m_y denote the mean of the time series x and y respectively.

Chapter 6

Model Creation: The Process

In this chapter we describe the creation of forecasting models that use Google Insight data. In order to build a Google Insight model we start by creating a traditional regression model that we refer to as the benchmark model. We then enhance the benchmark model by adding keyword elements.

For instance, AR(1) could serve as a benchmark model:

$$input_t = a_0 + a_1 * input_{t-1} \tag{6.1}$$

The Google Insight enhanced model could then be written as:

$$input_t = a_0 + a_1 * input_{t-1} + a_2 * keyword_{t-1} \tag{6.2}$$

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * weight_k) \tag{6.3}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{6.4}$$

Creating a Google Insight model requires to make four design decisions:

Structure of the benchmark model

What combination of classical model elements (constants, input elements) should constitute the benchmark model?

Selection of keywords

Which keywords should be included in the model (how to choose K'). In other words, how can one identify the keywords that have predictive power on the input time serie.

Keyword weights
How should one weight the importance of each individual keyword with respect to the other keywords?

keyword offsets
How far back in time should the keyword's data be considered?

We first investigate which structure to give to the benchmark model. We then handle the problem of choosing which Google Insight keywords to use in the model and which weights to give to each individual keyword. Finally, we look into selecting optimal keyword offsets.

6.1 Model Structure

A model structure defines the elements contained in the model. As seen in chapter 5, our models can contain three types of basis elements:

- Input elements.
- Keyword elements.
- Constants.

How should one combine these elements to create an efficient forecasting model? In this section we consider how to structure a classical regression model therefore not using keyword elements.

We assume in this paper that the problem of choosing a model structure will be solved manually and won't be automated. We believe that the problem isn't well suited for automation as creating an efficient model structure is more of an art than a science and remains highly dependent on the modeler's skills.

Defining and optimizing a model's structure is a general data mining problem. A good structure captures patterns observed in the input time serie. We don't intend to go into data mining in depth in the context of this paper. It would be out of scope. The interested reader can find information in [23] and [24].

6.2 Incorporating Keywords

The problem of selecting which keywords to incorporate in a Google Insight prediction model gets solved by identifying which keywords have an influence on the input time serie. It represents a computational challenge as there are potentially

50 million Google Insight keywords to choose from. The problem can be defined as finding a subset of keywords $K' \subset K$ that gives an optimal forecasting model.

Based on a set structure of a traditional forecasting model (see previous section 6.1) the enhanced model (i.e. the model incorporating Google Insight keywords) can be expressed in its general form as:

$$input_t = benchmark_t + a_0 * keyword_{t-1} \tag{6.5}$$

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * weight_k) \tag{6.6}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{6.7}$$

We present five methods or algorithms for identifying which keywords to incorporate in a model. In order to compare the methods we designed a ranking system based on three performance measures:

1. Model quality

   The measure estimates the accuracy of a model incorporating the keywords selected by the method. Accuracy is computed over the in-sample period as described in chapter 5.

2. Monotonicity

   A method exhibits monotonicity when the resulting model's quality improves (or remains the same) when increasing the size of the keyword universe K. In other words the more keywords a monotonic method can choose from the better will it be able to identify the relevant keywords.

3. Computational performance

   The measure estimates how much computational resource is needed to run a method.

We now present five methods to identify the relevant keywords to be included in a model. We make the assumption that the partition of data into the in-sample and out-of-sample sets is given. We assume that the in-sample set contains n data points. We also assume that the structure of the benchmark model has been defined and that the enhanced model has the general form.

We are concerned with finding a good subset $K' \subset K$ to constitute the keyword element. Beside, we assume that every keyword in K' has equal weight.

6.2.1 The Naive Method

The naive method consists in considering every subset $K' \subset K$ in turn. For each subset, the method creates and trains a model using the keywords in the subset and evaluates the resulting model in-sample. The subset giving the most accurate model gets returned as output. The naive method can be classified as a brute-force method. The pseudo-code for the method is:

Algorithm 1 The Naive Method

for all K' subset of K **do**
  Create model M:
  for $t = 1$ to n **do**
    Aggregate keywords:
    $keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} keyword_{k,t}$
    Create model:
    $M_t = Benchmark_t + a_0 * keyword_{t-1}$
  end for
  $Train(M)$
  $Accuracy = Evaluate(M)$
  if $Accuracy > BestAccuracy$ **then**
    $BestAccuracy = Accuracy$
    $Result = K'$
  end if
end for
return $Result$

By design the naive method will investigate every possible keyword combination and is therefore guaranteed to find the best keyword combination. The method allows for optimal model quality. Since the method guarantees finding the best set of keywords, monotonicity is also guaranteed. The trade-off for this naive solution comes in terms of computational resources.

The method will need to train and evaluate a model for each subset $K' \subset K$ except the subset containing no keywords. There is therefore a total of $2^{|K|} - 1$ subsets to consider. Thus we can conclude that the naive method for identifying relevant keywords has complexity $O(2^{|K|})$ which makes it unusable for practical purposes with large amount of keywords.

6.2.2 The Incremental Increase Method

Because of the high resource requirements of the naive method one needs to find a better method that can run in a scenario with limited resources. The incremental

increase method fulfills this role.

The incremental increase method works by first identifying the strong keywords: keywords that have a strong influence on the input time serie. It sorts the keyword by their strength and then attempts to combine the strongest keywords iteratively. If a keyword improves the result it gets added to the result set. The method can be classified as a greedy method. Optionally one can set a limit L on the number of keywords to select. The pseudo code for the incremental increase method can be written as:

Algorithm 2 The Monotonic Incremental Increase Method

for all keyword k in K **do**
  Create model M_k using only k:
  for $t = 1$ to n **do**
    $M_{k,t} = Benchmark_t + a_0 * k_{t-1}$
  end for
  $Train(M_k)$
  $Accuracy_k = Evaluate(M_k)$
end for
Rank keywords by accuracy
$Result = \emptyset$
for all ranked keyword k in K **do**
  if $|Result| < L$ **then**
    Create model M using keywords $Result \cup k$:
    for $t = 1$ to n **do**
      $M_t = Benchmark_t + a_0 * keyword_{t-1}$
    end for
    $Accuracy = Evaluate(M)$
    if $Accuracy > BestAccuracy$ **then**
      $BestAccuracy = Accuracy$
      $Result = Result \cup k$
    end if
  end if
end for
return $Result$

Contrary to the naive solution, the incremental increase method only analyzes a subset of all possible keyword combinations. Therefore the method is not guaranteed to identify the optimal subset. However by trying to combine the best ranked keywords, the method expects the resulting keyword set to form an acceptably good solution that might be a local minimum. For concrete empirical performance evaluation of this strategy please see chapter 7 where we present

empirical results.

As far as monotonicity goes, once the keywords have been ranked by accuracy, they only get added to the result if they improve the accuracy of the resulting model therefore enforcing monotonicity. We refer to this method as the monotonic incremental increase variant to contrast with the Top N variant (see section 6.2.3).

The huge advantage of this iterative method against the naive one lies in the number of model it has to compute. Indeed this greedy method first has to train and evaluate $|K|$ models in order to rank the keywords. It then has to compute an additional $|K|$ models in order to check whether adding a specific keyword to the result improves the resulting model's quality. When using a limit L it might be that the method compute less than $|K|$ models. We however consider the worst case. We can therefore conclude that the incremental increase method requires a total of $2 * |K|$ models to be trained and evaluate. In addition the method also requires to sort the keywords which can be performed in linear time with respect to $|K|$. Therefore the incremental increase method has linear complexity $O(|K|)$ which makes it usable in practice.

6.2.3 The Incremental Increase Top N Variant

In order to save computational time at the cost of a slight decrease in the resulting model's accuracy once keywords have been ranked by accuracy one can simply select the top N keywords without checking whether adding a keyword improves the resulting model. This variant needs to compute $|K|$ models only which is two times less than the original version of the method.

The pseudo-code for the top N variant is as follows:

Algorithm 3 The Incremental Increase Top N Variant

for all keyword k in K **do**
  Create model M_k using only k:
  for $t = 1$ to n **do**
    $M_{k,t} = Benchmark_t + a_0 * k_{t-1}$
  end for
  $Train(M_k)$
  $Accuracy_k = Evaluate(M_k)$
end for
Rank keywords by accuracy
$Result =$ top N ranked keyword
return $Result$

6.2.4 The Incremental Decrease Method

The incremental increase methods presented above work efficiently for identifying a subset out of millions of possible keywords. However one might also face a slightly different problem when creating a Google Insight prediction model. It can be that one already identified some keywords that have a strong influence on the input time serie (by intuition for instance) and is now faced with the challenge of identifying a good combination out of the reduced set of strong keywords K''.

The method works by first training and evaluating a model M incorporating all keywords in $K'' \subset K$. It then loops over all keywords $k \in K$' and computes a model M_k with the keywords $K''\backslash k$. The resulting model's accuracy allows to estimate k's contribution in M. If M_k's accuracy is less than M's accuracy then keyword k has a positive contribution in the model M. However, in the case where M_k's accuracy is less than M's accuracy we can conclude that k has a negative contribution and therefore discard k. Beside helping distinguish strong keywords from weaker keywords the contribution measure can help rank the stronger keywords. The contribution's magnitude can be used to weight keywords. We will see more on this later.

Similarly to the incremental increase methods, the incremental decrease method supports an optional limit L on the number of keywords to include in a model. The pseudo-code for the method is given in algorithm 4.

Because of keyword's contributions being computed in relation to the model including all keywords M, the incremental decrease method isn't monotonic. Indeed the bigger the initial keyword set, the noisier M. When the baseline model becomes too noisy the method won't distinguish keyword's contributions efficiently and will therefore give a lot of false negatives (keywords with positive contribution flagged as having negative contribution) and false positives (keywords with negative contribution flagged as having positive contribution) resulting in poor accuracy for the resulting model.

However, when the initial keyword set contains a limited number of influential keywords the method performs well and successfully manages to identify keywords playing a key role which results in excellent model quality.

For concrete empirical performance evaluation of this strategy please see chapter 7 where we present empirical data.

The incremental decrease method requires to train and evaluate $|K''| + 1$ models (one model for each keyword and the additional model incorporating all keywords). In order to analyze the method's performance we assume the worst case scenario where $|K''| = |K|$. The method therefore needs to compute $|K| + 1$ models. In addition, it also needs to rank the keywords which can be performed in linear time with respect to $|K|$. The incremental decrease method therefore has complexity $O(|K|)$.

Algorithm 4 The Incremental Decrease Method

Create model M using all keywords in $K'' \subset K$:
for $t = 1$ to n **do**
  $M_t = Benchmark_t + a_0 * keyword_{t-1}$
end for
$Train(M)$
$Accuracy = Evaluate(M)$
for all keyword k in K'' **do**
  Create model M_k using all keywords in K'' except k:
  for $t = 1$ to n **do**
    $M_{k,t} = Benchmark_t + a_0 * keyword_{t-1}$
  end for
  $Train(M_k)$
  $Accuracy_k = Evaluate(M_k)$
  $Contribution_k = Accuracy - Accuracy_k$
end for
Rank keywords by contribution
$Result = \emptyset$
for all Ranked keyword k in K'' **do**
  if $|Result| < L$ and $Contribution_k > 0$ **then**
    $Result = Result \cup k$
  end if
end for
return $Result$

6.2.5 The Correlation Method

We have now presented four methods for identifying keywords to use in a model out of the universe K. While one method has complexity $O(2^{|K|})$ and is therefore unusable for practical purposes the other three have complexity $O(|K|)$. Even though the complexity is linear with respect to the number of keywords, both methods still require to train and evaluate many models which is a resource-heavy operation.

