4_Twitter_Mining

Nicholas Mitchell

March 13, 2016

Contents

NOTE:
Any time you see a † symbol, it means there is a link attached that you can click on to go to a website. This
is to help people reading a printed version later on, as the links will all be in an appendix.

1 Social media data

There are many possible sources of social media data that could be incorporated into a statistical model,
and naturally it is the Big Three: Google, Facebook and Twitter, who spring to mind. While the means exist to obtain data from all three, there are also limitations that apply to each.
1

1.1 Google Trends

Google makes a lot of data freely available, for example the number of times a given word or phrase was
'Googled'. The search engine does, however, apply certain filter and pre-processing steps to the data before
making it available. What remains is a great tool for making comparisons between terms, plotting their relative popularity against one another over a long time period, an example of which is shown in FigX.

Figure 1: A placeholder for the real image

There are two issues that make Google Trends data difficult (but not impossible) to use in the context of this study. The first issue is that the frequency of the data is (at the time of writing) limited to weekly aggregates for timelines longer than three months. This means a method of interpolation would have to be
implemented before the data could be combined with daily financial market data over this study's desired
time-line of two or more years. Daily data is available for time-lines shorter that three months, which leads
nicely on to the second issue. The pre-processing of the data is not transparent; the exact methods used
are not published and so any interpretation could perhaps be fallacious. The data is clearly normalised, the

Comment [MS1]: I guess the word is fine, but maybe 'misleading' would be more straightforward for readers (especially those with English as a second language) to understand

maximum 'popularity' in each extracted data set being 100 and so naïve attempts to stitch many threemonth
data sets together - such as linear combinations - would be in vain, as the final time-line could not be considered homogeneous in scale. With a different objective in mind, the Google search data does present
an interesting case. Hamid and Heiden (2014) were able to show how Google search volumes could be used
to increase forecasting accuracy for market phases of relatively high volatility[1]. In the context of this study,
however, the task must be deferred as discussed in Future extensions.

1.2 Facebook

To use Facebook as a source of data, it is necessary to create a special account for software developers (which
is free). The downside, however, is that only the publicly available information of your own friends who also
have a developer account may freely be obtained. This is a large limitation, as it would significantly reduce
the amount of data available and narrows down the pool of social media data specifically to a biased subset of
users, i.e. data for people who are involved in software development and data mining. This is unfortunately
not the target group of this study and so rules out the use of data Facebook.

1.3 Twitter

The third option is Twitter, which has been extensively mined for its large flow of information [References
to several examples]. The following section explains why Twitter is such a popular choice as a source of
social media data, justifying its selection for this study. The current best practices of extracting data are
then summarised, along with a brief explanation of the procedure defined by this study.

[1]This is an interesting direction that could potentially be built upon with the Twitter data accumulated for this study

2 Twitter Mining

2.1 Why use Twitter?

The social media data used for sentiment analysis (see ChapterX) was sourced exclusively from the online
social media platform Twitter[y]. The first post (in Twitter terminology, a tweet) was made in March 2006 via short message service (SMS), the entire service running off of a single laptop. In the ensuing months the
platform began its ascent to popularity, steadily expanding its user-base after its official launch in summer
2006. Not only individuals, but everyone from news companies and sports teams to artists and presidents
use Twitter to update their follows, with the potential to reach anybody with an internet connection. A study into the monthly count of unique visitors that top digital media properties receive showed that Twitter
was ranked 14th with 118,000 unique visitors [2] in January 2016 alone. Perhaps somewhat unsurprisingly,
the top two spots for the same time period were occupied by Google and Facebook, each with ca. 245,000
and ca. 207,000 unique visitors, respectively.

Comment [MS2]: How are these numbers so low? I thought these would have millions of unique visitors??

There are several reasons why Twitter data is an attractive candidate as an explanatory variable in a study such as this one. First and foremost, it is content that is created on a continuous basis with almost
no filter [3]. In short, users may post their thoughts regarding any topic, at any time, for anyone to read. This makes the data an excellent tool for capturing the sentiments and emotions across extremely large
demographics of users, in real time.

