Llenguatges i Sistemes Informàtics
Master in Computing

Master Thesis

Using Twitter as a source of information for time series prediction

January 2012

Author: Ramon Xuriguera

Advisors: Marta Arias and Argimiro Arratia

Universitat Politècnica de Catalunya

Contents

1

Introduction

Recent years have seen a dramatic rise in the use of social networking platforms, Twitter being one of the most popular and successful ones. Since its creation in 2006, Twitter has been growing steadily and an average of over 200 million messages are currently posted everyday [43]. As a result, incredible amounts of information on an immense variety of domains are now available through their service. The accessibility to social networking data has attracted the attention of many researchers and has become an area under active investigation as new ways of using this data are being devised.

This project aims to assess whether tapping into the wealth of information that Twitter has to offer can positively affect the prediction of time series. We intend to accomplish our goal by means of applying multiple machine learning predictive models and text mining techniques, all organized within a framework that should be general enough to allow the realization of any similar task.

Given a set of past observations of a target time series with information on a given domain, we want to be able to predict future values of the series by looking at these past observations as well as external information relevant to the same domain. In our case, this additional information will be coming from Twitter. Diagrams illustrating an overview of the complete process are given in Figures 1.1 and 1.2.

As depicted in Figure 1.1, one of the first problems that will have to be addressed is data acquisition. Recollection of Twitter messages is not as trivial as it may seem because some restrictions apply (Chapter 2). The content of the messages will mainly consist of text and an associated time stamp, it will thus be necessary to apply some type of transformation to this data before being able to provide it to a predictive mathematical model. Given that all items have an associated time and date, it seems only natural that the result of this transformation should be another time series.

This additional series will be created by aggregating all the messages by time or date and then taking some piece of information from the collection of daily texts as the value for the series for that given day. The complexity of extracting this piece of information from the daily messages can vary widely, ranging from just counting the number of unique messages to more advanced text mining techniques including, but not limited to, text categorization, topic modelling or sentiment analysis.

One of the areas of Natural Language Processing (NLP) that has received renewed attention since the increase in popularity of online social networks is Sentiment Analysis. This type of analysis attempts to automatically identify and extract subjective opinions expressed in a given text. Sentiment Analysis is very appropriate for the type of application we are dealing with (Chapter 3). Once the conversion from messages is done, we will have two time series: the one that we are attempting to predict (the *target series*), and the one derived from Twitter data and that will hopefully contain information on the first one (the *additional series*).

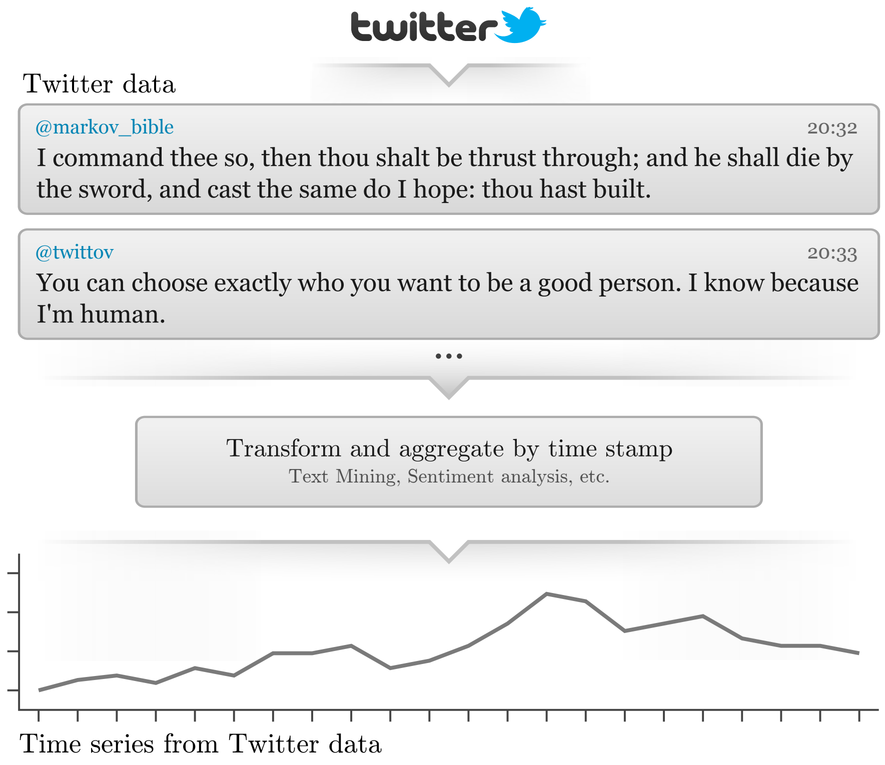

twitter
Twitter data
@markov_bible 20:32
I command thee so, then thou shalt be thrust through; and he shall die by the sword, and cast the same do I hope: thou hast built.
@twittov 20:33
You can choose exactly who you want to be a good person. I know because I'm human.
...
Transform and aggregate by time stamp
Text Mining, Sentiment analysis, etc.
Time series from Twitter data

Figure 1.1: From text to time series

The rest of the procedure is represented in Figure 1.2. A predictive model for the target time series is estimated by taking past values from both the target and the additional series. This is typically done by iterating through the set of available observations (in our case, these correspond to days) and, at each iteration, giving the model the value of the target series for that day as well as a finite number of past values, also known as *lags*. Ideally, these lags hold information on the output and can be of use for predicting. Once the model has been trained, it can be used for predicting future values of the series.

In order to assess whether the inclusion of Twitter data has any effect on the predictions, models are trained both using and not using the additional time series. This will allow us to see if there is a significant improvement when using additional information for the predictions (Chapter 4).

Experiments for the proposed approach will be done for data coming from two different domains. The first one is Stock Market prediction, where the target time series will correspond to the closing prices of a specific company or stock index. The second application is box office prediction, where we will attempt to predict the revenue of recently released movies.

1.1 Related work

Some novel attempts at incorporating external data sources for making predictions have been done during the last few years. Table 1.1 shows a comparison of previous work in this direction. As can be seen from the table, some of the approaches do not consider any prediction model and only look at correlation between the additional and the target data.

This master thesis was originally inspired by an article written by Johan Bollen et al. [8] (included in the comparison table) in which the general mood of the messages published on

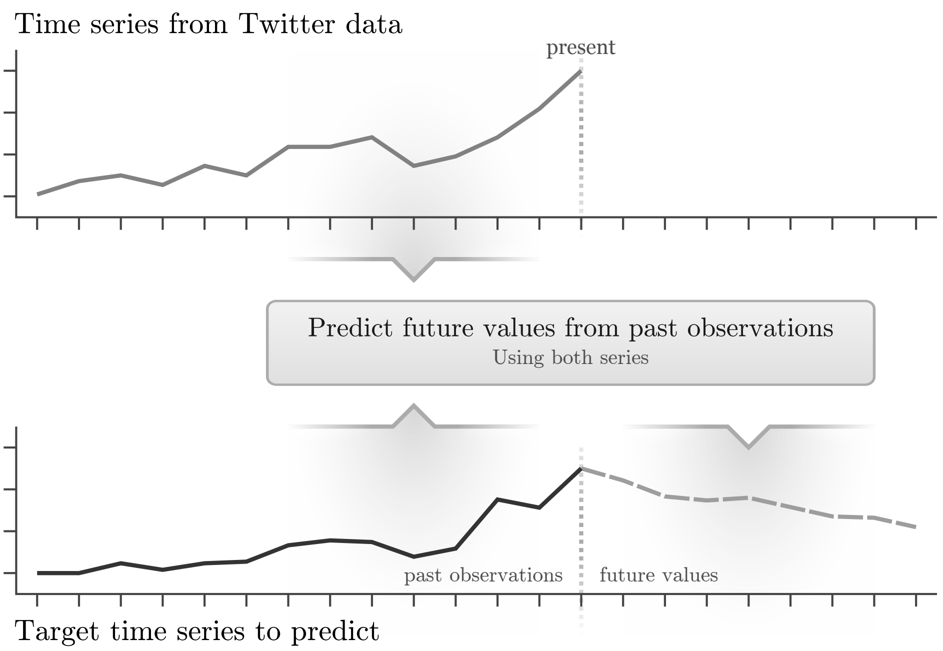

Time series from Twitter data
present
Predict future values from past observations
Using both series
past observations
future values
Target time series to predict

Figure 1.2: Time series forecasting

Twitter is estimated and later used to predict the Dow Jones Industrial Average (DJIA). In their article, the general mood is estimated with two different lists of words.

First, a general sentiment time series is created by computing the daily ratio between the amount of messages that contain positive words and the messages with negative ones. These positive and negative words are taken from the OpinionFinder[1] software, an open source system for performing subjectivity analysis of text. OpinionFinder comes with a corpus of polarity-tagged words where 2718 of the words are considered to be positive and 4912 are tagged as negative. OpinionFinder also distinguishes between weak and strong polarity, but this distinction is not taken into account in Bollen's work.

The other approach for estimating the general mood from Twitter messages is based on the *Profile of Mood States - Bipolar* (POMS for short) rating scale, with which six different mood factors are derived from a list of 72 adjectives. These factors are: calm, alert, sure, vital, kind and happy[2]. The POMS list of adjectives is too short to be used in text classification tasks and hence was extended by looking at co-occurrences from a collection of 4-grams and 5-ngrams provided by Google. This extended list, GPOMS, is not available to the general public, we have therefore not been able to incorporate these mood factors to our experiments. Once the general mood has been computed, they aim to predict the DJIA by providing these newly-created time series to a Self-Organizing Fuzzy Neural Network (SOFNN).

Along the same line, Sebastian Wolfram [46] also attempts to predict the price of some NASDAQ stock quotes by using Twitter as an additional source of information. His work differs in a variety of ways from ours, though. First, the features extracted from text are directly fed into the prediction model. That is, the intermediate step of analysing the messages' sen-

[1] `http://code.google.com/p/opinionfinder/`

[2] The six mood factors are identified with a different names in the original POMS rating scale.

Ref.	Event	Models	Corpus	Conclusion
[8]	DJIA	SOFNN	∼10M tweets, Stock market prices	An index of the calmness of the public is predictive of the DJIA and predictions can be significantly improved using a SOFNN.
[46]	NASDAQ stocks	SVM	Edinburgh Corpus, English, Relevant to stocks	Works with high freq. data. No sentiment analysis, but direct count of frequency of words
[47]	DJIA, S&P500, NASDAQ, VIX	n/a	English, with mood keywords	Finds correlations of tweet's emotions (hope, fear, worry) and the direction of the DJIA stock index.
[22]	Movie sales	n/a	Blog posts with links to IMDB, IMDB sales data	Considering the sentiment of blog posts improves the correlation between references to movies and their financial success.
[2]	Movie sales	Linear regression	∼2.9M tweets for 24 movies	The model built with the tweet rate time series outperforms the baseline that uses the Hollywood Stock Exchange (HSX).
[35]	Swine Flu Pandemic	SVM	Edinburgh Corpus, Newspaper articles	Adding features concerning historical context of a feature has a beneficial impact on forecast accuracy.
[24]	U.S. polls	n/a	10^9 tweets (omitting non-English), Public opinion polls	The evolution of Twitter sentiment correlates to periodical public polls on the presidential election and on the presidential job approval
[16]	Book sales	*Spikes predictor*	Blog posts, Amazon sales rank	There is correlation between an increase on the number of blog mentions of a book and a spike in its sales. Blog mentions data does help improve the ability to predict spikes. The article uses a custom *Spikes predictor*.

Table 1.1: Table comparison of other work on predicting using available data from the Internet to predict other areas

timent and creating a time series from it is omitted. Second, high frequency data from the stock market is used for the predictions. As stated in his work, historical data of this kind is not easily available, so predictions are only done for a two-week timespan. However, considering that consecutive observations are just one minute apart, this period is enough for testing the system. Regression is used to estimate the most immediate stock price. In addition, his work also incorporates a simulated trading engine to test how his system would perform in real life and how it would translate in terms of benefits.

The literature on box office prediction using messages from social networks is somewhat more limited in terms of forecasting. While both [22] and [2] take a similar approach in using sentiment analysis to explore the opinion of the general public in relation to specific movies, predictive power is not thoroughly explored. In [22], Nigam and Hurst only look at Pearson's r-correlation between some sentiment metrics derived from blog posts and the sales of 49 movies. Asur and Huberman [2] go a bit further and use linear regression to predict sales. Much more work can be done in this direction, and this is partly what this work is set out to do.

In this thesis, we want to test whether using Twitter helps in making better predictions. What differentiates us mainly from previous work is the fact that we have tested this hypothesis under a wide variety of conditions, applying an extensive list of predictive models with varying parameter settings and testing for two different domains under a unified set of techniques.

1.2 About this document

The remainder of this document has been divided into two parts. The first one defines a generic framework for assessing the influence of Twitter data on time series predictions. The second part focuses on experimenting with the framework in two different domains: stock market and movie sales. Each of the parts is in turn divided into chapters as follows:

- Part I: Methodology
  - Chapter 2 describes some characteristics of Twitter and the challenges to retrieve data.
  - In Chapter 3 we will go through the problem of sentiment analysis and some of the techniques that are used.
  - Chapter 4 completes the first part of the document by describing the set of time series prediction techniques that have been adopted.
- Part II: Applications
  - Chapter 5 describes the setup of some experiments and presents their results for stock market prediction.
  - Finally, Chapter 6 shows the result of applying the methodology to data from a different domain, namely, box office.

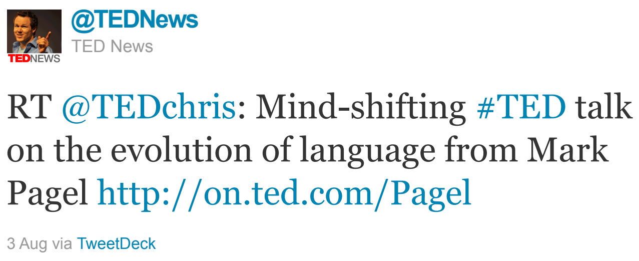
Part I

Methodology

2
Data Retrieval

2.1 Twitter Basics

Twitter is an online microblogging platform that allows its users to build social networks. Its core functionality is to easily and quickly share messages with the rest of the members of one's network. In the Twitter domain, these messages are known as *tweets* and are limited to a maximum of 140 characters by design. The social network is defined by unidirectional *follow* associations between users, meaning that a user will only be able to see tweets posted by the people he or she is following. Specific terminology is also used to refer to users depending on their relationship: given two users A and B with a directed relation from B to A, it is said that B is A's *follower* and that A is B's *friend*.



@TEDNews
TED News

RT @TEDchris: Mind-shifting #TED talk on the evolution of language from Mark Pagel http://on.ted.com/Pagel

3 Aug via TweetDeck

Figure 2.1: Sample tweet

The figure above shows an example of a tweet; it has been chosen because it has some parts worth noting:

- Other people can be *mentioned* or *replied to* by using the `@` symbol followed by their user name. User names are alphanumeric strings of up to 15 characters. Underscores are allowed as well.
- Words within a message preceded by the # symbol are known as *hashtags* and are mostly used to assign messages to topics or to mark keywords, like the example above. However, hashtags have become a very versatile tool and can also be used to accomplish other tasks such as indicating feelings, sarcasm or as a side commentary on the message, e.g. *Sarah Palin for President??!? #Iwouldratherhaveamoose* [25].
- Tweets beginning with the expression `RT @[\w_]{1,15}` are called *retweets* and they are a handy way to share something with the people who are following you.
- Finally, as it can be seen in the figure, the tweet also contains a URL. While this is not Twitter-specific, it is important to note that links are very common and are to be expected. For instance, in a database of 16,065,759 tweets collected from March to July 2011, 4,361,922 contained the string *http://*, which amounts to 27.15% of the total.

Twitter offers a variety of APIs to obtain tweets programatically, and even though messages are limited to 140 characters, abounding additional information is sent along with them.

This obviously includes the author's name, the tweet's id and the creation time stamp but it also includes other user-related data such as the number of followers or the time zone. See Listing 2.1 for an example of a JSON-formatted tweet retrieved with the REST API. Many fields have been omitted for brevity, but the full example is reproduced in Appendix A.1.

```
{
    "user": {
        "followers_count": 82751,
        "time_zone": "Eastern Time (US & Canada)",
        "lang": "en",
        "screen_name": "TEDNews",
        "id": 36843988,
         ...
    },
    "retweeted": false,
    "text": "RT @TEDchris: Mind-shifting #TED talk on the evolution
            of language from Mark Pagel http://on.ted.com/Pagel",
    "retweet_count": 7,
    "created_at": "Wed Aug 03 15:24:51 +0000 2011",
    "id": 98776348977414140
     ...
}
```

Listing 2.1: Same sample tweet retrieved with Twitter's REST API

2.2 Tweet retrieval

Even though working with Twitter data is becoming very common, one major problem is the lack of standard datasets. In April 2010 Twitter updated the API terms of service introducing a rule that does not allow third parties to redistribute Twitter Content, i.e. tweets, to third parties without prior written approval from Twitter [40, 39]. Therefore, attempts to release Twitter corpora, like the Edinburgh Corpus presented in [29], have failed. This has serious consequences on research since it makes it very hard to reproduce previous results. Moreover, we were left with the need to collect our own data using the provided APIs, which are briefly described below. Specific details on the data we collected is described separately for the sentiment analysis classifier in Section 3.1 and for each of the applications in Part II. A further point that has to be considered is the fact that an extremely large number of messages will be collected, easily amounting to millions. This data has to be stored in an efficient way so it can be easily retrieved afterwards.

2.2.1 APIs

The REST API is a web service implemented using the principles of REST (Representational State Transfer) that allows developers to interact with Twitter and use most of its features. The use of this API has a limited rate with a maximum of 350 requests per hour[1]. Taking into account that a maximum of 100 tweets can be retrieved in each request, the total number of daily messages available through the REST API amounts to 840000 tweets/day. That would

[1]Given limits are the ones we had in 2011 and are subject to change in the future

be a reasonable number of tweets if it were not because there is no easy way to get tweets containing specific words. That is what the Search API is for.

With the Search API it is possible to retrieve tweets that match a given query. Although it is also rate-limited, the limit is not applied after a certain number of requests per hour but depending on the complexity and frequency of the requests [42].

Finally, the Streaming API offers a small sampling of the tweets in the form of a stream. It is intended for developers with data intensive needs and it works by establishing a single HTTP long-lived connection that is kept alive indefinitely and over which new tweets are sent as they are being posted. Among other features, it offers the possibility to filter the stream with up to 400 words and/or 5000 user identifiers. Other restrictions also apply. For instance, the output of overly broad predicates is periodically limited [41]. This has some implications on the way the words should be chosen. Suppose that the stream is filtered by two words; one of them is very popular whereas the other is rarely used. Given the huge volume of tweets containing the popular word, by the time someone writes a message with the rare word, the periodic limit may have already been reached, ending up with almost no tweets for it.