The correlation method is an attempt to identify useful keywords without computing a single model. Instead of training and evaluating models the correlation method tries to identify the relevant keywords by computing the correlation between keyword time series and the input time serie. Computing correlation between series is much less resource-intensive than training and evaluating a model. The main idea behind the method comes from the intuition that a highly correlated keyword should have a strong influence on the input time serie. In or-

der to avoid look-ahead bias the correlations are computed on in-sample data. In addition we also shift the input time serie by one in order to take into account the keyword element bias $t-1$ in the general form of the enhanced model: $input_t = benchmark_t + a0 * keyword_{t-1}$

The correlation method works by first computing each keyword's (absolute) correlation with the input time serie. The keywords are then ranked by correlation. The method selects the top L keywords to form the prediction model. The pseudo-code for the correlation method is:

Algorithm 5 The Correlation Method

for all keyword k in K **do**
  $Correlation_k = Correlation(k, inputTimeSerie)$
end for
Rank keywords by correlation
$Result =$ top L ranked keyword
return $Result$

The correlation method isn't monotonic since combining keywords with high correlations to the input serie might lead to worse models than using the keywords separately. There is no direct relationship between a keyword's correlation and the resulting model's accuracy using the keyword. This fact also explains why the correlation method performance as far as model quality goes lacks in contrast to the other methods as we see in chapter 7.

However the correlation method truly shines when it comes to computational performance. Not only is the method linear with respect to $|K|$ but it only needs to compute $|K|$ correlation values instead of training and evaluating models. The correlation method has complexity $O(|K|)$.

6.3 Weighting Keywords

We have now investigated how to structure a benchmark model and we have presented the Google Insight model in its general form:

$$input_t = benchmark_t + a_0 * keyword_{t-1} \tag{6.8}$$

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * weight_k) \tag{6.9}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{6.10}$$

We have introduced five algorithms for determining which keywords should be used in the model. We now concentrate on the problem of weighting keywords. The problem of weighting keywords involves finding the weights $weight_k$ so that the resulting model is optimal in terms of accuracy.

The weights allow to give more relative importance to certain keywords.

We experienced a tendency of over-fitting when trying to find good keyword weights. Although weighting keywords can improve a model's performance in-sample, the over-fitting causes sharp declines in performance out-of-sample. The most robust and consistent weighting strategy consists of weighting keywords equally.

However we have found a strategy for weighting keywords efficiently when using the incremental decrease method for identifying relevant keywords. As part of this method we compute each keyword's contribution. The contributions can be used as weights to consistently improve the resulting model.

The formula for weighting keywords would be:

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * \frac{1}{|K'|} * Contribution_k) \tag{6.11}$$

6.4 Choosing Keyword Offsets

Choosing keyword offsets implies modifying the general form of the Google Insight prediction model.

The general form is:

$$input_t = benchmark_t + a_0 * keyword_{t-1} \tag{6.12}$$

The keyword element $keyword_{t-1}$ has default offset $t - 1$. However there might be better offsets to use, for example:

$$input_t = benchmark_t + a_0 * keyword_{t-2} \tag{6.13}$$

$$input_t = benchmark_t + a_0 * keyword_{t-15} \tag{6.14}$$

In addition one can also improve the general form by using several keyword elements (with different offsets) instead of a single one:

$$input_t = benchmark_t + a_0 * keyword_{t-1} + a_1 * keyword_{t-2} + a_2 * keyword_{t-3} \tag{6.15}$$

The problem of choosing keyword offsets consists of choosing how many keyword elements to use and with which offsets in order to maximize the resulting model's accuracy.

In the scope of this paper we assume that offsets bigger than a few periods won't be relevant. We therefore limit the set of possible combinations of keyword elements and offsets. Please note that it doesn't make sense to create a model with two distinct keyword elements that have the same offset. Thus, by limiting the number of offsets that we consider we also limit the number of different keyword elements. Please also note that offset zero and positive offsets $(t, t+1, t+2, ...)$ introduce look-ahead bias and are therefore not considered. With a maximum offset limited to a constant c, there are 2^c different keyword element combinations. We suggest to train and evaluate a model for every combination as the process can be conducted in constant time. The best combination would then be selected to form the prediction model.

Chapter 7

Model Creation: Experiment Part I

In this chapter we present the first part of an experiment to illustrate the process of creating Google Insight models. We go through all the design decisions introduced in chapter 6. The first part of the experiment investigates the performance of the five keyword identification methods.

7.1 Data

7.1.1 Input Data

We use three different time series as input data.

For the first input serie we use USA initial unemployment claims data. The data tracks the weekly number of initial claims for unemployment benefits in the USA. The numbers are reported weekly by the US Department of Labor [1]. Although initial claims data can be obtained back to 1987 we limit ourselves to the period 2004-2010 because Google Insight data goes back to 2004. The first input data therefore contains 349 data points (i.e. weeks) that range from 17th January 2004 to 18th September 2010.

As can be seen in figure 7.1 initial unemployment claims tend to peak at the beginning of each year. In order to simplify our analysis we remove the seasonal trend by considering the seasonally-adjusted time serie shown in figure 7.2.

In order to facilitate the computation of prediction models we normalize the data so that it falls between 0 and 100.

For the second input serie we use the number of arrivals in the Republic of Seychelles. The data can be obtained from the National Bureau of Statistics of the

[1] Initial unemployment claims data source: US Department of Labor
Historical Data: UI weekly claims

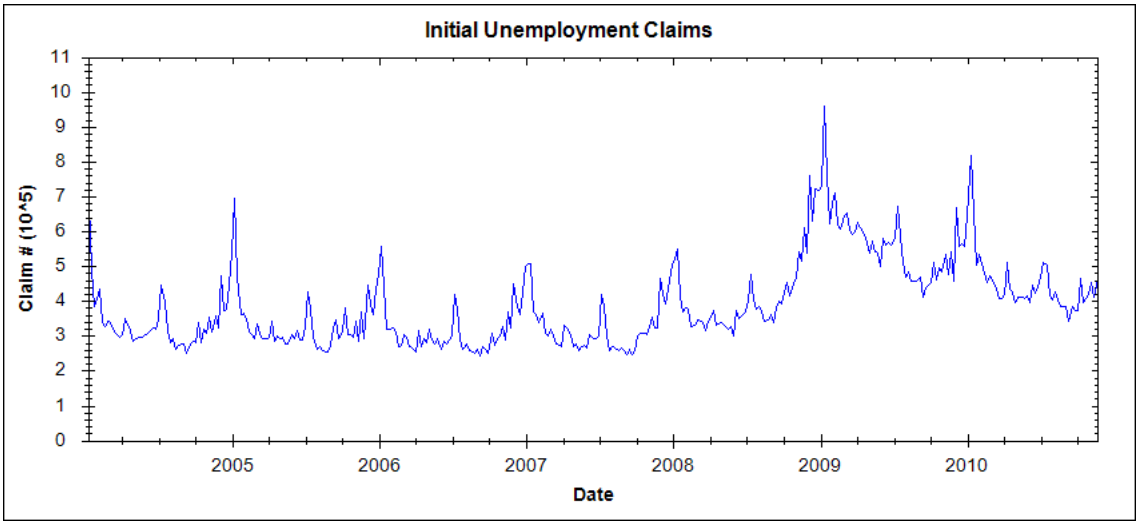

Figure 7.1: Initial Unemployment Claims in the USA

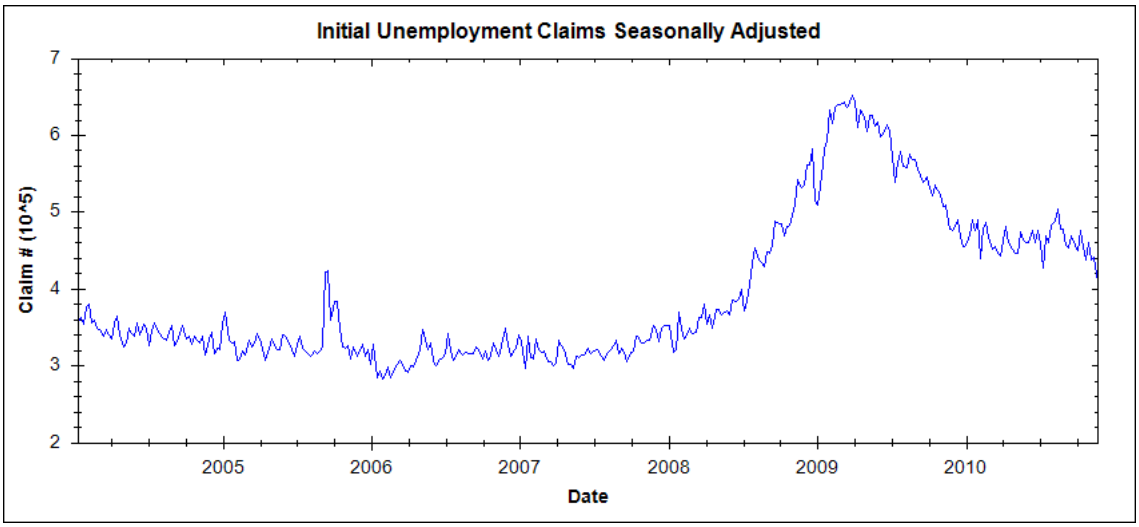

Figure 7.2: Seasonally-adjusted Initial Unemployment Claims in the USA

Republic of Seychelles [2]. It contains monthly data points. The second input data therefore contains 79 points (i.e. months) that range from January 2004 to July 2010. We show the time serie in figure 7.3.

The second input serie has a cyclical component. Contrary to the first input serie we decide to keep the seasonal trend in order to see how the methods perform

[2]Tourism statistics

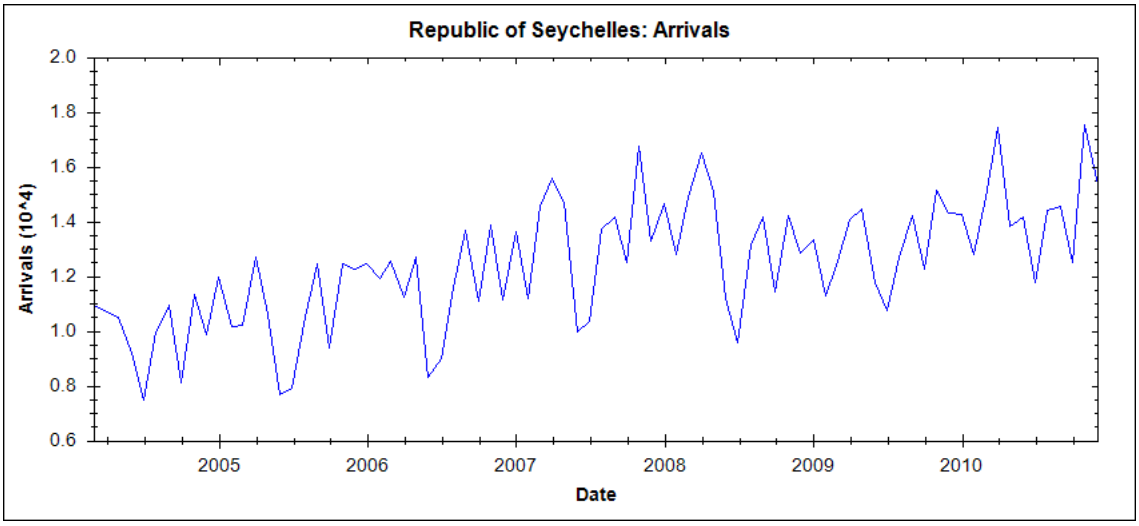

Figure 7.3: Arrivals in the Republic of Seychelles

on data with seasonal trends.

