8/10

2014 - 131 pages - 2021.11.01

[TO DO]

→ This is more of a text-book rather than paper

→ 2 relevant chapters to read !

Text and Context: Language Analytics in Finance

Text and Context: Language Analytics in Finance

Sanjiv Ranjan Das
Santa Clara University
Leavey School of Business
srdas@scu.edu

the essence of knowledge
Boston — Delft

Foundations and Trends® in Finance

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

S. R. Das. *Text and Context:*
Language Analytics in Finance. Foundations and Trends® in Finance, vol. 8, no. 3, pp. 145–260, 2014.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-60198-910-9

Foundations and Trends® in Finance
Volume 8, Issue 3, 2014

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Finance publishes survey and tutorial articles in the following topics:

- Corporate finance
  - Corporate governance
  - Corporate financing
  - Dividend policy and capital structure
  - Corporate control
  - Investment policy
  - Agency theory and information
- Financial markets
  - Market microstructure
  - Portfolio theory
  - Financial intermediation
  - Investment banking
  - Market efficiency
  - Security issuance
  - Anomalies and behavioral finance
- Asset pricing
  - Asset-pricing theory
  - Asset-pricing models
  - Tax effects
  - Liquidity
  - Equity risk premium
  - Pricing models and volatility
  - Fixed income securities
- Derivatives
  - Computational finance
  - Futures markets and hedging
  - Financial engineering
  - Interest rate derivatives
  - Credit derivatives
  - Financial econometrics
  - Estimating volatilities and correlations

Information for Librarians

Foundations and Trends® in Finance, 2014, Volume 8, 4 issues. ISSN paper version 1567-2395. ISSN online version 1567-2409. Also available as a combined paper and online subscription.

Foundations and Trends® in Finance
Vol. 8, No. 3 (2014) 145–260

DOI: 10.1561/0500000045

now
the essence of knowledge

Text and Context: Language Analytics in Finance

Sanjiv Ranjan Das
Santa Clara University
Leavey School of Business
srdas@scu.edu

Contents

Abstract

This monograph surveys the technology and empirics of text analytics in finance. I present various tools of information extraction and basic text analytics. I survey a range of techniques of classification and predictive analytics, and metrics used to assess the performance of text analytics algorithms. I then review the literature on text mining and predictive analytics in finance, and its connection to networks, covering a wide range of text sources such as blogs, news, web posts, corporate filings, etc. I end with textual content presenting forecasts and predictions about future directions.

S. R. Das. *Text and Context:*
Language Analytics in Finance. Foundations and Trends® in Finance, vol. 8, no. 3, pp. 145–260, 2014.

DOI: 10.1561/0500000045.

1

What is Text Mining?

> Howard: You know, I'm really glad you decided to learn Mandarin.
> Sheldon: Why?
> Howard: Once you're fluent, you'll have a billion more people to annoy instead of me.
>
> *"The Tangerine Factor"*
> *The Big Bang Theory, Season 1, Episode 17*

If you consider all the data in the universe, only some of it is in numerical form. There is certainly a lot more text.[1] If you read a financial news article, the quantity of text vastly outnumbers the quantity of numbers. Until recently, financial analysis was just based on numbers. Usage of text required human coding of attributes into numerical form before yielding to analysis. This was a slow process, and not exhaustive, given how much textual data is at hand. We are entering the age of

[1] We may also consider images, sound clips, and videos as data, in which case, numerical data comprises a very small portion of human expression and experience. See Mayew and Venkatachalam [2012] for the use of speech analysis in deciphering the emotive content of voice communications by managers of firms.

Big Text, and this monograph describes the current landscape of text analytics.

Text is versatile. It contains nuances and behavioral expression that is not possible to convey using numbers. Behavioral economics makes a case for considering these nuances that permeate human activity, in economics and finance. Advances in computer science have made text mining possible, and finance is replete with applications, and offers substantial payoffs for profit-making ideas using text mining tools.

There are several benefits to enhancing quantitative financial analysis with text mining analytics. First, text contains *emotive content* that may be useful in assessing sentiment in markets. There are several articles in mainstream journals that deal with this topic, both theoretical and empirical [for example, Admati and Pfleiderer, 2001, DeMarzo et al., 2003, Antweiler and Frank, 2004, 2005, Das and Chen, 2007, Tetlock, 2007, Tetlock et al., 2008, Mitra et al., 2008, Leinweber and Sisk, 2010].

Second, text contains *opinions and connections* that may be harvested and assessed for trading rules, or to corroborate other news, or for risk assessment. Many papers examine these issues as well, and present the benefits of such analysis, as in Das et al. [2005], Das and Sisk [2005], Godes et al. [2005], Li [2006], Hochberg et al. [2007].

Third, many facts do not lend themselves to quantitative expression. They may be intrinsically qualitative and better expressed in the form of text. Of course, most qualitative phenomena may be expressed as numerical quantities on a discrete support, but such abstraction results in a loss of holistic meaning. For example, a trading algorithm may examine a news report to determine a buy or sell signal, and text mining tools can use past data on news and trading outcomes to determine the best course of action in a seamless, efficient manner. Coding text using quantitative variables, i.e., dummy variables for the various attributes of text is clunky, spawns too many variables, and is less accurate.

Fourth, numbers tend to aggregate and summarize underlying phenomena, of infinite variety, and the nuances are better expressed in text, which is disaggregated. Numbers are not raw, original data, but

quantifications of characteristics of markets, often first expressed in textual form. For this reason, it is likely that text (such as news streams) contains information that is more timely than numerical financial information, and better suited to predictive analytics. There is evidence that textual information may be used to predict markets, as in Antweiler and Frank [2004], Tetlock [2007], Leinweber and Sisk [2010]. Analyzing large bodies of text enables operationalization of the wisdom of the crowds as discussed in the excellent book by Surowiecki [2004].

The benefits of text mining are easy to see without defining it formally, but it's time to attempt a formal definition. *Text mining is the large-scale, automated processing of plain text language in digital form to extract data that is converted into useful quantitative or qualitative information.* Hence, text mining is automated on big data that is not amenable to human processing within reasonable time frames. It entails extracting data that is converted into information of many types. Text mining may be simple as in key word searches and counts. Or it may require language parsing and complex rules for information extraction. It may be applied to structured text, such as the information in forms and some kinds of web pages, or it may be applied to unstructured text, a much harder endeavor. Text mining is also aimed at unearthing unseen relationships in unstructured text as in meta analyses of research papers, see Van Noorden [2012].[2]

A subfield of text mining is "news analytics." Wikipedia defines it as - "... the measurement of the various qualitative and quantitative attributes of textual (unstructured data) news stories. Some of these attributes are: sentiment, relevance, and novelty. Expressing news stories as numbers permits the manipulation of everyday information in a mathematical and statistical way. News analytics are used in financial modeling, particularly in quantitative and algorithmic trading. Further, news analytics can be used to plot and characterize firm behaviors over time and thus yield important strategic insights about rival firms. News analytics are usually derived through automated text analysis and ap-

[2]See the article by Gary Belsky, "Why Text Mining may be The Next Big Thing" in *TIME*:
`http://business.time.com/2012/03/20/why-text-mining-may-be-the-next-big-thing/print/`.

plied to digital texts using elements from natural language processing and machine learning such as latent semantic analysis, support vector machines, 'bag of words', among other techniques."

In the ensuing chapters we will examine several topics in financial text mining. In Chapter 2 we examine how text is extracted from various web sites and services. Chapter 3 deals with the basics of text analytics such as dictionaries, lexicons, mood scoring, and summarization of text. This is followed by the analytics of text classification in Chapter 4. The performance of text analytic algorithms is assessed using a range of metrics in Chapter 5. A survey of the empirical literature on text mining in finance and the commercialization of textual analytics is discussed in Chapter 6. Finally, we end with a look at the future of text analytics in Chapter 7.

2

Text Extraction

Amy: Did you hold the baby?
Sheldon: I did.
Amy: And how did it make you feel?
Sheldon: Looking into the blank, innocent eyes of a creature that couldn't begin to comprehend anything I was saying ... basically another day at the office.

"The Cooper Extraction"
The Big Bang Theory, Season 7, Episode 11

A substantial part of text mining comprises text extraction from unstructured text documents or web pages. We note that the title of this section is information extraction, which refers to organized data that is created after processing the text download. This means that we undertake a two part process, downloading textual data first, and then cleaning up and organizing unstructured text into structured text or numerical data arranged in tables.

Downloading text from the internet is called "web-scraping" and there are a host of tools available for this procedure. In the next few paragraphs we will review a few of the common approaches used to

download text from web pages, and perform rudimentary clean up. This list is by no means exhaustive.

2.1 Using R for text extraction

The R programming language is increasingly being used to download text from the web and then analyze it. The ease with which R may be used to scrape text from web site may be seen from the following simple command in R:

```
> text = readLines("http://online.wsj.com/news/
␣␣articles/SB10001424052702304179704579459161
␣␣61537152946?mod=WSJ_Home_largeHeadline&
␣␣mg=reno64-wsj")
```

Here, we downloaded the headline article about the Malaysian plane crash from the Wall Street Journal. (Note: the character ␣ in the code above stands for whitespace.)

The function `readLines()` reads in an entire page from the web, and stores each line as a character array. In fact `readLines()` is agnostic about whether the file is a web page, i.e., resident on a server, or a file on your disk. If you have a file on your drive, you may use the function in the same way, making to sure to point to the file location there. To figure out how many lines the page has issue the command:

```
> length(text)
[1] 1855
```

Let's examine the top 11 lines of the web page:

```
> head(text,11)
> head(text,11)
[1] "<!DOCTYPE␣html>"
[2] "<!--␣TESLA␣DESKTOP␣V1␣--><!--LOCAL-->"
[3] "<html␣itemscope=\"\"␣itemtype=\"http://schema.org/
  NewsArticle\"␣xmlns=\"http://www.w3.org/1999/xhtml\""
[4] "␣␣␣␣␣␣lang=\"en-US\"␣data-env=\"prod\"␣data-site=
␣␣\"wsj\"␣data-region=\"na,us\"␣data-layouttype=
␣␣\"article\">"
[5] "<head>"
[6] "␣␣<meta␣http-equiv=\"X-UA-Compatible\"␣content=
```

```
␣␣\"IE=edge\">"
[7] "␣␣<meta␣http-equiv=\"Content-Type\"␣content=
␣␣\"text/html; charset=UTF-8\">"
[8] "␣␣<title>Malaysian␣PM␣Says␣Malaysia␣Airlines␣Flight
␣␣370␣Ended␣in␣Indian␣Ocean␣-␣WSJ.com</title>␣␣"
[9] "␣␣<meta␣name=\"dj.asn\"␣content=\"ip--27\"␣>"
[10] "<meta␣name=\"user.type\"content=
\"nonsubscriber\">"
[11] "<meta␣name=\"user.exp\"content=\"default\"␣>"
```

This produces the usual html code we see at the top of a web page. Suppose we just want the sixth line, we do

```
> text[6]
[1] "␣␣<meta␣http-equiv=\"X-UA-Compatible\"
content=\"IE=edge\">"
```

And, to find out the character length of the sixth line we use the function

```
> library(stringr)
> str_length(text[6])
[1] 55
```

We have first invoked the library `stringr` that contains many string handling functions. In fact, we may also get the length of each line in the `text` vector by applying the function `str_length()` to the entire `text` vector.

```
> text_len = str_length(text)
> text_len[55]
[1] 123
> text_len[6]
[1] 55
```

We see that some lines are very long and are the ones we are mainly interested in as they contain the bulk of the story, whereas many of the remaining lines that are shorter contain html formatting instructions. Thus, we may extract the top three lengthy lines with the following set of commands.

```
> res = sort(text_len, decreasing=TRUE,
```

```
 index.return=TRUE)
> idx = res$ix[1:3]
> text = text[sort(idx)]
```

The first command above sorts the file by text length of each line, and the second command retrieves the line numbers of the three longest lines. The sort option `index.return=TRUE` returns into object `res` two lists, one the sorted values `res$x` and the index of the sorted values `res$ix`. The third command restricts `text` to the three longest lines.

In short, text extraction can be exceedingly simple, though getting clean text is not as easy an operation. Removing html tags and other unnecessary elements in the file is also a fairly simple operation. We undertake the following steps that use generalized regular expressions (i.e., `grep`) to eliminate html formatting characters.

```
> text = paste(text, collapse="\n")
> text = str_replace_all(text, "[<>&;,.]", " ")
```

This will generate one single paragraph of text, relatively clean of formatting characters. Such a text collection is also known as a "bag of words".

The `XML` package in R also comes with many functions that aid in cleaning up text and dropping it (mostly unformatted) into a flat file. This may then be further processed. Example code for this is as follows.

```
require(RCurl)
require(XML)

txt = getURL("http://www.thestreet.com/story/
12538792/1/
jim-cramers-madmoney-recap-the-enemy-is-us.html?
puc=yahoo&cm_ven=YAHOO", .encoding = "UTF-8")

page1 = htmlTreeParse(txt, useInternal = TRUE)

cleantxt = xpathApply(page1, "//body//text()
[not(ancestor::script)][not(ancestor::style)]
[not(ancestor::noscript)]", xmlValue)
```

```
write.table(cleantxt, "clean.txt", col.names =
FALSE, quote = FALSE, row.names = FALSE,
sep = "\t", fileEncoding = "UTF-8")
```

This code downloads Jim Cramer's page and strips it of all html tags and generates a clean text file that is saved to the hard drive as `clean.txt`. Later in this article, we present an example in which this clean text is presented as a visualization.

The `XML` package also provides functions that extract tables in html pages. See `readHTMLTable()`. Table information is closer to the goal of information extraction rather than mere text extraction.

2.2 Using the text mining package `tm`

The R programming language supports a text-mining package, succinctly named `tm`. Using functions such as `readDOC()`, `readPDF()`, etc., for reading DOC and PDF files, the package makes accessing various file formats easy.

Text mining involves applying functions to many text documents. A library of text documents (irrespective of format) is called a "corpus." The essential and highly useful feature of text mining packages is the ability to operate on the entire set of documents at one go. For example, let's invoke the `tm` package and create a corpus.

```
> library(tm)
> text = c("INTL␣is␣expected␣to␣announce␣good
␣␣␣earnings␣report",
"AAPL␣first␣quarter␣disappoints","GOOG␣announces
␣␣␣new␣wallet",
"YHOO␣ascends␣from␣old␣ways")
> text_corpus = Corpus(VectorSource(text))
> text_corpus
A corpus with 4 text documents
> writeCorpus(text_corpus)
```

The `writeCorpus` function in `tm` creates separate text files on the hard drive, and by default are names `1.txt`, `2.txt`, etc. The simple

program code above shows how text scraped off a web page and collapsed into a single character string for each document, may then be converted into a corpus of documents using the `Corpus()` function. It is easy to inspect the corpus as follows:

```
> inspect(text_corpus)
A corpus with 4 text documents

The metadata consists of 2 tag-value pairs and
a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
INTL is expected to announce good earnings report

[[2]]
AAPL first quarter disappoints

[[3]]
GOOG announces new wallet

[[4]]
YHOO ascends from old ways
```

This output shows that the corpus is a "list" object in R. To access individual documents, grab the required item in the list, i.e.,

```
> text_corpus[[3]]
GOOG announces new wallet
```

In order to see how a function may be applied across all documents in the corpus, see the following command, for example.

```
> text_upper = tm_map(text_corpus,toupper)
> inspect(text_upper)
A corpus with 4 text documents
```

```
The metadata consists of 2 tag-value pairs
and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
INTL IS EXPECTED TO ANNOUNCE GOOD EARNINGS REPORT

[[2]]
AAPL FIRST QUARTER DISAPPOINTS

[[3]]
GOOG ANNOUNCES NEW WALLET

[[4]]
YHOO ASCENDS FROM OLD WAYS
```

This is the power of processing text within a corpus, i.e., the ability to apply functions directly across all documents. You can easily see that this operation is possible to compute in parallel across many machines in a cluster or cloud.