All things considered, we chose the Streaming API because the `filter` method is very convenient for the task we are trying to accomplish. The retrieval of tweets (or *listening*) began on 22 March 2011.

2.2.2 Alternatives

One inherent problem of streaming APIs is the impossibility to retrieve information from the past. That is, to get a two-month dataset, it is necessary to listen to the stream for this same amount of time. Google Realtime Search, which was a Google Search feature that returned search results from sources such as Twitter or Facebook, seemed a good candidate to circumvent this problem as it allowed to navigate through time to see previously posted messages.

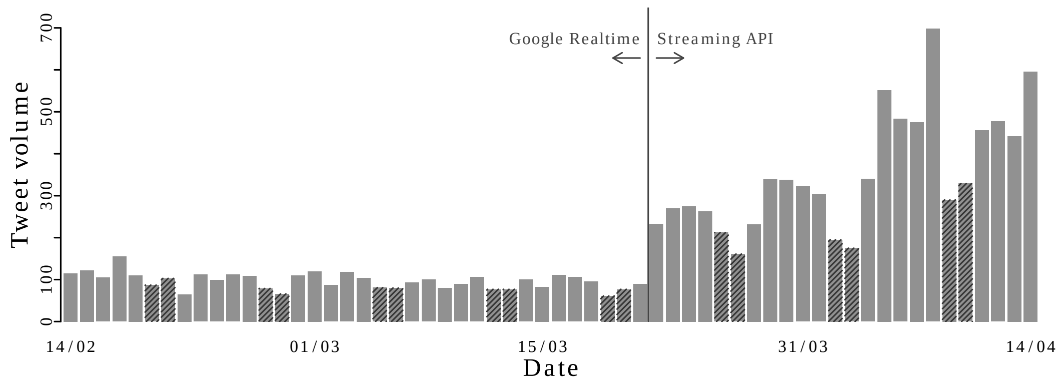


Figure 2.2: Number of tweets collected per day from 14 Feb. to 14 Apr.

We started retrieving tweets dating before the 22 March, time when we started listening to the stream. However, the amount of tweets per day retrieved for this period dropped significantly because only a small sample of the total number of tweets was offered. Figure 2.2 shows this difference when querying for tweets containing some company names. Shaded areas correspond to weekends, when activity is lower.

Moreover, Realtime Search went offline early July 2011 due to the expiration of Google's agreement with Twitter [36].

Finally, there is also the possibility to buy tweets from third parties that have an agreement with Twitter. As of December 2011, there were two official resellers: Gnip[2] and Datasift[3]. These were not an option for us due to the high monthly costs.

[2]http://gnip.com/twitter
[3]http://datasift.com

3

Sentiment Analysis

3.1 Introduction

The primary goal of this chapter is to describe a way of transforming Twitter messages into a time series. Sentiment analysis is a line of research that combines techniques from various fields such as Natural Language Processing and Machine Learning to extract, from a given piece of text, information on the author's personal impressions. Sentiment analysis can be divided into two main subtasks:

- Subjectivity Recognition: which is usually a binary problem that consists in deciding whether a given text contains personal impressions or not.

- Polarity Detection: usually a second step on the sentiment analysis problem, tries to extract concrete information from subjective writing.

As has become customary in most recent work that deals with social data, smileys are used to create labelled corpora for supervised training [14, 3]. This is the very same approach we take in this project, creating a dataset of tweets that are automatically tagged as positive if they contain one of these smileys: :-), :-D[1] or negative if they contain :-(.

Although there exists some work doing multi-class sentiment analysis, such as Ahkter and Soria in [1], where they classify Facebook messages into *Happy*, *Unhappy*, *Sceptical* and *Playful*, in this document we will restrict ourselves to binary classification.

The first problem that must be addressed is the need of a corpus from which to train a sentiment classifier. Since there are no publicly available datasets we have created our own. As noted in [14, 3], there are many more tweets containing positive smileys than negative ones, so in order to balance the number of instances for the two different classes, the total amount of tweets included in the sentiment classifier training set is limited by the number of available negative messages.

3.2 Preprocessing

Cleaning the data

The first transformation applied to the data is a case conversion to ensure that all text is in lower case. Afterwards, some replacements are done to keep some characteristics and expressions that we consider may give some hints on the sentiment of the message. The following regular expressions are substituted by predefined tags that will allow us to group multiple occurrences of the item they represent under the same feature.

URLs and e-mail addresses are also substituted but their expressions are too lengthy to reproduce them here. Some of the listed expressions, on the other hand, are very simple

[1]The actual list of tweets we consider is somewhat wider, using the regular expressions `[:8=][- ]?[)D]` and `[:8=][- ]?[(]` for positive and negative tweets respectively.

Field	Regular expression
Usernames	`@\w+`
Stock Market Tickers	`\$[a-z]{1,4}`
Hashtags	`#\w+`
Hearts	`<3|♡`
Laughter	`[jhaie]{4,}`
Ellipsis (...)	`[.]{3,}|...`
Question marks	`?+`
Exclamation marks	`!+`
Interrobang	`(TAG\_(QUESTION|EXCLAM)){2,}`
Currencies	`[\$€¥£¢]`
Percentages	`[+-]?\d+([.,]\d+)?%`
Time	`[0-2]?[0-9]([:.][0-6][0-9]){1,2}\s?([ap]\.?m\.?)?`
Ordinals	`\d+(st|nd|rd|th|[aeo])`
Other Numbers	`[-+]?\d+([.,]\d+)?`
Repeated characters	`(\w)\1{2,}`

Table 3.1: Substitutions during data cleaning

and make assumptions that might not always hold. For instance, any substring matching `[jhaie]{4,}` will be replaced by the Laughter tag. Surely, there exist (non-laughter) words solely formed by using letters *a, e, h, i* or *j*, but after the cleaning they will be considered as laughter.

The heart and the ellipsis after the bar correspond to ISO character entities. Interrobangs are series of both question and exclamation marks. All other symbols and smileys are removed.

One remaining question is whether to keep hashtags or not. As seen in the previous chapter, hashtags are usually the result of concatenating multiple words, resulting in new words that have not been seen before. Only when a hashtag is frequent enough will it be used to determine the polarity of the sentence. In contrast, grouping all hashtags under the same word and checking for their presence of hashtags may help predicting polarity the same way as URLs or usernames. Both approaches will be tested in the experiments described in Section 3.4.1.

Duplicate Removal
Most of the duplicates in our database correspond to retweets, and are not included in the dataset. However, after cleaning and applying the aforementioned substitutions we end up with a considerable amount of identical tweets. These have been removed as well from the corpus.

Language Detection
One of the characteristics of social data is that messages are written in a broad range of languages. This may not affect the aforementioned preprocessing steps, but it can be a problem when trying to apply more complex procedures to the data. For instance, one problem with multi-language data is that typical text-processing techniques such as stemming or stop word removal cannot be correctly applied because their performance is tightly tied to the language the text is written in.

As can be seen in Listing 2.1 users have a language associated to them. It must be noted that this does not correspond to the language they write in but to the language they have set their Twitter's interface to. At first thought, it might seem reasonable to consider that these two are the same most of the time. But even if we ignore the fact that lots of people use more than one language on a daily basis (e.g. mother tongue and English) we are still left with another problem. Twitter is currently translated to only 11 languages [44] and the default language is English. We therefore believe that it does make sense to do some sort of language detection.

We decided to use the Guess Language[2] library, to associate a language to each tweet at the time of retrieval. Internally, this tool looks at *trigrams* and applies some heuristics to choose a language from a predefined list. The brevity of the tweets means that this method may fail to detect the correct language much more often than what it would for more extensive texts. This problem worsens if we take into account that people tend to use some English words in their native languages.

Stop words removal

Stop words are those words so commonly used that their presence in a document does not give any clue to the text's topic. Some words that fall into this category are pronouns, prepositions, most used verbs, among others.

There is no standard stop list, for our experiments we have taken a rather short one from [34] for the English language. In case of the multi-language data, we have elaborated a special list combining stop lists for some of the most common languages, available also from [34].

Stemming

Stemming is the task of reducing words to their stem by removing common suffixes. The main reason for doing this is to map multiple words that share the same stem to one single feature. For example, the words *reject*, *rejected*, *rejecting*, *rejection*, *rejections* should be all mapped to *reject*.

We have used the Porter2 stemmer for the English language, which adds some improvements over the original Porter algorithm. Details of the algorithm can be found in [32, 31].

Handling Negation

Negation can play an important role in the task of sentiment classification. Consider the sentences *I think it was good* and *I think it was not good*. While they only differ in one word and would score highly in most of similarity measures, their sentiment polarities are completely opposite.

We have attempted a very simple form of negation handling for English texts by tagging words between common polarity shifters such as *not*, *don't* or *haven't*. For instance, the sentence from the previous example would become *I think it was not NOT_good*. With this transformation, a word and its negated counterpart are considered to be different words, significantly increasing the size of the vocabulary. This translates to a larger set of features and that a larger dataset is preferable. Moreover, this only covers a small subset of negations where there is a valence shifter involved.

[2]`http://code.google.com/p/guess-language/`

This is the same approach as the one used in [11], also described in [26], Section 4.2.5, although more complex techniques that take into account part of speech tags are also commented there. [30] and [33] study not only negatives but a wide range of words that may shift the valence of other nearby words in a much more subtle way: connectors, modal verbs, etc. However, none of these more advanced techniques have been tested in our experiments due to time constraints.

Tweet Representation: The Bag of Words Model

In the Bag of Words model each document is represented as an n-dimensional vector $d = (w_1, w_2, ..., w_n)$ where w_i indicates whether word i appears in document d or not. The length of the vector will be given by the size of the vocabulary $|\Sigma| = n$. This simplistic model assumes that word order does not matter. For instance, the sentences: *You are not ill, I am.* and *I am not ill, you are.* will result in the same document vector.

Rather than considering sets of words at random, multiple lists of words have been elaborated by finding all those words that occur a minimum of five times in the dataset.

As an alternative, the components of the document vector can represent the word frequency in the text instead of just the term presence.

3.3 Relevance Filter

One of the problems found by skimming through the collected data is that there is a considerable amount of tweets that, while containing one or more of the filtering keywords, are not relevant to the task we are trying to accomplish. This is usually caused by homonyms, polysemous words, proper names or words that are part of an idiom. For example, all the tweets below contain the word *apple* but belong to very different topics:

> Food:
> *"I really love eating an apple with peanut butter for breakfast. #soyummy"*
>
> Body hair:
> *"Thanks to some sloppy shaving, I developed the latest trend in facial hair: the apple patch. Like a soul patch, but over your Adam's apple."*
>
> Stock Market:
> *"Apple stock soared above $404 today."*
>
> New York City:
> *"In the big apple skipping down street. New York city is my soul mate. Mine. Mine. Mine."*

Latent Dirichlet Allocation (LDA)

We will attempt to alleviate this problem by using Latent Dirichlet Allocation (LDA), a generative probabilistic model – mostly used for topic modelling – that was first introduced by Blei et al [6], built upon Latent Semantic Indexing (LSI) and Probabilistic LSI (pLSI). LDA will be provided with relevant documents from the tweet collection in order to find latent descriptions of them. Unseen documents will only be kept if they conform to these descriptions.

The basic idea behind LDA is that documents consist of a mixture of topics. The generative process is depicted in Figure 3.1. Documents belong to topics in different proportions, determined by the histogram. A multi-sided coin is then thrown for each of the words in the document, assigning topics depending on the topic distribution. Finally a specific word is chosen from each topic. Highlighted words are color-coded depending on the topic they are associated with. We could say that the tweet in the diagram is a mixture of *politics*, *sociology* and *sound/music*.

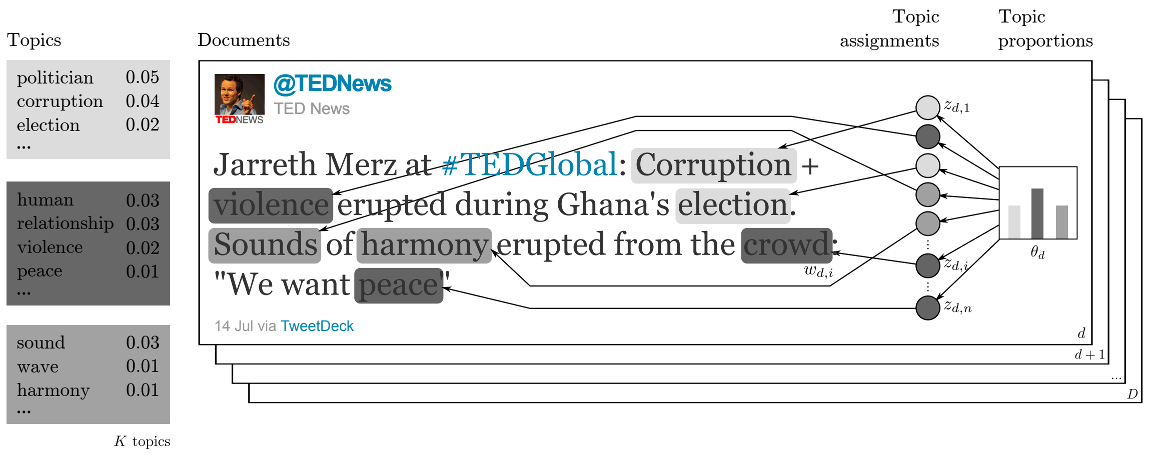


Figure 3.1: LDA generative process. Diagram adapted from Blei [5]

More formally, given a corpus of D documents, the goal is to infer the most probable model that generated it, the only observed variables being the words $w_{d,n}$ in the documents. Topics $\beta_{1..K}$ are defined as distributions over a fixed vocabulary and the number of possible topics K is considered to be known beforehand. The generative process goes as follows:

1. A distribution over the topics for a specific document d is drawn from a Dirichlet distribution $\theta_d \sim Dir(\alpha)$. A Dirichlet distribution is a distribution over multinomial distributions and is parametrized by a positive real-valued vector α.

2. For each of the n words in the document:

   - A topic is drawn from the per-document topic distribution: $z_{d,n} \sim Multinomial(\theta_d)$. It should be noted that topics are drawn independently, so we are assuming that word order does not contribute to the topic structure of a document.
   - A specific word is selected by making a single trial from the chosen topic: $w_{d,n} \sim Multinomial(\beta_{z_{d,n}})$

The conditional dependencies between the different variables are visually represented in plate notation in Figure 3.2. The shaded node corresponds to the observed variables while the rest are latent variables. Rectangles or *plates* are a concise way to represent collections. For instance, the K plate denotes that there are β_1 to β_K latent variables that depend on η.

Plate notation does not, however, show all the relations between variables. Notice that an observed word $w_{d,n}$ depends both on the topic assignment $z_{d,n}$ and all the topics $\beta_{1..K}$. In practice, $z_{d,n}$ acts as a switching variable that determines which of the topics is used.

Inference for estimating the posterior probabilities of the latent variables can be done with techniques such as Gibb's sampling or expectation maximization. We employ an implemen-

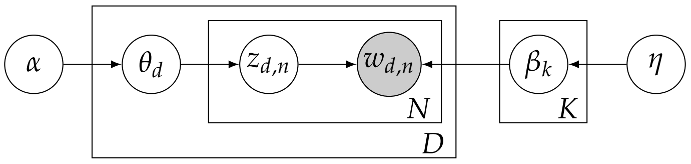


Figure 3.2: Plate notation for Latent Dirichlet Allocation

tation of the variational Bayes algorithm for online LDA introduced by Hoffman, Blei and Bach [19] and which is available at Hoffman's webpage[3].

3.3.1 Experimental Results

LDA has only been evaluated for Stock Market forecasting (see Chapter 5) because there is apparently much more noise contained in this domain than in the films application (Chapter 6). There is a couple of factors that could explain this difference between the two domains.

For most companies, tweets containing their name do not usually refer to their stock but their products, events, promotions, etc. This problem gets even worse when not only the company's name but also its ticker symbol are used in non-stock related sentences. Such is the case, for example, of McDonald's, whose ticker, MCD, is commonly used by people eating at their restaurants.

> *"Filling your bowl up with cereal without checking to see if there's milk in the fridge. #WorseFeeling. Guess it's* ***Mcd****'s for breakfast."*

There are companies whose tickers have other meanings too; searching for tweets with Caterpillar's ticker, CAT, will obviously yield a majority of results related to the animal instead of the company.

Finally, some more peculiar situations can take place. For instance, our collection contains over three thousand tweets wishing their followers a *goog* morning, which turns to be Google's ticker. Not to mention *goog night*, *goog luck* or *goog* time.

On the other hand, movies tend to have multi-word titles, which narrows the possibilities of ambiguity. The described problems obviously also arise in some cases, when titles are short or every-day words, like for the films *One Day*[4] or *Lucky*[5].

In order to find descriptions for relevant and non-relevant tweets and to test the performance of LDA for this task, we have created three different datasets. Doing it by manually classifying and filtering tweets would have been tedious and time-consuming, we have therefore automatically collected the datasets below taking advantage of some features.

- Some users precede stock tickers with the dollar symbol as some sort of hashtag [9]. One of our assumptions was that tweets containing these dollar-tagged symbols are much more likely to be related to the stock market than the others. Unfortunately, these tags are not used as often as it would be desirable and there are only around 16000 of them in our tweet collection. The first dataset contains these 16000 as well as another 16000 chosen at random.

[3] http://www.cs.princeton.edu/~mdhoffma
[4] http://www.imdb.com/title/tt1563738/
[5] http://www.imdb.com/title/tt1473397/

- Another dataset has been created by retrieving tweets from users that usually post to the stock-related messages. This second dataset contains 10000 random tweets from our collection plus a total of 9978 tweets from a list of twenty accounts including *FinancialTimes*, *BBCBusiness*, as well as some investors and bloggers like *Dan Tanner* or *Jim Cramer*.

- Finally, we have created a third dataset by just taking 300000 tweets from our collection. Most of them contain company names.

To measure the performance of the relevance filter we have also created a test set of 4000 tweets, 2000 tweets from the aforementioned stock-related accounts labelled as relevant and another 2000 supposedly irrelevant tweets from Twitter accounts we are fairly certain that do not focus on the stock exchange. These include accounts from musicians, comedians or personal bloggers among others. It should be noted, though, that as these labels have been set automatically, and so large errors may be expected.