As with the initial unemployment claims we normalize the arrivals data to fall between 0 and 100.

For the third time serie we use the VIX. We obtain data from Yahoo! Finance[3]. The data consists of 349 weeks ranging from 17th January 2004 to 18th September 2010 as shown in figure 7.4.

We also normalize VIX data so that the values fall between 0 and 100.

7.1.2 Keyword Data

In this experiment we use 778 keywords. The keywords consist of 768 common English words and 10 specifically selected words relating to finance, economics and tourism in the Republic of Seychelles. Please see appendix A for the complete list of keywords. The specifically selected keywords are displayed in table 7.1.

We are interested in seeing whether the specifically selected keywords have a predominant influence on the input data as could be expected from intuition. And we also want to test whether the automated methods for selecting keywords are able to identify the specifically selected keywords out of all keywords.

For each keyword we download the corresponding weekly time serie from Google Insight. The keyword series are then cut to the length of the input series. They are also synchronized with the input serie so that data points for all series occur on the same days.

[3]VIX

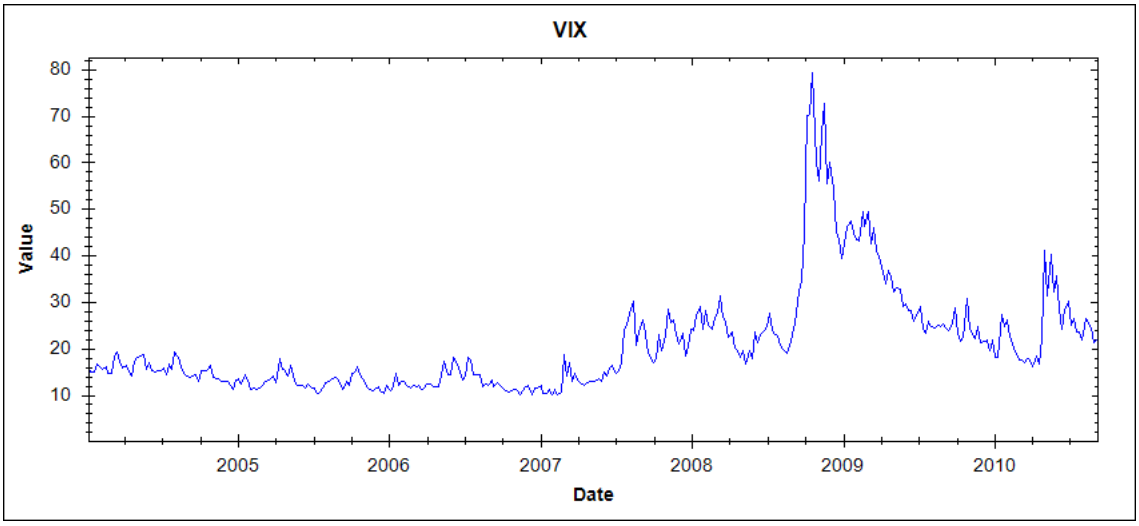

Figure 7.4: VIX implied volatility index

Keywords
crisis
recession
stock crash
stock market
buy gold
buy stocks
sell stocks
unemployment
unemployment benefits
seychelles hotels

Table 7.1: Specifically selected keywords

For the Republic of Seychelles the weekly keyword series are turned into monthly series. To do so, for each month we average the weekly data points contained in that month.

7.1.3 In-Sample Parition

We define 70% of the input data to be in-sample. In the case of unemployment claims and VIX we therefore have 244 data points to train our models on and 105 data points to test them out-of-sample. The in-sample data set goes up to October

2008. We therefore have the end-of-2005 peak and the beginning of the 2008-2009 peak in-sample. This choice allows for the out-of-sample data to contain all three important unemployment and volatility regimes: a period of claim/volatility rise, one of claim/volatility decline and one of claim/volatility stagnation.

Concerning the arrivals data we have 60 points in-sample and 27 points out-of-sample. The in-sample data goes up to December 2008.

7.1.4 Benchmark Structure

We decide to use an auto-regressive model for all three input series. We choose the ARCH(2) model as benchmark as it allows the different methods to pick relevant keywords efficiently in our experience.

The ARCH(2) model can be written as:

$$input_t = a_0 + a_1 * input_{t-1} + a_2 * input_{t-2} \tag{7.1}$$

In order to run the methods for identifying relevant keywords we enhance the benchmark model by adding a keyword element:

$$input_t = a_0 + a_1 * input_{t-1} + a_2 * input_{t-2} + a_3 * keyword_{t-1} \tag{7.2}$$

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * weight_k) \tag{7.3}$$

$$\sum_{k=1}^{|K'|} weight_k = 1 \tag{7.4}$$

7.2 Keyword Selection

We run the different keyword selection methods in order to compare their performances in terms of the resulting model's accuracy.

7.2.1 Initial Unemployment Claims

Incremental Increase Methods

We run the incremental increase methods on the 778 keywords. We decide to use a limit $L = 5$ on the selected keywords. We run both variants of the method.

The top 5 keywords having the most predictive power according to the methods are depicted in table 7.2.

Top 5 Keywords
unemployment benefits
recession
buy gold
spring
buy stocks

Table 7.2: Best keywords as selected by the incremental increase methods

It is interesting to note that although hidden in ”noisy” keywords (i.e. common language words), the method successfully identifies 4 out of 5 keywords relating to finance and economics. In particular the keyword ”unemployment benefits” comes at the top of the list which confirms our intuition of its predictive power.

The monotonic version selects a keyword only when using the keyword improves the performance of the final model. In this example the combination ”unemployment benefits” and ”recession” gives the best performance and no additional keyword improves the final result. The monotonic method therefore selects ”unemployment benefits” and ”recession” as output keywords.

Figure 7.5 shows how the aggregated two keywords relate to initial unemployment claims.

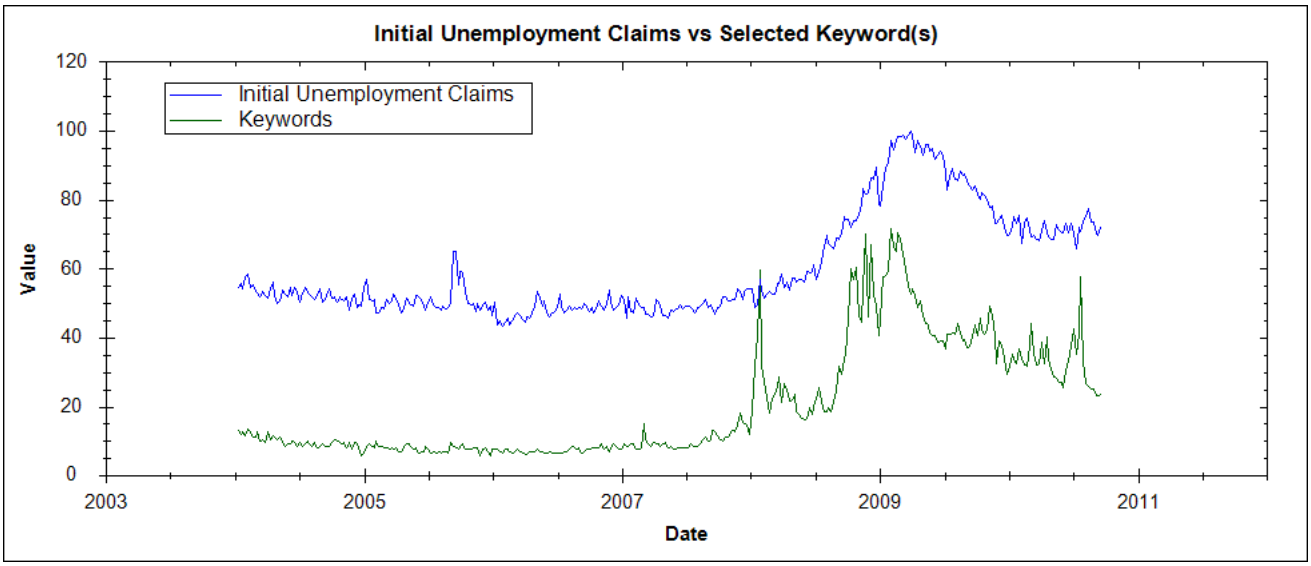

Figure 7.5: Keywords selected by the monotonic variant

We expect the resulting model’s performance to be excellent with the mono-

tonic variant method. Indeed, the performance should be better than for the incremental decrease method and the correlation method at the price of requiring more resources.

The performance is represented as the resulting model's accuracy over the whole in-sample period and its standard deviation. In the scope of the first part of this example we are not interested in a model's performance against a benchmark. We are only concerned with the relative performance of the various methods. We show the performance for the monotonic incremental increase method in table 7.3

Performance	
Accuracy	98.12%
Standard Deviation	1.5%

Table 7.3: Performance of the monotonic incremental increase method

In contrast to the monotonic version, the top N variant method selects the top 5 keywords. Figure 7.6 shows how the aggregated keywords relate to the input serie.

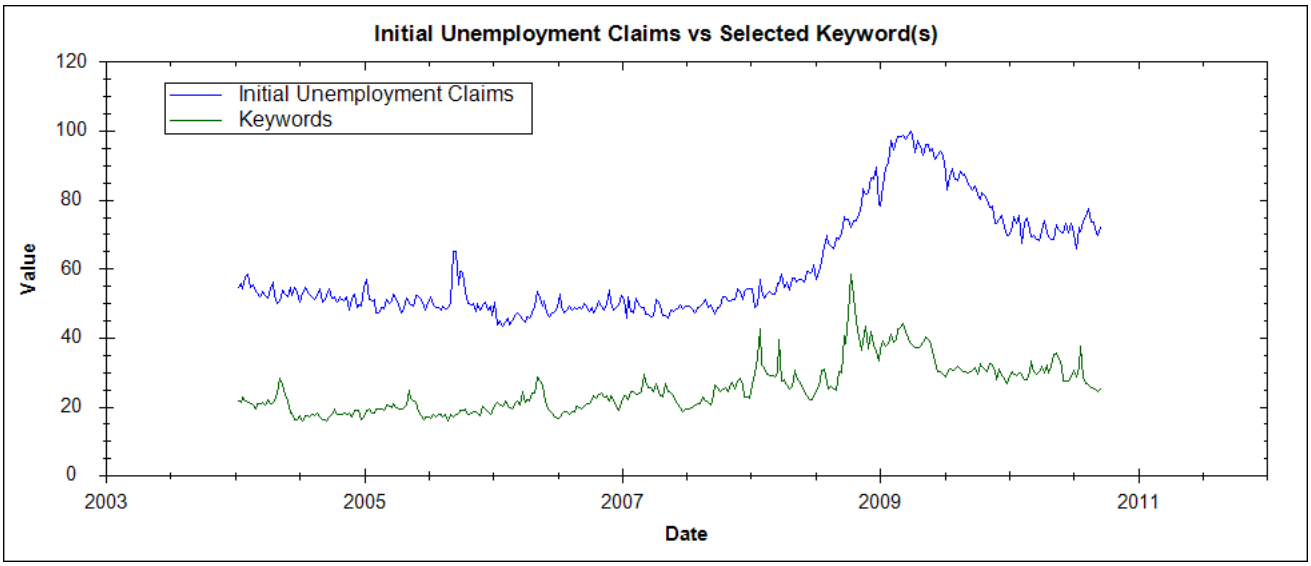

Figure 7.6: Keywords selected by the top N variant

We expect the top N variant to perform slightly worse than the monotonic version as the variant doesn't check whether adding a keyword improves the resulting model's accuracy. Performance for the top N variant is presented in table 7.4.