An important step in some procedures for text mining is to "stem" words so that similar words are not treated as different words. For example, "quote" and "quotation" should be treated the same. In order to make them the same, we need to stem these words using a stemmer, of which a very popular one is that of Porter [1980]. Application of the stemmer would reduce both words to the value "quot".

2.3 Term Document Matrix (Indexing)

An essential output from a corpus is the "term document matrix" or TDM for short. This is a table that provides the frequency count of every word (term) in each document. The number of rows in the TDM

is equal to the number of unique terms, and the number of columns is equal to the number of documents.

```
> tdm = TermDocumentMatrix(text_corpus)
> inspect(tdm)
A term-document matrix (19 terms, 4 documents)

Non-/sparse entries: 19/57
Sparsity           : 75%
Maximal term length: 11
Weighting          : term frequency (tf)

             Docs
Terms         1 2 3 4
  aapl        0 1 0 0
  announce    1 0 0 0
  announces   0 0 1 0
  ascends     0 0 0 1
  disappoints 0 1 0 0
  earnings    1 0 0 0
  expected    1 0 0 0
  first       0 1 0 0
  from        0 0 0 1
  good        1 0 0 0
  goog        0 0 1 0
  intl        1 0 0 0
  new         0 0 1 0
  old         0 0 0 1
  quarter     0 1 0 0
  report      1 0 0 0
  wallet      0 0 1 0
  ways        0 0 0 1
  yhoo        0 0 0 1

> dim(tdm)
[1] 19  4
```

This is an interesting TDM as each word (term) appears just once. The TDM is a matrix object in R and hence is available for any sort of matrix manipulation that may be required. It is the starting point for different kinds of textual analyses that we will consider later on in this article.

2.4 Visualizing Text

A popular approach to visualizing text is the representation of terms in a "word cloud". We use the facile `wordcloud` package in R. There are three steps here: extract text, construct a term document matrix, and generate the word cloud. Example code based on Bloomberg's markets page is as follows:

```
> library(RCurl)
Loading required package: bitops
Warning message:
package "RCurl" was built under R version 2.15.2
> library(XML)
> txt = getURL("http://www.bloomberg.com/
news/markets/")
> mypage = htmlTreeParse(txt, useInternal=TRUE)
> cleantxt = xpathApply(mypage, "//body//text()
[not(ancestor::script)][not(ancestor::style)]
[not(ancestor::noscript)]",xmlValue)

> library(stringr)
> cleantxt = paste(cleantxt,collapse="\n")
> cleantxt = str_replace_all(cleantxt,"\n"," ")

> library(wordcloud)
Loading required package: Rcpp
Loading required package: RColorBrewer
> w = str_split(cleantxt," ")
> w = w[[1]]
> idx = which(w!="")
```

```
> w = w[idx]
> words = unique(w)
> wcount = NULL
> for (j in 1:n) {
+ count = length(which(w==words[j]))
+ wcount = c(wcount,count)
+ }
> wc_sort = sort(wcount,decreasing=TRUE,
    index.return=TRUE)
> wcount = wc_sort$x
> words = words[wc_sort$ix]
> wordcloud(words,wcount)
```

This set of code generates a visual word cloud shown in Figure 2.1. The size of each word reflects the frequency with which it appears on the web page. We have used the `XML` package, the string handling package `stringr`, and the `wordcloud` package. A visual depiction of text in this manner provides a good summary of the content at a first glance. In ensuing sections, we will describe other approaches to making meaning of text using a computational approach.

Word clouds are useful ways of summarizing text. But, they are also useful in detecting errors in parsing or classifying text. In later sections of this manuscript, we will explore the mood scoring and classification of text and news into bullish, neutral, and bearish signals. Many techniques use words and their positive and negative connotations. Sometimes these may be misinterpreted because in finance they are used differently, or they are taken out of context. Using the R package `wordcloud` allows for a visual screen to see whether the scoring of text maps well into the visualized word picture.

2.5 Using Twitter Feeds

Twitter is an increasingly popular source for sentiment mining in finance. We will later explore the literature on using Twitter for trading. Twitter provides an API for downloading tweets into R using the OAuth protocol, and in R this is handled in the `ROAuth` package. The

Figure 2.1: Word cloud generated from Bloomberg's Markets main page.

functions that download and process information from Twitter reside in the `twitteR` package.

The process for connecting R to Twitter involves some complicated handshaking, and the brief code is provided here, though you need to first set up a developer's account on Twitter. This is easy to do, and free. The program code for setting this all up is then given by:

```
> library(twitteR)
> library(ROAuth)
> library(RCurl)
> download.file(url="http://curl.haxx.se/
␣␣␣ca/cacert.pem",destfile="cacert.pem")
> cKey = "xxxx"
> cSecret = "xxxx"
> reqURL = "https://api.twitter.com/
␣␣␣oauth/request_token"
> accURL = "https://api.twitter.com/
```

```
␣␣␣oauth/access_token"
> authURL = "https://api.twitter.com/
␣␣␣oauth/authorize"
> cred = OAuthFactory$new(consumerKey=cKey,
    consumerSecret=cSecret,
    requestURL=reqURL, accessURL=accURL,
    authURL=authURL)
> cred$handshake(cainfo="cacert.pem")
To enable the connection, please direct
your web browser to:
https://api.twitter.com/oauth/authorize?
oauth_token=KExaICHYtTpISKEaUuGnXxWlzXoVdkoXC
When complete, record the PIN given to you and
   provide it here: xxxxxx

> registerTwitterOAuth(cred)
[1] TRUE
> save(list="cred",file="twitteR_credentials")
```

The `cKey` and `cSecret` are obtained from your account on the Twitter developer web site. Enter those values instead of the `xxxx` shown in the program above.

When you issue the `cred$handshake()` command, R will prompt for a pin code, which is also obtained from the developer site. Once that is entered, your R session is then connected to Twitter. The `oauth_token` above is randomly generated each time. At the end of the program, the credentials are saved for repeated use. This set of handshaking commands is somewhat laborious, but subsequently, the use of Twitter is rendered fairly straightforward and simple once the set up is done. Then the entire startup process is then as follows:

```
> library(twitteR}
> load("twitteR_credentials")
> registerTwitterOAuth(cred)
[1] TRUE
```

There are various functions that make extraction of data from tweets very simple. First, users enter hashtags for topics, stock tickers, etc. One can extract tweets that contain a specific hashtag. For example, to pull all tweets that contain the ticker for Apple Computer, issue the command:

```
> atweets = searchTwitter("#AAPL")
> atweets
[[1]]
[1] "jopocop: #Apple's #A7 Processor Truly
'Desktop Class', #iOS Apps Don't Take Full
Advantage via @TechNewsTube
#aapl http://t.co/CO1DjHozFI"

[[2]]
[1] "jopocop: #Apple's #Cyclone microarchitecture
detailed via @TechNewsTube #aapl
http://t.co/1PuzxRhznw"

[[3]]
[1] "jopocop: What CIOs need to know about
#Office365 and #Office for #iPad http://t.co/
ayLkSUHzTw via @Appy_Geek #msft #aapl"

[[4]]
[1] "jopocop: #iPhone6 #phablet concept is
impossibly thin and impossible to build
http://t.co/p4Bn7sXD3a via @Appy_Geek #aapl"

[[5]]
[1] "jopocop: If you use #Facebook a lot, this
terrific #iPhone tweak is a must-have
http://t.co/smWY2fQKhG via @Appy_Geek #fb #aapl"

[[6]]
[1] "jopocop: #AppleID #Phishing Scam Won't
```

```
Take␣Fake␣Data␣http://t.co/dIWvXorz6A
via␣@Appy_Geek␣#aapl"
```

Here `atweets` is a list object containing the most recent 25 tweets (by default). One may then proceed to text analyze these tweets.

To get a list of friends for any user, proceed as follows:

```
> me = getUser("srdas")
> me$getFriends(n=4)
$‘2303751216‘
[1] "FiveThirtyEight"

$‘144592995‘
[1] "Rbloggers"

$‘107563743‘
[1] "profkeithdevlin"

$‘579299426‘
[1] "stevenstrogatz"
```

And, my tweets are really easy to access, as follows:

```
> mytweets = userTimeline("srdas",n=3)
> mytweets
[[1]]
[1] "srdas:␣Want␣a␣Job?␣Learn␣Front-End␣Developing.
http://t.co/jjP69f02gg␣via␣@ozy"

[[2]]
[1] "srdas:␣How␣Your␣Tweets␣Reveal␣Your␣Home
Location␣&gt;@TechReview␣http://t.co/tpF4mJMRmZ"

[[3]]
[1] "srdas:␣RT␣@hnycombinator:␣Leslie␣Lamport
awarded␣Turing␣Award␣http://t.co/ZvusRRVdak
(cmts␣http://t.co/BsRfS62qPJ)"
```

There are many other useful functions in the `twitteR` package. Since the returned results are in the form of lists, it is easy to convert them into a corpus.

2.6 Using Facebook Feeds

As with Twitter, Facebook is also accessible using the `OAuth` protocol but with somewhat simper handshaking. The required packages are `Rfacebook`, `SnowballC`, and `Rook`. Of course the `ROAuth` package is required as well.

To access Facebook feeds from R, you will need to create a developer's account on Facebook, and the current URL at which this is done is: `https://developers.facebook.com/apps`. Visit this URL to create an app and then obtain an app id, and a secret key for accessing Facebook. The commands to complete your handshaking with Facebook are as follows:

```
 library(Rfacebook)
Loading required package: httr
Loading required package: rjson
Warning message:
package 'Rfacebook' was built under
R version 2.15.3
> library(SnowballC)
Warning message:
package 'SnowballC' was built under
R version 2.15.3
> library(Rook)
Loading required package: tools
Loading required package: brew
> library(ROAuth)
Loading required package: RCurl
Loading required package: bitops
Loading required package: digest
Warning messages:
1: package 'RCurl' was built under
```

```
R␣version␣2.15.2
2:␣package␣'digest'␣was␣built␣under
R␣version␣2.15.2
>␣app_id␣=␣"xxxx"
>␣app_secret␣=␣"xxxx"
>␣fb_oauth␣=␣fbOAuth(app_id,␣app_secret,
␣extended_permissions=TRUE)
Copy␣and␣paste␣into␣Site␣URL␣on␣Facebook
␣App␣Settings:␣http://localhost:1410/
When␣done,␣press␣any␣key␣to␣continue...
Waiting␣for␣authentication␣in␣browser...
Authentication␣complete.
```

The last command requires some handshaking through the developer web site, which is easy to do. The code here is standard and is directly taken from the user manual of the `Rfacebook` package. One may save the authorization file for reuse as follows:

```
> save(fb_oauth, file="fb_oauth")
> load("fb_oauth")
```

For example, one may extract the news feed for the Bloomberg News page on Facebook as follows:

```
> bbn = getUsers("bloombergnews",token=fb_oauth)
> names(bbn)
 [1] "id"          "name"        "username"
"first_name" "last_name"
 [6] "gender"      "locale"      "category"
"likes"      "picture"
> page=getPage(page="bloombergnews",token=fb_oauth)
100 posts
> names(page)
 [1] "from_id"         "from_name"
"message"         "created_time"
 [5] "type"            "link"
"id"              "likes_count"
 [9] "comments_count" "shares_count"
```

```
> page[1:5,3]
[1] "Sales of co-ops and condominiums in the first
quarter jumped 35 percent from a year
earlier to 3,307."
[2] "Scrutiny of high-frequency trading is
intensifying, with federal investigators examining
whether firms violate U.S. laws by acting on
nonpublic information."
[3] "Nobody likes \"patent trolls,\" even if
they're not quite sure what they are. "
[4] "In a saturated coffee market, the company
is trying to entice more customers to add a pastry
or croissant to their latte orders. "
[5] "The first phase of Obamacare ended yesterday
much the same way it began."
```

In this code, we see the attributes of the user `bbn` obtained using the `names(bbn)` function. We also download the 100 most recent posts on this page and can examine its attributes and then print the top 5 messages as well. These messages may then be subjected to text mining using the analytics we subsequently explore later in this article. For example we may score the mood of the text.

2.7 Alternate Programming Languages

R is one of the languages of choice for text mining (and data science in general), but it is certainly not the only one. (I have often used C, Java, and Python instead of R, depending on the computing environment.) R is certainly easy to use. The `readLines` function is so simple to run, is versatile, and is part of the main distribution, one is up and running without needing the installation of any additional packages for web scraping. The scraped text is tokenized in R automatically making manipulation extremely facile. As we have seen, the `tm` package is replete with all the functions needed for a corpus-based view of language analytics. And because R is a full-fledged econometrics environment, extracted textual data may be instantly visualized, quantified, and ana-

lyzed statistically, and inference engines may be built within the same environment. An excellent reference book for machine learning in R that includes use of the `tm` package is Conway and White [2012].

Python is probably the most other preferred text mining language. It's use is growing exponentially, along with R, since both languages are widely used by major tech firms such as Google and Facebook. In fact, the triumvirate of C, Python, and R forms the bulk of Google's computing stack. Web crawling in Python uses the `urllib` and `urllib2` packages, which require a few more lines of code than does R's `readLines`. But the web scraping package Beautiful Soup (`http://www.crummy.com/software/BeautifulSoup/`) is a useful tool for grabbing data for further analysis in Python.

Python has the `text mining` package that is useful for generating term document matrices. These may then be analyzed in Python or imported into R for further processing. Conceptually this package is similar to the `tm` package in R, and hence, programmers in one language may seamlessly move from one environment to the other. Overall, both Python and R are more or less the same in functionality, and both contain many tools for text analytics. These two languages are easier to work with than C and Java.

A less functional but more user-interface driven tool for text mining is SAS `Text Miner`. It allows extracting text and uncovering themes within text. Textual data can be scored numerically and then combined with numerical data for further analysis in SAS. If one is already a SAS user, then it may be helpful to use this utility.