LDA models have been trained and tested using the described datasets and setting the number of latent topics to 2. Data is preprocessed following the steps from Section 3.2 except for negation handling, which is not applied. Results are shown in the table below.

Dataset	Accuracy	Precision	Recall	F-measure
Dollar-tagged	54.88%	53.86%	**67.98%**	60.10%
Stock-related accounts	64.21%	64.35%	63.68%	64.01%
300K from collection	**76.16%**	**83.31%**	65.43%	**73.29%**

Table 3.2: LDA performance by dataset

Training the model on the large dataset results in a much better accuracy than with the other two. Still, it would be interesting to run the same experiments with a similar amount of data for the other approaches.

Precision and recall are two metrics typically used in information retrieval and classification for evaluating the relevance of the results. Precision represents the fraction of retrieved messages that are actually relevant, while recall is the fraction of messages that have been retrieved from all the relevant ones. These two metrics are computed as follows:

$$precision = \frac{|\{relevant\ tweets\} \cap \{retrieved\ tweets\}|}{|\{retrieved\ tweets\}|}$$

$$recall = \frac{|\{relevant\ tweets\} \cap \{retrieved\ tweets\}|}{|\{relevant\ tweets\}|}$$

As shown in Table 3.2, the precision of the last model is quite high, meaning that most of the tweets that pass through the filter are, in fact, relevant. However, the recall is rather low and consequently, lots of relevant tweets are not accepted and missed. Thus, the experiments for the stock market prediction application (Chapter 5) will be performed with both filtered and unfiltered data. F-measure is a combination of the two and is formulated as follows

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

3.4 Sentiment Classifier

Naïve Bayes Classifier

In a text classification scenario, let D be the set of all possible documents and let $C = \{c_1,...,c_l\}$ be the set of class labels. Assuming that each of the documents can only belong to one class, C can be seen as a discrete random variable with l possible values. A document $d \in D$ is represented by an n-dimensional vector $(w_1,...,w_n)$ of Bernoulli-distributed variables indicating whether word w_i occurs in the document, just like in the Bag of Words model.

Upon receiving a document the Naïve Bayes classifier assigns the class maximizing the conditional probability given that document. More formally, the output of the classifier, given document d, is

$$\arg\max_{c_i} P(C = c_i | D = d)$$

Applying Bayes' rule we have:

$$P(c_i|d) = \frac{P(d|c_i)P(c_i)}{P(d)}$$

If we knew these posterior probabilities we would have an optimal classifier. We do not, and so it is necessary to estimate them from sample data. Given a training set it is straightforward to compute $\widehat{P(c_i)}$ by just counting the number of documents class c_i out of all the instances of the sample.

Computing $\widehat{P(d|c_i)}$ is more complex, especially if we take into account the fact that the number of features describing the document, i.e. the length of d, can be extremely high. By making the strong assumption that the presence of a feature w_j is independent of the occurrence of any other feature w_k we can write:

$$P(d|c_i) = \prod_{w \in d}^{n} P(w|c_i)$$

Finally, since we are only interested in comparing the posterior probabilities for all c_i, denominator $P(d)$ can be ignored.

Thus, in order to apply this model the only thing we need to do is to estimate the quantities $P(w|c_i)$, which can be done by simple counting with optional smoothing correction.

Multinomial Naïve Bayes

Multinomial Naïve Bayes attempts to improve Naïve Bayes with the idea that the number of times a word appears in a document may give information on the document's class [21]. In this setting, the elements of the document vectors contain the word frequency instead of just indicating the presence. The words of a document are drawn with n independent trials from a multinomial distribution over words.

Let n_{w_i} be the number of times that word w_i appears in a given document. The conditional probability of a document given a class is then defined as:

$$P(d|c_i) = \prod_{w \in d}^{n} P(w|c_i)^{n_w}$$

3.4.1 Experimental Results

Multiple training datasets have been extracted from our tweet collection with an equal number of positive and negative instances in each of them. The total amount of tweets is limited by the number of negatives in the collection. As commented at the beginning of this chapter, retweets have not been included in these datasets. As for the validation datasets

Tweet language	Smiley Location	Training Instances
English (en)	End of tweet	380000
English	Anywhere	600000
Multi-language (ml)	End of tweet	1300000
Multi-language	Anywhere	1800000

Table 3.3: Properties and size of training datasets

that will be used for model selection, two different sets have been created for English and multi-language messages and containing 50000 and 200000 tweets respectively. It should be emphasized that these datasets are also automatically labelled using the smiley approach so large errors are expected.

Multiple experiments have been performed to each of the just mentioned datasets. They basically differ on the preprocessing steps applied to the data and the set of features chosen to represent the documents. The different options used in tables 3.4 and 3.5 are briefly described below:

- Feature List: frequent words with at least 5 occurrences in the data have been extracted from the sentiment analysis training datasets (*Sentiment*) Two additional feature sets have been elaborated for multi-language data, using two million tweets related to the stock market (*Stock*) and another two million for films (*Films*) We consider that while these domain-oriented features might not perform well for a general purpose classifier, they may make a difference when applied adequately
- Hashtag policy: tags can be replaced by a common word `TAG_HASHTAG` (*Replace*)or kept in the data (*Keep*)
- Stemming: whether stemming has been applied to the data (*True/False*)
- Negation: whether the negation handling procedure described in 3.2 has been applied or not (*True/False*)
- Vector values: This column indicates what the values of the document-vector represent. We have tested with term presence (*Presence*), term count within the document (*Count*) and term frequency (*Frequency*)

Pang et al. [28, 27] found that term presence outperforms word frequency in sentiment analysis tasks. Nevertheless, we have executed the tests using term frequency because the different nature of the data may yield different results. In their study they use movie reviews, we, in contrast, use single-sentenced tweets. A priori, though, we do not expect the results to vary much; term presence and term counts will often agree because of the brevity of tweets.

Tables 3.4 and 3.5 show the best ten performing models on the validation set for English and multi-language data. Results have been obtained doing cross-validation on the training sets and using Multinomial Naïve Bayes. The top performing classifiers for each of the different feature lists above are tagged on the right of the tables. The complete list of results can be found in the appendices A.2.2 and A.2.3.

1-6 Smiley Loc.	Hashtag	Stemming	Negation	Vect.Values	Accuracy ▼	
End	Replace	False	False	Frequency	**76.4909%**	(*C-En*)
End	Keep	False	False	Frequency	76.4759%	
End	Keep	True	False	Frequency	76.2107%	
End	Keep	False	False	Count	76.1707%	
End	Replace	True	False	Frequency	76.1707%	
End	Replace	False	False	Count	76.1207%	
End	Keep	False	True	Frequency	76.0656%	
End	Replace	False	True	Frequency	76.0156%	
End	Replace	False	False	Presence	76.0106%	
End	Keep	False	False	Presence	76.0006%	

Table 3.4: Top sentiment classifiers by accuracy on English tweet datasets

Smiley Location	Keep Hashtags	Feat.List	Vect.Values	Accuracy ▼	
Any	Keep	Sentiment	Frequency	**79.5155%**	(*C-Ml*)
Any	Replace	Sentiment	Frequency	79.2775%	
Any	Keep	Sentiment	Presence	78.7368%	
Any	Keep	Sentiment	Count	78.6719%	
Any	Replace	Sentiment	Presence	78.5853%	
		...			
Any	Keep	Stock	Frequency	76.0978%	(*C-Stk*)
Any	Keep	Stock	Presence	76.0978%	
Any	Keep	Stock	Count	76.0112%	
Any	Replace	Films	Presence	76.0112%	(*C-Flm*)
Any	Replace	Films	Frequency	75.8815%	
Any	Replace	Films	Count	75.8166%	

Table 3.5: Top sentiment classifiers by accuracy on multi-language tweet datasets

For both test sets, the best models are obtained by using term frequency instead of presence, although the difference between the two is of only 0.48% and 0.778% for the English and the multi-language datasets respectively. Most surprising is the fact that applying stemming did not have a positive effect on the classifier performance; we are a bit sceptical about this result. Classifiers trained with tweets that initially contained their labelling smiley at the end of the tweet outperform those where the smiley location did not matter. There does not seem to be any significant difference between the two hashtag policies.

In [8], the authors describe a very simple approach on how to use OpinionFinder's[6] sentiment-labelled dictionary to compute the general mood for a given day. The dictionary consists of 2718 positive and 4912 negative words labelled as *weak positive*, *strong positive*, *weak negative* and *strong negative*. For each tweet containing one or more words from the dictionary –no matter whether weak or strong–, the scores of positive and negative messages are increased

[6] `http://www.cs.pitt.edu/mpqa/opinionfinderrelease/`

accordingly. A day's mood is computed as the ratio of positive versus negative messages appearing in the tweets of the given day. Analogously, the sentiment of a tweet following this procedure can be computed by doing this count at the tweet level. Applying this technique to our English datasets resulted in an accuracy of 66% for the tweets with a score different than zero, but only 34% on the total number of tweets in the set.

A Latent Dirichlet Allocation model like the ones in the previous section can also be trained to be used as sentiment classifier. The idea is that by setting the number of topics to 2, LDA should be able to find descriptions for the latent *positive* and *negative* categories. The results for these experiments were a bit deceiving and barely exceeded 50% accuracy. For the multi-language dataset, the most probable words for the first topic were in English while the ones for the second topic belonged from a mix of languages. Thus, the model was not distinguishing messages in terms of sentiment but more in terms of language.

3.5 Sentiment Index

Using the sentiment classifier described in this chapter we can obtain a daily sentiment index that will represent the evolution of the general mood towards a specific item, expressed in terms of one or more filtered words.

This index is represented as a time series where every single value corresponds to the daily percentage of positive tweets over the total number of messages that were posted concerning a specific item.

Other alternatives are left for future work. It would have been interesting to weigh the different messages depending on the number of followers of the author, the idea being that the higher the potential audience of a message, the more confidence or relevance that message should have. Currently, a similar weighting by the number of retweets is indirectly done. During the prediction phase, duplicate tweets are not removed and, since Twitter treats retweets as new messages, we end up with many instances containing the same text. These are obviously assigned to the same class and affect the final value of the index.

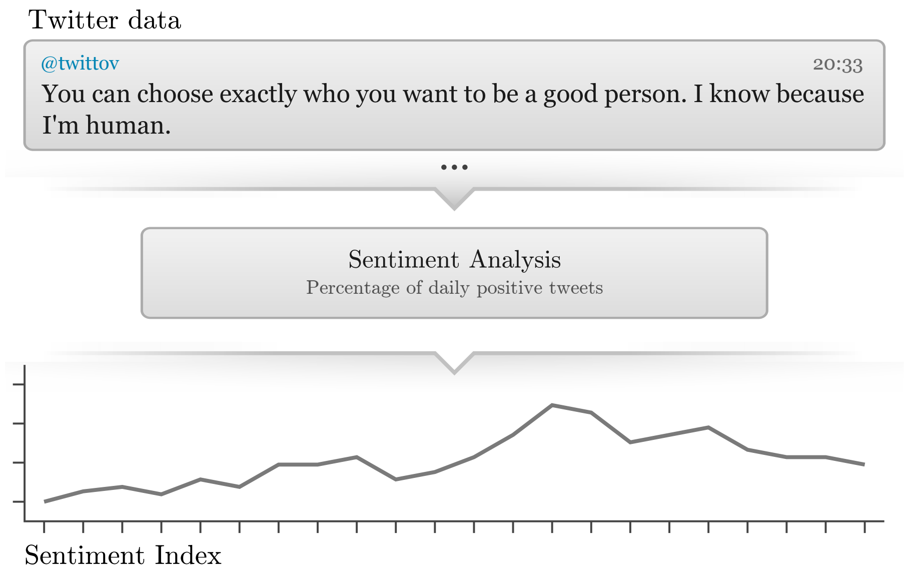

Twitter data
@twittov
20:33
You can choose exactly who you want to be a good person. I know because I'm human.
...
Sentiment Analysis
Percentage of daily positive tweets
Sentiment Index

Figure 3.3: From tweets to Sentiment Index

4

Forecasting time series

4.1 Introduction

The primary goal of this chapter is to describe our framework for prediction of a time series' future values. After processing the messages as it has been explained in Chapter 3, we are left with two different time series: a target time series that we are attempting to predict and the additional series derived from Twitter. The process of forecasting and its validation is done in the three steps described in the following sections (model adequacy, prediction and evaluation).

4.2 Model Adequacy

Although modelling time series involves subjective judgement, some general guidelines should be drawn through statistical testing. Thus, in order to have a better grasp of the adequacy of using Twitter as part of a forecasting model, we run some widely accepted tests to assess, first, if there is a nonlinear relationship between the time series and second, whether there is causality at different lags.

4.2.1 Neglected nonlinearity

The goal of the test for nonlinearity is to ascertain whether a time series or group of time series appears to be generated by a linear model or if they are nonlinearly related. This information can be later used to test for causality with nonlinear tests as well as to experiment with nonlinear models.

The specific meaning of *linearity* in this context is *linearity in conditional mean*. Given the following regression model

$$y_t = m(X_t) + \epsilon_t$$

where X_t is a k-dimensional vector that may contain lagged values of y and $m(X_t) \equiv E(y_t|X_t)$ is the true unknown regression function. y_t is linear in mean conditional on X_t if

$$P\left[E(y_t|X_t) = X_t'\theta^*\right] = 1 \quad \text{for some} \quad \theta^* \in \mathbb{R}^k$$

where θ^* is the parameter vector of the optimal linear least squares approximation to $E(y_t|X_t)$. The null hypothesis of linearity and the alternate hypothesis of interest are as follows:

$$H_0: \quad P\left[E(y_t|X_t) = X_t'\theta^*\right] = 1 \quad \text{for some} \quad \theta^* \in \mathbb{R}^k$$

$$H_1: \quad P\left[E(y_t|X_t) = X_t'\theta^*\right] < 1 \quad \text{for all} \quad \theta \in \mathbb{R}^k$$

When the null hypothesis is rejected the model is said to suffer from *neglected nonlinearity*, meaning that a nonlinear model may provide better forecasts than those obtained with the linear model. The neural network test for neglected nonlinearity formulated by White in [45, 20] uses a single hidden layer feedforward neural network with nonlinear activation functions capable of approximating an arbitrary nonlinear mapping.

For the experiments presented in this document the Teräsvirta linearity test is used. Teräsvirta's test was introduced in [37] and it is based on White's neural network test [45]. An implementation of this algorithm is available in the `tseries` R library.

4.2.2 Granger Causality

The following step is to assess whether there is a causality relationship between the two time series. For this task, we consider the Granger causality test [15]. Given two time series X and Y, X is said to Granger-cause Y if past values of X can be used to get better predictions of Y than using past values of Y alone.

Suppose we are given a bivariate linear autoregressive model M_u on X and Y, where Y depends on p past values from both X and Y as well as another linear autoregressive model M_r that only depends on Y.

$$M_u: \quad y_t = a_0 + a_1 y_{t-1} + ... + a_p y_{t-p} + b_1 x_{t-1} + ... + b_p x_{t-p} + \epsilon_t$$

$$M_r: \quad y_t = a_0 + a_1 y_{t-1} + ... + a_p y_{t-p} + \epsilon_t$$

The null hypothesis that X does not cause Y can be stated as follows:

$$H_0: \quad b_1 = b_2 = ... = b_p = 0$$

Since the unrestricted model M_u has more parameters it should be able to fit the data at least as well as the restricted one M_r. To determine whether M_u is significantly better than M_r a F-test with the Residual Sums of Squares (RSS) can be used:

$$F = \frac{\left(\frac{RSS_r - RSS_u}{p_u - p_r}\right)}{\left(\frac{RSS_u}{n - p_u}\right)}, \quad \text{where} \quad RSS = \sum_{t=1}^{n} (y_t - f(x_t))^2$$

Non-parametric Granger Causality

Note that the parametric Granger test defined above assumes linear dependence between X and Y as well as that the data is normally distributed. This will seldom be the case with the data used in our experiments, hence a non-parametric causality test is preferable. The Granger causality relation between the two time series X and Y can be generally formulated, without assuming any specific model, in terms of the distribution of future values $(Y_t + 1, ..., Y_{t+k})$. X is said to Granger-cause Y if the distribution of these future values conditional on the past observations X_s and Y_x, $s \leq t$, is not equivalent to the distribution of those future values conditional on the past values of Y_s alone. Thus, the tests basically consist in comparing the conditional distribution of Y with and without past values of X.

The tests are limited to detecting Granger causality for $k = 1$ (only one future value is considered) and using finite lags $l_X, l_Y \geq 1$. Having $X_t^{l_X} = (X_{t-l_X+1}, ..., X_t)$ and $Y_t^{l_Y} = (Y_{t-l_Y+1}, ..., Y_t)$ the null hypothesis that X does not Granger-cause Y can be formulated as follows:

$$H_0: \quad Y_{t+1} | (X_t^{l_X}; Y_t^{l_Y}) \sim Y_{t+1} | (Y_t^{l_Y})$$

A non-parametric Granger causality test was formulated by Hiemstra and Jones in [18] although the details on the statistical test for accepting or rejecting this hypothesis are left out of this document. For the experiments described herein, an improved test proposed by Diks and Pachenko in [12] is used. A `C` implementation of the statistical test is available through the author's webpage[1]

4.3 Models

The predicted values for the target time series are obtained by training a machine learning model and providing them with past observations (*lags*) of both time series. Our predictions are limited to guessing a times series' immediate direction, that is, whether the next future value will be higher or lower than the last observation. This is a binary classification task, so the models considered in the experiments have been chosen accordingly. The models that we are going to use are among the most popular and effective in machine learning; more detailed descriptions can be found in standard textbooks such as [23, 17].

4.3.1 Linear Regression

The simplest of the models used in this section is Linear Regression, where the relation between the input and the output is modelled as a linear transformation of the input features.

Given a training sample $\{(y_i, x_{i1}, ... x_{ik}), i = 1, ..., n\}$, the relation between input features x_{ik} and target values y_i can be expressed as:

$$y_i = w_0 + w_1 x_{i1} + ... + w_k x_{ik}$$

The process of fitting the model consists in minimizing the residual sum of squares (RSS) described in the previous section. Once the model has been fitted and the regression coefficient estimations $\widehat{w}_k$ have been found, predictions $\widehat{y}$ for unseen instances can be found by replacing x_k with the unseen instance's features.