2.2 Requirements on data

At the time of writing there are several clear ways in which it is possible to obtain Twitter data, each outlined in the section Sources of Twitter data[y] along with their corresponding benefits and drawbacks. When considering which route to take, it should be kept in mind exactly what kind of analysis will ultimately be performed on the data. The context of this study necessitates that the information obtained fulfils certain criteria. Regarding the data of each individual tweet, there are two criteria:

Criterion 1: each tweet must contain the tweet text

Criterion 2: each tweet must contain a timestamp

The last requirements concerns the population of tweets obtained:

Criterion 3: the collective corpus of tweets must span a timeline of at least two years

In order to perform sentiment analysis on the Twitter data, it is imperative that the text string is obtained, fulfilling Criterion 1. If only meta-data were to be received, e.g. the creation time and point of origin of a tweet, sentiment analysis would be impossible. Criterion 2 ensures that the Twitter data (and therefore the results of the sentiment analysis) can be reliably aggregated into daily data. This allows for coherent usage with daily financial market data. Although the timeline specified by Criterion 3 may appear somewhat arbitrary (and it is!), a minimum timeline of several years is commonly desired for time-series analysis of financial markets. For discussion as to why this is the case, please refer to sectionX ('Time series analysis - N versus P').

2.3 Sources of Twitter data

2.3.1 Twitter API

Twitter offers an application programme interface (API) to allow programmatic connections to its databases. This is commonly achieved using languages such as Python, JavaScript and R, but can be implemented using any language capable of establishing an API connection. The service is free, requiring only that users create

[2]Statistics compiled by comScore[y]. Unique signifies a user with identifiable IP address. Total numbers of hits, i.e. simple page visits, including those from machines and referals over other sites reach around 3 billion per month according to SimilarWeb[y].

[3]The only limit imposed on users is the 140-character limit placed on each tweet. Twitter's actual definition is slightly more detailed[y].

a developer account, obtaining secure identification methods using a token system. Furthermore, the tweet data is very clean and there are many tools[4] already available that parse and display that data.

There are two restrictions placed on the API. The first is to safeguard the Twitter servers from being overrun, namely that each user may make only a certain number of requests[y] in a given time-frame, which translates into a limit of approximately 10,000 tweets in a fifteen-minute timeframe. The second restriction limits the API's reach into the past to approximately seven days. This means that it is impossible to collect and create a time-series of the required length for this study. While it is possible to implement and automate a script to collect tweets at a given frequency[5], one would have to still wait e.g. two years minus seven days to obtain a time-series that is two years in length. For this reason, the Twitter API methodology was not a feasible option for this study.

2.3.2 Third Party Providers

It is possible to gain access to the complete Twitter archives, spanning back to Twitter's inception. This is facilitated by a third party company called Union Metrics via their Echo product line[y]. There are interactive

analytics tools built in to the console, which allow the slicing and drilling of the entire database with visual representations. This is aimed at commercial users needing to make strategic marketing decisions, rather than perform statistical analysis or make quantitative forecasts.

Although the product is extensive and offers many features, it has two drawbacks. Firstly it is not a free service; requiring a corporate level monthly subscription. Secondly, the offering is not optimised for independent data analysis, as restrictions on exporting the data would impede full usage of the data within alternative software packages. Both constraints rule out this is a valid otion for this study, with the second constraint being particularly large for any parties interested in quantitative forecasting.

Comment [MS3]: Does it also introduce a greater risk of the data being incomplete/inaccurate because you are relying on a third party handling the data before you get it?

2.3.3 Twitter Advanced Search

The Twitter Advanced Search[5] (TAS) web interface allows any user to search for tweets in any time period, displaying tweets that match a given search term. The tweets are displayed in reverse chronological order (the most recent tweet is at the top of the webpage) and each tweet is displayed with its key information.

The HTML code being rendered, however, holds additional information, matching all that is available via the API and third party options. There isn't only the tweet text, username and timestamp, but rather a whole host of meta data including e.g. the number of times the tweet has since been retweeted (re-posted or shared by another user) or favourited (marked as a favourite by another user) and even the longitude-latitude coordinates of the user at the time of posting[6]. SectionX goes into more detail about how this data may be located and extracted from the HTML code.

The web interface is free to use, contains the entire Twitter archive and also, being Twitter's advanced search, allows for filtering of tweets beyond a date range. For example, the natural language of the tweets (English, Portuguese, etc.) can be used as a filter as well as a longitude-latitude coordinates from which a tweet was posted. Tweets for individual users or containing specific hashtags[7] can also be selected. This study uses solely the common search function, returning all tweets that contain the user-specified word(s).