3

Basic Text Analytics

> Sheldon: Everyone at the university knows I eat breakfast at 8:00 and move my bowels at 8:20.
> Leonard: Yes, how did we live before Twitter?
>
> *"The Monopolar Expedition"*
> *The Big Bang Theory, Season 2, Episode 23*

In this new age one may frankly say, "I tweet, therefore I am." In this chapter, I present the basic ideas of text analytics, and demonstrate using examples, how one may determine text sentiment. This sentiment may then be used for prediction. In ensuing chapters, we will assess the value of this approach in prediction in finance.

3.1 Dictionaries and Lexicons

3.1.1 Dictionaries

Webster's defines a "dictionary" as "a reference source in print or electronic form containing words usually alphabetically arranged along with information about their forms, pronunciations, functions, etymologies, meanings, and syntactical and idiomatic uses." Computerized text

mining uses all these features of a dictionary, from mere word counts to parsing into nouns, adjectives, verbs, etc., or connotations of words, such as good versus bad, positive versus negative, and so on.

The fact that dictionaries are available in electronic form has made text analytics extremely facile. Programs are able to sweep through large bodies of text and produce summaries and sentiment in mere seconds using specific dictionaries that are adapted to the particular text analytics under consideration. For example, in order to score an article as being optimistic or pessimistic about the economy, a special dictionary containing words with optimistic and pessimistic tags is required. The Harvard General Inquirer (`http://www.wjh.harvard.edu/ inquirer/`) is one such dictionary. We will see an example of the use of this later on in this manuscript. The advantage of an external dictionary such as the Harvard Inquirer is that its composition is beyond the control of the researcher, and hence, the results are less likely to be manipulated to match the textual data set being used.

A cause and consequence of the increased use of dictionaries by text analysis software is the explosion in the types of electronic dictionaries. We have standard dictionaries, essentially electronic versions of the original paper tomes, and these may be accessed online, for example `dictionary.com`, and `www.merriam-webster.com`. On the other hand, specialized dictionaries deal with specific subject matter such as the computer dictionary at `http://www.hyperdictionary.com/computer` that contains about 14,000 computer related words, such as "byte" or "hyperlink". Or a math dictionary, such as `http://www.amathsdictionaryforkids.com/dictionary.html` or a medical dictionary, see `http://www.hyperdictionary.com/medical` for words like "aneurism".

Internet lingo dictionaries may be used to complement standard dictionaries with words that are not usually found in standard language, for example, see `http://www.netlingo.com/dictionary/all.php` for words such as `2BZ4UQT` which stands for "too busy for you cutey" (LOL). When extracting text messages, postings on Facebook, or stock

message board discussions, internet lingo does need to be parsed and such a dictionary is very useful.

Associative dictionaries are also useful when trying to find context, as the word may be related to a concept, identified using a dictionary such as `http://www.visuwords.com/`. See Figure 3.1 for a screenshot of the interlinkages for the word "sentiment". This dictionary doubles up as a thesaurus, as it provides alternative words and phrases that mean the same thing, and also related concepts.

Value dictionaries deal with values and may be useful when only affect (positive or negative) is insufficient for scoring text. The Lasswell Value Dictionary[1] may be used to score the loading of text on the eight basic value categories: Wealth, Power, Respect, Rectitude, Skill, Enlightenment, Affection, and Well being. Within these eight categories is a large listing of subcategories. These are described on the home page of Harvard Inquirer.

3.1.2 Lexicons

A "lexicon" is defined by Webster's as "a book containing an alphabetical arrangement of the words in a language and their definitions; the vocabulary of a language, an individual speaker or group of speakers, or a subject; the total stock of morphemes in a language." This suggests it is not that different from a dictionary. And a "morpheme" is defined as "a word or a part of a word that has a meaning and that contains no smaller part that has a meaning."

In the text analytics realm, we will take a lexicon to be a smaller, special purpose dictionary, containing words that are relevant to the domain of interest. For example, if we are analyzing discussion group data on cancer to understand the concerns of patients, we will construct a lexicon that contains words that are related to medicine and cancer in particular, but we may also include words that are reflective of the stress and emotions that cancer patients experience. To take another

[1] This is part of Harvard Inquirer. For the structure and description of this dictionary, see `http://www.wjh.harvard.edu/~inquirer/lasswell.htm`.

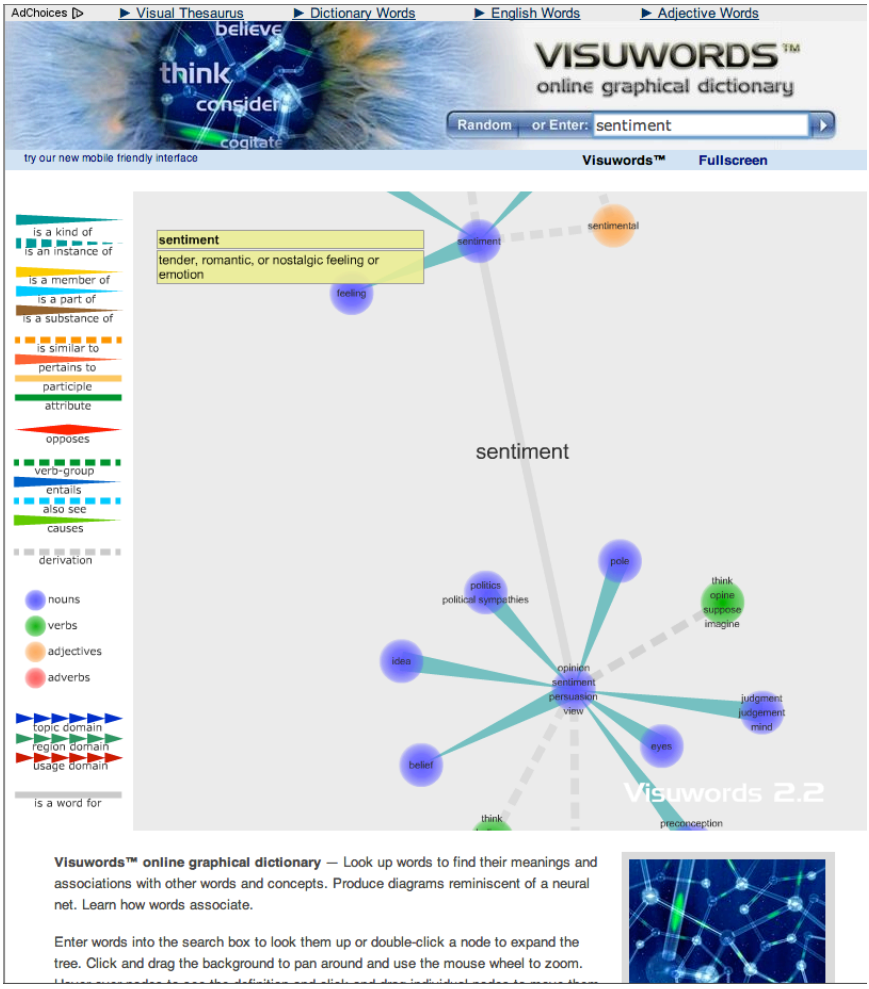

Figure 3.1: Visuwords: An online graphical associative dictionary.

example, if we are looking at stock chat rooms to tease out market sentiment, then we will include finance related words as well as words that reflect optimism and pessimism.

The benefit of a lexicon is that it enables focusing only on words that are relevant to the analytics and discards words that are not. Another benefit is that since it is a smaller dictionary, the computational effort required by text analytics algorithms is drastically reduced. We will discuss this in greater detail when we look at various algorithms for text classification.

3.1.3 Constructing a lexicon

There are many approaches used to construct a lexicon. A first approach is to do this by hand. This is an effective technique and the simplest. It calls for a human reader who scans a representative sample of text documents and culls important words that lend interpretive meaning. For example, if a message about stocks says "I am bullish on this stock" then the word "bullish" is likely to be a good candidate for the lexicon.

A second approach is to use the term document matrix discussed earlier. Scan the entire corpus of sample text, and then examine the term document matrix for most frequent words, and pick the ones that have high connotation for the classification task at hand. For example, in a data set like `CrunchBase` (`http://www.crunchbase.com/`) that is based on reported new venture financings, the words "funding" or "round" may appear many times in the term document matrix, and may be suggestive of positive fortunes for a startup firm. These words would be extracted by hand and included in the lexicon.

A third approach to constructing a lexicon requires pre-classified documents in a text corpus. We analyze the separate groups of documents to find words whose difference in frequency between groups is highest. Such words are likely to be better in discriminating between groups. We will visit this idea more formally later.

If the lexicon is used for classification, it may be further enhanced to include a flag for each group in the classification. For example, if news articles are being used to gauge economic sentiment, then the words

in the lexicon may be tagged to suggest their primary connotation as positive or negative.

Lexicons are often described as word lists. In finance, researchers have begun compiling word lists that are used for specialized purposes. Das and Chen [2007] constructed a lexicon of about 375 words that are useful in parsing sentiment from stock message boards. This lexicon also introduced the notion of "negation tagging" into the literature. Thus the word "good" is treated as having a positive connotation in the lexicon, but the phrase "... not good..." appearing in a sentence suggests that the word good should be treated with the opposite sign for sentiment. Hence, in their algorithm, when a negation word such as "not", "never", etc., appears in a sentence, the words after it are tagged negatively, for example, "good_n", and then the lexicon also contains these additional negated word with the opposite sign. Hence, the lexicon doubles in size after negation tagged words are inserted. However carrying these additional negation tagged words speeds up text processing as the text may be altered to add a negation tag to each word in a sentence, where appropriate, and then all that is needed for a word count of sentiment is to match words to the negation enhanced lexicon.

In recent work, Loughran and McDonald [2011] created tools for textual analysis that include word lists. They tested standard lexicons and found them to be unsuitable for financial text. In fact, word lists for other disciplines lead to misclassification of financial documents. Taking a sample of 50,115 firm-year 10-Ks from 1994 to 2008, they found that almost three-fourths of the words identified as negative by the Harvard Inquirer dictionary (see Section 3.2 below) are not typically negative words in a financial context.

Therefore, they specifically created separate lists of words by the following attributes of words: negative, positive, uncertainty, litigious, strong modal, and weak modal. Modal words are based on Jordan [1999]'s categories of strong and weak modal words. Strong modal words are emphatic such as always, highest, must, and will. Weak modal words express uncertainty such as depend-

ing, might, and possibly. These word lists may be downloaded from http://www3.nd.edu/∼mcdonald/Word_Lists.html.

As textual analysis becomes more widespread, word lists will become more specialized. These are critical ingredients in textual analysis. In the next section we will show how to mood score a body of financial text.

3.2 Mood scoring using Harvard General Inquirer

In this section we explore the mood scoring of text to generate an optimism or pessimism score. In order to do this we may directly use a word list of positive or negative words, but here we show how to create a lexicon of such words from a standard dictionary, in this case the Harvard Inquirer.

The following news report from Bloomberg about the patent law suit between Apple and Samsung is an interesting document to mood score.

```
> text = readLines("http://www.bloomberg.com/news/
print/2014-04-14/samsung-calls-one-of-its-own-at-2-
billion-apple-patent-trial.html")
```

We then eliminate all lines with html code that is for markings and so on, and contains text that we are not interested in.

```
> text = text[setdiff(seq(1,length(text)),
grep("<",text))]
> text = text[setdiff(seq(1,length(text)),
grep(">",text))]
> text = text[setdiff(seq(1,length(text)),
grep("]",text))]
> text = text[setdiff(seq(1,length(text)),
grep("{",text))]
> text = text[setdiff(seq(1,length(text)),
grep("}",text))]
> text = text[setdiff(seq(1,length(text)),
grep("[",text))]
> text = text[setdiff(seq(1,length(text)),
```

```
grep("_",text))]
> text = text[setdiff(seq(1,length(text)),
grep("\\/",text))]
```

At the end of this sequence of commands, there is a variable `text` that is relatively free of html characters and codes. `text` is actually an array of lines from the original web page. Each line (an element of the array) is a string of characters. It is this text that is then subjected to a comparison of words between the text and a list of positive and negative words.

Therefore, the next step is compiling the positive and negative world lists using the Harvard Inquirer. See Figure 3.2 for a screenshot of the dictionary, where the words are signed with "Pos" and "Neg" tags. We can then write code to extract a list of positive and negative words for use from this file.

At the top of the file is a header and it is followed by each word, one per line along with related emotive tags. We see many more words here that are negative, such as ABANDON, ABATE, ABJECT, etc., than positive words such as ABLE, ABIDE.

Here is how we create word lists from the Harvard Inquirer using the following R script:

```
HGI = readLines("HGI.txt")
hgi_pos = HGI[grep("Pos",HGI)]
pos = NULL
for (s in hgi_pos) {
  s = strsplit(s,"#")[[1]][1]
  pos = c(pos, strsplit(s," ")[[1]][1])
}
hgi_neg = HGI[grep("Neg",HGI)]
neg = NULL
for (s in hgi_neg) {
  s = strsplit(s,"#")[[1]][1]
  neg = c(neg, strsplit(s," ")[[1]][1])
}
```

Figure 3.2: Screen shot of the Harvard General Inquirer. After the header, each word has its own line, and after the word, there are a series of tags that represent the emotive content of each word. We focus on only two tags, Pos and Neg.

```
pos = toupper(unique(pos))
neg = toupper(unique(neg))
> print(c(length(pos),length(neg)))
[1] 1647 2121
```

The code above reads in the file `HGI.txt` which may be downloaded from the Harvard Inquirer web site (the file has been renamed here). Using generalized regular expressions (i.e., `grep`) we extract all lines containing `Pos` and `Neg` tags, thereby cleaning up the file and restricting it to words with positive and negative connotations. There are some string handling arguments in the code above that clean up the file and store the words with Neg and Pos tags into the `neg` and `pos` arrays, respectively. As we see, there are more negative words (2121 in number) than positive words (1647 in number) in these word lists which form the lexicon for our next analyses.

The following program code counts the positive and negative values.

```
> text = unlist(strsplit(text," ")
> posmatch = match(text,pos)
> negmatch = match(text,neg)
> print(c(length(posmatch),length(negmatch)))
[1] 1019  1019
```

This code snippet takes the text and splits into an array of separate words. Then the `match` function is used to match the words to the lexicon separately for positive and negative words. A simple count of matched words leads to an indication of sentiment. Interestingly, there are equal numbers of positive and negative word matches in the dictionary. Hence, the mood score is neutral.

Just to round things out, it is also useful to point out that looking for similar words is often useful in textual analysis. For example, if we start with a small set of positive connotation words in finance, and then wish to explore a document for all positive words, a thesaurus like lexicon such as WordNet may be used to expand the small set of words into all associated words with similar positive meaning. WordNet is a large database of words in English, i.e., a lexicon. See Miller [1995], Fellbaum [1998]. The repository is at `word net.princeton.edu`. WordNet

groups words together based on their meanings (synonyms) and hence may be used as a thesaurus. WordNet is also useful for natural language processing as it provides word lists by language category, such as noun, verb, adjective, etc.