Performance	
Accuracy	97.88%
Standard Deviation	1.9%

Table 7.4: Performance of the top N incremental increase method

As expected the variant performs slightly worse than the monotonic version. The variant also requires less computation time.

Incremental Decrease Method

We run the incremental decrease method with a limit $L = 5$ on the number of keywords. In addition to identifying relevant keywords the method also allows to rank the keywords according to their relevance by assigning keyword weights. The selected keywords and their weights are shown in table 7.5.

Keyword	Weight
recession	0.236
stock crash	0.204
unemployment benefits	0.2
buy gold	0.188
earth	0.174

Table 7.5: Best keywords as selected by the incremental decrease method

As with the incremental increase methods the incremental decrease method manages to identify four keywords related to finance and economics. However due to the lack of monotonicity for this method and the additional noise when aggregating many keywords we expect poorer performance when increasing the size of the keyword universe. Nonetheless, in this example the method manages to hold ground with the incremental increase methods in terms of performance as shown in table 7.6.

Performance	
Accuracy	97.9%
Standard Deviation	1.89%

Table 7.6: Performance of the incremental decrease method

Correlation Method

As opposed to the methods illustrated above, the correlation algorithm can be run without computing a single prediction model which allows for big gains in terms of resource performance. However the performance in terms of accuracy might not suffice for practical purposes. We run the method with a limit $L = 5$ on the number of keywords to identify. The selected keywords are presented in table 7.7.

Top 5 Keywords
fertile
price
delicate
ray
harbour

Table 7.7: Best keywords as selected by the correlation method

Unfortunately the correlation method fails at identifying keywords related to finance and economics. It seems to have fallen prey to "noisy" keywords. The method's performance shown in table 7.8 is disappointing compared to the other methods.

Performance	
Accuracy	97.68%
Standard Deviation	2.04%

Table 7.8: Performance of the correlation method

The poor performance of the correlation method seems to indicate that high correlation between a keyword time serie and an input serie doesn't necessarily leads to a successful model. It can be explained in part by the fact that the correlation between keywords and input series are non-stationary. A keyword might have a strong positive correlation with unemployment claims at a specific time period and a strong negative correlation at another time period.

One could extend the correlation method by computing a rolling correlation (instead of a correlation over the whole in-sample period) and keep keywords that show strong and robust correlations over the different rolling windows. We believe it would increase the method's performance at the cost of longer computation times. However we don't investigate this point in the scope of this paper.

7.2.2 Arrivals in the Republic of Seychelles

Incremental Increase Methods

We run the incremental increase methods on the 778 keywords in our universe in order to identify the ones that have predictive power for the Seychelles arrivals time serie. We decide to run the method with a limit $L = 5$ on the number of keywords. We run both variants of the method.

The top 5 keywords having the most predictive power according to the incremental increase methods are depicted in table 7.9.

Top 5 Keywords
snow
sell stocks
weather
seychelles hotels
winter

Table 7.9: Top 5 keywords according to the incremental increase method

The methods manage to identify the keyword related to tourism in the Republic of Seychelles. Also the method ranks three keywords related to the winter period (”winter”, ”weather” and ”snow”) in the top five keywords. It can probably be explained by the fact that surges in the arrivals in Seychelles occur during winter time. Please note however that we are not concerned with explaining why a particular keyword gives good results. We are mostly interested in identifying in a non-discriminatory manner which keywords perform well.

The monotonic version of the incremental increase method starts with a model containing ”snow” as only keyword and adds additional keywords only when they improve the final result.. No other keyword improves the result and the monotonic version returns ”snow” as output.

Figure 7.7 shows how the keyword relates to the arrivals serie.

We report in table 7.10 the performance of the monotonic method on Seychelles data.

Performance	
Accuracy	91.62%
Standard Deviation	7.96%

Table 7.10: Performance of the monotonic incremental increase method

The top N variant selects all 5 keywords in table 7.9 as output. Figure 7.8 shows how the five aggregated keyword look in relation to the input serie.

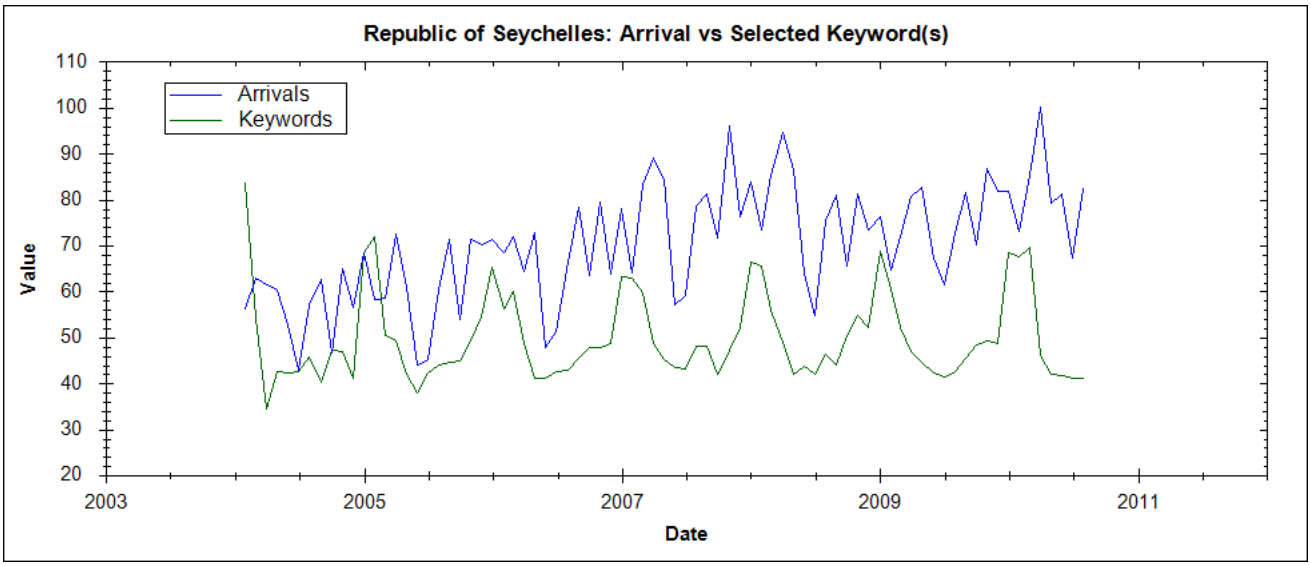

Figure 7.7: Keyword selected by the monotonic version of the incremental increase method

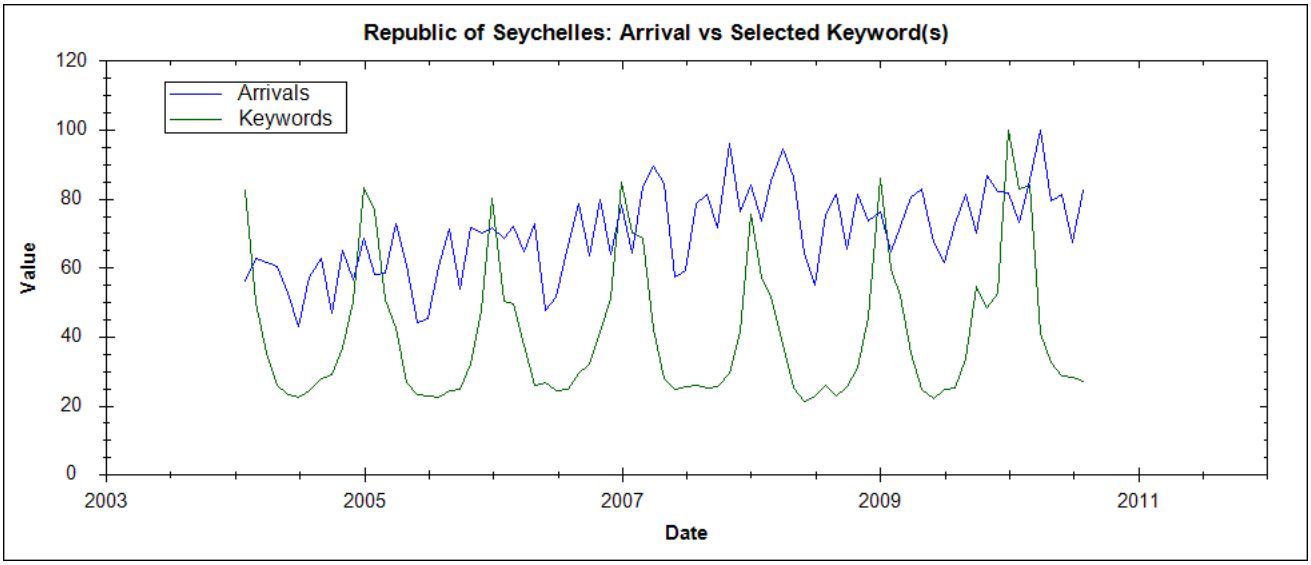

Figure 7.8: Keyword selected by the top N variant of the incremental increase method

The performance of the top N variant presented in table 7.11 confirms our expectation that the method performs slightly worse than the monotonic version.

The efficient keyword ranking of the incremental increase methods nonetheless allows the top N variant to perform well with respect to the other methods.

Performance	
Accuracy	91.55%
Standard Deviation	8%

Table 7.11: Performance of the top N incremental increase method

Incremental Decrease Method

We run the incremental decrease method with a limit $L = 5$ on the number of keywords. In addition to identifying relevant keywords the method also allows to rank the keywords according to their relevance by assigning keyword weights. The selected keywords and their weights for the arrivals in Seychelles are shown in table 7.12. And the aggregated weighted keywords can be seen in relation with the input serie in figure 7.9. One can observe how the aggregated keywords tend to lead the input time serie.

Keyword	Weight
snow	0.234
sell stocks	0.218
weather	0.192
seychelles hotels	0.18
unemployment benefits	0.176

Table 7.12: Best keywords as selected by the incremental decrease method

The selected keywords confirm our results obtained with the initial unemployment claims. Indeed the incremental decrease method is able to find relevant keywords similar to the ones obtained with the incremental increase methods out of a relative large pool of 778 candidates. Also due to the weighting mechanism the method actually achieves better performances than the top N incremental increase on the arrivals serie as shown in table 7.13. Although unexpected this good performance shows that the incremental decrease method is potentially excellent on a limited universe of keywords. In our experience, the smaller the universe the better the performance. We however expect a decrease in performance relative to the incremental increase methods for a very large universe with millions of keywords.