4.3.2 Neural Networks

An artificial neural network is a directed graph with neurons on its nodes. Each artificial neuron is a computational unit such that, given two n-dimensional vectors of inputs $(x_1, ..., x_n)$ and weights $(w_1, ..., w_n)$, will compute a certain function y and output a scalar. This function, in turn, is often a composition of a linear combination of the inputs and weights, and a nonlinear activation function φ:

$$y = \varphi\left(\sum_{i=0}^{n} w_i x_i\right)$$

For convenience let w_0 be an additional weight representing a possible bias and set $x_0 = 1$. In practice the activation function φ is usually a sigmoid function like the logistic function or the hyperbolic tangent.

During the training phase, the artificial neural network is adjusted by finding the weights that allow for a better approximation of the target function. Again, the weight adjustment is done by minimizing the residual sum of squares. Weights of the neurons on the hidden layers can be estimated with backpropagation.

[1] `http://research.economics.unsw.edu.au/vpanchenko/`

A simple feedforward neural networks is used for the experiments presented in this document. In these type of networks, each of the neurons in one layer is connected to all the neurons in the previous and next layers. The literature, though, offers an exceptionally large number of other possibilities.

4.3.3 Support Vector Machines

Support vector machines (SVMs) are the last machine learning technique used in this document for obtaining a model from training data. The main idea behind SVMs is that of large margin classification. Suppose we are given a set of linearly separable instances from two classes, then, the simplest solution that classifies the data correctly and that will presumably work best for unseen instances is the one that is equidistant from both classes, i.e. the one with largest margin.

Let $\{(x_i, y_i), i = 1, ..., n\}$ be a set of observed points $x_i \in \mathbb{R}^d$ with an associated class label $y_i \in \{-1, +1\}$. In a geometrical representation of the data, the classes are separated by a hyperplane, the points on which will satisfy $wx_i + b = 0$, where w is a vector of weights normal to the plane and b is the bias. Classification is done with the rule $f(x) = \text{sgn}(wx + b)$ given that all the observed instances must satisfy

$$wx_i + b \geq 1 \quad \text{for } y_i = +1$$
$$wx_i + b \leq 1 \quad \text{for } y_i = -1$$

Parameters w and b can be scaled so that the closest points to the separating hyperplane satisfy $|wx_i + b| = 1$. These points are called *support vectors* and the margin is defined as twice their distance to the hyperplane, $\frac{2}{||w||}$. Consequently, in order to maximize the margin we should minimize $||w||$, subject to the linear separability constraints.

When data is not linearly separable, it can be mapped into another inner product space using a nonlinear function $\phi : \mathbb{R}^d \to F$ where the points will hopefully be much more separable. This procedure is known as the *Kernel Trick* and allows to do the mapping without having to explicitly compute it.

4.4 Evaluation

The last of the steps is to compute some metrics in order to assess the performance of the different models. Time series prediction evaluation is typically done by holding out an independent data set from the training data. The amount of available data in our collection, though, is rather limited in terms of the number of daily observations (see Part II). For this reason, we have taken a prequential approach [13, 4] for evaluating the experiments. For each prediction, a model is fitted with all the available past data. Once the actual value is known, it is included in the training set so it can be used for the next prediction. After repeating this process for all the available observations, we get a contingency table of hits and misses like the one below.

Actual	Predicted		
	Up	Down	
Up	m_{11}	m_{12}	m_{10}
Down	m_{21}	m_{22}	m_{20}
	m_{01}	m_{02}	m

The performance of each of the tested models is then measured with three different metrics:

- **Accuracy:** computed as a simple percentage of correct predictions

  $$accuracy = \frac{m_{11} + m_{22}}{m}$$

- **Cohen's Kappa** [10]**:** this measure takes into account the probability of random agreement between the predicted and the actual observed values and it is computed as

  $$\kappa = \frac{P(a) - P(e)}{1 - P(e)}$$

  where $P(a)$ is the observed agreement and $P(e)$ is the probability of random agreement, that is, the probability that the actual and the predicted coincide assuming independence between predictions and actual values

  $$P(a) = \frac{m_{11} + m_{22}}{m}$$

  $$P(e) = \frac{m_{10}m_{01} + m_{20}m_{02}}{m^2}$$

  Therefore, $\kappa = 1$ when the predicted and the actual values completely agree.

- **Directional Measure:** this metric has been taken from Tsay [38], and it is computed out from the contingency table as

  $$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\left(m_{ij} - m_{i0}m_{0j}/m\right)^2}{m_{i0}m_{0j}/m}$$

  Similar to Cohen's Kappa, large values of χ^2 tell us that the model outperforms the chance of random agreement. χ^2 behaves like a chi-squared distribution with 1 degree of freedom, and we can use this information to compute the quantile with respect to a given significance level.

Part II

Applications

5

Stock Market

5.1 Stock-related Twitter Data

In order to forecast the stock market with Twitter data, the following technological companies were selected for tracking with the methods described in Chapter 2: Apple (AAPL), Google (GOOG), Yahoo! (YHOO), Microsoft (MSFT) Both the company name and ticker symbols in parentheses were tracked covering a time span of eight months from 20 March to 20 November 2011. This translates to roughly 170 working days, after discarding weekends and U.S. federal holidays such as Memorial Day or Independence Day, etc.

The focus on technological companies is due to the availability of a higher volume of user generated messages than, for instance, companies from the energy or healthcare sectors. The reason not to track more companies was mainly the rate limit described in Section 2.2.1.

Apart from using all messages to train the predictors, we also split them into different datasets depending on the company they are related to. The number of available tweets in each dataset is shown below:

Dataset	Multi-language	English
Yahoo!	2881410	2048301
Google	2353057	1076108
Apple	1588157	1204255
Microsoft	29790	21262
All previous four	6852414	4349926

Table 5.1: Available tweets per company

Some stock market indices such as Standard&Poor's S&P100 and S&P500 are obtained by combining the prices of large capitalization corporations and weighting them based on the market capitalizations (share price $\times$ number of shares). The companies listed above are components of these S&P's indices and both Apple and Microsoft are in the top ten market capitalization companies. Therefore, we believe that the combination of the tweets related to these companies can be of good use to predict these indices as well.

5.2 Stock Market Data

There are currently a wide variety of websites that offer daily stock market data for download. The historic prices used in this project were retrieved from Yahoo! Finance[1]. Among the multiple daily values offered by this service, we are particularly interested on the Adjusted Close which corresponds to the closing price once it has been updated to include any

[1] http://finance.yahoo.com

dividend payment and corporate actions, such as splits of the value, occurred at any time prior to the next day's open.

The following tickers are targeted: Apple (AAPL), Google (GOOG), Yahoo! (YHOO), Microsoft (MSFT), S&P100 (OEX), S&P100's implied volatility (VXO), S&P500 (GSPC) and S&P500's implied volatility (VIX).

Price Returns

Instead of directly working with adjusted closes we will focus on the price returns, which is the difference in the price, or the benefit. Returns are computed using the following equation:

$$R_t = \frac{P_t}{P_{t-1}} - 1$$

Assuming that the prices come from a log normal distribution, then their logarithm is normally distributed. Thus, using logarithmic returns can be more convenient when working with statistical methods that assume normality. Moreover, for small changes, returns are approximately equal to their logarithmic counterparts, which are computed as follows:

$$r_t = \ln(1 + R_t) = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

Volatility

Historic daily volatility for a specific company can be estimated with the price returns of the *m* past days for that company. In our experiments a 30-day exponentially-weighted moving average (EWMA) is used. The main idea behind EWMA is that the returns of the last few days should have a greater impact on the volatility than the returns of last month. This is achieved by exponentially decreasing the weight of the price returns as we get further in the past.

$$Vn = (1 - \lambda) \sum_{i=1}^{m} \lambda^{i-1} R_{n-i}^2$$

One of the advantages of using EWMA in favour of other options is that it can be computed recursively with just the previous days' volatility and squared return:

$$Vn = \lambda V_{n-1} + (1 - \lambda) R_{n-i}^2$$

where λ is usually set to 0.94.

Implied volatilities for stock indices such as S&P100 (OEX) or S&P500 (GSPC) are much more complex and are computed with a pricing model such as Black&Scholes. Implied volatilities are available from Yahoo! Finance as separate indices. S&P100 and S&P500 have implied volatilities VXO and VIX, respectively.

Standardization

The standard score *z* is employed for transforming the time series to have zero mean and a standard deviation of one. This kind of normalization is important when various attributes, such as tweet volume and adjusted close are expressed in different units. Applying the formula below will bring all the series to the same scale.

$$z = \frac{x - \mu}{\sigma}$$

One of the drawbacks of standardizing is that if the data has outliers, it will scale to a small interval. Winsorization could be a better alternative to avoid this. With this transformation, the most extreme values, e.g. 5% percentiles, are moved towards the center. However, only basic standardization is used for the experiments in this project.

5.3 Parameter Combinations

With the intention of doing an extensive study on the predictability of stock market time series, many combinations of the parameters listed below have been tested.

- Predicted symbol: the ticker symbol of the target company or stock index for which we want to make predictions.
- Predicted Series: this parameter establishes the target time series that we are attempting to predict. In the case of companies this can be set to *Adjusted Close* or *Volatility*. For implied volatility indices, we only consider the volatility, which is given as their closing price.
- Twitter Dataset: defines from which of the datasets defined in Section 5.1 should the additional time series be extracted.
- Additional Series: sets what kind of transformation should be applied to Twitter data in order to create a time series from it. Four possible values are defined: daily tweet volume time series, the Sentiment Index, the sentiment computed using OpinionFinder and a fourth possibility combining values from the volume and the computed Sentiment Index.
- LDA: whether Latent Dirichlet Allocation (LDA) has been applied to filter out non-relevant messages before creating the additional series to add to the predictions.
- Classifier: this parameter describes the classifier used to perform the sentiment predictions. The experiments in this section have been done by employing the three best scoring classifiers for English (*C-En*), Multi-language (*C-Ml*) and Stock (*C-Stk*) collections and feature lists. Specific parameters of these classifiers can be found in Section 3.4.1 with the abbreviations next to them.
- Model Family: refers to one of the three models described in Section 4.3: support vector machines, with its different kernel flavours polynomial, radial and sigmoid; different-degree neural networks or linear models.
- Lags: number of the most recent past values which are provided to the classifier. It should be noted that the higher the number of lags, the less days that are available for training.

5.4 Case Study: Apple Inc.

From the many experiments that have been performed, one of them has been selected for this section, to be studied in more depth. Below are the results for predicting the AAPL ticker using the Apple dataset after applying LDA-filtering to remove non-relevant tweets. The *C-Ml* classifier from Section 3.4.1 is used to build the Sentiment Index.

5.4.1 Graphical Representation

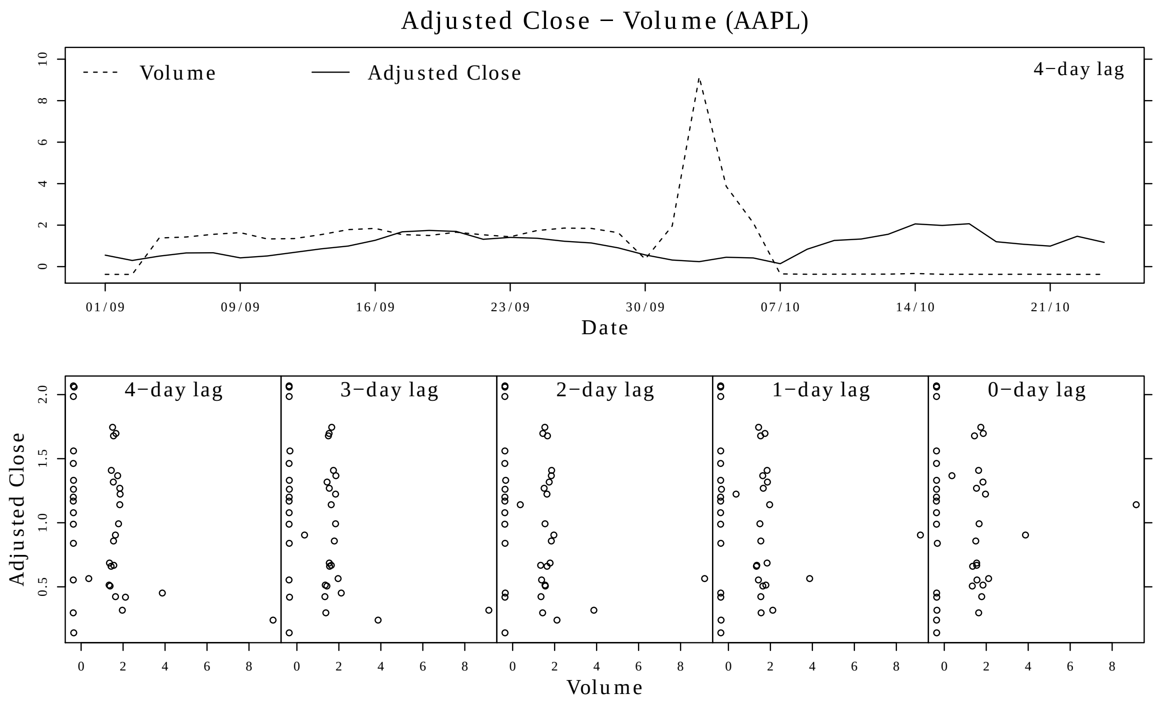


Figure 5.1: Adjusted close versus daily tweet volume for the Apple stock

The adjusted close and tweet volume time series for the period between 1 September and 31 October 2011 are depicted in Figure 5.1. The peak corresponds to the death of a former CEO on 5 October. It may seem that tweet volume is 0 from the peak onwards; this is because of the standardization applied to the time series.

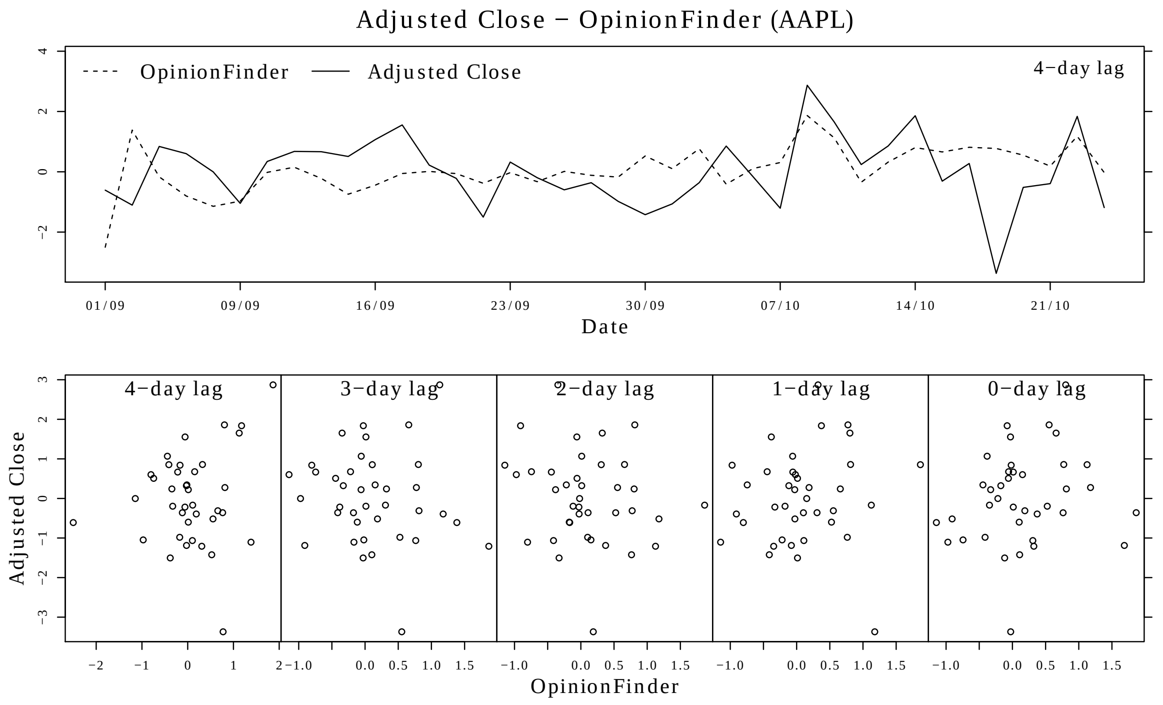


Figure 5.2: Adjusted close versus OpinionFinder sentiment for the Apple stock

The evolution of the adjusted close log returns along with the sentiment index is represented in Figure 5.3, for the same period as before, and in Figure 5.4 for the whole period for which we collected messages.