3.3 Stemming and Stop Words

Stemming reduces words to their root morphological forms, and leaves a stem of the word in place. This may be best seen with an example. Here is an extract from the financial news wires for April 15, 2014.

```
> text = "U.S. stocks rose a second day, after
equities posted the worst week since 2012, as
Coca-Cola Co. and Johnson & Johnson
reported earnings and investors weighed
developments in Ukraine. The Nasdaq Composite
Index gained 0.1 percent, erasing an earlier drop
of 1.9 percent after touching its 200 day average
price. Coca-Cola gained 3.6 percent as global
volume sales increased. Johnson & Johnson
climbed 1.7 percent as the company raised its
forecast for the year."
```

We demonstrate how to stem this paragraph and remove stop words. We proceed as follows.

```
> library(tm)
> ctext = Corpus(VectorSource(text))
> ctext = tm_map(ctext, removePunctuation)
> ctext = tm_map(ctext, removeNumbers)
> ctext = tm_map(ctext, stemDocument)
> inspect(ctext)
[[1]]
US stock rose a second day after equiti
post the worst week sinc as CocaCola Co
and Johnson Johnson report earn and
investor weigh develop in UkraineTh
Nasdaq Composit Index gain percent eras
```

```
an earlier drop of  percent after touch it  day
averag price CocaCola gain  percent as global
volum sale increas Johnson  Johnson climb
percent as the compani rais it forecast for the
year
```

The result of the stemming operation is quite self-explanatory. Note that we removed all punctuation and numbers.

"Stop words" are non-contextual words, i.e., not germane to interpretation of the text that are removed from the data before conducting textual analysis. This is believed to improve the quality of the analysis on text. Some stop words such as "the" are simple to see as being redundant, but others may be more subtle. There are lists of stop words that may be used for this parsing procedure, such as "a", "are", "as", "to", etc.

Using the same example as above, we remove stop words as follows:

```
> stoplist = c(stopwords("english"))
> ctext = tm_map(ctext,removeWords,stoplist)
> inspect(ctext)
[[1]]
US stocks rose  day  equities posted  worst week
CocaCola Co  Johnson  Johnson reported earnings
investors weighed developments  UkraineThe Nasdaq
Composite Index gained  percent erasing earlier
drop percent  touching  day average price CocaCola
gained  percent  global volume sales increased
Johnson Johnson climbed  percent company
raised forecast
```

The `tm` package comes with an English stop word list and it may be enhanced by adding more words to it as needed and as the domain requires. Here we used the `tm_map()` function with the `removeWords` option to eliminate all stop words from the passage of text.

3.4 Text Summarization

Digital news feeds generate more text than we can hope to consume and comprehend. Summaries of documents may aid in our handling of text manually, by making reading tasks more efficient. Moreover, automated trading systems that are based on text signals need to distill down text in order to reduce noise and extract a signal. One way to find a needle in a haystack is to reduce the size of the haystack, and this is what text summarization is analogous to.

The simplest form of text summarizer we may build operates on a sentence-based model that sorts sentences in a document in descending order of word overlap with all other sentences in the text. Hence, the first sentence in a summary is the one that has the greatest commonality to all other sentences; The second sentence is the one with the second greatest word overlap, and so on.

A document D is comprised of m sentences $s_i, i = 1, 2, ..., m$, where each s_i is a set of words. We compute the pairwise overlap between sentences using the Jaccard [1901] similarity index:

$$J_{ij} = J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} = J_{ji} \tag{3.1}$$

The overlap is the ratio of the size of the intersect of the two word sets in sentences s_i and s_j, divided by the size of the union of the two sets. The similarity score of each sentence is computed as the row sums of the Jaccard similarity matrix.

$$\mathcal{S}_i = \sum_{j=1}^{m} J_{ij} \tag{3.2}$$

Once the row sums are obtained, they are sorted and the summary is the first n sentences based on the $\mathcal{S}_i$ values.

An alternate approach to using row sums is to compute centrality using the Jaccard matrix J, and then pick the n sentences with the highest centrality scores.

The procedure is best illustrated with a news article from the financial markets. The sample text is taken from Bloomberg on April 21, 2014, at the following URL:

http://www.bloomberg.com/news/print/2014-04-21/wall-street-bond-dealers-whipsawed-on-bearish-treasuries-bet-1-.html. The full text spans 4 pages and is presented in Appendix 8.

This article is read using a web scraper (as seen in preceding sections), and converted into a text file with a separate line for each sentence. We call this file summary_text.txt and this file is then read into R and processed with the following parsimonious program code.

```
# TEXT SUMMARIZATION

# FUNCTION TO RETURN n SENTENCE SUMMARY
# Input: array of sentences (text)
# Output: n most common intersecting sentences
text_summary = function(text, n) {
  m = length(text)  # No of sentences in input
  jaccard = matrix(0,m,m)  #Store match index
  for (i in 1:m) {
    for (j in i:m) {
      a = text[i]; aa = unlist(strsplit(a," "))
      b = text[j]; bb = unlist(strsplit(b," "))
      jaccard[i,j] = length(intersect(aa,bb))/
                          length(union(aa,bb))
      jaccard[j,i] = jaccard[i,j]
    }
  }
  similarity_score = rowSums(jaccard)
  res = sort(similarity_score, index.return=TRUE,
          decreasing=TRUE)
  idx = res$ix[1:n]
  summary = text[idx]
}

# READ IN THE TEXT FILE AND REMOVE BLANK LINES
text = readLines("summary_text.txt")
idx = which(text!="")
text = text[idx]
```

```
# CALL TEXT SUMMARIZER
res = text_summary(text,10)
print(res)
```

The code is a function that generates a ten-sentence summary of a four page article. The following output is generated, and is a succinct representation of the original article.

```
> source("text_summarizer.R")
[1] "After␣surging␣to␣a␣29-month␣high␣of␣3.05
percent␣at␣the␣start␣of␣the␣year,␣yields␣on␣the
10-year␣note␣have␣declined␣and␣were␣at␣2.72
percent␣at␣7:42␣a.m.␣in␣New␣York."
[2] "Average␣daily␣trading␣has␣also␣dropped␣to
$551.3␣billion␣in␣March␣from␣an␣average␣$570.2
billion␣in␣2007,␣even␣as␣the␣outstanding␣amount
of␣Treasuries␣has␣more␣than␣doubled␣since
the␣financial␣crisis,␣according␣data␣from␣the
Securities␣Industry␣and␣Financial␣Markets
Association."
[3] "While␣the␣Fed's␣decision␣to␣inundate␣the␣U.S.
economy␣with␣more␣than␣$3␣trillion␣of␣cheap␣money
since␣2008␣by␣buying␣Treasuries␣and␣mortgaged-
backed␣bonds␣bolstered␣profits␣as␣all␣fixed-income
assets␣rallied,␣yields␣are␣now␣so␣low␣that␣banks
are␣struggling␣to␣make␣money␣trading␣government
bonds."
[4] "They␣collectively␣amassed␣$5.2␣billion␣of
wagers␣in␣March␣that␣would␣profit␣if␣Treasuries
fell,␣the␣first␣time␣they␣had␣net␣short␣positions
on␣government␣debt␣since␣September␣2011,
data␣compiled␣by␣the␣Fed␣show."
[5] "The␣world's␣largest␣economy␣added␣fewer␣jobs
on␣average␣in␣the␣first␣three␣months␣of␣the␣year
than␣in␣the␣same␣period␣in␣the␣prior␣two␣years,
data␣compiled␣by␣Bloomberg␣show.␣"
```

```
[6] "LaVorgna, who has the highest estimate among
the 66 responses in a Bloomberg survey, said
stronger economic data will likely cause investors
to sell Treasuries as they anticipate a rate
increase from the Fed."
[7] "The wagers may include market-making, which
is the business of using the firmÕs capital to buy
and sell securities with customers while profiting
on the spread and movement in prices."
[8] " "During the crisis, the Fed went to great
pains to save primary dealers," Christopher
Whalen, banker and author of ``Inflated: How
Money and Debt Built the American Dream,"
said in a telephone interview. "
[9] "In the past, "calling the direction of the
market and what you should be doing in it was
a lot easier than it is today, particularly for
the dealers.""
[10] "Treasuries (USGG10YR) have confounded
economists who predicted 10-year yields would
approach 3.4 percent by year-end as a strengthening
economy prompts the Fed to pare its unprecedented
bond buying."
```

This chapter showed how extracted text (using tools from Chapter 2) may be characterized and modified using dictionaries, lexicons, and text-mining tools to clean up and summarize text. In the next chapter we explain how sentiment is extracted and the way in which documents are classified based on sentiment.

4

Text Classification

> Penny: You keep him there a little longer, and when you get to the party, I'll point out which of my friends are easy.
> Howard: Don't toy with me, woman.
> Penny: I got a hot former fat girl with no self-esteem. I got a girl who punishes her father by sleeping around, and an alcoholic who's 2 tequila shots away from letting you wear her like a hat.
> Howard: Thy will be done.
>
> *"The Peanut Reaction"*
> *The Big Bang Theory, Season 1, Episode 16*

Modern text mining is as much about reading and understanding text as it is about profiling and bucketing humans and there opinions into categories for commercial applications.

Pick up the Wall Street Journal and start reading. After scanning through article after article about one company and another, stop and reflect on your thought process. One salient aspect is that for each firm that you read about, your mind arrived at a sentiment value for that

firm. What you are engaged in is text mining, mapping each article to a simple rubric of positive, neutral, or negative sentiment for the firm being reported on. It is only natural for humans to want to automate something as simple as this. This chapter details various approaches for classifying documents (of various types such as press releases, Dow Jones news feeds, chat room discussion, etc.) by sentiment into simple categories.

We call this process "sentiment extraction" but it is nothing new. In an older incarnation, for a very different domain, we have known it as spam filtering. When we receive email, our computer "reads" the mail and "comprehends" whether it is spam or not, a simple binary choice. Spam and other filters are modern day electronic secretaries that decide what gets read and what doesn't. Adapting spam filtering to document classification for sentiment extraction is a simple process and has been extensively used. We discuss this in the following sections.

Machine classification is, from a layman's point of view, nothing but learning by example. In new-fangled modern parlance, it is a technique in the field of "machine learning". Learning by machines falls into two categories, supervised and unsupervised. When a number of explanatory X variables are used to determine some outcome Y, and we train an algorithm to do this, we are performing supervised (machine) learning. The outcome Y may be a dependent variable (for example, the left hand side in a linear regression), or a classification (i.e., discrete outcome). When we only have X variables and no separate outcome variable Y, we perform unsupervised learning. For example, cluster analysis produces groupings based on the X variables of various entities, and is a common example.

Machine learning usually involves a subsample of the data that is used to fit the algorithm, and then another subsample used to test its predictive accuracy. The former subsample of data is known as the "training" data and the latter is the "test" data (out of sample). There may even be a third hold-out sample, known as "validation" data, used for final testing of models that have passed muster on test data.

4.1 Bayes classifiers

In Bayes classification, we use the training data to define the prior probabilities of the classification and then these priors are used with additional cases to determine the classification of these cases from the posterior probabilities of them falling into the specified categories.

Bayes classification extends the Document-Term model we encountered in Section 2.3 with a document-term-classification model. These are the three entities in the model and we denote them as (d, t, c). Assume that there are D documents to classify into C categories, and we employ a dictionary/lexicon (as the case may be) of T terms or words. Hence we have $d_i, i = 1, ..., D$, and $t_j, j = 1, ..., T$. And correspondingly the categories for classification are $c_k, k = 1, ..., C$.

Suppose we are given a text corpus of stock market related documents (tweets for example), and wish to classify them into bullish (c_1), neutral (c_2), or bearish (c_3), where $C = 3$. We first need to train the Bayes classifier using a training data set, with pre-classified documents, numbering D. For each term t in the lexicon, we can compute how likely it is to appear in documents in each class c_k. Therefore, for each class, there is a T-sided dice with each face representing a term and having a probability of coming up. These dice are the prior probabilities of seeing a word for each class of document. We denote these probabilities succinctly as $p(t|c)$. For example in a bearish document, if the word "sell" comprises 10% of the words that appear, then $p(t = \text{sell}|c = \text{bearish}) = 0.10$.

In order to ensure that just because a word does not appear in a class, it has a non-zero probability we compute the probabilities as follows:

$$p(t|c) = \frac{n(t|c) + 1}{n(c) + T} \tag{4.1}$$

where $n(t|c)$ is the number of times word t appears in category c, and $n(c) = \sum_t n(t|c)$ is the total number of words in the training data in class c. Note that if there are no words in the class c, then each term t has probability $1/T$.

A document d_i is a collection or set of words t_j. The probability of seeing a given document in each category is given by the following

multinomial probability:

$$p(d|c) = \frac{n(d)!}{n(t_1|d)! \cdot n(t_2|d)! \cdots n(t_T|d)!} \times p(t_1|c) \cdot p(t_2|c) \cdots p(t_T|c) \quad (4.2)$$

where $n(d)$ is the number of words in the document, and $n(t_j|d)$ is the number of occurrences of word t_j in the same document d. These $p(d|c)$ are the prior probabilities in the Bayes classifier, computed from all documents in the training data. We then use these to compute the posterior probabilities using documents from the test data. The probabilities are computed for each document in the test data as follows:

$$p(c|d) = \frac{p(d|c)p(c)}{\sum_k \ p(d|c_k)p(c_k)}, \forall k = 1, \ldots, C \quad (4.3)$$

Note that we get C posterior probabilities for document d, and assign the document to class $\max_k c_k$, i.e., the class with the highest posterior probability for the given document. In essence, this approach may be used to classify any text document into pre-assigned categories for which training data is available. This is also, in essence, how a spam filter works, where the training data is accumulated when assign emails to spam.