The single disadvantage of this approach is that it involves using an interactive interface, i.e. it is not designed to be utilised programmatically. This created significant challenges within the scope of this study, including the development of a customised web-scraper, as shall be explained in the following section.

3 Constructing a web-scraper

3.1 What is a web-scraper?

To explain this, a good analogy between the internet and an encyclopaedia can be used. Imagine we would like to find all the pages in the encyclopaedia that contain information regarding a topic of interested, for example "chocolate". We would look in the index for our search term and find all topics involving chocolate to be listed with their page and section numbers. The term given to such a mechanism is web-crawler[y]

[4] The most useful implementation in R is currently the twitteR package, which is a one-stop-shop for cleanly extracting tweets, ready for analysis with common R functions.
[5] The author has already implemented such as system, available on request.
[6] The coordinates accuracy is approximately a 1.5 km radius, which should guarantee some level of privacy.
[7] Hashtags provide and unmoderated way to help to link tweets from different users and locales by theme.

and is (simplistically speaking) approximately what search engines such as Google, Yahoo and Bing carry
out each time somebody uses their search functions. They look all the pages in their encyclopaedia and the
returned search results are those (web-)pages containing the word "chocolate"[8]. The data that this study is
interested in, however, is not the page number i.e. the internet address of certain information, but rather
the contents held at those addresses.
Assuming the ~~the~~ information provided by a web-crawler is already known (in our case the internet address of TAS, using our analogy, we visit the specified page and make a copy of all the information that
is stored there. Just as one could write out a copy of any information visible in an encyclopaedia, it is possible to make a copy of all visible information (plus additional background meta data) presented on a
website. This is because, in order for the website to be displayed in a browser, all the required information
must first be transferred (downloaded) to the local device and stored in the form of HTML code, which the
browser then interprets and renders. It is then this HTML code that is copied, or scraped, leading to the
term web-scraper [9].
In order to obtain all required information from TAS, the first major objective of this study was to create
a web-scraper that was able to visit the TAS interface, manipulate the webpage and make a copy of the
underlying information i.e. the HTML code.

Comment [MS4]: Is TAS defined anywhere?

3.1.1 Types of web-scraping

Web-scraping can be performed in two ways: with a visible browser interface (e.g. what a user sees when
using Microsoft Internet Explorer or Google Chrome), or via a headless browser. The latter refers to a method whereby a computer connects to a web-address and collects the information held there (the HTML
code), but does not render that code in a browser, meaning the user does not see any actual webpages[10].
This method is preferable over the former as it does not require as much computational power and does
not consume much working memory on the local device, meaning it can be executed relatively quickly and
for a large number of websites. In such a framework it is the connection speed between local device and
target that is the limitation. Headless browsers are however (at the time of writing[11]) limited to static web-addresses, meaning that the information is held at an address and does not change. However as was
previously mentioned, TAS has a dynamically loading interface and so requires the former approach, which
is described in the following section.

3.2 How does our web-scraper work?

To provide the functionality required to manipulate a browser via its graphical user interface (GUI) - as the
case is using TAS - a software development tool called Selenium WebDriver was used[12]. This facilitated the
automation of web-page manipulation. To name several examples, Selenium WebDriver is able to perform
actions such as clicking, scrolling and entering text into text-fields - all specified programmatically.
As inputs, TAS takes a search term (plus any filters that a user adds) along with a date range. As output,
the youngest 20 tweets in the date range are return, all of which contain that search term. Once the user

has scrolled to the bottom of the page, the next 10 tweets are loaded. This process continues until the end
of the date range is reached, i.e. once the oldest tweet within the date range has been loaded and displayed.
At this point any attempt by the user to keep scrolling will have no effect - no more tweets will be loaded.
A date range and the search term, as well as a filter to receive only tweets written in English[13] are all able to be specified simply through their inclusion within the target URL[14]. Selenium then enters this URL