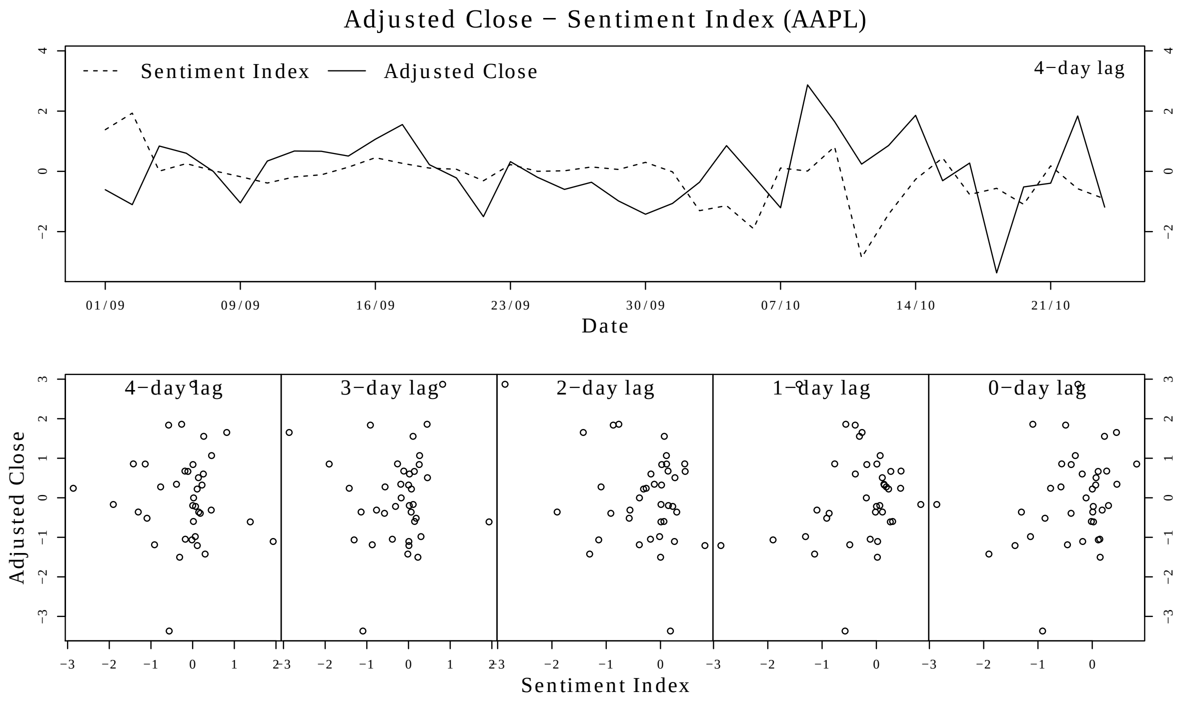


Figure 5.3: Adjusted close versus the Sentiment Index for the Apple stock

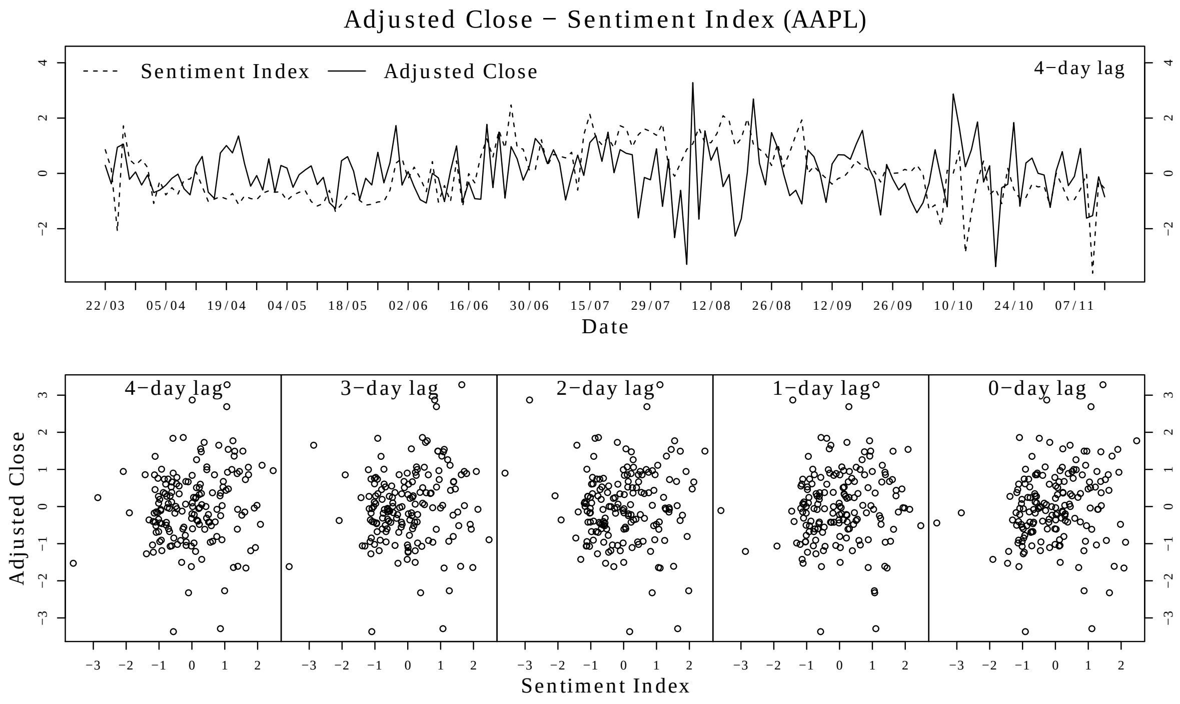


Figure 5.4: Adjusted close versus the Sentiment Index for the Apple stock. This chart shows data from the full period in which tweets were collected.

The scatter plots in the lower part of the figures set out to make it easier to see a visual relationship between the series. However, it is difficult to see a clear correlation from any of the previous charts, specially from the last one, since sentiment data appears to be very irregular. Thus, in the next section the statistical tests introduced in Chapter 4.2 will be used for a more in-depth study of the relation between the time series.

5.4.2 Model Adequacy

Nonlinearity

Teräsvirta's neural network test for neglected non-linearity yielded the p-values shown in the table below. For a 95% confidence interval, the tests suggest that there is neglected nonlinearity between the volatility and the tweet volume as well as the volume and the adjusted close (the experiments are actually done for the logarithmic returns of the adjusted close, but we refer to them as adjusted close for the sake of brevity). Reducing the confidence to 90% p-values for the test between the Sentiment Index and the adjusted close are also below the significance level.

Volume $\downarrow$ Volatility	Volatility $\downarrow$ Volume	Close $\downarrow$ Volume	Volume $\downarrow$ Close
0.4998	**0.0036**	0.4362	**0.0130**

OpinionFinder $\downarrow$ Volatility	Volatility $\downarrow$ OpinionFinder	Close $\downarrow$ OpinionFinder	OpinionFinder $\downarrow$ Close
0.9543	0.8067	0.9236	0.2786

Sentiment $\downarrow$ Volatility	Volatility $\downarrow$ Sentiment	Close $\downarrow$ Sentiment	Sentiment $\downarrow$ Close
0.4509	0.9046	**0.0603**	**0.0802**

Table 5.2: p-values for the neglected nonlinearity tests

Causality

The three tables below reproduce the p-values for the causality tests between target and additional time series. The tests are done in both directions, to and from the additional series, and by shifting the values from 1 to 5 days. In the tables, arrows indicate the direction of the causality and all p-values lower than 0.1 are shown in bold.

The parametric Granger tests suggest that there is causality from all three additional time series to the adjusted close with a lag of 5 days. For the Sentiment Index, while not in the 95% confidence interval, the p-value is still very low, much lower than the p-value for the opposite direction. This results are in the line with the idea that what happens in Twitter will reflect on the market with some delay.

The non-parametric version of the tests only show causality for the OpinionFinder series (Table 5.4), but not in the direction of interest. The tests indicate that there might be causality from the adjusted close to the additional series for a 5-day lag, although p-values are also quite low for lags from 1 to 4. When applied to the Sentiment Index and a lag greater than 2, these tests yield similar non-significant p-values for both of the target time series and in both directions.

	Volatility ↓ Volume	Volume ↓ Volatility	Close ↓ Volume	Volume ↓ Close
Lag	**Granger Causality**			
1	0.6015	0.3973	0.2326	0.4570
2	0.8184	0.7958	0.2805	0.6653
3	0.9357	0.3695	0.2622	0.7518
4	0.9009	**0.0027**	0.3506	**0.0041**
5	0.8157	**0.0018**	0.4372	**0.0098**
Lag	**Non-parametric Granger**			
1	0.8102	0.6735	0.2300	0.6354
2	0.5934	0.8480	0.2026	0.6824
3	0.4680	0.3579	0.2808	0.2792
4	0.1248	0.2984	0.2581	0.1291
5	0.1599	0.3202	0.6239	0.1505

Table 5.3: Tweet Volume. This table shows the p-values for both the Granger and non-parametric Granger causality tests between the Tweet Volume time series and the two targets, volatility and adjusted close.

	Volatility ↓ OpFinder	OpFinder ↓ Volatility	Close ↓ OpFinder	OpFinder ↓ Close
Lag	**Granger Causality**			
1	**0.0737**	0.4946	0.2589	0.6977
2	0.1968	0.7682	0.1921	0.3413
3	0.3879	0.8458	0.3487	**0.0219**
4	0.4404	0.9451	0.5110	**0.0405**
5	0.2325	0.9355	0.6316	**0.0318**
Lag	**Non-parametric Granger**			
1	0.5567	0.8293	0.3228	0.1619
2	0.1721	0.9627	**0.0842**	**0.0617**
3	0.2676	0.9517	0.1631	0.3230
4	0.1734	0.8784	**0.0809**	0.3646
5	**0.0572**	0.7750	**0.0208**	0.3404

Table 5.4: Causality results for the OpinionFinder time series.

Lag	Volatility ↓ Sentiment	Sentiment ↓ Volatility	Close ↓ Sentiment	Sentiment ↓ Close
	Granger Causality			
1	0.4154	0.2930	0.2144	0.6660
2	0.7505	0.5591	0.4691	0.9296
3	0.6457	0.7595	0.6786	0.8263
4	0.5470	0.7938	0.6263	0.4453
5	0.7408	0.8084	0.5988	**0.0793**
	Non-parametric Granger			
1	0.5983	0.9742	0.6660	0.6544
2	0.3033	0.4125	0.3675	0.4677
3	0.3109	0.4503	0.3084	0.1974
4	0.4656	0.5307	0.3292	0.2526
5	0.7158	0.7490	0.3105	0.3413

Table 5.5: Causality results for our Sentiment Index.

5.4.3 Model fitting and Evaluation

We begin by showing, in Table 5.6, some results obtained by adding the daily tweet volume information for the prediction of AAPL's adjusted close. Even though the statistical tests suggest causality for the greater lags from the daily message volume to the adjusted close, the models do not succeed in predicting the direction of the adjusted close when combining it with Tweet Volume. We do not find this surprising given the little amount of information contained in the number of daily messages .

The accuracies are not consistent with the causality tests from the previous section and both the Kappa and Directional Measures give a very low confidence for lags greater than 1.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.5272	0.5878	0.0513	0.1747	0.5036	0.0246
2	0.5151	0.5030	0.0252	0.0048	0.7363	0.9502
3	0.4878	0.4451	-0.0317	-0.1081	0.6727	0.1646
4	0.5214	0.4723	0.0400	-0.0527	0.6054	0.4962
5	0.5350	0.4840	0.0702	-0.0309	0.3783	0.6951

Table 5.6: Effect of adding past values of daily message volume for prediction of the Apple's stock adjusted close. SVM with a sigmoid kernel. The table shows a comparison of accuracies (Acc.), Cohen's Kappa statistics (Kappa) and Directional Measure p-values (DM) depending on whether Twitter information was used (w/) or not (w/o) for the predictions.

As for the prediction of the volatility, the results are slightly better. As can be seen in Tables 5.7 and 5.8, accuracies improve in a consistent way to the causality tests in the previous section. For those lags with a high p-value in the causality tests, the predictions drop dramatically. In contrast, lags with a low p-value result in better accuracies when adding the volume information. Cohen's Kappa and the directional measure also improve, indicating that the positive effect of adding tweet volume is not due to chance.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.6385	0.6445	-0.0338	0.0231	0.4745	0.6962
2	0.6606	0.6000	0.0666	-0.0645	0.2505	0.3216
3	0.6158	0.6524	0.0088	0.1032	0.8989	0.1374
4	0.6257	0.6196	0.0287	0.0377	0.6733	0.5987
5	0.6172	0.6234	0.0454	0.0654	0.5337	0.3734

Table 5.7: Effect of adding past values of the Tweet Volume when predicting the direction of the Apple's stock volatility. Linear Regression.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.5757	0.5333	-0.0396	-0.1379	0.5975	0.0671
2	0.5757	0.5090	0.0000	-0.1198	1.0000	0.1234
3	0.5364	0.5695	-0.0432	0.0434	0.5926	0.5922
4	0.5629	0.5894	-0.0006	0.0760	0.9939	0.3469
5	0.5629	0.5960	0.0079	0.0463	0.9211	0.5505

Table 5.8: Effect of adding past values of the Tweet Volume when predicting the direction of the Apple's stock volatility. SVM with a sigmoid kernel.

The rest of the experiments focus on using Twitter sentiment instead of the message volume. Table 5.9 shows how the accuracy of the predictions decreases when using past values of the the Sentiment Index. This is not surprising because the causality tests from Table 5.5 already indicated that no causal relationship should be expected between the two series. Results of adding OpinionFinder sentiment index for predicting the volatility are similar and are not included here.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.6909	0.6121	0.3825	0.2233	0.0000	0.0040
2	0.6181	0.5454	0.2233	0.0863	0.0037	0.2510
3	0.6135	0.5460	0.2248	0.0884	0.0037	0.2495
4	0.5828	0.5582	0.1642	0.1158	0.0355	0.1389
5	0.5493	0.5432	0.0993	0.0860	0.2053	0.2735

Table 5.9: Effect of adding past values of the Sentiment Index when predicting the direction of the Apple's stock volatility. SVM with a sigmoid kernel.

Finally, tables 5.10 and 5.11 show the effect of adding OpinionFinder and the Sentiment Index for the prediction of AAPL's adjusted close. As before, these results are largely consistent with the output of the causality tests from the previous section, having a large improvement for a 5-day-lag.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.6000	0.5575	0.2015	0.1138	0.0091	0.1427
2	0.5818	0.5515	0.1646	0.1031	0.0337	0.1851
3	0.5609	0.5914	0.1209	0.1807	0.1214	0.0201
4	0.5214	0.5705	0.0426	0.1408	0.5863	0.0722
5	0.5370	0.6172	0.0715	0.2331	0.3608	0.0029

Table 5.10: Effect of adding sentiment information from the OpinionFinder series when predicting the direction of Apple's adjusted close. SVM with sigmoid kernel.

Lag	Acc. w/o	Acc. w/	Kappa w/o	Kappa w/	DM w/o	DM w/
1	0.6000	0.5878	0.2015	0.1762	0.0091	0.0234
2	0.5818	0.6424	0.1646	0.2857	0.0337	0.0002
3	0.5609	0.6280	0.1209	0.2558	0.1214	0.0010
4	0.5214	0.6012	0.0426	0.2009	0.5863	0.0100
5	0.5370	0.6358	0.0715	0.2713	0.3608	0.0005

Table 5.11: Effect of adding the Sentiment Index series when predicting the direction of Apple's adjusted close. SVM with sigmoid kernel.

5.5 Summary Tree

All the different parameter combinations result in a battery of 43200 different experiments that produce tables similar to the ones in the previous section. Combining all these tables and distinguishing the rows by adding the set of parameters that were used to generate them, we end up in another table similar to the one below.

Target	Dataset	R.Filter	Model	Lag	...	Acc. w/o	Acc. w/	...
GOOG	GOOG	TRUE	SVM(sigmoid)	2	...	0.567	0.578	...
GOOG	GOOG	TRUE	SVM(sigmoid)	3	...	0.578	0.589	...
GOOG	GOOG	TRUE	SVM(sigmoid)	4	...	0.589	0.612	...
...	...	...	...	...	...	...	...	
VIX	YHOO	TRUE	SVM(sigmoid)	3	...	0.521	0.601	...
VIX	YHOO	TRUE	SVM(sigmoid)	4	...	0.537	0.635	...
...	...	...	...	...	...	...	...	...

Attempting to draw any conclusions by just looking at such a long table of numbers is overwhelming and hence a way of summarizing this data is needed. We have tackled this problem by automatically generating a decision tree that will tell us, in general, which of the different parameter combinations lead to an increase of predictive power. Each of the intermediate nodes of the tree have a parameter assigned to them. Different branches spring from these nodes depending on the value of the parameter assigned to the given node. The leaves of the tree are tagged with the result (improves/does not improve) of the given branch. Leaves also contain information on how many of the instances in that branch behave in the same way.

The tree is generated using `REPTree`, a decision tree learner available in Weka[2] that builds trees by greedily selecting, for each node, which of the parameter will result in a higher information gain. One of the characteristics of the REPTree algorithm is that the height of the tree can be limited in advance.

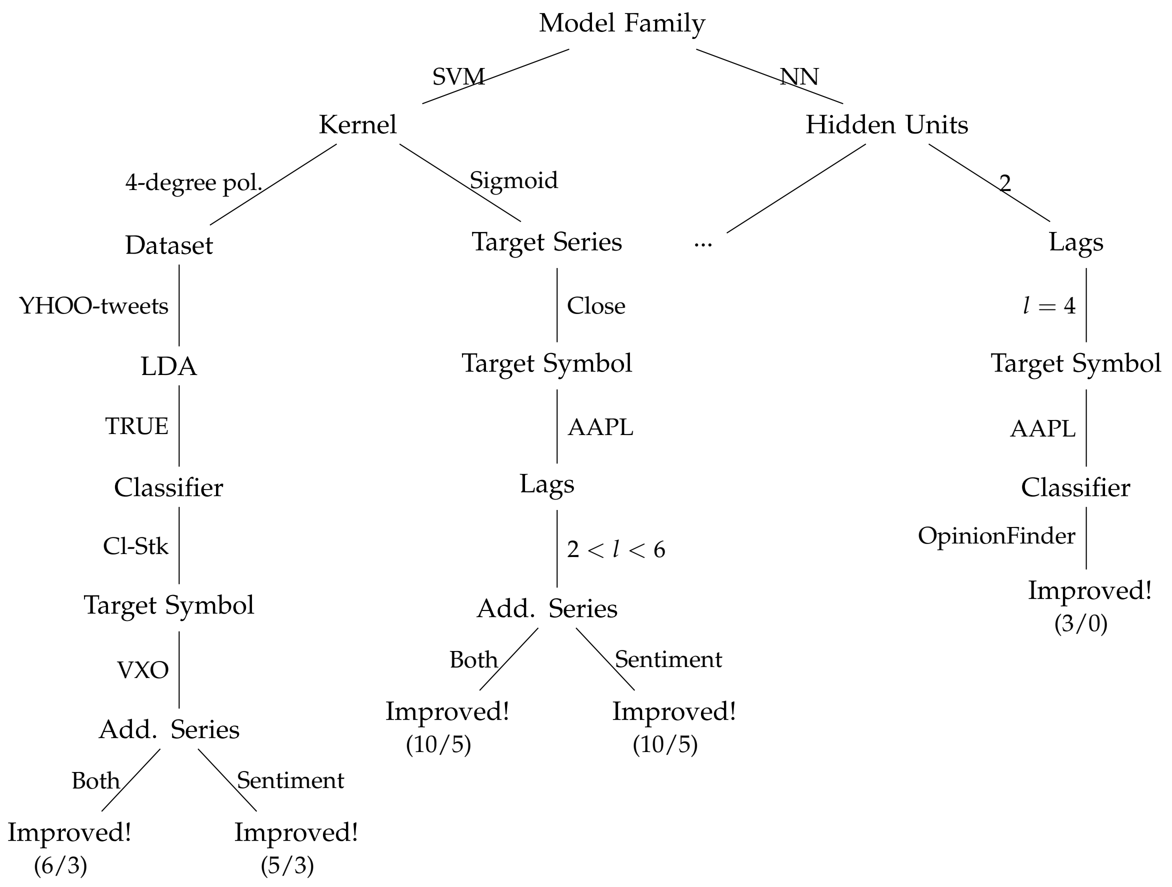


Figure 5.5: Summary tree for accuracy improvements of at least 5%. See A.2.4 for the complete tree.

For the tree in Figure 5.5, a model is considered to have improved when it has an accuracy greater than 50% and adding the additional time series from Twitter results in an increase in prediction accuracy of 5% or more. For the sake of readability, we have pruned the tree to only show those paths that lead to an increase of performance. The complete tree is reproduced in Appendix A.2.4 in a more space-efficient format.