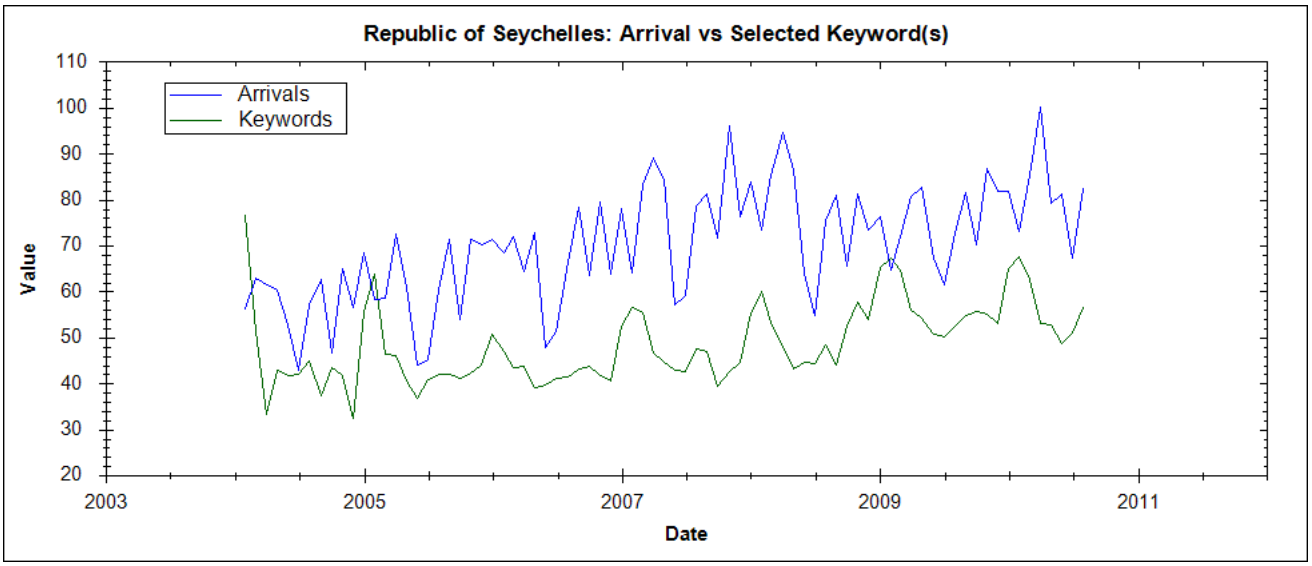

Figure 7.9: Keyword selected by the incremental decrease method

Performance	
Accuracy	91.59%
Standard Deviation	7.98%

Table 7.13: Performance of the incremental decrease method

Correlation Method

We run the correlation method with a limit $L = 5$ on the number of keywords to identify. The selected keywords are presented in table 7.14.

Top 5 Keywords
even
dark
fowl
head
frame

Table 7.14: Best keywords as selected by the correlation method

The correlation method fails to identify the ”seychelles hotels” keyword. It also fails to find the keywords related to the winter season that the other methods managed to identify. This disappointing results confirm our expectations and our

results obtained with the initial unemployment claims. The aggregated top five keywords according to the correlation method can be seen in figure 7.10. The keywords fail to correlate with the peaks and valleys of the arrivals serie and seem to be very noisy which leads us to expect bad performances for the method.

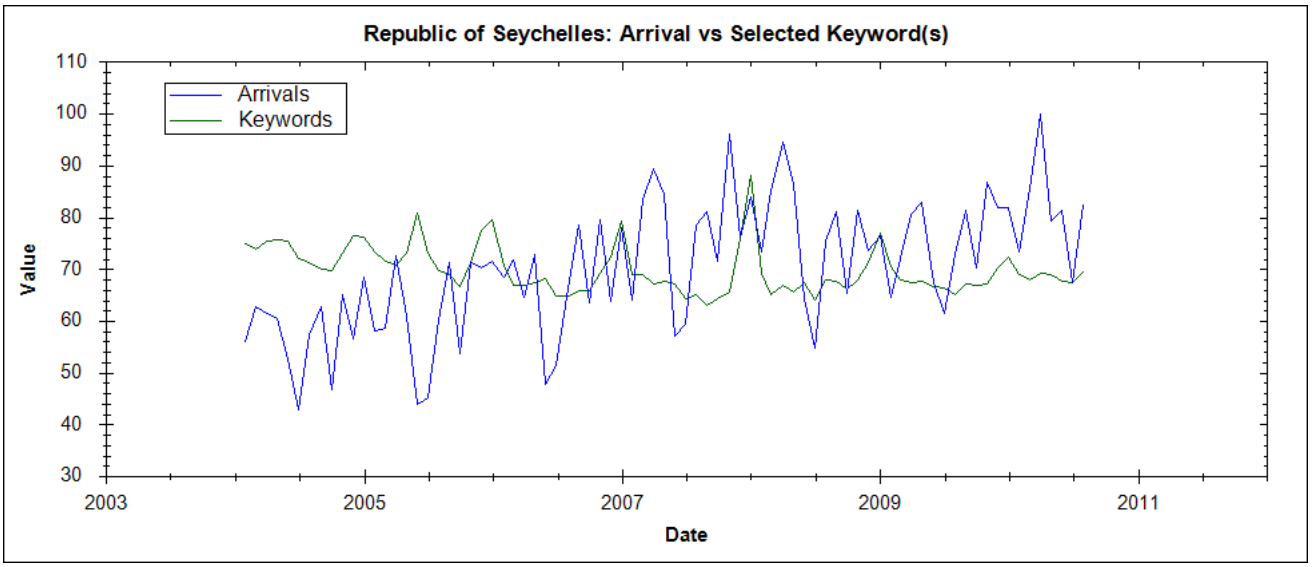

Figure 7.10: Keywords selected by the correlation method

Indeed, the correlation method comes out as the worse performer as with the unemployment serie. The performance can be seen in table 7.15.

Performance	
Accuracy	91.31%
Standard Deviation	8.2%

Table 7.15: Performance of the correlation method

7.2.3 Implied Volatility

We now run the methods for identifying keywords on VIX data.

Incremental Increase Method

We run the incremental increase methods on the 778 keywords in order to identify the ones that have predictive power on the VIX time serie. We decide to run the

methods with a limit $L = 5$, which means that we want to identify a maximum of 5 keywords. We run both variants.

The top 5 keywords having the most predictive power according to the method are depicted in table 7.16.

Top 5 Keywords
recession
stock crash
unemployment benefits
buy gold
earth

Table 7.16: Top 5 keywords according to the incremental increase method

The method successfully manages to identify the keywords related to finance.

The monotonic version of the incremental increase method starts with a model containing "recession" as only keyword and adds additional keywords only when they improve the final result. No other keyword improves the result and the monotonic incremental increase returns the keyword "recession" as output.

Figure 7.11 shows how the keyword relates to the VIX time serie.

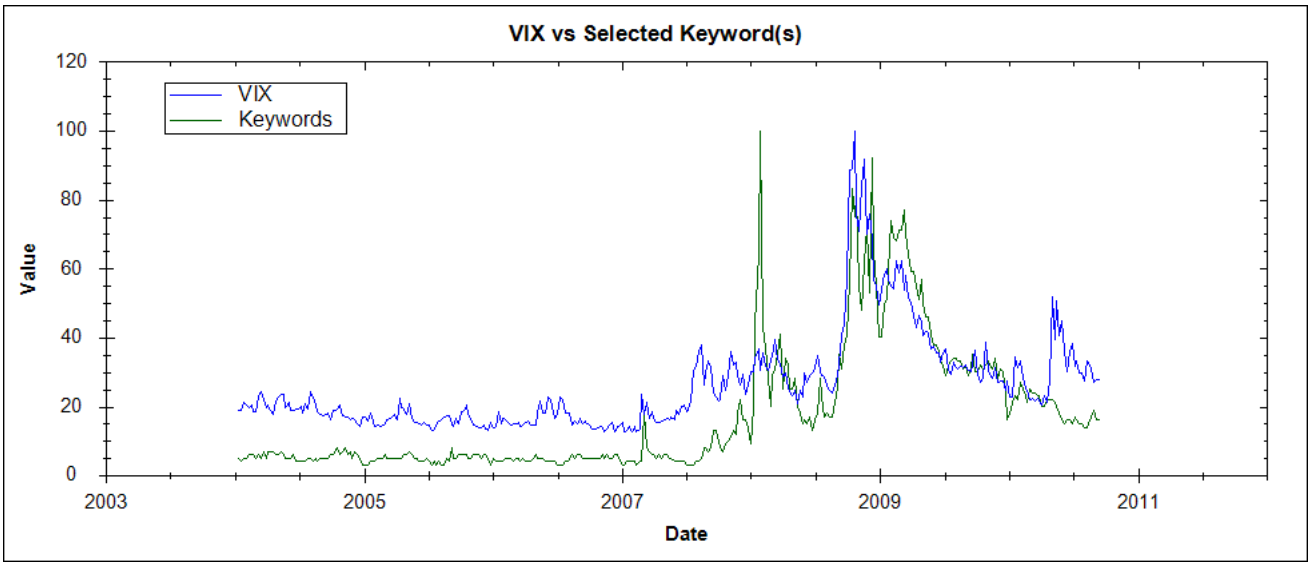

Figure 7.11: Keyword selected by the monotonic version of the incremental increase method

We report in table 7.17 the performance of the monotonic method on VIX

data.

Performance	
Accuracy	93.29%
Standard Deviation	4.56%

Table 7.17: Performance of the monotonic incremental increase method

The top N variant of the method selects all top 5 keywords as output. Figure 7.12 shows how the five aggregated keywords look in relation to the VIX.

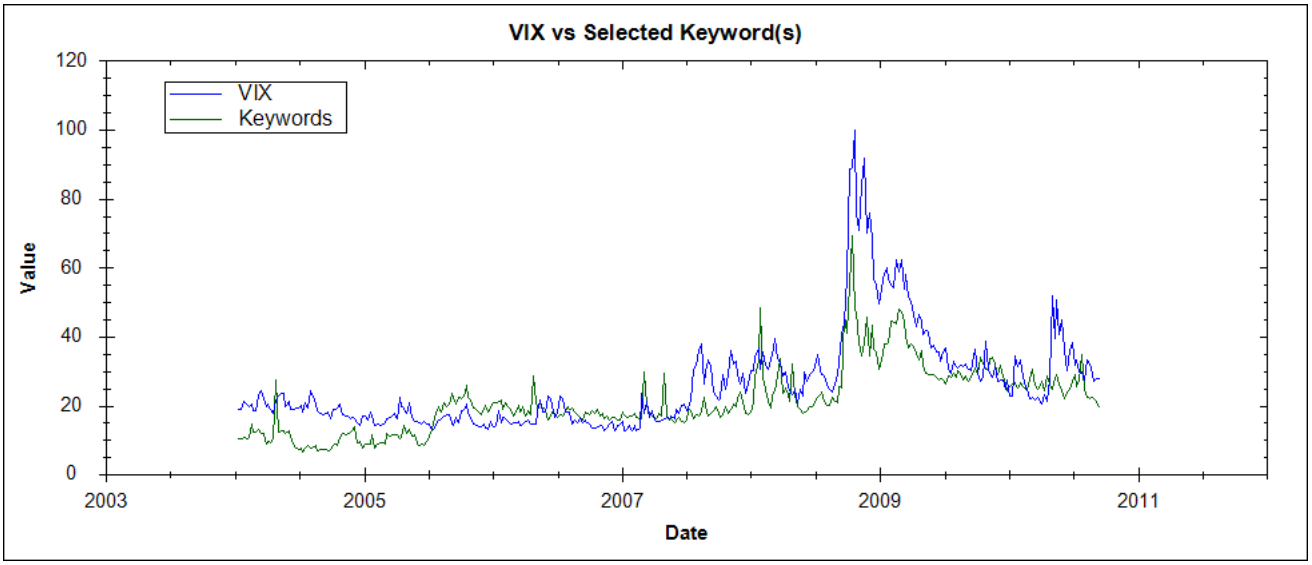

Figure 7.12: Keyword selected by the top N variant of the incremental increase method

The top N variant performance presented in table 7.18 shows as expected slightly worse performance than the monotonic version. Nonetheless the top N variant performs well with respect to the other methods.

Performance	
Accuracy	92.57%
Standard Deviation	4.94%

Table 7.18: Performance of the top N incremental increase method

Incremental Decrease Method

We run the incremental decrease method with a limit $L = 5$ on the number of keywords. In addition to identifying relevant keywords the method also allows to rank the keywords according to their relevance by assigning keyword weights. The selected keywords and their weights for the VIX are shown in table 7.19. And the aggregated weighted keywords can be seen in relation with the input serie in figure 7.13.