[8]Web-crawling also includes how the search engines obtain their information (i.e. the encyclopaedia) to begin with. An explanation of this does not lie within the scope of this study. Heydon and Najork (1999)y provide a good starting point.
[9]Also referred to as web harvesting and web data extraction.
[10]Headless browsing is a technique often used for debugging purposes, as errors can be detected without visualisation i.e. without rendering the underlying information. This accelerates the process of web-development.
[11]Progress is being madey in the development of headless browsers for tasks such as scrolling dynamically loaded webpages
[12]A detailed technical explanation of this step shall not be provided here.
[13]Although Twitter includes this as an option within TAS, it is not guaranteed to classify the language with 100% accuracy.
[14]A Uniform Resource Locator (URL) can contain several elements, but usually essential are a protocol (http:) and a hostname (www.twitter.com). More specific locations are then appended as necessary, commonly separated by a forward slash.

into the browser's address field and visits the page.
Once the browser has reached the URL and the first 20 tweets have been loaded, a basic process is followed and can be summarised by the following steps:
1. scroll to the bottom of the page
2. wait long enough for the next 10 tweets to be loaded
3. scroll to the bottom of the page again
4. Repeat steps 2. and 3. until no more tweets load
A programme to automate this process was written in Python, importing the Selenium WebDriver package.
A full description of the automation process is described by Algorithms 1 and 2 .

3.3 Stability considerations

As previously mentioned there are computational constraints to consider when working with a browser. In
the case of this task it was the working memory[15] that posed this problem. Because the web browser receives,
stores and renders the information for all tweets, the amount of memory required climbs very quickly. Certain
steps can however be taken to reduce this burden, and can be divided into two branches: programmatic and
organisational.
In terms of organisation, it was necessary to create batch-processes to perform scrolling sessions, which
provided control and stability when scrolling downwards over the extremely long dynamically loaded webpage,
TAS. Due to the fact that the number of tweets posted that contain a given search term - over any given
time-span - cannot be known in advance, the size of the batches had to be determined heuristically. The size
determined by the user as a variable defined as the scroll.limit, which tells tells Selenium how many times
to scroll down - pausing for a given time between each scroll to allow TAS to respond and load the next 10
tweets.
The greatest gain in performance made through programmatic technique was gained by creating a custom
broswer profile that the Selenium package then called upon when opening the browser. Within such a profile
(depending on the choice of browser used[16], it is possible to make tweaks such as to prevent images from
being downloaded and rendered, which is of course the main cause of memory allocation. Furthermore, one

can provide a chosen identity to present a target address with, which can determine the form of the data
a target supplies to a visitor. Presenting oneself, for example, as a 2008 version of a browser could limit
the quality of certain meta data that a target sends, with lower quality meaning less information, leading to
lower memory requirements. These 'tricks' were necessary to allow each scrolling session to run as long as
possible before significantly eroding performance or possibly crashing, losing all progress.
In order to design an algorithm capable of recursing over the desired time-span of over two years on the
interface, it was necessary to first carry out many tests to obtain some heuristic parameters for a batchprocess.
When the Twitter timeline is being scrolled, it is thought of as reaching ever further back into the past. Each time the user scrolls downards and loads more tweets, the newly loaded tweets are older than
the previous tweets.
The target URL is composed manually for each batch, including start and end dates for one of the two week time blocks and one of the thirteen seach terms. This URL is then fed into
The scroll.count variable is initialised to zero. The scroll.limit must be sufficient for periods with many tweets to be fully scrolled, not to lose any tweets. This limit of course differs with each search term as some
return many more tweets than others[17]. It was heuristically determined that a value of approximately 500
Algorithm 1 describes the iterative process performed by the web scraper for each scroll session defined.

[15]Random Access Memory (RAM).
[16]Drivers for Mozilla Firefox, Google Chrome and others exist, however ChromeDrivery proved itself to be the most reliable when highly customised.
[17]Compare the number of tweets obtained for search terms 'interest rates' and 'bull market'.

Input: target URL, scroll.count, scroll.limit
Output: HTML code
while scroll.count less than scroll.limit do
scroll to bottom of page;
if at end of page then
wait 3 seconds for next tweets to load;
current position becomes this one;
else
scroll to bottom of page;
end
end
begin
copy HTML source code
end

Algorithm 1: Iterative web-scraping algorithm for a dynamically loading website

[Algorithm 2] shows how [Algorithm 1] is then extrapolated into a batch process that will scrape tweets over the desired time-span. Each scroll session covers a data range of two weeks.