It should be noted that branches corresponding to unsuccessful models must also be considered in order to draw any conclusions from the tree. For instance, for the stock market application we see that using linear regression almost never leads to an improvement in the performance. From all the experiments done with this kind of models, only 27 were successful in contrast to 1973 for which the predictions did not increase, or did so to a much lesser extent.

This kind of summary trees contribute to our analysis in that they help identifying sets of parameters – defined by the branches – that lead to similar results. Even if the branches

[2] http://www.cs.waikato.ac.nz/ml/weka/

might be over-specific they can still serve as a guide for deciding in what direction should we attempt to derive more general conclusions.

Taking the parameters defined by the leftmost branch, we have analysed how do the same set of parameters perform for the three different model families we consider (linear models, support vector machines, and neural networks). The results table contains 110 experiments for predicting the VXO index by using LDA-filtered tweets from the YHOO dataset and the *Cl-Stk* classifier along with the Tweet Volume (additional series is set to *Both*). In general, 56 out of these 110 experiments, that is 50.90%, yield successful results, but by separately looking at the different model families, some more information on the real capabilities of each family can be inferred. This is shown in Table 5.12 and, as can be seen, a majority of SVMs have successful results for this specific set of parameters.

Model family	Successful	Unsuccessful	Success rate
Linear Regression	4	6	40.00%
Neural Networks	17	23	42.50%
Support Vector Machines	35	25	58.33%

Table 5.12: Success rates by model family for predicting the VXO index by using the described set of parameters.

Analogously, if we take the rightmost branch of the tree and compare it for the different model families, we have a general success rate of 54.54%. Looking at the different models separately, we have that SVMs perform very poorly for this second set of parameters, with just a 29.16% success rate. Results are listed in table 5.13. The use of the model family just serves as an example. This same analysis could be done by generalising any of the other parameters.

Model family	Successful	Unsuccessful	Success rate
Linear Regression	3	1	75.00%
Neural Networks	14	2	87.50%
Support Vector Machines	7	17	29.16%

Table 5.13: Success rates by model family for predicting the VXO index by using the second set of parameters.

6
Box Office

6.1 Film-related Twitter Data

The task of collecting tweets in the movie sales domain is a bit more complex. In contrast to the stock market, films have a short life span and are only mentioned for a limited time. Thus, the selection of titles that were tracked evolved over time.

A list of upcoming releases is maintained at *IMDB*[1]. Using this list, our tracking system started retrieving messages for upcoming movies two weeks before the release date and it kept collecting data for a minimum of four weeks. Movies still in the top ten box office ranking after two weeks from release, were still tracked. Due to Twitter API restrictions the number of movies was limited to 50.

The box office dataset that we collected spans from late June to August 2011 and consists of more than 100 million tweets from a total of 121 different films. However, only a small fraction of this data is used in the experiments presented in this section. Different datasets of tweets for eight of the most popular movies were created and are listed in the table below. For the sake of brevity, we have assigned a two-letter code to each of the selected films, we adhere to this abbreviations for the rest of this chapter.

Film	Code	Tweets	Days	Tweets/Day ▼
Harry Potter and the Deathly Hallows: Part 2	*hp*	1323779	41	32287
Captain America: The First Avenger	*ca*	572064	34	16825
Rise of the Planet of the Apes	*ra*	258760	20	12938
Super 8	*s8*	491017	75	6546
Cars 2	*c2*	284155	50	5683
The Smurfs	*sf*	144847	27	5364
Cowboys & Aliens	*cw*	136838	27	5068
Green Lantern	*gl*	325582	69	4718

Table 6.2: Available data from release until 24 August 2011

6.2 Box Office Data

The other piece of information needed for the experiments is the weekly movie sales. *The Numbers*[2] offers daily U.S. box office information of the top 50 movies being screened on that day. Among other information, this webpage offers the daily gross revenue, change with respect to the previous day, number of theatres in which the film is screened, number of days since release and the total gross revenue. We scraped this data on a daily basis.

As was the case for the stock market, instead of directly working with the gross revenue, the logarithmic returns of the series has been computed.

[1] http://www.imdb.com/movies-in-theaters/

[2] http://www.the-numbers.com/charts/today.php

6.3 Parameter Combinations

The parameters for the box office experiments are, to a large extent, the same than the ones for stock market prediction. Only the following parameters differ:

- Predicted film: defines which of the films' box office listed in Table 6.2 we are trying to predict.
- Predicted Series: Unlike the stock market experiments, here we only consider the daily gross revenue for the movie under study. Therefore, this parameter does not apply in this context.
- Classifier: The three best-scoring sentiment classifiers are tested. These are the ones obtained from the English (*C-En*) and multi-language (*C-Ml*) datasets, and for the film feature list (*C-Flm*). These classifiers are highlighted in tables 3.4 and 3.5 in Section 3.4.1.

6.4 Case Study: Captain America

Among the eight films for which we performed the experiments, two case studies have been selected to be discussed in-depth. The first of the movies is *Captain America: The First Avenger* and the results in this section correspond to the experiments using the non-LDA-filtered version of the tweets dataset and the *C-Flm* sentiment classifier.

6.4.1 Graphical Representation

Figure 6.1 shows the graphical representation of the gross revenue and tweet volume time series for the period commencing from 22 July to 23 August 2011. *Captain America* was released on 22 July in the U.S. As can be seen in the figure, the volume and the gross are apparently closely related.

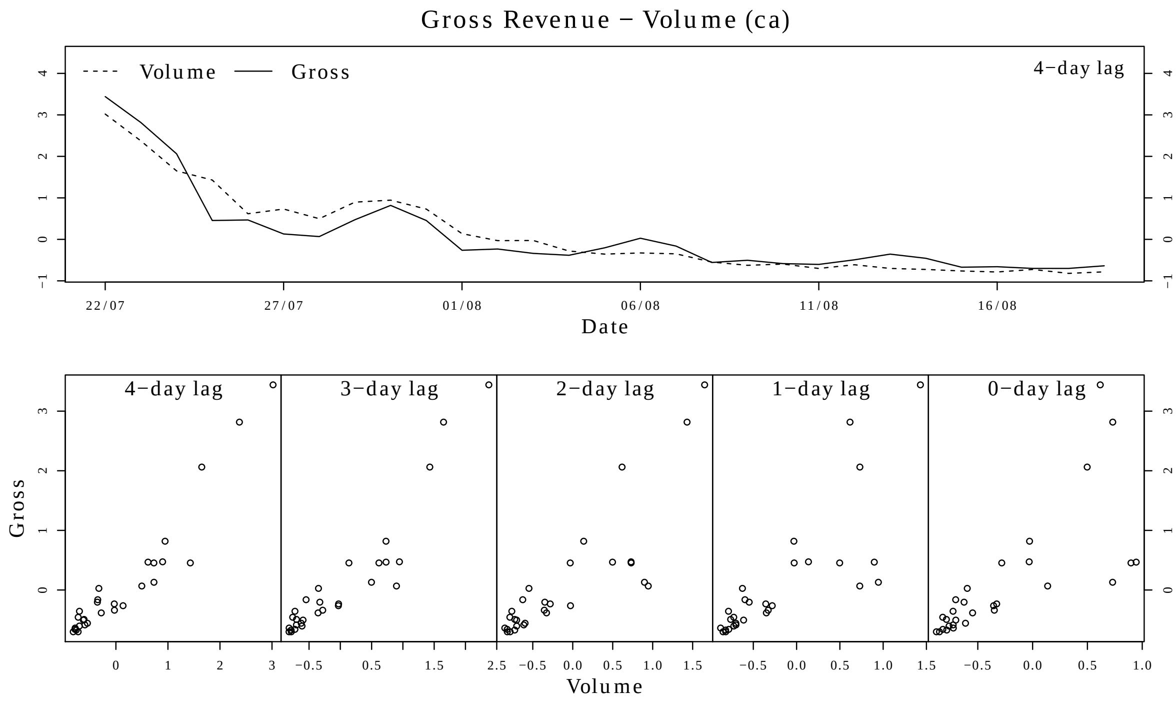


Figure 6.1: Gross revenue versus daily tweet volume for Captain America

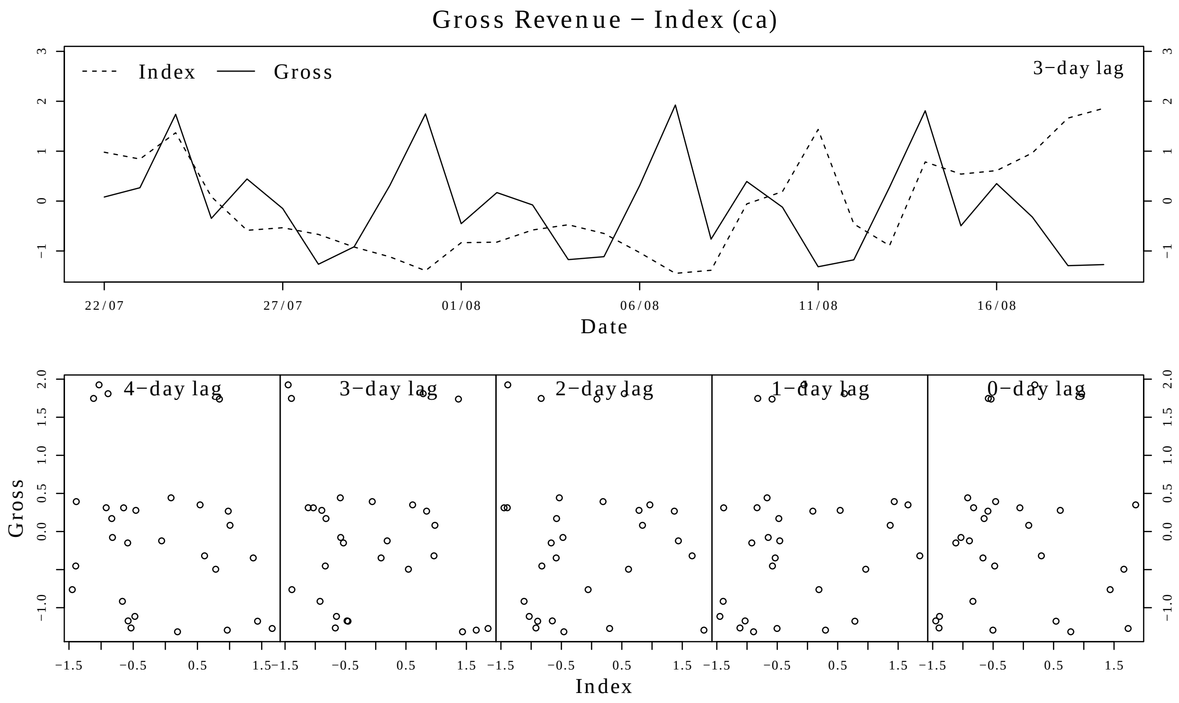


Figure 6.2: Gross revenue versus sentiment index for Captain America

6.4.2 Model Adequacy

Nonlinearity

Teräsvirta's neural network test for neglected nonlinearity yielded the p-values shown in the table below. The results clearly point to a nonlinear relationship between volume and gross revenue, indicated by a strong reject of the null hypothesis of linearity.

Gross ↓ Volume	Volume ↓ Gross	Gross ↓ Sentiment	Sentiment ↓ Gross
0.0004	**6.02e-05**	0.6505	0.9865

Causality

The causality tests results in the tables below suggest that there is causality between the daily volume of messages and the gross revenue.

Lag	Gross ↓ Volume	Volume ↓ Gross	Gross ↓ Sentiment	Sentiment ↓ Gross
	Granger Causality			
1	0.8826	**0.0335**	0.5223	0.3050
2	**0.0002**	**0.0034**	0.3050	0.5254
3	**0.0003**	**3.55e-05**	0.4005	0.8679
4	**9.02e-05**	**0.0017**	0.4766	0.8319

While the results of the Granger test show causality in both directions, the non-parametric version of the tests give a much lower p-value for de direction from volume to gross revenue, which is the direction of interest.

	Gross $\downarrow$ Volume	Volume $\downarrow$ Gross	Gross $\downarrow$ Sentiment	Sentiment $\downarrow$ Gross
Lag	Non-parametric Granger			
1	0.8403	0.1547	0.3594	0.5332
2	0.7969	0.1527	0.6302	0.6087
3	0.7986	0.1522	0.7470	0.9064
4	0.7541	0.1501	0.2705	0.7074

Table 6.3: p-values for the Granger and non-parametric Granger causality tests between the Tweet Volume, the Sentiment Index and the daily gross.

6.4.3 Model fitting and Evaluation

Below are the results after predicting Captain America's box office with the addition of the Tweet Volume and the Sentiment Index. Table 6.4 shows the difference in accuracies when the Tweet Volume is added to the prediction using linear regression. The results are fairly consistent with the causality suggested by the parametric Granger test. Only for lag 4 there is a drop in the accuracy. That might be explained by a much lower causality p-value (0.000092 in contrast to 0.0017) in the direction from gross revenue to Volume.

Lag	Acc w/o	Acc w/	Kappa w/o	Kappa w/
1	0.4137	0.4827	-0.0979	-0.0847
2	0.3571	0.4642	-0.1290	-0.1290
3	0.2962	0.4814	-0.3902	-0.0161
4	0.6538	0.5000	0.3157	0.0451

Table 6.4: Accuracy and Kappa comparison after predicting the direction of the daily gross by adding the Tweet Volume. Linear Regression.

As can be seen in the table, there is a 10% increase in accuracy only for lag one. Trying to add older sentiment information has a terrible effect on the results. Results vary much from one model to the other, we suspect that such a low performance might be influenced by the fact that training is done with very few instances, between 26 and 29, depending on the lag.

Lag	Acc w/o	Acc w/	Kappa w/o	Kappa w/
1	0.5517	0.6551	0.0407	0.2892
2	0.8571	0.7500	0.7142	0.5050
3	0.8888	0.6296	0.7665	0.1614
4	0.8400	0.5600	0.6774	0.0350

Table 6.5: Accuracy and Kappa comparison after predicting the direction of the daily gross by adding sentiment information. SVM with a radial kernel.

6.5 Case Study: Super 8

The second of the case study is dedicated to *Super 8*. Once again, the results in this section are the ones obtained by using the non-LDA-filtered version of the tweets dataset and the *C-Flm* sentiment classifier.

6.5.1 Graphical Representation

Figures 6.3 and 6.4 show a comparison between the target and additional time series for the film under study for the period commencing on 11 June to 23 August 2011. In the first figure we can see one of the problems of working with non-localized Twitter messages and use them together with country-specific data. *Super 8* was released on 10 June in the U.S., which accounts for the first weeks depicted in the graph where both time series gradually decrease. Then, on 4 August, the film was released in the United Kingdom and the tweet volume increased again.

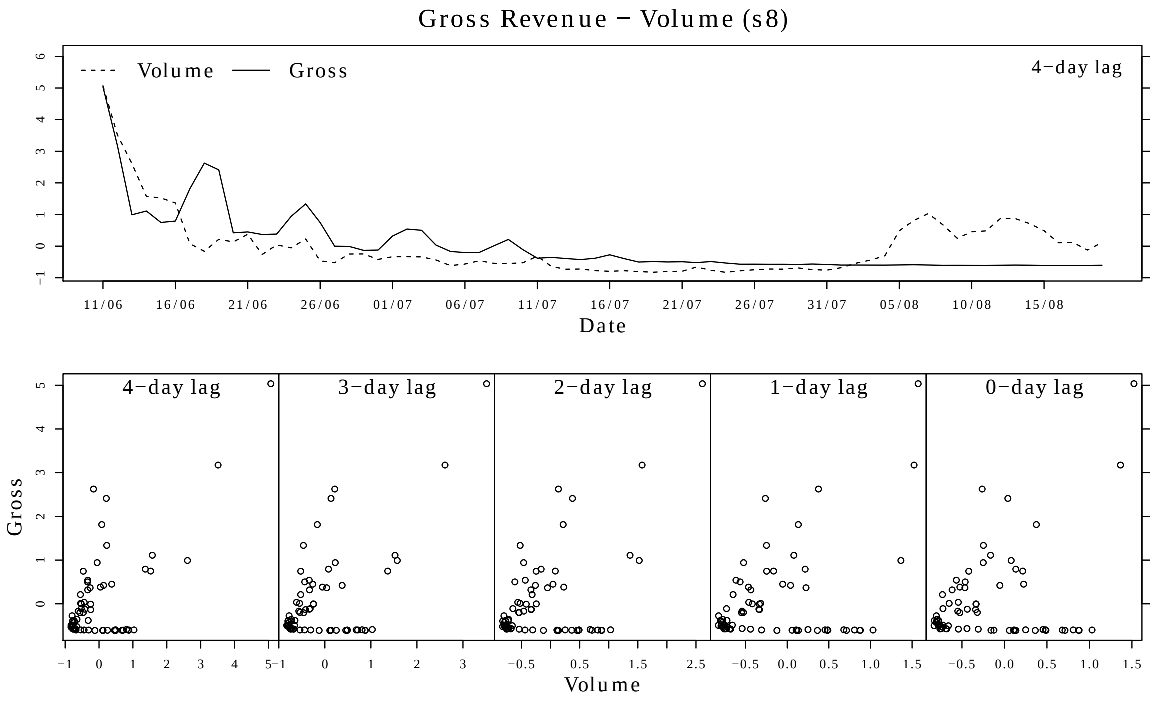
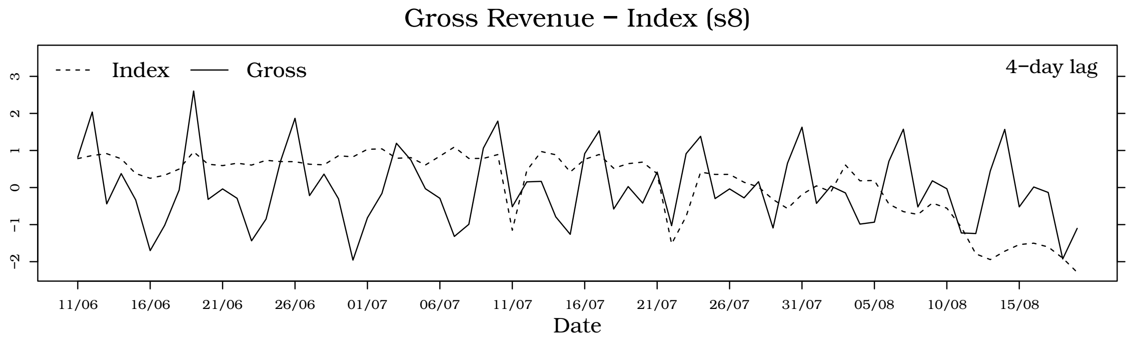


Figure 6.3: Gross revenue versus daily tweet volume for Super 8



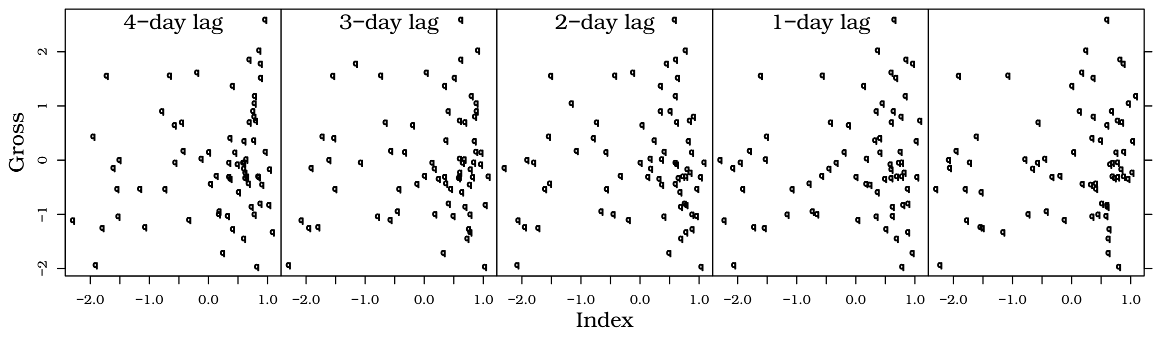


Figure 6.4: Gross revenue versus sentiment index for Super 8

6.5.2 Model Adequacy

Nonlinearity

As was the case for *Captain America*, Teräsvirta's neural network tests for neglected nonlinearity point to a nonlinear relationship between volume and gross revenue. The null hypothesis is strongly rejected as can be seen from the low p-values.