Keyword	Weight
sell stocks	0.288
yahoo	0.2
tired	0.18
cry	0.168
bite	0.166

Table 7.19: Best keywords as selected by the incremental decrease method

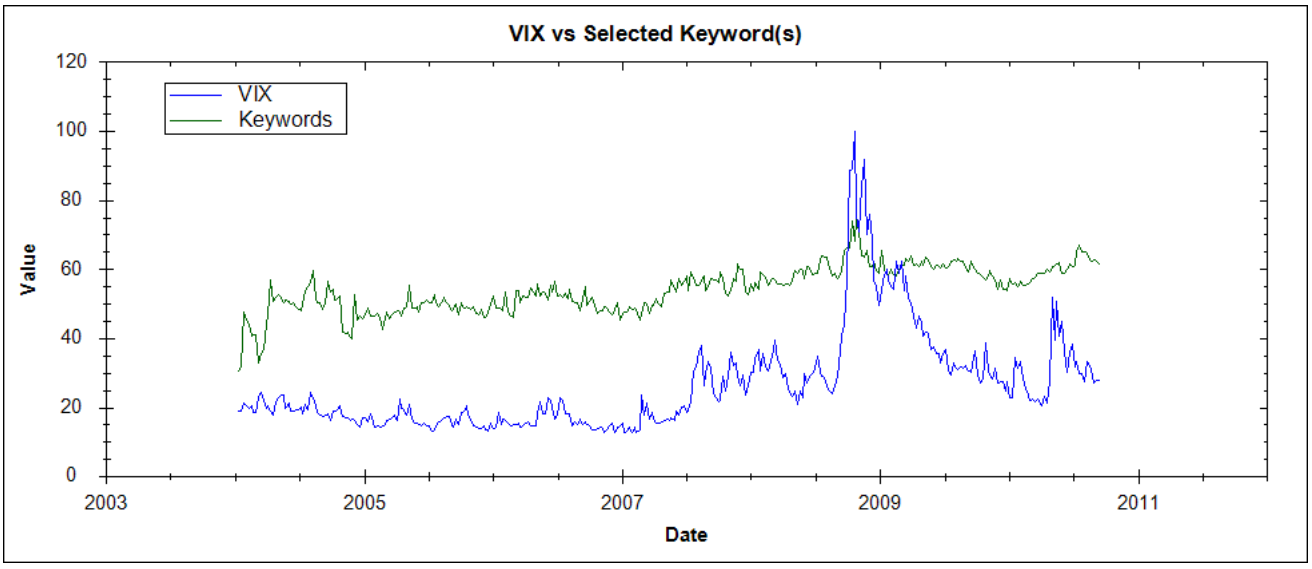

Figure 7.13: Keyword selected by the incremental decrease method

The selected keywords give a result slightly different than with unemployment and arrivals in Seychelles. Indeed with the VIX the method seems to perform less favorably. It still manages to identify "sell stocks" as relevant keyword but the four other selected keywords have no intuitive relevance to the VIX. The performance

shown in table 7.20 confirms the initial analysis. Contrarily to the other two input time series for the VIX 778 keywords is too many for the incremental decrease method to perform well.

Performance	
Accuracy	89.9%
Standard Deviation	7.13%

Table 7.20: Performance of the incremental decrease method

Correlation Method

We run the correlation method with a limit $L = 5$ on the number of keywords. The identified keywords are presented in table 7.21.

Top 5 Keywords
same
delicate
neck
room
bed

Table 7.21: Best keywords as selected by the correlation method

As with the other input time series, the correlation method performs poorly and identifies no keyword related to finance. The aggregated top five keywords according to the correlation method can be seen in figure 7.14. The keywords fail to correlate with the peaks and valleys of the VIX and seem to be very noisy.

Indeed, the correlation method comes out as the worse performer as with the two other input series. The performance can be seen in table 7.22.

Performance	
Accuracy	89.86%
Standard Deviation	8.73%

Table 7.22: Performance of the correlation method

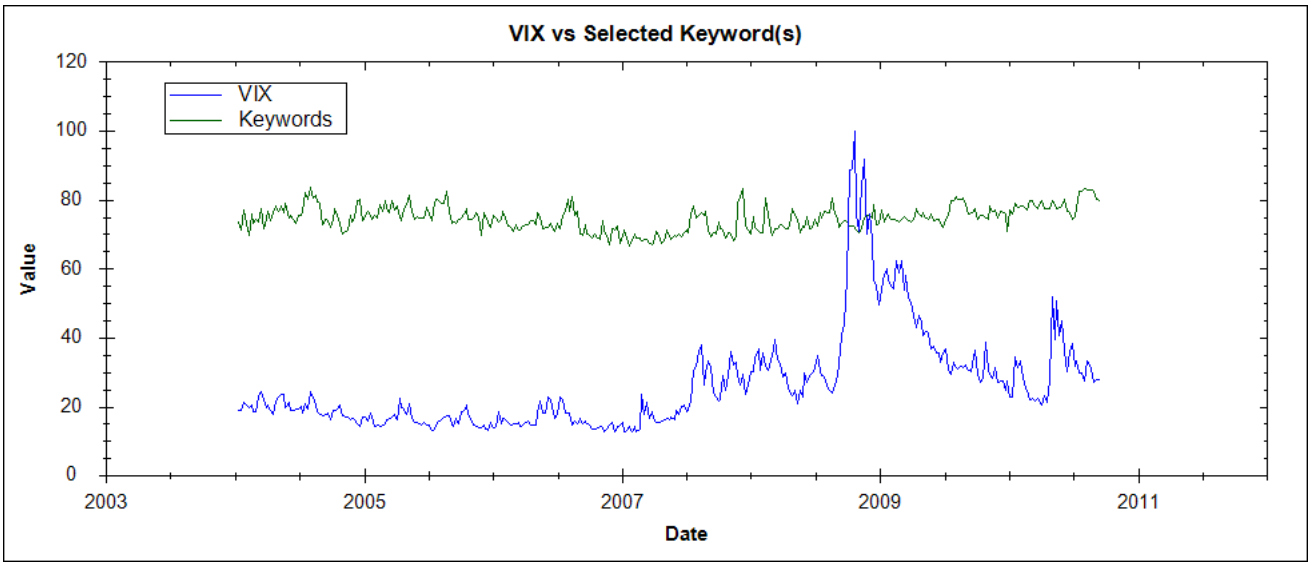

Figure 7.14: Keywords selected by the correlation method

Chapter 8

Model Creation: Experiment Part II

In the second part of the example we investigate performances of Google Insight models against a benchmark model.

8.1 Data

8.1.1 Input Data

We use the VIX as input data. Our data consists of 349 weekly data points ranging from 17th January 2004 to 18th September 2010.

8.1.2 Keyword Data

We use the same 778 keywords as for the first part of the experiment.

8.1.3 In-Sample Partition

As with the first part of the experiment we train our models with 70% of the data points in-sample. We then also test our models with 80% and 90% of the points in-sample in order to test for consistency with regard to the in-sample partitioning.

8.2 Benchmark

We use AR(1) model as benchmark model:

$$input_t = a_0 + a_1 * input_{t-1} \tag{8.1}$$

AR(1) gives good results when modeling volatility because of high auto-correlation in the data. The model is often used for modeling volatility in the literature. Our

challenge in this part of the example is to beat the AR(1) benchmark with a Google Insight model in order to show that by enhancing AR(1) with keywords one can improve its forecasts. We are not interested in competing with other models forecasting volatility. Our goal is to investigate the effect of the keywords.

8.3 Results: Benchmark

We report performances for the AR(1) benchmark model with 70% of the points in-sample in table 8.1. The performance is represented as the model's accuracy out-of-sample as well as its standard deviation. In addition we also show the distribution of the model's error in figure 8.1. The distribution allows us to see how large errors tend to cluster together due to the memory effect of volatility. We can also see how the error distribution for volatility has much fatter tails than Gaussian distribution. In particular we can identify multiple sigma 6+ events which should be much less probable with Gaussian distribution.

Benchmark Performance 70% In-Sample	
Accuracy	93.14%
Standard Deviation	6.51%

Table 8.1: Performance of the benchmark model on VIX data with 70% of the data in-sample

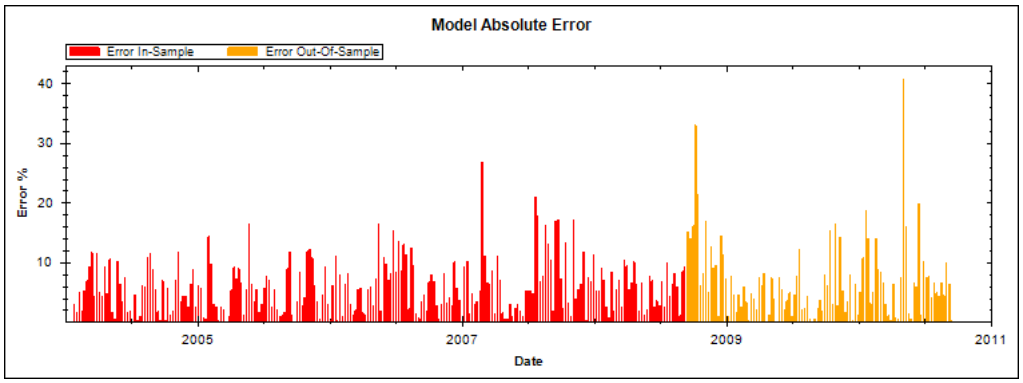

Figure 8.1: Error of the benchmark model on VIX data with 70% of the data in-sample

In order to test Google Insight model's ability to beat the AR(1) benchmark reliably we need to conduct consistency testing for different in-sample partition-

ing. We therefore also report the benchmark's performance with 80% and 90% of the points in sample in table 8.2.

Benchmark Performance 80% In-Sample	
Accuracy	93.7%
Standard Deviation	6.3%
Benchmark Performance 90% In-Sample	
Accuracy	92.43%
Standard Deviation	7.58%

Table 8.2: Performance of the benchmark model on VIX data with 80% and 90% of the data in-sample

8.4 Results: Machine-Created Model with Automatic Keyword and Offset Selection

In this section we inspect the performances of a Google Insight model generated by using the incremental increase method for selecting keywords and by computing the best keyword offsets as explained in chapter 6. In the first part of this experiment we showed that the monotonic incremental increase method has the best and most reliable performance. We therefore use that method for computing our machine-created model. The keywords are weighted equally.

As for the first part of the experiment we run the incremental increase method on the 778 keyword universe with ARCH(2) as traditional model. For VIX data the method selects "recession" as relevant keyword.

We then compute the best keyword offset which is $t - 1$ for all three series. We extend the AR(1) model by adding a keyword element with the corresponding offset:

$$input_t = a_0 + a_1 * input_{t-1} + a_2 * keyword_{t-1} \tag{8.2}$$

The results for this model for all three in-sample partitioning options are shown in table 8.3. Figure 8.2 shows the model's error when using 70% of the data in-sample.