Input: time-span start and end dates
Output: HTML code of TAS results
while scroll.count less than scroll.limit do
scroll to bottom of page;
if at end of page then
wait 3 seconds for next tweets to load;
current position becomes this one;
else
scroll to bottom of page;
end
end
begin
copy HTML source code

end
Algorithm 2: Batch-process algorithm to recursively scrape over desired time-span
7

3.4 Parsing the HTML code

The parts of the data that are useful for this study are summarised in Table 1 below. The Description column outlines the data available for each tweet, whereas the Usage column describes what was ultimately
extracted for use in the modelling.

Data	Description	Usage
timestamp	A millisecond accurate timestamp	The calendar day
tweet-ID	A unique identifier	Remove any duplicates
tweet text	The text string (max.140 characters)	Sentiment analysis
times retweeted	Number of times a tweet was retweeted	As variable and weighting factor*
times favourites	Number of times a tweet was favourited	As variable and weighting factor*

Table 1: Summary of Twitter data usage
* This is explained in detail in SectionXX

3.5 How is the HTML code parsed?

As was alluded to in Section, TAS represents the most accessible means of obtaining the desired Twitter
data. Before a suitable web-scraper can be outlined, a description of the interface offered by TAS must be
given. TAS is a dynamically loading webpage interface to a database. This means, that it holds a great deal
of information, but when it called upon it displays only a small portion of the results to begin with; the next portion of results being loaded as soon as the user has scrolled to the bottom of the webpage. This
is common feature implemented by many websites that host data-heavy content, as it enhances the user
experience by delivering a lazy evaluation or just in time approach - data is loaded only at the moment it is
required. Other examples are the Google image search results page and a Facebook user's main news feed.
In the case of TAS, roughly 20 results (i.e. 20 tweets) for a given search are first returned, with the next
10 being loaded dynamically once the user has scrolled to bottom of the page. Given this, it immediately
becomes clear, which difficulties are introduced by using TAS when trying to obtain data over a time-span
of several years: it is necessary to scroll to the bottom of a webpage for every 10 tweets.
To put this into perspective, the Twitter data that was used in the final analysis of this study contained 2,350,217 tweets.

Search term	Total tweets	Time-span coverage (days)	(% of 982 days)
bear market	47,924	963	98.1
bull market	74,937	965	98.3
dow jones	250,112	982	100.0
dow SPDR	1,628	700	71.3
dow wallstreet	26,395	921	93.8
federal reserve	378,970	904	92.1
financial crisis	261,500	922	93.9
goldman sachs	289,485	909	92.6
interest rates	396,765	857	87.3
market volatility	60,858	970	98.8
obama economy	202,654	908	92.5
oil prices	219,766	785	79.9
stock prices	139,223	982	100.0

Table 2: Breakdown of the total number of tweets extracted by search term
8

3.6 Pre-processing tweet text

1. Here we show what kind of things had to be removed (example of the hex code for smileys etc.)

Comment [MS5]: Add in section number

2. Some things left in because certain sentiment models actually use them, e.g. :-)
3. Explanation of how regex code using Perl engine and hex-codes was used with a code snippet linked
in the appendix.
4. Before and after of one tweet.

3.7 Final output for sentiment analysis

Everything can be interpreted by the SA models to produce reliable results. Here we summarise the final
product of the work detailed in the chapter and summarise how it is stored. Description of overall process.
"Gone from HTML code to raw text, cleaned text, ready to process. Next chapter explains how SA was
performed." Link here to the flow-charts in appendix?

4 Time-series analysis - N versus P

When conducting time-series analysis, there are no hard-and-fast rules governing how many time-periods
must be included to guarantee model robustness. It is a question whose answer changes depending on the
data being used. There is a trade-off to be found between three main components: the number of periods
available, the number of covariates used (i.e. the number of model parameters to be estimated) and lastly
the level of noise within the data. [Reference - Book by Hyndman?].
There are other ~~considerations~~ factors that should be taken into consideration in the context of financial markets,
and that is of trends and cycles. There are times in which an asset (e.g. a single company stock, oil prices
or an entire index) tends to move in one directions. It shows some level of momentum. The event of such a
cycle changing may be labelled a fraction or break in the assets price-path. The approach taken here to deal
with this facet of financial time-series is to make extensive use of a model parameters named frame-size,
which describes how many time-periods are used for each model that is fitted. Its usage is explained in more
details in [section: modelling, parameters, shifting the time-frame]