Gross ↓ Volume	Volume ↓ Gross	Gross ↓ Sentiment	Sentiment ↓ Gross
0.0020	**0.0007**	0.7635	0.7439

Causality

Conversely, the causality tests do not suggest any causation for *Super 8*. The test results are reproduced in the table below.

	Gross ↓ Volume	Volume ↓ Gross	Gross ↓ Sentiment	Sentiment ↓ Gross
Lag	Granger Causality			
1	0.2049	0.3507	0.3893	0.4156
2	0.7476	0.3544	0.8033	0.1946
3	0.4209	0.1527	0.4477	0.1427
4	0.2246	0.3253	0.3163	0.3510
Lag	Non-parametric Granger			
1	0.9169	0.13985	0.55772	0.2574
2	0.88533	0.19902	0.74132	0.19526
3	0.91866	0.30412	0.77438	0.2528
4	0.92256	0.33405	0.83877	0.64435

Table 6.6: p-values for the Granger and non-parametric Granger causality tests between tweet volume, sentiment index and daily gross revenue.

6.5.3 Model fitting and Evaluation

Predicting daily grosses with an SVM with a polynomial kernel of degree 3 we get the results shown in the table below. The increase in accuracy coincides with those lags for which the sentiment and the gross have a low p-value.

Lag	Acc. w/o	Acc w/	Kappa w/o	Kappa w/
1	0.6000	0.6571	0.1778	0.2917
2	0.5217	0.5942	0.0189	0.1600
3	0.6865	0.7014	0.3760	0.3886
4	0.7761	0.7313	0.5515	0.4498

Table 6.7: Accuracy and Kappa comparison after predicting the direction of the daily gross by adding sentiment information. SVM with a radial kernel.

The results obtained for this movie are slightly better and more consistent among the different tested models. We believe this is because there where much more instances – between 65 and 70 – for training the classifier than for *Captain America*.

6.6 Summary Tree

In a similar way to what has been done for the stock market application, below is a decision tree summarizing which parameter combinations lead to an improvement in the accuracy when using additional data from Twitter. The addition of Twitter data is considered to improve the results only when the accuracy is higher than 50% and when it increases by at least 5% after adding the Sentiment Index.

For this application we had already limited the number of lags to 4, but as can be seen at the root of the tree, the results suggest that lower lags should be considered. The fact that the second most important parameter corresponds to the target film shows that the success of predictions is closely related on the input data.

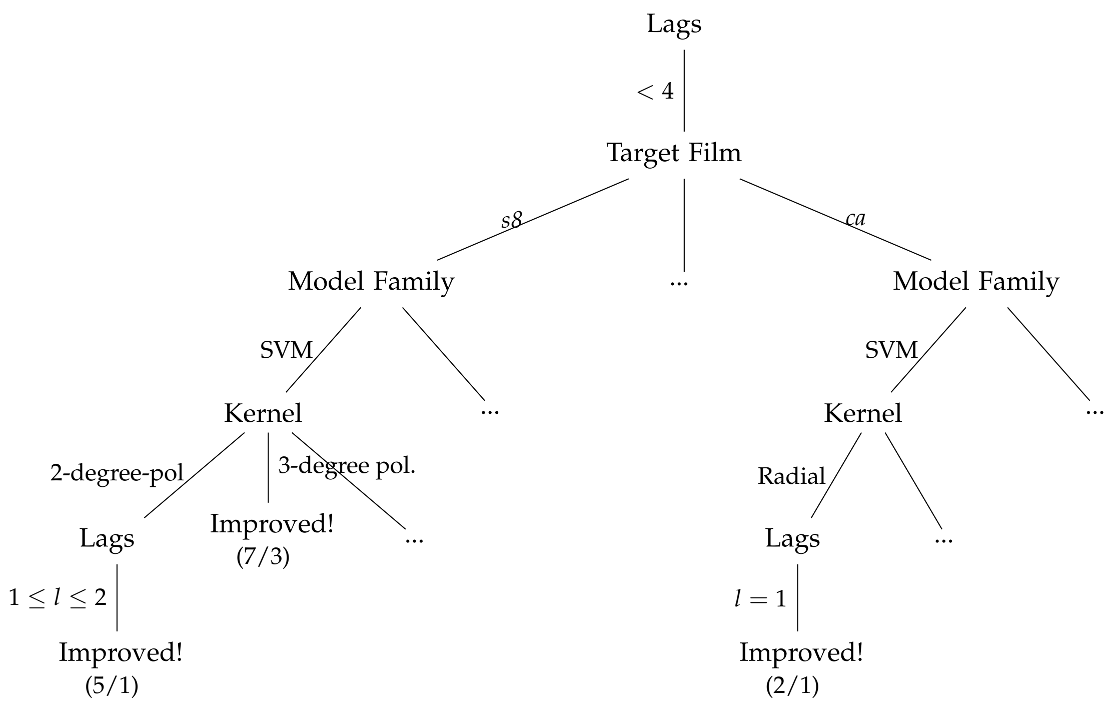


Figure 6.5: Decision tree paths leading to an increase in accuracy when adding sentiment index information to the model. See A.2.5 for the complete tree.

7

Thoughts and Conclusions

This study set out to assess whether the addition of external information extracted from Twitter can be of use for obtaining better time series predictions. A fairly number of studies have been published on the usage of data derived from social networks for improving predictions; hence two of the most popular applications, stock market and movie sales, have been selected to be tested under a unified framework.

Due to the lack of standard datasets for this kind of applications, we have started this work by retrieving our own collection of tweets, a task that has proved much more difficult than it ought to be. In spite of the many limitations imposed by Twitter we were able to collect several millions of messages over the course of eight months.

The next obstacle that we had to overcome is how to conveniently extract useful information from such a large number of messages. For this purpose, we have applied some Natural Language Processing techniques, namely Sentiment Analysis, that allow us to estimate the sentiment polarity –positive or negative– of a given text. Multiple sentiment classifiers have been built for this task, using tweets that contain smileys as training instances. All messages are then fed to these classifiers and, after aggregating the output predictions by date, we end up with a new time series, the Sentiment Index.

The second phase of the project has focused on using this newly generated time series for the forecasting of target time series in the stock market of movie sales domains. Special attention has been put into the study of nonlinearity and causality relationships between the target and the additional time series. These give us information on the adequacy of the predictive models that have been chosen. Experiments have been done with three different predictive models: linear regression, neural networks and support vector machines. One of the main contributions of this work is the definition of a general methodology for the study of time series forecasting by using external information.

After examining the results of our experiments, we cannot confidently determine, in advance, whether the addition of Twitter-derived information will translate into an improvement on the output predictions for a given target time series. In absolute terms, the output of our experiments is largely unsuccessful, not leading to any improvements. However, the number of instances for which the prediction accuracy increases is still high. By the use of a decision tree, we have been able to summarise the results and group the experiments into different sets of parameters that lead to similar results. Consequently, our primary conclusion is that the success of the results is tightly tied to many of the parameters, mainly the input data.

For any given task, it will be convenient to assess the quality of the external data and determine how it relates to the target time series before embarking on the task of adding it to the predictions.

7.1 Future work

Unfortunately, not all of our ideas have made it into the final work, so it would be interesting to continue exploring in the following directions:

- Training a specific sentiment classifier for each of the applications would have surely resulted in better sentiment indices. One problem, though, is the lack of enough labelled data to train these classifiers. As time goes by and more data is collected, further research in these lines might lead to better sentiment indices.
- Happy smileys are much more common than sad ones. Thus, in order to train a sentiment classifier with a balanced dataset, the total number of instances is limited by the negative smileys. Co-training [7] might alleviate the problem caused by the lack of labelled data. It would be interesting to investigate how this technique can be applied for sentiment analysis tasks.
- Currently, the Sentiment Index is computed as a simple percentage of daily messages written on a given topic. We can think of many improvements for calculating the index. For instance, each of the messages could be weighted by its potential audience, e.g. the number of followers of the author. The idea is that users with many followers are bound to be more influential than others with less followers.
- Limiting the predictions to the direction of the time series has allowed us to restrict the scope of our experiments. However, further research should be done using regression techniques for predicting the actual values of the target series.
- Instead of just using one model for predicting the future of the time series, we could use many of them and decide the final prediction by weighted majority vote. Weights would be assigned to each of the experts depending on their accuracy.
- It would be interesting to perform paired difference tests (e.g. paired t-test) in order to measure whether the differences on the accuracies between the models that use the additional series and the ones that do not are statistically significant. This should be done for the different branches of the summary trees presented in Chapters 5 and 6.

References

[1] J.K. Ahkter and Steven Soria. Sentiment Analysis: Facebook Status Messages. *nlp.stanford.edu*, pages 1–19.

[2] Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. *CoRR*, abs/1003.5699, 2010.

[3] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in Twitter streaming data. In *Discovery Sciencefile:///home/rxuriguera/Baixades/readings5.pdf*, pages 1–15. Springer, 2010.

[4] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[5] David M. Blei. Introduction to probabilistic topic models. Available at Blei's webpage `http://www.cs.princeton.edu/~blei/papers/Blei2011.pdf`, 2011.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[7] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. pages 92–100. Morgan Kaufmann Publishers, 1998.

[8] Johan Bollen, Huina Mao, and Xiao-jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, pages 1–8, 2011.

[9] Michael Calore. On Twitter, $ is the new #. `http://www.wired.com/epicenter/2009/02/on-twitter-is-t/`, 2009. [Accessed 02-Jan-2012].

[10] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[11] Sanjiv Das and Mike Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *In Asia Pacific Finance Association Annual Conf. (APFA)*, 2001.

[12] C Diks and V Panchenko. A new statistic and practical guidelines for nonparametric Granger causality testing. *Journal of Economic Dynamics and Control*, 30(9-10):1647–1669, September 2006.

[13] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, page 329, New York, USA, 2009. ACM Press.

[14] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 2009.

[15] Clive W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–38, 1969.

[16] Daniel Gruhl, R. Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. The predictive power of online chatter. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 78–87, New York, NY, USA, 2005. ACM.

[17] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003.

[18] C. Hiemstra and J.D. Jones. Testing for linear and nonlinear granger causality in the stock price-volume relation. *Journal of Finance*, pages 1639–1664, 1994.

[19] Matthew D. Hoffman, David M. Blei, and Francis Bach. Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[20] Tae-Hwy Lee, Halbert White, and Clive W.J. Granger. Testing for neglected nonlinearity in time series models:: A comparison of neural network methods and alternative tests. *Journal of Econometrics*, 56(3):269–290, April 1993.

[21] A. McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[22] Gilad Mishne and Natalie Glance. Predicting Movie Sales from Blogger Sentiment. 2005.

[23] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[24] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: linking text sentiment to public opinion time series. In *Proceeding of the International AAAI Conference on Weblogs and Social Media*, pages 122–129, 2010.

[25] Susan Orlean. Hash. `http://www.newyorker.com/online/blogs/susanorlean/2010/06/hash.html`, 2010. [Accessed 02-Jan-2012].

[26] Bo Pang and Lillian Lee. *Opinion Mining and Sentiment Analysis*. PhD thesis, 2008.

[27] Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, January 2008.

[28] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[29] S. Petrović, Miles Osborne, and Victor Lavrenko. The Edinburgh Twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26. Association for Computational Linguistics, 2010.

[30] Livia Polanyi and Annie Zaenen. Contextual valence shifters. *Computing attitude and affect in text: Theory and applications*, 20:1–10, 2006.

[31] Martin Porter. The english (porter2) stemming algorithm. `http://snowball.tartarus.org/algorithms/english/stemmer.html`, 2002. [Accessed 05-Aug-2011].

[32] M.F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[33] Christopher Potts. On the negativity of negation. In *Proceedings of Semantics and Linguistic Theory*, volume 20, 2010.

[34] Ranks.nl. Stopword lists. `http://www.ranks.nl/resources/stopwords.html`. [Accessed 02-Jan-2012].

[35] Joshua Ritterman, Miles Osborne, and Ewan Klein. Using prediction markets and Twitter to predict a swine flu pandemic. In *Proceedings of the 1st International Workshop on Mining Social Media*, November 2009.

[36] Danny Sullivan. As deal with Twitter expires, Google Realtime Search goes offline. `http://searchengineland.com/as-deal-with-twitter-expires-google-realtime-search-goes-offline-84175`, 2011. [Accessed 02-Jan-2012].

[37] Timo Terasvirta, Chien-Fu Lin, and Clive W. J. Granger. Power of the neural network linearity test. *Journal of Time Series Analysis*, 14:209–220, 1993.

[38] Ruey S. Tsay. *Analysis of Financial Time Series*. John Wiley and Sons, 3rd edition, 2010.

[39] Twitter. Developer rules of the road. `http://webarchive.nationalarchives.gov.uk/20100520095247/dev.twitter.com/pages/api_terms`. [Archived on 20-May-2010].

[40] Twitter. Draft: Twitter rules for API rules. `http://webarchive.nationalarchives.gov.uk/20100409154700/http://twitter.com/apirules`. [Archived on 09-Apr-2010].

[41] Twitter. Streaming API concepts. `https://dev.twitter.com/docs/streaming-api/concepts#filter-limiting`. [Accessed 02-Jan-2012].

[42] Twitter. Using the Twitter Search API. `https://dev.twitter.com/docs/using-search`. [Accessed 02-Jan-2012].

[43] Twitter. 200 million tweets per day. `http://blog.twitter.com/2011/06/200-million-tweets-per-day.html`, 2011. [Accessed 02-Jan-2012].

[44] Twitter. It takes a community to translate Twitter. `http://blog.twitter.com/2011/08/it-takes-community-to-translate-twitter.html`, 2011. [Accessed 05-Aug-20101].

[45] H. White. An additional hidden unit test for neglected nonlinearity in multilayer feed-forward networks. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 451–455. IEEE, 1989.

[46] M. Sebastian. A Wolfram. Modelling the stock market using Twitter. Master’s thesis, School of Informatics, University of Edinburgh, 2010.

[47] X. Zhang, H. Fuehres, and P. A. Gloor. Predicting stock market indicators through Twitter "I hope it is not as bad as I fear". In *Collaborative Innovation Networks (COINs)*, 2010.

A

Additional Results

A.1 Complete Tweet

Below is a tweet formatted in JSON:

```
{
    "favorited": false,
    "in_reply_to_status_id_str": null,
    "user": {
        "friends_count": 6774,
        "profile_sidebar_fill_color": "DDEEF6",
        "protected": false,
        "default_profile_image": false,
        "lang": "en",
        "statuses_count": 5231,
        "profile_background_tile": false,
        "profile_image_url": "http://a3.twimg...",
        "name": "TED News",
        "verified": true,
        "is_translator": false,
        "utc_offset": −18000,
        "profile_link_color": "0084B4",
        "followers_count": 82751,
        "screen_name": "TEDNews",
        "follow_request_sent": false,
        "profile_sidebar_border_color": "C0DEED",
        "location": "New York, NY",
        "time_zone": "Eastern Time (US & Canada)",
        "default_profile": true,
        "following": true,
        "profile_use_background_image": true,
        "url": "http://www.ted.com",
        "description": "ALL the news from #TED: TEDTalks, TED
                        Conferences, the TED Prize and more.",
        "contributors_enabled": true,
        "profile_background_color": "C0DEED",
        "id_str": "36843988",
        "listed_count": 5965,
        "profile_background_image_url_https": "https://si0...",
        "profile_background_image_url": "http://a0.twimg...",
        "created_at": "Fri May 01 01:32:57 +0000 2009",
        "show_all_inline_media": false,
        "geo_enabled": false,
            "notifications": false,
            "profile_image_url_https": "https://si0.twimg...",
            "id": 36843988,
            "favourites_count": 49,
            "profile_text_color": "333333"
        },
        "place": null,
        "in_reply_to_screen_name": null,
        "in_reply_to_user_id_str": null,
        "contributors": null,
        "geo": null,
        "retweeted": false,
        "truncated": false,
        "coordinates": null,
        "text": "RT @TEDchris: Mind-shifting #TED talk on the evolution
                of language from Mark Pagel http://on.ted.com/Pagel",
        "in_reply_to_user_id": null,
        "retweet_count": 7,
        "in_reply_to_status_id": null,
        "id_str": "98776348977414144",
        "source": "TweetDeck",
        "created_at": "Wed Aug 03 15:24:51 +0000 2011",
        "possibly_sensitive": false,
        "id": 98776348977414140
    }
```

A.2 Sentiment Classifier

A.2.1 Control set

An extremely small control set has been manually tagged and tested against the classifiers to ensure that they can indeed predict the polarity of obvious sentences. The table below shows the 20 tweets conforming the set. Punctuation and grammar has been left as in the originals.