The results are disappointing as the Google Insight model barely succeeds at beating the AR(1) model. Although the keyword-aware model beats the benchmark model in-sample by a healthy margin it fails to do so out-of-sample. We

Incremental Increase Model Performance 70% In-Sample	
Accuracy	93.14%
Standard Deviation	6.5%
Incremental Increase Model Performance 80% In-Sample	
Accuracy	93.71%
Standard Deviation	6.29%
Incremental Increase Model Performance 90% In-Sample	
Accuracy	92.45%
Standard Deviation	7.58%

Table 8.3: Performance of the machine-created model on VIX data with 70%, 80% and 90% of the data in-sample

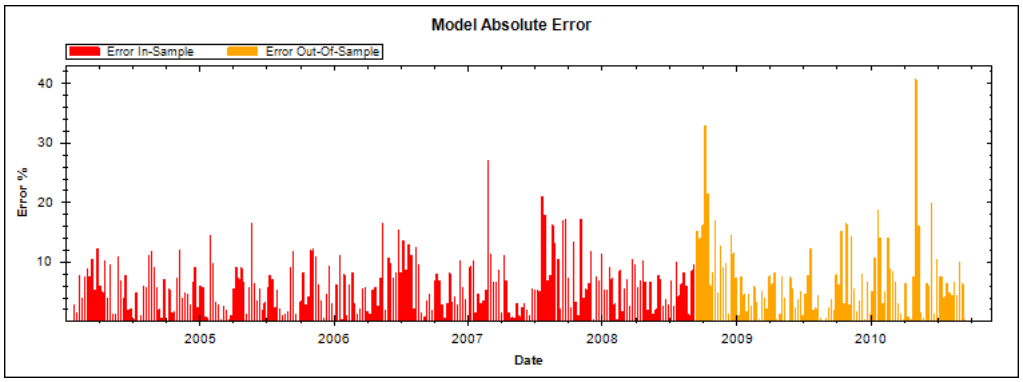

Figure 8.2: Error of the machine-created model on VIX data with 70% of the data in-sample

believe the cause might be over-fitting of the in-sample data. We will see more on this later.

Although the results obtained by the machine-created model are disappointing, one should not dismiss the various methods for identifying relevant keywords. One needs to consider that these forecasting problems remain very challenging and especially when it comes to stock-market volatility. The methods for selecting keywords might give better results on different time series. The automated identification of keywords can also be useful when one has no a-priori knowledge or intuition of the keywords that will make a model successful. They can also help uncover keywords or groups of keywords that have a non-obvious influence on the input serie. We also believe that a machine-generated model, even if it gives satis-

factory results, can still be outperformed by a carefully crafted manually designed model.

8.5 Result: Model with Manual Keyword and Offset Selection

As we have seen in previous sections, the major challenge when trying to forecast volatility lies in over-fitting the in-sample data. Our machine-created models tend to over-fit the input data and give poor performance out-of-sample. We now design a model manually whose main goal is to tackle the problem of over-fitting.

We start by selecting the keywords for our model.

We select the following keywords as they have obvious intuitive relevance to stock-market volatility:

- crisis
- buy gold
- buy stock
- recession
- stock crash
- stock market

Contrarily to the automatic selection of keywords we intentionally select a larger set of keywords in order to limit the chances of over-fitting. For the same reason we decide to weight the keywords equally.

For the selection keyword offsets we decide to limit ourself to a maximum offset of $t - 5$. Due to initially strong and then fading over time auto-correlation in volatility we suspect that going further in time adds little value. We know from the previous section that offset $t - 1$ performs favorably in-sample. However we believe that using only one keyword element is too restrictive and prone to over-fitting. And since we do not know which keyword elements will give better results and don't want to find out by over-fitting training data we decide to use all five elements to form the following model:

$$\begin{aligned} input_t = \quad & a_0 + a_1 * input_{t-1} + a_2 * keyword_{t-1} + a_3 * keyword_{t-2} \\ & + a_4 * keyword_{t-3} + a_5 * keyword_{t-4} + a_6 * keyword_{t-5} \end{aligned}$$

$$keyword_t = \frac{1}{|K'|} * \sum_{k=1}^{|K'|} (keyword_{k,t} * weight_k) \tag{8.3}$$

$$weight_k = 1/|K'| \tag{8.4}$$

We show the model's performance for 70%, 80% and 90% of the data in-sample in table 8.4. We also show the distribution of the model's error in figure 8.3. The error spike at the beginning of the test period occurs because of initialization.

Incremental Increase Model Performance 70% In-Sample	
Accuracy	93.64%
Standard Deviation	6.75%
Incremental Increase Model Performance 80% In-Sample	
Accuracy	94.32%
Standard Deviation	6.38%
Incremental Increase Model Performance 90% In-Sample	
Accuracy	93.17%
Standard Deviation	7.93%

Table 8.4: Performance of the manual model on VIX data with 70%, 80% and 90% of the data in-sample

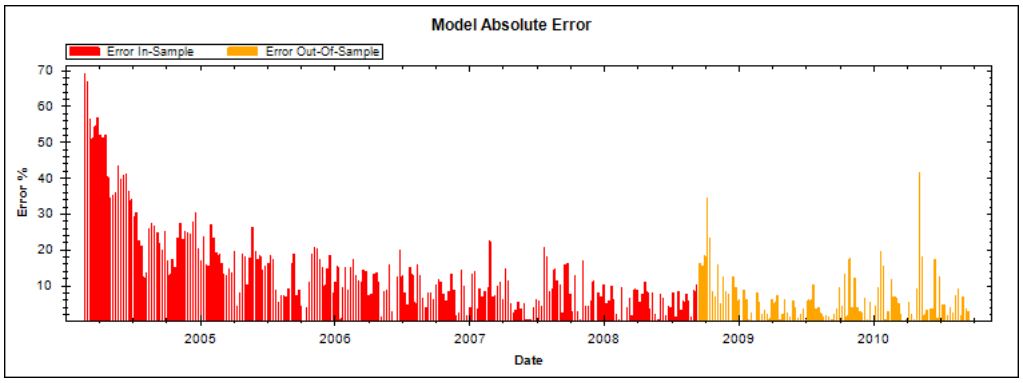

Figure 8.3: Error of the manual model on VIX data with 70% of the data in-sample

The results show how the model manages to significantly beat its benchmark over all time periods.

8.6 Results: Rolling-Window Model

In this section we attempt to improve results of the machine-created model. As we have seen in section 8.4 we believe that poor results for the machine-created model are caused for a large part by over-fitting of the input serie. The influence of a particular keyword on volatility evolves over time. A keyword selected by the incremental increase method for a specific in-sample partition will not necessarily get selected again for another in-sample partition. In order to account for that fact we decide to re-select keywords to use in the model at every period. Instead of keeping the set of identified keywords constant over the whole testing period we recompute the set before forecasting each data point. We call this new model the rolling-window model.

Rolling-Window Model	
Accuracy	94.79%
Standard Deviation	4.13%

Table 8.5: Performance of the rolling-window model

As can be seen in table 8.5 the results for the rolling-window model are better by a large margin than when keeping a constant set of keywords. The model manages to significantly beat the benchmark model. Recomputing the set of keywords to use in our model after each forecasted data point seems to give an efficient solution for machine-created models that can compete against manually designed ones.

Chapter 9

Google Insight vs Twitter

In this chapter we compare Google Insight data with Twitter data. For Google Insight we use the same keyword data as in the previous chapters. For Twitter we collected tweets in real-time from November 20th 2010 to February 18th 2011. We computed weekly keyword statistics as explained in chapter 3 in order to match Google Insight weekly data, which gave us a sample size of 12 weeks.

We use the same keyword universe (see appendix A) as in the previous chapters.

As the sample size with Twitter consists of twelve data points for each keyword, we renounce creating models using Twitter keywords and choose to focus instead on comparing Google Insight and Twitter data directly. To do so we inspect the correlation between keyword statistics obtained with Google Insight and statistics for the same keyword obtained with Twitter. We aggregate these correlations first for every keyword in our keyword universe and then for the ten specifically picked keywords shown in table 9.1.

We are interested in seeing whether Twitter data could be used as an alternative to Google Insight data. Doing so presents several advantages:

- Twitter data can be observed directly and raw contrarily to Google Insight data that is already processed.
- Twitter data is available in real-time and allows for any granularity while Google Insight publishes its data with a delay of one week and with weekly granularity.
- Google Insight puts a limit on the number of keyword statistics one can download from its website. As for Twitter once the tweets have been recorded one can process them and extract data for any number of keywords.

Keywords
crisis
recession
stock crash
stock market
buy gold
buy stocks
sell stocks
unemployment
unemployment benefits
seychelles hotels

Table 9.1: Specifically selected keywords

9.1 Results for All Keywords

Table 9.2 and figure 9.1 show the correlation statistics. The average correlation over all keywords is positive and relatively high considering that most keywords in the universe do not carry much information as they are common English words. The little information carried by some keywords also explains in our opinion the strong negative correlations for some keyword pairs. They can be attributed to noise. Also please keep in mind one needs to be careful when considering correlation for a single keyword pair due to the randomness induced by the low sample size. Thus it is statistically more relevant to consider the average over all keyword pairs. While some keyword pairs have strong negative correlation the average over all pairs clearly shows positive correlation.

In the light of these results we believe that Twitter data can potentially replace (or complement) Google Insight in a modeling framework.

	Value	Keyword
Average	28.85%	
Median	37.40%	
Max	96.91%	yellow
Min	-88.14%	bath

Table 9.2: Correlation Statistics- All Keywords

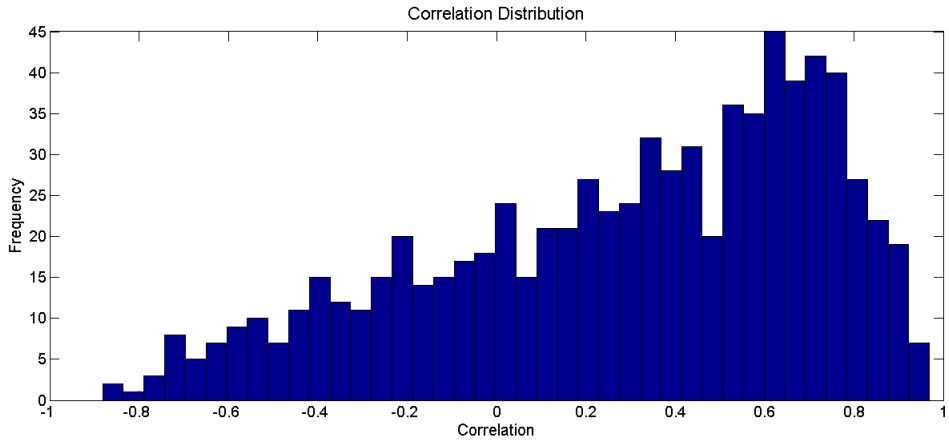

Figure 9.1: Distribution of the correlations

9.2 Results for the specific Keywords Subset

We now consider the correlations for the ten specifically picked keywords shown in table 9.1.

We except the correlations for these keywords to be stronger because they carry more information. Indeed as can be seen in table 9.3 the correlations are significantly stronger when considering the subset rather than the whole universe. Also no keyword pair has negative correlation. We provide all correlations in table 9.4.

Data for the keyword ”seychelles hotels” is not available as there were not enough occurrences of it in the tweets. We therefore excluded the keyword when computing the average and the median.

	Value	Keyword
Average	43.79%	
Median	50.86%	
Max	88.84%	unemployment
Min	2.60%	stock market

Table 9.3: Correlation Statistics- Keyword subset

Keyword	Correlation
buy gold	52.50%
buy stocks	22.60%
crisis	59.68%
recession	50.86%
sell stocks	26.59%
seychelles hotels	n.a.
stock crash	16.29%
stock market	2.60%
unemployment	88.84%
unemployment benefits	74.57%

Table 9.4: Correlation Statistics- Keyword subset

Chapter 10

Conclusion & Future Work

We have provided a framework for working with Google Insight forecasting models including methods for identifying subsets of relevant keywords for a given problem. We have then investigated the methods performances and we have shown how they manage to identify relevant keywords for three different applications: volatility, unemployment claims and visitors in the Republic of Seychelles. We then proceeded to show how the framework can be used to improve volatility forecasts over the AR(1) model. We managed to beat the benchmark out-of-sample for several time horizons with both manually and automatically created models.