The algorithm may be best explained with a simple example. First, assume a very simple lexicon of just 6 terms.

```
> lexicon = c("buy","sell","hold","good",
"bad","okay")
```

Then we generate ten documents and prepare a document-term matrix. The documents are along the rows, and the columns contain the number of times each word in our lexicon appears in each document.

```
> train_data = matrix(rpois(60,3),10,6)
> train_data
      [,1] [,2] [,3] [,4] [,5] [,6]
 [1,]    2    1    1    3    3    8
 [2,]    1    4    4    1    3    4
 [3,]    8    1    1    2    6    3
 [4,]    4    5    5    3    3    3
 [5,]    3    3    7    2    5    3
```

```
 [6,]    4    3    4    6    5    3
 [7,]    3    6    3    1    1    2
 [8,]    0    1    4    1    3    1
 [9,]    2    2    1    3    7    2
[10,]    2    4    2    3    5    2
```

We also have a vector that contains the known classes of this training data for these ten documents. The class vector is (1 = bullish, 2 = neutral, 3 = bearish):

```
> train_class = as.matrix(c(2,3,1,2,2,1,3,2,3,3))
> train_class
      [,1]
 [1,]    2
 [2,]    3
 [3,]    1
 [4,]    2
 [5,]    2
 [6,]    1
 [7,]    3
 [8,]    2
 [9,]    3
[10,]    3
```

The `e1071` package in R contains the Bayes classifier, and we invoke it to train the classification of text documents. In order to do this we pass it the training data set and also the pre-assigned classification of documents in the training data.

```
> library(e1071)
> model = naiveBayes(train_data,train_class)
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = train_data, y = train_class)
```

```
A-priori probabilities:
train_class
  1   2   3
0.2 0.4 0.4

Conditional probabilities:
           V1
train_class [,1]      [,2]
          1 6.00 2.8284271
          2 2.25 1.7078251
          3 2.00 0.8164966

           V2
train_class [,1]     [,2]
          1  2.0 1.414214
          2  2.5 1.914854
          3  4.0 1.632993

           V3
train_class [,1]     [,2]
          1 2.50 2.121320
          2 4.25 2.500000
          3 2.50 1.290994

           V4
train_class [,1]      [,2]
          1 4.00 2.8284271
          2 2.25 0.9574271
          3 2.00 1.1547005

           V5
train_class [,1]      [,2]
          1  5.5 0.7071068
          2  3.5 1.0000000
          3  4.0 2.5819889
```

```
             V6
train_class [,1]     [,2]
          1 3.00 0.000000
          2 3.75 2.986079
          3 2.50 1.000000
```

The trained classifier contains the unconditional probabilities $p(c)$ of each class, which are merely frequencies with which each document appears, i.e., class 1 appears twice, class 2 appears four times, as does class 3. For each of the six terms in the lexicon we have conditional probability distributions $p(t|c)$ given as the mean and standard deviation of the occurrence of these terms in each class.

We may take this trained model and re-apply to the training data set to see how well it does. We use the `predict` function for this.

```
> pred = predict(model, train_data, type="raw")
> pred
                1            2            3
 [1,] 9.472240e-06 9.999895e-01 1.001314e-06
 [2,] 2.610521e-07 4.808679e-01 5.191318e-01
 [3,] 9.999998e-01 2.277438e-07 2.671533e-15
 [4,] 2.335902e-01 6.923859e-01 7.402396e-02
 [5,] 9.924273e-01 7.402110e-03 1.705938e-04
 [6,] 9.999986e-01 4.230290e-07 9.845067e-07
 [7,] 5.459330e-14 8.106116e-03 9.918939e-01
 [8,] 1.515014e-06 9.448175e-01 5.518095e-02
 [9,] 1.444991e-04 1.642963e-03 9.982125e-01
[10,] 6.715189e-05 3.945796e-02 9.604749e-01
```

We see above the posterior probabilities $p(c|d)$ of each class for each of the ten documents. We assign the class based on the maximum of these three probabilities in each row. The results are as follows.

```
> train_fitted = as.matrix(c(2,3,1,2,1,1,3,2,3,3))
> cbind(train_class, train_fitted)
      [,1] [,2]
 [1,]    2    2
```

```
 [2,]    3    3
 [3,]    1    1
 [4,]    2    2
 [5,]    2    1
 [6,]    1    1
 [7,]    3    3
 [8,]    2    2
 [9,]    3    3
[10,]    3    3
```

We printed out the original class for each document alongside the one predicted by the model. We see that only the fifth document was misclassified (as bullish instead of neutral).

This model was intentionally limited to a very small set of documents and a tiny lexicon so as to make the ideas clear. In practice the training data set may be very large, the lexicon may run into a few hundreds of words, and the classes are usually few. The coding steps essentially remain the same. And the processing time of even very large data sets is extremely economical.

4.2 Support vector machines

Support vector machines (SVMs) are efficient classifiers based on partitioning space using varied metrics. The original idea and work is attributed to Vladimir Vapnik, and a series of seminal works detail the approach, see Vapnik and Lerner [1963], Vapnik and Chervonenkis [1964], for the underlying statistical learning theory, and Vapnik [1995] for the SVM. Vapnik began this work in the Soviet Union, and then later at AT&T Bell Labs, when he moved to the US in the 1990s.

The goal of the SVM is to map a set of entities with inputs $X = \{x_1, x_2, \ldots, x_n\}$ of dimension n, i.e., $X \in R^n$, into a set of categories $Y = \{y_1, y_2, \ldots, y_m\}$ of dimension m, such that the n-dimensional X-space is divided using hyperplanes, which result in the maximal separation between classes Y. A hyperplane is the set of points $\mathbf{x}$ satisfying the equation

$$\mathbf{w} \cdot \mathbf{x} = b$$

where b is a scalar constant, and $\mathbf{w} \in R^n$ is the normal vector to the hyperplane, i.e., the vector at right angles to the plane. The distance between this hyperplane and $\mathbf{w} \cdot \mathbf{x} = 0$ is given by $b/||\mathbf{w}||$, where $||\mathbf{w}||$ is the norm of vector $\mathbf{w}$.

This set up is sufficient to provide intuition about how the SVM is implemented. Suppose we have two categories of data, i.e., $y = \{y_1, y_2\}$. Assume that all points in category y_1 lie above a hyperplane $\mathbf{w} \cdot \mathbf{x} = b_1$, and all points in category y_2 lie below a hyperplane $\mathbf{w} \cdot \mathbf{x} = b_2$, then the distance between the two hyperplanes is $\frac{|b_1 - b_2|}{||\mathbf{w}||}$. The goal of the SVM is to maximize the distance (separation) between the two hyperplanes, and this is achieved by minimizing norm $||\mathbf{w}||$. This naturally leads to a quadratic optimization problem.

$$\min_{b_1, b_2, \mathbf{w}} \frac{1}{2}||\mathbf{w}|| \tag{4.4}$$

subject to $\mathbf{w} \cdot \mathbf{x} \geq b_1$ for points in category y_1 and $\mathbf{w} \cdot \mathbf{x} \leq b_2$ for points in category y_2. Note that this program may find a solution where many of the elements of $\mathbf{w}$ are zero, i.e., it also finds the minimal set of "support" vectors that separate the two groups. The "half" in front of the minimand is for mathematical convenience in solving the quadratic program.

Of course, there may be no linear hyperplane that perfectly separates the two groups. This slippage may be accounted for in the SVM by allowing for points on the wrong side of the separating hyperplanes using cost functions, i.e., we modify the quadratic program as follows:

$$\min_{b_1, b_2, \mathbf{w}, \{\eta_i\}} \frac{1}{2}||\mathbf{w}|| + C_1 \sum_{i=1}^{n} \eta_i + C_2 \sum_{i=1}^{n} \eta_i \tag{4.5}$$

where C_1, C_2 are the costs for slippage in groups 1 and 2, respectively. Often implementations assume $C_1 = C_2$. The values η_i are positive for observations that are not perfectly separated, i.e., lead to slippage. Thus, for group 1, these are the length of the perpendicular amounts by which observation i lies below the hyperplane $\mathbf{w} \cdot \mathbf{x} = b_1$, i.e., lies on the hyperplane $\mathbf{w} \cdot \mathbf{x} = b_1 - \eta_i$. For group 1, these are the length of the perpendicular amounts by which observation i lies above the hyperplane $\mathbf{w} \cdot \mathbf{x} = b_2$, i.e., lies on the hyperplane $\mathbf{w} \cdot \mathbf{x} = b_1 + \eta_i$.

For observations within the respective hyperplanes, of course, $\eta_i = 0$. The optimizer decides which observations are allowed to "slip" out of the hyperplane restriction, hence the η_i become choice variables in the optimization.

In R, the package `e1071` also provides the function `sum` to undertake classification. To illustrate, we apply it on the same data we used for the Naive Bayes case earlier.

```
> library(e1071)
> model = svm(train_data, train_class)
> model

Call:
svm.default(x = train_data, y = train_class)

Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1666667
    epsilon:  0.1

Number of Support Vectors:  10
```

We then take the trained model and apply it to generate the predicted classification of the ten documents. The predicted values when rounded deliver the fitted category. See the output next.

```
> pred = predict(model, train_data, type="raw")
> pred
       1        2        3        4
5        6        7        8
       9       10
2.078806 2.921133 1.481483 2.078716
2.078895 1.479892 2.921324 2.079185
2.907141 2.920822
> out = table(pred,train_class)
```

```
> print(out)
                   train_class
pred                1 2 3
  1.47989177959641  1 0 0
  1.48148340348819  1 0 0
  2.07871568663517  0 1 0
  2.07880601903982  0 1 0
  2.07889543046682  0 1 0
  2.07918461069813  0 1 0
  2.9071410716688   0 0 1
  2.92082236130375  0 0 1
  2.92113327188399  0 0 1
  2.92132365807947  0 0 1
> train_fitted = round(pred,0)
> train_fitted
 1  2  3  4  5  6  7  8  9 10
 2  3  1  2  2  1  3  2  3  3
> cbind(train_class,train_fitted)
      train_fitted
1  2              2
2  3              3
3  1              1
4  2              2
5  2              2
6  1              1
7  3              3
8  2              2
9  3              3
10 3              3
```

The SVM fits the model in-sample with perfect accuracy. Hence, it works better than the Bayes classifier. This is a common experience with SVMs, they tend to be more flexible and deliver better predictive accuracy, hence their widespread popularity.

4.3 Word count classifiers, adjectives, and adverbs

Given a lexicon of selected words, one may sign the words as positive or negative, and then do a simple word count to compute net sentiment or mood of text. We already considered examples of this in Section 3.2. By establishing appropriate cut offs, one can determine the classification of text into optimistic, neutral, or pessimistic. These cut offs are determined using the training and testing data sets.

Word count classifiers may be enhanced by focusing on "emphasis words" such as adjectives and adverbs, especially when classifying emotive content. One approach used in Das and Chen [2007] is to identify all adjectives and adverbs in the text and then only consider words that are within ± 3 words before and after the adjective or adverb. This extracts the most emphatic parts of the text only, and then mood scores it.

4.4 Fisher's discriminant-based word count

Each body of text contains words as atomic elements. When comparing (say) optimistic text with pessimistic text, some words discriminate the two texts better than others, i.e., have greater discriminating ability. This discriminating ability is measured by the Fisher [1936] discriminant.

Fisher's discriminant is simply the ratio of the variation of a given word across groups to the variation within group. If we have two groups of text, spam versus non-spam, and the word "Nigerian" appears on average seven times in the spam group emails, and on average once in the non-spam, and the variation within groups is minimal, then the word will have a high Fisher score. If it so turns out that the word "Nigerian" occurs exactly seven times in each spam email, and never in the non-spam, then the Fisher score will be infinity, because within group variation (in the denominator of the Fisher discriminant calculation) is zero. Such a word would be the perfect discriminant.

More formally, Fisher's discriminant score $F(w)$ for word w is

$$F(w) = \frac{\frac{1}{K}\sum_{j=1}^{K}(\bar{w}_j - \bar{w}_0)^2}{\frac{1}{K}\sum_{j=1}^{K}\sigma_j^2} \tag{4.6}$$

where K is the number of categories and $\bar{w}_j$ is the mean occurrence of the word w in each text in category j, and $\bar{w}_0$ is the mean occurrence across all categories. And σ_j^2 is the variance of the word occurrence in category j. This is just one way in which Fisher's discriminant may be calculated, and there are other variations on the theme as well, as there are alternate ways in which we may compute the ratio of across group variation to within group variation.

We may compute $F(w)$ for each word w, and then use it to weight the word counts of each text, thereby giving greater credence to words that are better discriminants. See Das and Chen [2007] for an application of this technique and its performance.

4.5 Vector distance classifiers

In Section 2.3 we represented any document as a vector of word counts. This vector representation may be generalized to include standard dictionaries, discussed in Section 3.1.

Standard English dictionaries contain 50,000 to 100,000 words. Assume we have a dictionary with $n = 100,000$ words. Each word is identified with a specific position in a vector of length n. Any text may be represented by a vector of word frequencies. For example, the text string "Text mining is easy easy" will contain $(n - 4)$ zeroes and 4 slots for the words in the text that will have value 2 in the slot for the word "easy" and value 1 in the other three slots. This vector representation positions the document as a point at the end of a vector in n-dimensional space.

Suppose we have 500 documents in each of two categories, bullish and bearish. These 1,000 documents may all be placed as points in n-dimensional space. It is more than likely that the points in each category will lie closer to each other than to the points in the other category. Now, if we wish to classify a new document, with vector D_i, the obvious idea is to look at which cluster it is closest to, or which point

in either cluster it is closest to. The closeness between two documents i and j is determined easily by the well known metric of cosine distance, i.e.,

$$1 - \cos(\theta_{ij}) = 1 - \frac{D_i^\top D_j}{||D_i|| \cdot ||D_j||} \tag{4.7}$$

where $||D_i|| = \sqrt{D_i^\top D_i}$ is the norm of the vector D_i. The cosine of the angle between the two document vectors is 1 if the two vectors are identical, and in this case the distance between them would be zero.

In Das and Chen [2007] this approach was used to classify posting on Yahoo! stock message boards. Their paper used a lexicon instead of dictionary, making the size of the word vector economical and the algorithm improved in efficiency with no degradation in classification ability.

5

Metrics

> Leonard: How can 5 not be worse than 1?
> Raj: Yeah, Star Trek 5 is worse than 1.
> Sheldon: Okay, first of all that is a comparison of quality not intensity. Secondly, Star Trek 1 is orders of magnitude worse than Star Trek 5.
> Raj: Are you joking? Star Trek 5 is the standard against which all badness is measured.
>
> *"The Lizard-Spock Expansion"*
> *The Big Bang Theory, Season 2, Episode 8*

An old adage goes - "What can be measured can be managed." The performance (accuracy and efficiency) of text extraction and classification is evaluated using various metrics. In this chapter we present most of the common metrics, as implemented in Das and Chen [2007] and other papers, though some of these have been widely used previously in other contexts than text mining. Metrics need to satisfy a statistical property such as being a significantly able to distinguish between classification categories. Predictive accuracy is important. Does the predictive tool perform well in-sample and out-of-sample? We also

should consider the stability of the algorithm across different data sets. An algorithm that is too specialized may not be robust across data sets.

Metrics are extremely useful for checking the validity of the text mining algorithms. Behind the scenes, even though the code is running, there may be logical errors that are not easy to notice and trace. Metrics offer an indirect way to check on the correctness of algorithms. In the ensuing sections, we consider many such metrics. However, there is often no substitute for hand-checking many use cases. The author undertook such an exercise in the analysis in the Das and Chen [2007] paper. Graduate students were asked to score stock bulletin board messages and their scoring was compared to that of the algorithm. Whereas the algorithm classified postings into bullish, neutral, and bearish correctly two-thirds of the time, human agreement amongst themselves was in fact a little higher, around 72 percent. Given this baseline, judgment from hand-checking suggested that the algorithms did not have major flaws in their logic.