Positive	I love watching teen mom
	lets talk about the good times til it takes us back
	@TeamLaylaa yeah that'd be nice babe!
	Quorra in Tron Legacy is really cute.
	Oh this feels so good.. My bed
	Hey guess what? I'm feeling better now!
	That was a great movie
	@TatumBateman That's crazy. Im happy you're back on twitter
	@bingethinking they are truly stunning, as are most of the ladies
	@janiedean Have lots and lots of fun. He puts on a good show.
Negative	I hate sleeping alone
	my heart just skipped a beat. can't believe it. RIP Dunn.
	Really missin Trey today
	Going home to study for some math
	don't like saying bye to my gramps

feeling guilty with her, perhaps that she'll fine tomorrow
i have anger issues
I can't fucking live with 124 songs
Ughh i hate itt when kayla and dom gangs up on meeee
@tuxeltje He fucking tortured me!

The tweet in italics is the only sentence that is misclassified by the *C-En* classifier. We believe that the *bed* is the word causing this misclassification because of the many negative messages coming from people who have problems trying to sleep or that do not want to wake up in the morning.

A.2.2 English tweet datasets

Smiley Loc.	Hashtag	Stemming	Negation	Vect.Values	Accuracy ▼
End	Replace	False	False	Frequency	**76.4909%**
End	Keep	False	False	Frequency	76.4759%
End	Keep	True	False	Frequency	76.2107%
End	Keep	False	False	Count	76.1707%
End	Replace	True	False	Frequency	76.1707%
End	Replace	False	False	Count	76.1207%
End	Keep	False	True	Frequency	76.0656%
End	Replace	False	True	Frequency	76.0156%
End	Replace	False	False	Presence	76.0106%
End	Keep	False	False	Presence	76.0006%
End	Keep	False	True	Presence	75.9406%
End	Keep	True	True	Frequency	75.9406%
End	Keep	False	True	Count	75.8955%
End	Replace	True	True	Frequency	75.8755%
End	Replace	False	True	Presence	75.8355%
End	Replace	False	True	Count	75.7855%
End	Keep	True	False	Presence	75.7404%
End	Keep	True	False	Count	75.6754%
Any	Keep	False	False	Frequency	75.6604%
End	Replace	True	False	Count	75.6254%
End	Replace	True	False	Presence	75.5753%
Any	Replace	False	False	Frequency	75.5603%
Any	Keep	False	True	Frequency	75.4203%
End	Keep	True	True	Count	75.4152%
Any	Replace	False	True	Frequency	75.3752%
Any	Keep	True	False	Frequency	75.3302%
End	Keep	True	True	Presence	75.3252%
End	Replace	True	True	Count	75.3152%
End	Replace	True	True	Presence	75.2652%
Any	Replace	True	False	Frequency	75.1851%
Any	Keep	False	True	Presence	75.1151%
Any	Keep	False	False	Count	75.1051%
Any	Keep	True	True	Frequency	75.1051%
Any	Replace	True	True	Frequency	75.0901%

Continued on next page

Smiley Loc.	Hashtag	Stemming	Negation	Vect.Values	Accuracy ▼
Any	Replace	False	False	Count	75.0600%
Any	Keep	False	True	Count	75.0250%
Any	Keep	False	False	Presence	75.0200%
Any	Replace	False	True	Presence	75.0050%
Any	Replace	False	False	Presence	74.9650%
Any	Replace	False	True	Count	74.9500%
Any	Keep	True	False	Presence	74.8349%
Any	Keep	True	False	Count	74.7098%
Any	Replace	True	False	Presence	74.6898%
Any	Keep	True	True	Count	74.6148%
Any	Keep	True	True	Presence	74.6148%
Any	Replace	True	False	Count	74.5848%
Any	Replace	True	True	Presence	74.5697%
Any	Replace	True	True	Count	74.4647%

A.2.3 Multilanguage tweet datasets

Smiley Location	Keep Hashtags	Feat.List	Vect.Values	Accuracy ▼
Any	Keep	Sentiment	Frequency	**79.5155**%
Any	Replace	Sentiment	Frequency	79.2775%
Any	Keep	Sentiment	Presence	78.7368%
Any	Keep	Sentiment	Count	78.6719%
Any	Replace	Sentiment	Presence	78.5853%
Any	Replace	Sentiment	Count	78.4555%
End	Keep	Sentiment	Frequency	77.2516%
End	Replace	Sentiment	Frequency	76.9418%
End	Keep	Sentiment	Presence	76.8533%
End	Replace	Sentiment	Presence	76.6984%
End	Keep	Sentiment	Count	76.6099%
End	Replace	Sentiment	Count	76.4992%
Any	Keep	Stock	Frequency	76.0978%
Any	Keep	Stock	Presence	76.0978%
Any	Keep	Stock	Count	76.0112%
Any	Replace	Films	Presence	76.0112%
Any	Replace	Films	Frequency	75.8815%
Any	Replace	Films	Count	75.8166%
Any	Keep	Films	Frequency	73.8914%
Any	Keep	Films	Count	73.7833%
Any	Keep	Films	Presence	73.7400%
Any	Replace	Stock	Presence	73.6535%
Any	Replace	Stock	Count	73.6102%
End	Keep	Stock	Frequency	73.6004%
End	Keep	Stock	Presence	73.6004%
Any	Replace	Stock	Frequency	73.5886%

Continued on next page

Smiley Location	Keep Hashtags	Feat.List	Vect.Values	Accuracy ▼
End	Keep	Stock	Count	73.5340%
End	Replace	Films	Frequency	73.4012%
End	Replace	Films	Presence	73.4012%
End	Replace	Films	Count	73.2906%
End	Keep	Films	Presence	71.0998%
End	Keep	Films	Frequency	71.0113%
End	Keep	Films	Count	70.9892%
End	Replace	Stock	Presence	70.7015%
End	Replace	Stock	Count	70.5245%
End	Replace	Stock	Frequency	70.5245%

A.2.4 Stock Market

Summary Decision Tree (Accuracy)

5-level decision tree showcasing the parameter settings that allow for an accuracy improvement of at least 5% when adding Twitter information to the model. Accuracy must also surpass 50%. Paths that lead to a `TRUE`-tagged leaf describe models that improve with Twitter data.

```
model = glm : FALSE (1973/27) [1957/39]
model = nnet−2
|   lags < 3.5 : FALSE (1191/45) [1167/55]
|   lags ≥ 3.5
|   |   target_symbol = goog : FALSE (42/4) [38/4]
|   |   target_symbol = gspc : FALSE (171/32) [149/19]
|   |   target_symbol = msft : FALSE (46/6) [34/4]
|   |   target_symbol = oex : FALSE (147/22) [161/25]
|   |   target_symbol = vix : FALSE (147/8) [157/9]
|   |   target_symbol = vxo : FALSE (145/18) [175/13]
|   |   target_symbol = yhoo : FALSE (36/1) [44/2]
|   |   target_symbol = aapl
|   |   |   classifier = Cl−En : FALSE (10/0) [14/1]
|   |   |   classifier = Cl−Ml : FALSE (12/0) [12/0]
|   |   |   classifier = Cl−Stk : FALSE (12/0) [12/2]
|   |   |   classifier = opinion_finder
|   |   |   |   lags < 4.5 : TRUE (3/0) [1/0]
|   |   |   |   lags ≥ 4.5 : FALSE (4/0) [0/0]
model = nnet−3
|   lags < 4.5 : FALSE (1568/107) [1576/100]
|   lags ≥ 4.5
|   |   target_symbol = goog : FALSE (22/8) [18/4]
|   |   target_symbol = gspc : FALSE (91/20) [69/14]
|   |   target_symbol = msft : FALSE (18/1) [22/1]
|   |   target_symbol = oex : FALSE (79/11) [75/7]
|   |   target_symbol = vix : FALSE (73/7) [79/16]
|   |   target_symbol = vxo
|   |   |   additional_series  = both :  FALSE (22/2) [26/7]
|   |   |   additional_series  = sentiment :  FALSE (29/5) [35/7]
|   |   |   additional_series  = volume
|   |   |   |   target_series  = close  :  FALSE (16/1) [8/0]
|   |   |   |   target_series  =  volatility   :  TRUE (14/2) [10/0]
|   |   target_symbol = yhoo : FALSE (19/0) [21/5]
|   |   target_symbol = aapl :  FALSE (17/1) [23/1]
model = nnet−4 : FALSE (1998/172) [1932/183]
model = nnet−5
|   target_symbol = goog : FALSE (103/3) [97/7]
|   target_symbol = gspc
|   |   target_series  = close  :  FALSE (192/9) [208/15]
|   |   target_series  =  volatility
|   |   |   dataset  = goog :  FALSE (19/3) [31/3]
|   |   |   dataset  = googlda :  FALSE (24/3) [26/3]
|   |   |   dataset  = msft :  FALSE (18/1) [32/3]
|   |   |   dataset  = msftlda :  FALSE (25/0) [25/0]
|   |   |   dataset  = yhoo
|   |   |   |   classifier   = Cl−En : FALSE (8/0) [7/0]
|   |   |   |   classifier   = Cl−Ml : TRUE (3/2) [12/7]
|   |   |   |   classifier   = Cl−Stk : FALSE (7/0) [8/2]
|   |   |   |   classifier   = opinion_finder :  TRUE (4/1) [1/0]
|   |   |   dataset  = yhoolda : FALSE (27/6) [23/7]
|   |   |   dataset  = aapl  :  FALSE (25/4) [25/5]
|   |   |   dataset  = aapllda  :  FALSE (30/4) [20/2]
|   target_symbol = msft :  FALSE (98/16) [102/19]
|   target_symbol = oex :  FALSE (396/49) [374/55]
|   target_symbol = vix :  FALSE (382/33) [378/17]
|   target_symbol = vxo : FALSE (403/48) [397/42]
|   target_symbol = yhoo : FALSE (106/3) [94/5]
|   target_symbol = aapl :  FALSE (102/3) [98/6]
model = svm−poly−2 : FALSE (1909/41) [2016/29]
model = svm−poly−3 : FALSE (1927/36) [1998/34]
model = svm−poly−4
|   dataset  = goog :  FALSE (233/2) [227/0]
|   dataset  = googlda :  FALSE (252/0) [248/0]
|   dataset  = msft :  FALSE (235/0) [265/0]
|   dataset  = msftlda :  FALSE (251/0) [249/0]
|   dataset  = yhoo : FALSE (262/17) [238/10]
|   dataset  = yhoolda
|   |   classifier   = Cl−En : FALSE (78/1) [72/0]
|   |   classifier   = Cl−Ml : FALSE (68/0) [82/0]
|   |   classifier   = Cl−Stk
|   |   |   additional_series  = both
|   |   |   |   target_symbol = goog : FALSE (0/0) [0/0]
|   |   |   |   target_symbol = gspc :  FALSE (6/1) [4/1]
|   |   |   |   target_symbol = msft :  FALSE (0/0) [0/0]
|   |   |   |   target_symbol = oex :  FALSE (3/0) [7/1]
|   |   |   |   target_symbol = vix :  FALSE (8/1) [2/0]
|   |   |   |   target_symbol = vxo : TRUE (6/3) [4/1]
|   |   |   |   target_symbol = yhoo : FALSE (6/0) [4/2]
|   |   |   |   target_symbol = aapl :  FALSE (0/0) [0/0]
|   |   |   additional_series  = sentiment
|   |   |   |   target_symbol = goog : FALSE (0/0) [0/0]
|   |   |   |   target_symbol = gspc :  FALSE (5/0) [5/2]
|   |   |   |   target_symbol = msft :  FALSE (0/0) [0/0]
|   |   |   |   target_symbol = oex :  FALSE (6/1) [4/0]
|   |   |   |   target_symbol = vix :  FALSE (4/1) [6/0]
|   |   |   |   target_symbol = vxo : TRUE (5/2) [5/2]
|   |   |   |   target_symbol = yhoo : FALSE (6/2) [4/0]
|   |   |   |   target_symbol = aapl :  FALSE (0/0) [0/0]
|   |   |   additional_series  = volume : FALSE (21/0) [29/1]
|   |   classifier   = opinion_finder :  FALSE (23/0) [27/0]
|   dataset  = aapl  :  FALSE (233/10) [232/3]
|   dataset  = aaplIda  :  FALSE (259/1) [241/1]
model = svm−poly−5 : FALSE (1994/36) [1931/34]
model = svmradial : FALSE (1980/28) [1945/23]
model = svmsigmoid
|   target_series  = close
|   |   additional_series  = both
|   |   |   target_symbol = goog : FALSE (11/0) [19/0]
|   |   |   target_symbol = gspc :  FALSE (57/9) [63/14]
|   |   |   target_symbol = msft :  FALSE (16/0) [14/1]
|   |   |   target_symbol = oex :  FALSE (55/10) [60/19]
|   |   |   target_symbol = vix :  FALSE (60/13) [55/4]
|   |   |   target_symbol = vxo :  FALSE (52/5) [68/14]
|   |   |   target_symbol = yhoo : FALSE (19/1) [11/2]
|   |   |   target_symbol = aapl
|   |   |   |   lags  < 2.5  :  FALSE (5/0) [7/1]
|   |   |   |   lags  ≥ 2.5  :  TRUE (10/5) [8/3]
|   |   additional_series  = sentiment
|   |   |   target_symbol = goog : FALSE (23/1) [17/0]
|   |   |   target_symbol = gspc :  FALSE (69/10) [91/20]
|   |   |   target_symbol = msft :  FALSE (22/1) [18/2]
|   |   |   target_symbol = oex :  FALSE (81/19) [74/15]
|   |   |   target_symbol = vix :  FALSE (76/12) [74/11]
|   |   |   target_symbol = vxo :  FALSE (81/15) [79/12]
|   |   |   target_symbol = yhoo : FALSE (19/3) [21/1]
|   |   |   target_symbol = aapl
|   |   |   |   lags  < 2.5  :  FALSE (5/0) [11/1]
|   |   |   |   lags  ≥ 2.5  :  TRUE (10/5) [14/6]
|   |   additional_series  = volume : FALSE (291/10) [299/13]
|   target_series  =  volatility   :  FALSE (981/53) [979/42]
```

A.2.5 Box Office

Summary Decision Tree

As with the stock market case, below is the decision tree showcasing the parameter settings that allow for an accuracy improvement of at least 5% when adding Twitter information to the model. Accuracy must also surpass 50%.

```
lags < 3.5
|   target_film = gl : FALSE (64/3)
|   target_film = s8
|   |   model = glm : FALSE (7/0)
|   |   model = nnet−2 : FALSE (6/0)
|   |   model = nnet−3 : FALSE (3/0)
|   |   model = nnet−4 : FALSE (8/0)
|   |   model = nnet−5 : FALSE (7/0)
|   |   model = svm−polynomial−2
|   |   |   lags < 2.5 : TRUE (5/1)
|   |   |   lags ≥ 2.5 : FALSE (2/0)
|   |   model = svm−polynomial−3 : TRUE (8/4)
|   |   model = svm−polynomial−4 : TRUE (7/3)
|   |   model = svm−polynomial−5 : TRUE (5/2)
|   |   model = svm−radial : FALSE (6/0)
|   |   model = svm−sigmoid : FALSE (3/0)
|   target_film = c2 : FALSE (67/8)
|   target_film = sf : FALSE (67/4)
|   target_film = hp : FALSE (64/3)
|   target_film = cw : FALSE (68/8)
|   target_film = ra
|   |   classifier = C−En
|   |   |   model = glm : FALSE (3/1)
|   |   |   model = nnet−2 : FALSE (3/1)
|   |   |   model = nnet−3 : FALSE (3/1)
|   |   |   model = nnet−4 : TRUE (2/1)
|   |   |   model = nnet−5 : FALSE (0/0)
|   |   |   model = svm−polynomial−2 : TRUE (3/1)
|   |   |   model = svm−polynomial−3 : TRUE (2/1)
|   |   |   model = svm−polynomial−4 : TRUE (0/0)
|   |   |   model = svm−polynomial−5 : TRUE (3/1)
|   |   |   model = svm−radial : FALSE (1/0)
|   |   |   model = svm−sigmoid : FALSE (1/0)
|   |   classifier = C−Ml : FALSE (24/0)
|   |   classifier = C−Flm : FALSE (24/1)
|   target_film = ca
|   |   model = glm : FALSE (5/0)
|   |   model = nnet−2 : FALSE (6/0)
|   |   model = nnet−3 : FALSE (5/0)
|   |   model = nnet−4 : FALSE (6/0)
|   |   model = nnet−5 : FALSE (6/0)
|   |   model = svm−polynomial−2 : FALSE (5/0)
|   |   model = svm−polynomial−3 : FALSE (6/0)
|   |   model = svm−polynomial−4 : FALSE (8/0)
|   |   model = svm−polynomial−5 : FALSE (5/0)
|   |   model = svm−radial
|   |   |   lags < 1.5 : TRUE (2/1)
|   |   |   lags ≥ 1.5 : FALSE (6/0)
|   |   model = svm−sigmoid : FALSE (7/0)
```

B
Software

This appendix is dedicated to a brief description of the sentiment analysis software that has been used.

B.1 Existing Software

There are currently many open source alternatives that can be used for text mining. Among them we find Knime[1], OpinionMiner[2] or Kepler[3] with which one can build workflows to process information. A workflow can be defined as a graph of tasks that share their results with the rest of tasks connected to their output.

We really like the workflow idea and, while Knime looks very powerful and it is very easy to use, it has some drawbacks that prevented us from using it. Mainly, it has a poor Weka integration, trying to convert the whole output of a node into an ARFF file before using it. This should not be needed for online classifiers and it is a problem when working with large datasets. Next, remote workflows are also not supported in the free version of the software. Finally, stream processing cannot be easily implemented.

NLTK[4] is used for some of the typical text processing tasks. As for the implementations of machine learning models, we use Weka[5] as well as some R[6] packages. Both of these software solutions offer a wide range of classification and clustering algorithms. Moreover, many statistical tests are also implemented in R.

B.2 Our system

We have developed a very simple framework in Python by following the idea of workflows. This allows us to easily test different combinations of the text processing tasks, making it very easy to alter the flux of the information. Creating new workflows is straightforward and has a minimum cost in terms of the amount of code. Finally, it works by processing mini-batches of data, so it can be used for large datasets or online algorithms.

A handful of R scripts have been written in order to run the statistical tests presented in this work as well as for the prediction of time series. Finally, the last part of our software consists of a set of shell scripts that enable us to run our experiments on a cluster.

[1]`http://www.knime.org/`
[2]`http://opendover.nl/opinion-miner-0`
[3]`https://kepler-project.org/`
[4]`http://www.nltk.org/`
[5]`http://www.cs.waikato.ac.nz/ml/weka/`
[6]`http://cran.r-project.org/`