We have also investigated the relation between Google Insight and Twitter keyword statistics. Our results show a significantly positive correlation between the two data sources. They also show that for keywords carrying more information the correlation is stronger.

Future work could be conducted to expand the framework to non-linear models. In particular incorporating keyword statistics within a GARCH model for volatility could lead to excellent results. Also interesting would be to go further with the comparison of the data sources Google and Twitter and potentially others (Yahoo!, Facebook, ...). In particular efficient methods for identifying relevant keywords could be derived from the correlations between keywords from different data sources.

References

[1] Hyunyoung Choi, Hal Varian, *Predicting the Present with Google Trends*. Google Inc., 2009.

[2] Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski, Larry Brilliant, *Detecting Influenza Epidemics using Search Engine Query Data*. Google Inc., 2009.

[3] Hyunyoung Choi, Hal Varian, *Predicting Initial Claims for Unemployment Benets*. Google Inc., 2009.

[4] Nikolaos Askitas, Klaus F. Zimmermann, *Google Econometrics and Unemployment Forecasting*. Discussion Paper No. 4201, IZA, 2009.

[5] Guy Judge, Chris Hand, *Searching for the Picture: Forecasting UK Cinema Admissions making use of Google Trends Data*. Discussion Paper No 162, 2010.

[6] Francesco D'Amuri, Juri Marcucci, *"Google it!" Forecasting the US Unemployment Rate with a Google Job Search Index*. 2009.

[7] Francis X. Diebold, Roberto S. Mariano, *Comparing Predictive Accuracy*. University of Pennsylvania, 1995.

[8] Sharad Goel, Jake M. Hofman, Sebastien Lahaie, David M. Pennock, Duncan J. Watts, *What Can Search Predict?*. Yahoo! Research, 2010.

[9] Johan Bollen, Huina Mao, Xiao-Jun Zeng, *Twitter Mood predicts the Stock Market*. Journal of Computational Science, 2011.

[10] Sitaram Asur, Bernardo A. Huberman, *Predicting the Future with Social Media*. Hp Labs, 2010.

[11] Katja Ahoniemi, *Modeling and Forecasting Implied Volatility - an Econometric Analysis of the VIX Index*. Helsinki University, Discussion Paper No. 129, 2006.

[12] *THE CBOE VOLATILITY INDEX - VIX*. CBOE, 2003.

[13] Dimitri N. Politis *Model-free vs. Model-based Volatility Prediction*. University of California at San Diego, 2004.

[14] Dimitri N. Politis *Random Walk in Stock-Market Prices*. University of Chicago, 1965.

[15] Charlotte Christiansen, Maik Schmeling, Andreas Schrimpf, *A Comprehensive Look at Financial Volatility Prediction by Economic Variables*. University of Chicago, 2011.

[16] Tim Bollerslev *Generalized Autoregressive Conditional Heteroskedasticity*. Journal of Econometrics,31, 307-327, 1986.

[17] Robert F. Engle *Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation*. Econometrica, 50, 987-1007, 1982.

[18] John C. Hull, *Options, Futures and Other Derivatives*. 7th Edition, Prentice Hall India, 2008.

[19] Neil A. Chriss, *Black-Scholes and Beyond: Option Pricing Models*. McGraw-Hill, 1996.

[20] Paul Wilmott, *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge University Press, 1995.

[21] James D. Hamilton, *Time Series Analysis*. Cambridge University Press, 1994.

[22] Michael J.A. Berry, Gordon S. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. Wiley Computer Publishing, 2nd Edition, 2004.

[23] Trevor Hastie, Robert Tibshirani, J.H. Friedman, *The Elements of Statistical Learning*. Springer, 2003.

[24] Robert Nisbet, John Elder IV, Gary Miner, *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press, 2009.

Appendix A

Keyword Universe

- able
- account
- acid
- across
- act
- addition
- adjustment
- advertisement
- agreement
- air
- airplane
- amount
- amusement
- angle
- angry
- animal
- answer
- ant
- apparatus
- apple
- approval
- arch
- argument
- arm
- army
- art
- attack
- attempt
- attention
- attraction
- authority
- automatic
- baby
- bag
- balance
- ball
- band
- base
- basin
- basket
- bath
- beatles
- beautiful
- because
- bed
- bee
- behaviour
- belief
- berry
- between
- bird
- birth
- bite
- bitter
- black

- blade
- blood
- blow
- blue
- board
- boat
- body
- boiling
- bone
- book
- boot
- bottle
- box
- boy
- brain
- brake
- branch
- brass
- bread
- breath
- brick
- bridge
- bright
- broken
- brown
- brush
- bucket
- building
- bulb
- burn
- burst
- business
- butter
- button
- buy gold
- buy stocks
- cake
- camera
- canvas
- card
- care
- carriage
- cart
- cartoon characters
- cat
- cause
- certain
- chain
- chalk
- chance
- change
- cheap
- cheese
- chemical
- chest
- chief
- chin
- church
- circle
- clean
- clear
- clock
- cloth
- cloud
- coal
- coat
- cold
- collar
- colour
- comb
- come
- comfort
- committee
- common
- company
- comparison
- competition
- complete
- complex
- condition
- connection
- conscious

- control
- cook
- copper
- copy
- cord
- cork
- cotton
- cough
- country
- cover
- cow
- crack
- credit
- crime
- crisis
- cruel
- crush
- cry
- current
- curtain
- curve
- cushion
- damage
- danger
- dark
- daughter
- day
- dead
- death
- debt
- decision
- deep
- degree
- delicate
- dependent
- design
- destruction
- detail
- development
- different
- digestion
- direction
- dirty
- discovery
- discussion
- disease
- disgust
- distance
- distribution
- division
- dog
- door
- doubt
- down
- drain
- drawer
- dress
- drink
- driving
- drop
- dry
- dust
- ear
- earth
- east
- edge
- education
- effect
- egg
- elastic
- electric
- engine
- enough
- equal
- error
- espn
- even
- event
- example
- exchange
- existence
- expansion

- experience
- expert
- eye
- fact
- fall
- family
- farm
- father
- fear
- feather
- feeble
- feeling
- female
- fertile
- fiction
- field
- fight
- finger
- fire
- first
- fish
- fixed
- flag
- flame
- flat
- flight
- floor
- flower
- fly
- fold
- food
- foolish
- foot
- force
- fork
- form
- forward
- fowl
- frame
- free
- frequent
- friend
- from
- front
- fruit
- full
- future
- garden
- general
- get
- girl
- give
- glass
- glove
- goat
- gold
- good
- government
- grain
- grass
- great
- green
- grey
- grip
- group
- growth
- guide
- gun
- hair
- hammer
- hand
- happy
- harbour
- hard
- harmony
- hat
- hate
- head
- healthy
- hear
- hearing
- heart

- heat
- help
- high
- history
- hole
- hollow
- hook
- hope
- horn
- horse
- hospital
- hour
- house
- how
- how to
- humour
- ice
- idea
- ill
- important
- impulse
- increase
- industry
- ink
- insect
- instrument
- insurance
- interest
- invention
- iron
- island
- jelly
- jewel
- join
- journey
- judge
- jump
- keep
- kettle
- key
- kick
- kind
- kiss
- knee
- knife
- knot
- knowledge
- land
- language
- last
- late
- laugh
- law
- lead
- leaf
- learning
- leather
- left
- leg
- letter
- level
- library
- lift
- light
- like
- limit
- line
- linen
- lip
- liquid
- list
- little
- living
- lock
- long
- look
- loose
- loss
- loud
- love
- machine
- make

- male
- man
- manager
- map
- mark
- market
- married
- mass
- match
- material
- may
- meal
- measure
- meat
- medical
- meeting
- memory
- metal
- middle
- military
- milk
- mind
- mine
- minute
- mist
- mixed
- money
- monkey
- month
- moon
- morning
- mother
- motion
- mountain
- mouth
- move
- muscle
- music
- nail
- name
- narrow
- nation
- natural
- near
- necessary
- neck
- need
- needle
- nerve
- net
- new
- news
- night
- noise
- normal
- north
- nose
- note
- number
- nut
- observation
- offer
- office
- oil
- open
- operation
- opinion
- opposite
- orange
- order
- organization
- oven
- over
- owner
- page
- pain
- paint
- paper
- parallel
- parcel
- part
- past

- paste
- payment
- peace
- pen
- pencil
- person
- physical
- picture
- pig
- pipe
- place
- plane
- plant
- plate
- play
- playstation
- please
- pleasure
- plough
- pocket
- point
- poison
- polish
- political
- poor
- porter
- position
- possible
- pot
- potato
- powder
- power
- present
- price
- print
- prison
- private
- probable
- process
- produce
- profit
- property
- prose
- protest
- public
- pull
- pump
- punishment
- purpose
- push
- quality
- question
- quick
- quiet
- quite
- rail
- rain
- range
- rate
- ray
- reaction
- reading
- ready
- reason
- receipt
- recession
- record
- red
- regret
- regular
- relation
- religion
- representative
- request
- respect
- responsible
- rest
- reward
- rhythm
- rice
- right
- ring

- river
- road
- rod
- roll
- roof
- room
- root
- rough
- round
- rub
- rule
- run
- sad
- safe
- sail
- salt
- same
- sand
- say
- scale
- school
- science
- scissors
- screw
- sea
- sears
- seat
- second
- secret
- secretary
- seed
- selection
- self
- sell stocks
- send
- sense
- separate
- serious
- servant
- seychelles hotels
- shade
- shake
- shame
- sharp
- sheep
- shelf
- ship
- shirt
- shock
- shoe
- short
- shut
- side
- silk
- silver
- simple
- sister
- size
- skin
- skirt
- sky
- sleep
- slip
- slope
- slow
- small
- smell
- smile
- smoke
- smooth
- snake
- sneeze
- snow
- soap
- society
- sock
- soft
- solid
- some
- son
- song
- sort

- sound
- soup
- south
- space
- spade
- special
- sponge
- spoon
- spring
- square
- stage
- stamp
- star
- start
- statement
- station
- steam
- steel
- stem
- step
- stick
- sticky
- stiff
- still
- stitch
- stock crash
- stock market
- stocking
- stomach
- stone
- stop
- store
- story
- straight
- strange
- street
- stretch
- strong
- structure
- substance
- sudden
- sugar
- suggestion
- summer
- sun
- support
- surprise
- sweet
- swim
- system
- table
- tail
- take
- talk
- tall
- taste
- tax
- teaching
- tendency
- test
- theory
- thick
- thin
- thought
- thread
- throat
- through
- thumb
- thunder
- ticket
- tight
- till
- time
- tin
- tired
- toe
- together
- tomorrow
- tongue
- tooth
- top
- town

- trade
- train
- transport
- tray
- tree
- trick
- trouble
- trousers
- true
- turn
- twist
- under
- unemployment
- unemployment benefits
- unit
- use
- vacation
- value
- verse
- very
- vessel
- view
- violent
- visa
- voice
- waiting
- walk
- wall
- war
- warm
- wash
- waste
- water
- wave
- wax
- way
- weather
- week
- weight
- west
- wet
- wheel
- whip
- whistle
- white
- wide
- will
- wind
- window
- wine
- wing
- winter
- wire
- wise
- woman
- wood
- wool
- word
- work
- wound
- writing
- wrong
- yahoo
- year
- yellow
- yesterday
- young