5.1 Confusion Matrix

The confusion matrix is a workhorse tool for assessing classification accuracy. Given K categories, the matrix is of dimension $K \times K$. By convention, the columns relate to the category assigned by the classifier algorithm and the rows refer to the actual category in which the text resides. Each cell (i, j) of the matrix contains the number of text messages that were of type i and were classified as type j. The cells on the diagonal of the confusion matrix state the number of times the algorithm got the classification right. All other cells are instances of classification error. If an algorithm has no classification ability, then the rows and columns of the matrix will be independent of each other. Under this null hypothesis, the statistic that is examined for rejection is as follows:

$$\chi^2[dof = (K-1)^2] = \sum_{i=1}^{K} \sum_{j=1}^{K} \frac{[O(i,j) - E(i,j)]^2}{E(i,j)} \tag{5.1}$$

where $O(i,j)$ are the actual numbers observed in the confusion matrix, and $E(i,j)$ are the expected numbers, assuming no classification ability under the null hypothesis. If $M(i)$ represents the total across row i of the confusion matrix, and $M(j)$ the column total, then

$$E(i,j) = \frac{M(i) \times M(j)}{\sum_{i=1}^{K} M(i)} \equiv \frac{M(i) \times M(j)}{\sum_{j=1}^{K} M(j)} \tag{5.2}$$

The degrees of freedom of the χ^2 statistic is $(K-1)^2$. This statistic is very easy to implement and may be applied to models for any K. A highly significant statistic is evidence of classification ability.

For illustration, suppose we have 100 documents classified into $K = 3$ categories $\{1,2,3\}$. The observed categories are given in vector `x` below and the algorithm classifies them into categories displayed in vector `y`.

```
> x
  [1] 2 2 3 2 2 1 1 2 2 2 3 1 2 1 1 3 2 1 3 3 1 1 2
   1 2 2 3 1 2 2 3 2 2 3 2 1 3 2 2 3 3 2 3
 [44] 1 2 2 1 3 3 2 2 3 2 2 1 1 3 3 3 2 1 2 3 3 3 2
  1 1 1 2 1 1 2 3 2 1 3 2 3 2 2 2 1 2 3 2
 [87] 2 2 2 3 2 2 1 1 2 2 3 3 1 2
> y
  [1] 2 2 3 2 2 1 1 2 2 2 3 1 2 1 1 3 2 1 3 3 2 1 2
  1 2 2 2 1 2 2 3 2 2 3 2 1 3 2 2 3 3 2 3
 [44] 1 2 2 2 2 3 2 2 3 2 2 1 2 3 2 3 2 1 2 3 3 3 2
  1 1 1 2 1 1 2 3 2 1 3 2 3 2 2 2 3 1 3 2
 [87] 2 2 2 3 2 2 1 1 3 2 3 3 1 2
```

The following command easily generates the observed confusion matrix:

```
> Omatrix = table(x,y)
> Omatrix
   y
x     1  2  3
  1  22  3  1
  2   1 44  1
  3   0  3 25
```

The diagonal is quite heavy indicating that this algorithm has good classification ability. How good this is, and whether it is statistically significant may be determined by computing the χ^2 statistic for the confusion matrix which is provided in equation (5.1).

```
> rsum = rowSums(Omatrix)
> rsum
 1  2  3
26 46 28
> csum = colSums(Omatrix)
> csum
 1  2  3
23 50 27
> Ematrix = t(matrix(kronecker(rsum,csum)/
                sum(Omatrix),3,3))
> Ematrix
      [,1] [,2]  [,3]
[1,]  5.98   13  7.02
[2,] 10.58   23 12.42
[3,]  6.44   14  7.56
> test_stat = sum((Omatrix-Ematrix)^2/Ematrix)
> test_stat
[1] 149.435
> 1 - pchisq(test_stat,4)
[1] 0
```

The p-value at 4 degrees of freedom is zero, indicating that the classification achieved is highly statistically significant.

5.2 Accuracy

Algorithm accuracy over a classification scheme is the percentage of text that is correctly classified. This may be done in-sample or out-of-sample. To compute this off the confusion matrix, we calculate

$$\text{Accuracy} = \frac{\sum_{i=1}^{K} O(i,i)}{\sum_{j=1}^{K} M(j)} = \frac{\sum_{i=1}^{K} O(i,i)}{\sum_{i=1}^{K} M(i)}$$

We should hope that this is at least greater than $1/K$, which is the accuracy level achieved on average from random guessing. In practice, I find that accuracy ratios of 60–70% are reasonable for text that is non-factual and contains poor language and opinions.

For the example in the previous section, we compute the accuracy for illustration as follows.

```
> sum(diag(Omatrix))/sum(Omatrix)
[1] 0.91
```

The accuracy is high, 90%.

5.3 False Positives

Improper classification is worse than a failure to classify. In a 2×2 (two category, $n = 2$) scheme, every off-diagonal element in the confusion matrix is a false positive. When $n > 2$, some classification errors are worse than others. For example in a 3–way buy, hold, sell scheme, where we have stock text for classification, classifying a buy as a sell is worse than classifying it as a hold. In this sense an ordering of categories is useful so that a false classification into a near category is not as bad as a wrong classification into a far (diametrically opposed) category.

The percentage of false positives is a useful metric to work with. It may be calculated as a simple count or as a weighted count (by nearness of wrong category) of false classifications divided by total classifications undertaken. In experiments on stock messages in Das and Chen [2007], we found that the false positive rate for the voting scheme classifier was about 10%.

For example, assume that in the example above, category 1 is BULLISH and category 3 is BEARISH, whereas category 2 is NEUTRAL. The false positives would arise from mis-classifying category 1 as 3 and vice-versa. We compute the false positive rate for illustration.

```
> Omatrix
   y
x     1  2  3
  1 22  3  1
  2  1 44  1
```

```
   3   0   3 25
> (Omatrix[1,3]+Omatrix[3,1])/sum(Omatrix)
[1] 0.01
```

The false positive rate is just 1%.

5.4 Sentiment Error

When many articles of text are classified, an aggregate measure of sentiment may be computed. Aggregation is useful because it allows classification errors to cancel—if a buy was mistaken as a sell, and another sell as a buy, then the aggregate sentiment index is unaffected.

Sentiment error is the percentage difference between the computed aggregate sentiment, and the value we would obtain if there were no classification error. In Das and Chen [2007] sentiment error varied from 5-15% across the various data sets used. Leinweber and Sisk [2010] show that sentiment aggregation gives a better relation between news and stock returns.

In a 3-way classification scheme, where category 1 is BULLISH and category 3 is BEARISH, whereas category 2 is NEUTRAL, we can compute this metric as follows.

$$\text{Sentiment Error} = 1 - \frac{M(j=1) - M(j=3)}{M(i=1) - M(i=3)} \tag{5.3}$$

In our illustrative example, we may easily calculate this metric.

```
> rsum = rowSums(Omatrix)
> csum = colSums(Omatrix)
> rsum
 1  2  3
26 46 28
> csum
 1  2  3
23 50 27
> 1 - (-3)/(-2)
[1] -0.5
```

The classified sentiment from the algorithm was $-3 = 23 - 27$, whereas it actually should have been $-2 = 26 - 28$. The percentage error in sentiment is 50%.

5.5 Disagreement

Das et al. [2005] introduced a disagreement metric to gauge the level of conflict in the discussion from stock text messages. The metric uses the number of signed buys and sells in the day (based on a sentiment model) to determine how much difference of opinion there is in the market. The metric is computed as follows:

$$\text{DISAG} = \left| 1 - \left| \frac{B - S}{B + S} \right| \right|$$

where B, S are the numbers of classified buys and sells. Note that DISAG is bounded between zero and one.

Using the true categories of buys (category 1 BULLISH) and sells (category 3 BEARISH) in the same example as before, we may compute disagreement.

```
> DISAG = abs(1-abs((26-28)/(26+28)))
> DISAG
[1] 0.962963
```

Since there is little agreement (26 buys and 28 sells), disagreement is high.

5.6 Correlations

How well does the sentiment from news correlate with financial time series? Is there predictability? Leinweber and Sisk [2010] examine investment signals derived from news and show that there is a significant difference in cumulative excess returns between strong positive sentiment and strong negative sentiment days over prediction horizons of a week or a quarter. The effect appears to be robust across many countries.

We may plot the movement of a stock series, alongside the cumulative sentiment series. The latter is generated by taking classified text,

and counting all 'buys' as +1, 'holds' as zero, and 'sells' as −1, and plotting the cumulative total of scores of the messages. See Figure 5.1 where we plot the sentiment graph and stock graph for Dell Computers, where the sentiment and stock series track each other quite closely. We coin the term "sents" for the units of sentiment.

A casual study of correlations between each stock in the Morgan Stanley High Tech 35 index and the sentiment index for each stock showed an average correlation of 18.8%, ranging from a high of 49.3% (Dell) to a low of −12.4% (Intuit), with most correlations being positive. However, when an index of sentiment is constructed, much of the noise cancels out, and the correlation of the aggregate sentiment across these thirty-five stocks and the MSH35 index is a robust 48.6%, see Das [2011]. This suggests that cross-sectional information on sentiment may be exploited to generate better aggregate sentiment indexes. We will consider more information on correlations in assessments of the empirical work in financial text mining later in this monograph.

5.7 Phase lag metrics

Even though textual information is abundant, mining it results in conversion into condensed numerical scores. This mapping of broad text to narrow numbers may lose the forest for the trees, or give an inherent perception of false precision. Therefore, rather than zero in on debatable fine-grained numerical attributes, it may be easier to examine coarse features of the numerical scores. For instance, in Das [2011] I present a phase lag metric derived from the geometry of the sentiment time series.

A time series may be represented as taking on one of eight "shapes" derived from the features of the plot. We get precisely eight distinct shapes when considering four features of a sentiment plot: the beginning point, end point, maximum, and minimum points. If the maximum and minimum points do not coincide with the beginning and end points, then we get an "Up-Down" plot if the maximum comes before the minimum. If the minimum point comes before the maximum point, then we call it a "Down-Up" plot. Such plots have two points of inflection.

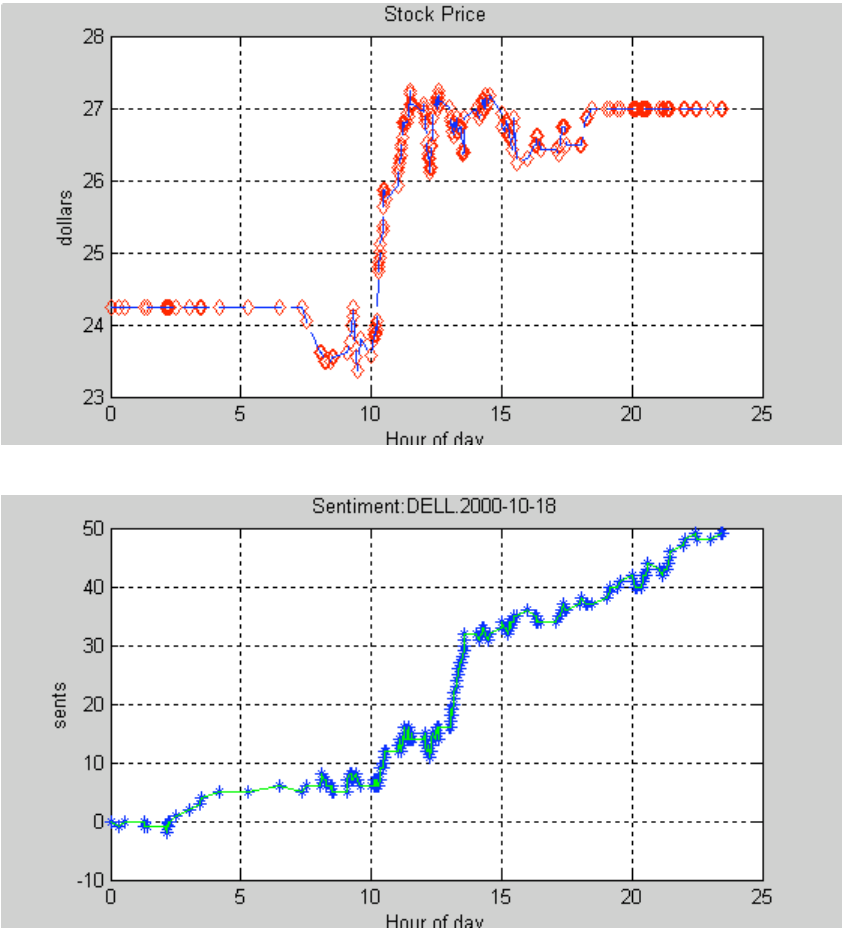

Figure 5.1: Plot of stock series (upper graph) versus sentiment series (lower graph). The correlation between the series is high. The plot is based on messages from Yahoo! Finance and is for a single twenty-four hour period.

When the maximum and minimum points both coincide with the beginning and ending points, there are no points of inflection, and the plots are called “Max-Min" (maximum precedes minimum) or "Min-Max" (minimum precedes maximum).

The remaining four graphs have just one point of inflection. The “Min-Up” plot starts at the minimum and then goes up (there cannot be a “Min-Down” plot for obvious reasons). The “Max-Down” plot starts at the maximum and proceeds downwards to the point of inflection before returning upwards. The “Up-Min” plot moves up from the beginning point, reaches the maximum and then moves down to the minimum. Finally, the “Down-Max” plot moves down from the beginning point, reaches the minimum, and then returns back up to end at the maximum. These eight plots are shown in Figure 5.2.

Phase lag plots are useful for detecting whether sentiment data generated from text may be used to predict stock returns or prices. These eight patterns may be explored in sentiment charts and stock charts simultaneously to determine in which chart the pattern appears first, and with how much lead time. Das [2011] reports that in most cases the stock chart leads the sentiment chart by about 1-2 hours, suggesting that stock prices are not easy to predict with text-based sentiment.

5.8 Readability

Financial documents are often long-winded and loaded with jargon. Average word lengths are higher, and sentences tend to be longer. “Readability” is a metric of how easy it is to comprehend text. Given a goal of efficient markets, regulators want to foster transparency by making sure financial documents that are disseminated to the investing public are readable. Hence, metrics for readability are very important and are recently gaining traction.

Gunning [1952] developed the Fog index. The index estimates the years of formal education needed to understand text on a first reading. A fog index of 12 requires the reading level of a U.S. high school senior (around 18 years old). The index is based on the idea that poor read-

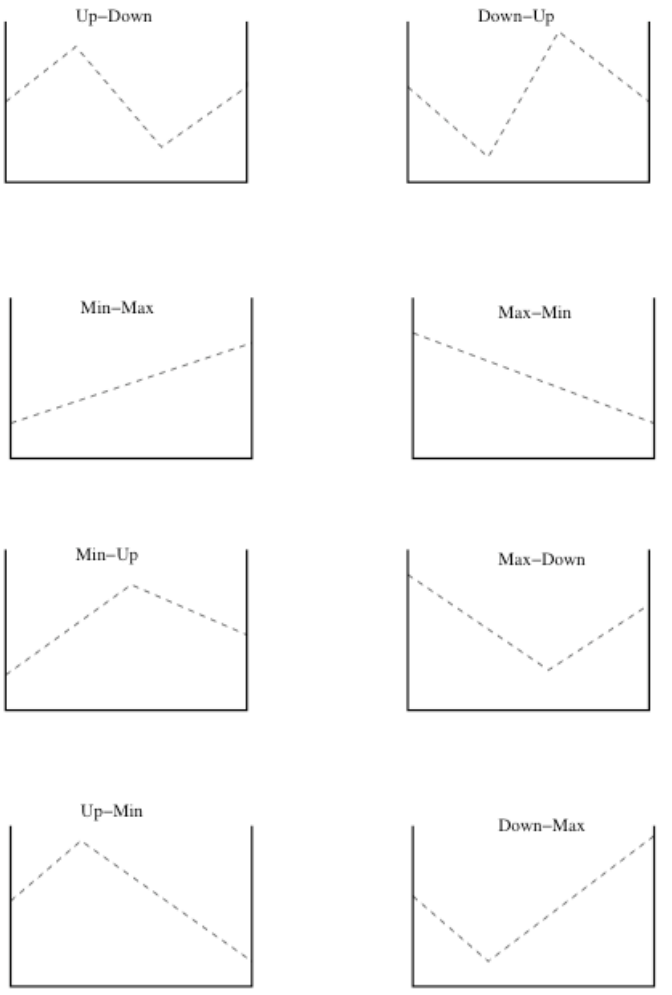

Figure 5.2: Eight different possible shapes of times series when four points are considered, the beginning, ending, maximum, and minimum points. See Das [2011].

ability is associated with longer sentences and complex words. Complex words are those that have more than two syllables. The formula for the Fog index is

$$0.4 \cdot \left[\frac{\#\text{words}}{\#\text{sentences}} + 100 \cdot \left(\frac{\#\text{complex words}}{\#\text{words}} \right) \right]$$

Alternative readability scores use similar ideas. The Flesch Reading Ease Score and the Flesch-Kincaid Grade Level also use counts of words, syllables, and sentences.[1] The Flesch Reading Ease Score is defined as

$$206.835 - 1.015 \left(\frac{\#\text{words}}{\#\text{sentences}} \right) - 84.6 \left(\frac{\#\text{syllables}}{\#\text{words}} \right)$$

With a range of 90-100 easily accessible by a 11-year old, 60-70 being easy to understand for 13-15 year olds, and 0-30 for university graduates.

The Flesch-Kincaid Grade Level is defined as

$$0.39 \left(\frac{\#\text{words}}{\#\text{sentences}} \right) + 11.8 \left(\frac{\#\text{syllables}}{\#\text{words}} \right) - 15.59$$

which gives a number that corresponds to the grade level. As expected these two measures are negatively correlated. Various other measures of readability use the same ideas as in the Fog index. For example the Coleman and Liau [1975] index does not even require a count of syllables, as follows:

$$CLI = 0.0588L - 0.296S - 15.8$$

where L is the average number of letters per hundred words and S is the average number of sentences per hundred words.

Standard readability metrics may not work well for financial text. Loughran and McDonald [2014] find that the Fog index is inferior to simply looking at 10-K file size.

Out of interest, and as an example, I ran readability scores for this monograph you are reading using `https://readability-score.com/`. It has readability scores shown in Table 5.1 below:

[1] See `http://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests`.

Table 5.1: Readability score of this monograph

Readability Formula	Grade
Flesch-Kincaid Grade Level	9.6
Gunning-Fog Score	11.8
Coleman-Liau Index	11.9
SMOG Index	9.5
Automated Readability Index	8.1
Average Grade Level	10.2
Note: The grade level (based on the US education system) is the number of years of education a person has to have had to read this document. The Flesch-Kincaid Reading Ease score for this monograph is 52.8, which is approximately high-school reading level.	

6

Applications and Empirics

> Beverly Hofstadter: Your scan data will be very helpful to my research. You have a remarkable brain.
> Sheldon: I know. Although I've always hated how my right frontal lobe looks in pictures.
> Beverly Hofstadter: Common complaint among men. Nothing's ever big enough, except when they get a tumor. Then you never hear the end of it.
>
> *"The Maternal Capacitance"*
> *The Big Bang Theory, Season 2, Episode 15*

The focus of text mining has been prediction. Well-known forecaster Paul Saffo criticized prediction as a wasted exercise in chasing uncertainty where we are looking for one certain point estimate. In contrast to prediction, a forecast is a statement about the future "cone of uncertainty" where various scenarios are developed with attached probabilities. Still, the term "predictive analytics" appears to be prevalent, and we will use the terms predictions and forecasts interchangeably, i.e., point estimates with error around them.

The literature has accessed varied sources of text to make predictions. Lu et al. [2010] categorize finance-related textual content into three categories: (a) forums, blogs, and wikis; (b) news and research reports; and (c) content generated by firms. Early work focused on extracting sentiment and other information from messages posted to stock message boards such as Yahoo!, Motley Fool, Silicon Investor, Raging Bull, etc., see Tumarkin and R.Whitelaw [2001], Antweiler and Frank [2004], Antweiler and Frank [2005], Das et al. [2005], Das and Chen [2007]. Other news sources have also been accessed such as Lexis-Nexis, Factiva, Dow Jones News, etc., see Das et al. [2005]; Boudoukh et al. [2012]. The Heard on the Street column in the *Wall Street Journal* has been used in work by Tetlock [2007], Tetlock et al. [2008]; see also the use of Wall Street Journal articles by Lu et al. [2010]. Other work uses the Thomson-Reuters NewsScope Sentiment Engine (RNSE) based on Infonics/Lexalytics algorithms and varied data on stocks and text from internal databases, see Leinweber and Sisk [2011]. Zhang and Skiena [2010] develop a market neutral trading strategy using news media such as tweets, over 500 newspapers, Spinn3r RSS feeds, and LiveJournal.

More recently, a string of papers has attempted to exploit Twitter feeds (and to a lesser extent Facebook data). One of the earliest papers is that by Bollen et al. [2010] showing that stock direction of the Dow Jones Industrial Average can be predicted using tweets with 87.6% accuracy. Bar-Haim et al. [2011] attempt to predict stock direction using tweets by detecting and overweighting the opinion of expert investors. Brown [2012] looks at the correlation between tweets and the stock market via several measures. Logunov [2011] uses OpinionFinder to generate many measures of sentiment from tweets. Twitter based sentiment developed by Rao and Srivastava [2012] is found to be highly correlated with stock prices and indexes, as high as 0.88 for returns. Sprenger and Welpe [2010] find that tweet bullishness is associated with abnormal stock returns and tweet volume predicts trading volume. Above average tweeters are retweeted more often too. Sprenger [2011] integrates data from text classification of tweets, user voting, and a proprietary stock game to extract the bullishness of online investors; these ideas are behind the site `TweetTrader.net`. Tweets also pose

interesting problems of big streaming data discussed in Pervin et al. [2013].

Other text mining approaches have been directed at questions that are more in the corporate finance and risk management space. Data used here is from filings such as 10-Ks, etc., (Loughran and McDonald [2011]; Burdick et al. [2011]; Bodnaruk et al. [2013]; Jegadeesh and Wu [2013]; Loughran and McDonald [2014]).

6.1 Predicting Market Movement

Beating the market is the goal of stock prediction. This is not consistently sustainable in an efficient market, since all information (including that extractable from text mining) is accurately and quickly incorporated into prices. Text mining algorithms that really predict market movements are refutations of the efficient market hypothesis. In this section we review the evidence that this is the case.

Sentiment extraction from text mining comprises both quality (meaning) and quantity (volume) of text. The former is harder to analyze because linguistic algorithms are needed. The latter is easier to parse as it entails counting (of words or messages).

An early study by Wysocki [1999] found that for the 50 top firms in message posting volume on Yahoo! Finance, message volume predicted next day abnormal stock returns. Using a broader set of firms, he also found that high message volume firms were those with inflated valuations (relative to fundamentals), high trading volume, high short seller activity (given possibly inflated valuations), high analyst following (message posting appears to be related to news as well, correlated with a general notion of "attention" stocks), and low institutional holdings (hence broader investor discussion and interest), all intuitive outcomes.

Text mining also opens up the variety of sources of information. For example, Bagnoli et al. [1999] examined earnings "whispers", unofficial crowd-sourced forecasts of quarterly earnings from small investors, posted to web pages and news stories that report whisper forecasts. Interestingly, they find that whisper forecasts are more accurate than

that of First Call analyst forecasts. In a negative result, Tumarkin and R.Whitelaw [2001] examined self-reported sentiment on the Raging Bull message board and found no predictive content, either of returns or volume.

Antweiler and Frank [2004] studied more than 1.5 million messages on the Yahoo! Finance and Raging Bull stock message boards. They analyzed about 45 companies in the Dow Jones Industrial Average, as well as the Dow Jones Internet Index. They constructed a "bullishness index" using text mining, and found that they could not predict stock returns, though they could predict stock volatility.

Antweiler and Frank used the Naive Bayes algorithm for classification, implemented in the `Rainbow` package of Andrew McCallum [1996]. They also repeated the same using Support Vector Machines (SVMs) as a robustness check. Both algorithms generate similar empirical results. Once the algorithm is trained, they use it out-of-sample to sign each message as $\{Buy, Hold, Sell\}$. Let n_B, n_S be the number of buy and sell messages, respectively. Then $R = n_B/n_S$ is just the ration of buy to sell messages. Based on this they define their bullishness index

$$B = \frac{n_B - n_S}{n_B + n_S} = \frac{R-1}{R+1} \in (-1, +1)$$

This metric is independent of the number of messages, i.e., is homogenous of degree zero in n_B, n_S. An alternative measure is also proposed, i.e.,

$$\begin{aligned} B^* &= \ln\left[\frac{1+n_B}{1+n_S}\right] \\ &= \ln\left[\frac{1+R(1+n_B+n_S)}{1+R+n_B+n_S}\right] \\ &= \ln\left[\frac{2+(n_B+n_S)(1+B)}{2+(n_B+n_S)(1-B)}\right] \\ &\approx B \cdot \ln(1+n_B+n_S) \end{aligned}$$

This measure takes the bullishness index B and weights it by the number of messages of both categories. This is homogenous of degree between zero and one. And they also propose a third measure, which is

much more direct, i.e.,

$$B^{**} = n_B - n_S = (n_B + n_S) \cdot \frac{R-1}{R+1} = M \cdot B$$

which is homogenous of degree one, and is a message weighted bullishness index. They prefer to use B^* in their algorithms as it appears to deliver the best predictive results. Finally, Antweiler and Frank [2004] produce an agreement index,

$$A = 1 - \sqrt{1 - B^2} \in (0, 1)$$

Note how closely this is related to the disagreement index in Section 5.5.

The bullishness index does not predict returns, but returns do explain message posting. More messages are posted in periods of negative returns, but this is not a significant relationship. As expected, a contemporaneous relation between returns and bullishness is present. Overall, Antweiler and Frank [2004] present some important results that are indicative of the results in this literature, confirmed also in subsequent work. First, that message board postings do not predict returns. Second, that disagreement (measured from postings) induces trading. Third, message posting does predict volatility at daily frequencies and intraday. Fourth, messages reflect public information rapidly. Overall, they conclude that stock chat is meaningful in content and not just noise.

Das and Chen [2007] developed a systematic algorithm for classifying stock board messages. Their approach comprised five algorithms that individually classified each posting into buy, hold, and sell categories. A voting system was overlaid to classify messages based on majority verdict across the five algorithms. The voting approach is shown to improve the signal to noise ratio of the overall classification system, as was an overlaid ambiguity filter. Classification accuracy was found to be much better than random (the confusion matrix had a significant χ^2 statistic), the false positive rate was low, and error in aggregate sentiment was also small. Despite the good quality of the extracted, text-mined sentiment, its predictive value was low. Hence, as in the previously cited work, the prognosis for predicting returns is poor.

Interestingly though, there are some useful contemporaneous findings. Das and Chen [2007] find strong contemporaneous correlation of sentiment and returns. They find that sentiment is inversely related to disagreement (defined in Section 5.5 of this paper). As disagreement increases, sentiment drops. Or interpreted differently, there is greater disagreement when sentiment is falling than when it is rising. Sentiment is also correlated to high posting volume, suggesting that increased discussion is a symptom of favorable opinion (indeed, Antweiler and Frank [2004] name their sentiment index a bullishness index).

Whereas stock message board postings offer one approach to eliciting sentiment information, news stories are another. Boudoukh et al. [2012] (BFKR) conduct an interesting analysis of news by analyzing its textual content for relevance, and for tone. The algorithm they implement determines whether news is relevant or not. They find that days where news is not relevant are not different than days on which there is no news. On the other hand days on which news is relevant have a different impact on returns and volatility. This paper questions the long standing literature that suggests that stock price movements are the result of irrational noise trading or trading based on private information, but not news. In contrast, they posit that when news is correctly identified as relevant versus irrelevant, then news does impact stock prices. They analyze data from 2000–2009 for all S&P500 companies using 1.9 million stories from Dow Jones Newswire (a total of 791 firms that existed at different times in the S&P500). Of these about 50% are identified as relevant. Their analysis does not rely purely on word counts using a dictionary but also on phrase level patterns to determine the tone of the news article.

BFKR do not use machine learning (Naive Bayes, etc.) to classify messages. Instead they determine tone using words, as in much of the preceding literature, and phrases, accounting also for negation. They employ *The Stock Sonar* (TSS) algorithm of Feldman et al. [2011]. The tone is determined using a sentiment score defined as

$$S = \frac{n_{pos} - n_{neg}}{1 + n_{pos} + n_{neg}}$$

where n_{pos}, n_{neg} are the number of positive and negative words, respectively. This algorithm also parses out the context of the news story by eliciting 14 event categories and 56 subcategories. These categories were determined in a rule-based manner using over 4000 rules.

The main results in the Boudoukh et al. [2012] paper are as follows. First, stock volatility on relevant news days is double that of days with irrelevant or no news. This clearly supports the idea that news does move stock prices, once it is correctly classified as relevant. Second, Roll [1988] theoretically posited that a low market model R-square indicates that there is more firm-specific information being priced in the stock, or conversely, on days with no news, stock movement is less idiosyncratic. Consistent with this, BFKR find that market models regressions estimated on days with relevant news have median R^2s of 16%, compared to 28% on days with no news or irrelevant news, or on all days taken together. Third, analysis of text for tone delivers intuitive results, for example, legal announcements tend to be negative, and announcements of deals render positive tone. Fourth, large stock price moves on no news or irrelevant news days are usually followed by reversals as the market corrects its erroneous interpretation, but reversals are not seen after large stock moves on relevant news days, in fact, continuation is more likely. Using this separation of effects a simple portfolio strategy to exploit this yields a Sharpe ratio of 1.7. Taken together, these results suggest that relevant news does move stock prices.

6.1.1 Twitter and News Based Prediction

There is a plethora of papers using Twitter feeds for sentiment construction and prediction. The ease of accessing Twitter talk comes from the facile API interface provided by Twitter and the easy download of messages for a given firm or topic using hashtags. In addition tweets are restricted to 140 characters, and lead to the intuition that the extent of meaning per quantum of text is likely to be much higher for tweets than most other forms of meaningful text. However, these microblog posts exhibit language usage that is often different than standard, and hence, mood scoring might require specific lexicons. For example, the

words "wicked stock" might be exaggerating the value of a stock even though the word "wicked" normally has a negative mood score.

The cost of Twitter sentiment construction is much lower than that of a poll. Further, the sheer quantity of tweets is so large that it lends new scale to the meaning of "wisdom of the crowd." Given the fact that modern computing has made all this possible, we may in fact refer to it as the "wisdom of the cloud." For example, Pervin et al. [2013] develop a system called `TrendMiner` to uncover trending topics on Twitter, showing how useful a high velocity stream from a microblog can be for crowdsourcing opinion. In many cases, tweets reference news items, and hence, aggregate more than just short statements.

Bollen et al. [2010] undertake an extensive analysis of Twitter feeds and relate textual information within tweets to the stock market, in particular the Dow Jones Industrial Average (DJIA). They are able to predict the sign of the daily move in the DJIA correctly 87.6% of the time. They construct two mood-tracking tools. The first, called OpinionFinder (OF)[1] measures positive and negative sentiment. The second, named Google-Profile of Mood States (GPOMS) takes the view that sentiment is more multifaceted than just positive and negative, and therefore measures mood on six dimensions: {Calm, Alert, Sure, Vital, Kind, Happy}. GPOMS is an extension of the well known Profile of Mood States (POMS) in McNair et al. [2003]. These public mood states are predictive of the DJIA, and are cross-validated using Presidential election results and Thanksgiving day outcomes.

Overall, sentiment from the GPOMS appears to be more predictive than sentiment from OpinionFinder. Regressing the OF normalized score on the six normalized mood states from GPOMS shows that the Sure and Happy mood states are significantly related to OF. The OF sentiment predicts the DJIA at a one-day lag, whereas the GPOMS (only the Calm state) predicts the DJIA two to 6 days ahead.

Zhang and Skiena [2010] use Twitter feeds and also three other sources of text: over 500 nationwide newspapers, RSS feeds from blogs,

[1] `http://mpqa.cs.pitt.edu/opinionfinder/`. The system contains corpora and lexicons for open use. See Wilson et al. [2005].

and LiveJournal blogs. These are used to compute two metrics.

$$\begin{aligned} \text{polarity} &= \frac{n_{pos} - n_{neg}}{n_{pos} + n_{neg}} \\ \text{subjectivity} &= \frac{n_{pos} + n_{neg}}{N} \end{aligned}$$

where N is the total number of words in a text document, n_{pos}, n_{neg} are the number of positive and negative words, respectively. They find that the number of articles is predictive of trading volume. Subjectivity is also predictive of trading volume, lending credence to the idea that differences of opinion make markets. Stock return prediction is weak using polarity, but tweets do seem to have some predictive power. Various sentiment driven market neutral strategies are shown to be profitable, though the study is not tested for robustness.

Logunov [2011] uses tweets data, and applies OpinionFinder and also developed a new classifier called Naive Emoticon Classification to encode sentiment. This is an unusual and original, albeit quite intuitive use of emoticons to determine mood in text mining. If an emoticon exists, then the tweet is automatically coded with that sentiment of emotion. Four types of emoticons are considered: Happy (H), Sad (S), Joy (J), and Cry (C). Polarity is defined here as

$$\text{polarity} = A = \frac{n_H + n_J}{n_H + n_S + n_J + n_C}$$

Values greater than 0.5 are positive. A stands for aggregate sentiment and appears to be strongly autocorrelated. Overall, prediction evidence is weak.

Sprenger and Welpe [2010] construct a bullishness index from 250,000 daily tweets and find it to be associated to contemporaneous abnormal stock returns. Message volume is not related to abnormal returns. Interestingly, users that provide better tweet advice tend to be retweeted more than others, thereby expanding their influence. Classification of tweets is undertaken using the Naive Bayes classifier discussed in Section 4.1. The technology for this has been developed in another paper that describes the microblogging forum `TweetTrader.net`, see Sprenger [2011]. The bullishness index is defined as in Antweiler and

Frank [2004]:

$$B = \ln\left[\frac{1 + \#\text{Buy Messages}}{1 + \#\text{Sell Messages}}\right]$$

Of course, the interesting question is whether bullishness can predict returns. The paper finds that the evidence for this is lacking, and in fact, returns appear to predict bullishness, in line with the findings of Das and Chen [2007], and Antweiler and Frank [2004].

Rao and Srivastava [2012] analyze more than 4 million tweets between June 2010 and July 2011 for the DJIA, NASDAQ-100 and 11 other big cap technology stocks. A Naive Bayes classifier is used to determine sentiment, using a service called `Sentiment140` from Stanford University.[2] Twitter sentiment and returns have an 88% correlation. Granger causality regressions show that tweets are predictive of stock and index movements. They implement the Expert Model Mining System (EMMS) to demonstrate that forecasted DJIA returns (R-square of 0.952) have low Maximum Absolute Percentage Error (MAPE) of 1.76%.

Bar-Haim et al. [2011] also generate sentiment using tweets, but in the spirit of Sprenger and Welpe [2010], they build an algorithm to identify expert investors whose tweets are weighted more highly. They also divide tweets (as done in Das et al. [2005]) into questions, noise or obscure items, and the remaining tweets are divided into facts and opinions.

Facts are further classified into news, chart patterns, trade, and trade outcome. News items are useful, but are less valuable than the main source that has been compressed into a tweet. Chart patterns are tweets that describe how the stock moved, and since they are backward looking, are of limited predictive value. A trade reports an actual buy or sell by the poster, and is deemed valuable information by the authors. Trade outcome is the reversal of an existing trade and as such, have limited information over and above the original trade.

Opinions are also broken down into four categories: speculation, chart prediction, recommendation, and sentiment. Speculation is a user opinion provided without basis, and is least useful. Chart prediction is

[2] `http://help.sentiment140.com/`.

a forward estimate of movement based on technical analysis. Recommendation is a prompt to take some action, and is less valuable than a trade fact. Sentiment tweets express pure emotion without any factual content.

Using this taxonomy of tweet types, a small sample was analyzed to come up with the proportion of tweets in each category. A quarter of the tweets were deemed irrelevant and 7% were questions. About 40% of tweets were facts, and the remaining 28% are opinions.

User u has expertise if a large number of his/her bullish tweets are followed by a stock rise. Bar-Haim et al. [2011] exclude bearish tweets followed by a stock fall. This assumption is debatable, though the authors argue that stock selling may be made for reasons other than bearishness, an argument that may also apply to stock purchases. Be that as it may, each tweet t is assessed for its predictive value over the next trading day by examining the sign of stock movement. The precision of user u is then scored as follows:

$$P_u = \frac{C_u}{C_u + I_u}$$

where C_u is the number of correct tweets, i.e., bullish tweets followed by a stock price increase, and I_u is the number of incorrect tweets, i.e., bullish tweets not followed by an uptick in price. The precision of a user is then compared with the average precision of all users in the sample and experts are users with statistically higher precision than average. Classification into bullish, bearish, and neutral is undertaken using a SVM, described in Section 4.2. Overall, the authors find that identifying experts and using user-specific unsupervised learning via SVM leads to better prediction than otherwise. In a manner of speaking, these results would be somewhat circular, by choosing users whose predictions are better, they have to be better predictors than the entire set of users, but there is strong evidence that this works well out-of-sample. Therefore, this paper makes a strong case for the "wisdom of a crowd of experts."

6.2 Predicting risk, volatility, volume

Wysocki [1999] examined message posting volume to see whether it relates to stock market activity. He found that the top 50 firms by message volume (period: January to August 1998) forecasted next day trading volume.

Antweiler and Frank [2004] developed a bullishness index to predict stock returns, but failed to find significant predictability. However, they did find that volatility could be predicted for 45 stocks in the Dow Jones Industrial Average. Consistent with the argument that differences in opinion make markets, they found that disagreement among posted messages was positively correlated with greater trading volume. Das and Chen [2007] also find a significant relationship between message volume and volatility.

The notion of relevance for a news article [Boudoukh et al., 2012] also relates to the concept of *certainty*, a state of being free from doubt. Certainty recognition, discussed in Lu et al. [2010], may be used to distinguish hard facts from doubtful ones in information retrieval. It is closely related to extracting risk related statements in finance textual data. Lending firms may monitor text about their clients continually to elicit risk related warnings. A statement is defined as risk related if there is information about firm value, future timing, and uncertainty. Lu et al. [2010] implemented two statistical learning approaches (SVM and elastic-net logistic regressions) to study 1529 sentences from the Wall Street Journal and found that 47% of these were risk related (with an accuracy level of classification of 70%).

The finding that message volume and trading volume are related appears to be a robust finding in the literature. Zhang and Skiena [2010] find that article volume in news predicts trading volume. Sprenger and Welpe [2010] also find the same relationship between tweet volume and trading volume.

6.3 Text Mining Company Reports

Whereas much of financial text analysis uses messages posted to finance boards like Yahoo!, Motley Fool, or to blogs such as Twitter, Facebook, etc., other textual analysis has been applied to company reports and filings. In the former case, analysis is usually of a time series nature, whereas in the latter, text analysis is undertaken across companies in a cross-section.

The quality of text in company reports is much better than in message postings, and hence, we should expect richer meaning extraction in this domain. Researchers in both Accounting and Finance have begun exploiting filing information. Textual analysis in this area has also resulted in technical improvements. Rudimentary approaches such as word count methods have been extended to weighted schemes, where weights are determined in statistical ways. For instance, in Das and Chen [2007], the discriminant score of each word across classification categories is used as a weighting index for the importance of words.

There is a proliferation of word-weighting schemes in the finance domain. One intuitive idea is that articles that contain words that are uncommon across all documents are more likely to be similar to other articles that contain those words, whereas articles that have common joint words are less likely to be similar. This introduces the idea of "inverse document frequency" (idf) as a weighting coefficient. Hence, the idf for word j would be

$$w_j^{idf} = \ln\left(\frac{N}{df_j}\right)$$

where N is the total number of documents, and df_j is the number of documents containing word j. This scheme was proposed by Manning and Schütze [1999].

Loughran and McDonald [2011] use this weighting approach to modify the word (term) frequency counts in the documents they analyze. The weight on word j in document i is specified as

$$w_{ij} = \max[0, 1 + \ln(f_{ij}) w_j^{idf}]$$

where f_{ij} is the frequency count of word j in document i. This leads naturally to a document score of

$$S_i^{LM} = \frac{1}{1+\ln(a_i)} \sum_{j=1}^{J} w_{ij}$$

Here a_i is the total number of words in document i, and J is the total number of words in the lexicon. (The LM superscript signifies the Loughran and McDonald [2011] weighting approach.)

Whereas the *idf* approach is intuitive, it does not have to be relevant for market activity. An alternate and effective weighting scheme has been developed in Jegadeesh and Wu [2013] (JW) using market movements. Words that occur more often on large market move days are given a greater weight than other words. JW show that this scheme is superior to an unweighted one, and delivers an accurate system for determining the "tone" of a regulatory filing. They also conduct robustness checks that suggest that the approach is quite general, and applies to other domains with no additional modifications to the specification. Indeed, they find that tone extraction from 10-Ks may be used to predict IPO underpricing.

JW create a "global lexicon" merging multiple word lists from Harvard-IV-4 Psychological Dictionaries(Harvard Inquirer), the Lasswell Value Dictionary, the Loughran and McDonald [2011] lists, and the word list in Bradley and Lang [1999]. They test this lexicon for robustness by checking (a) that the lexicon delivers accurate tone scores and (b) that it is complete by discarding 50% of the words and seeing whether it causes a material change in results (it does not). This approach provides a more reliable measure of document tone than preceding approaches. Their measure of filing tone is statistically related to filing period returns after providing for reasonable control variables. Tone is significantly related to returns for up to two weeks after filing, and it appears that the market under reacts to tone, and this is corrected within this two week window.

The tone score of document i in the JW paper is specified as

$$S_i^{JW} = \frac{1}{a_i} \sum_{j=1}^{J} w_j f_{ij}$$

where w_j is the weight for word j based on its relationship to market movement. (The JW superscript signifies the Jegadeesh and Wu [2013] weighting approach.) The following regression is used to determine the value of w_j (across all documents).

$$\begin{aligned} r_i &= a + b \cdot S_j^{JW} + \epsilon_i \\ &= a + b \left(\frac{1}{a_i} \sum_{j=1}^{J} w_j f_{ij} \right) + \epsilon_i \\ &= a + \left(\frac{1}{a_i} \sum_{j=1}^{J} (b w_j) f_{ij} \right) + \epsilon_i \\ &= a + \left(\frac{1}{a_i} \sum_{j=1}^{J} B_j f_{ij} \right) + \epsilon_i \end{aligned}$$

where r_i is the abnormal return around the release of document i, and $B_j = bw_j$ is a modified word weight. This is then translated back into the original estimated word weight by normalization, i.e.,

$$w_j = \frac{B_j - \frac{1}{J} \sum_{j=1}^{J} B_j}{\sigma(B_j)}$$

where $\sigma(B_j)$ is the standard deviation of B_j across all J words in the lexicon.

Abnormal return r_i is defined as the three-day excess return over the CRSP value-weighted return.

$$r_i = \prod_{t=0}^{3} ret_{it} - \prod_{t=1}^{3} ret_{VW,t}$$

Instead of r_i as the left-hand side variable in the regression, one might also use a binary variable for good and bad news, positive or negative 10-Ks, etc., and instead of the regression we would use a limited dependent variable structure such as logit, probit, or even a Bayes classifier. However, the advantages of r_i being a continuous variable are considerable for it offers a range of outcomes, and simpler regression fit.

Jegadeesh and Wu [2013] use data from 10-K filings over the period 1995–2010 extracted from SEC's EDGAR database. They ignore

positive and negative words when a negator occurs within a distance of three words, the negators being the words "not, no, never". Word weight scores are computed for the entire sample, and also for three roughly equal concatenated subperiods. The correlation of word weights across these subperiods is high, around 0.50 on average. Hence, the word weights appear to be quite stable over time and different economic regimes. As would be expected, when two subperiods are used the correlation of word weights is higher, suggesting that longer samples deliver better weighting scores. Interestingly, the correlation of JW scores with LM *idf* scores is low, and therefore, they are not substitutes.

JW examine the market variables that determine document score S_i^{JW} for each 10-K with right-hand side variables as the size of the firm, book-to-market, volatility, turnover, three day excess return over CRSP VW around earnings announcements, and accruals. Both positive and negative tone are significantly related to size and BM, suggesting that risk factors are captured in score. Volatility is also significant and has the correct sign, i.e., that increases in volatility make negative tone more negative and positive tone less positive. The same holds for turnover, in that more turnover makes tone pessimistic. The greater the earnings announcement abnormal return, the higher the tone, though this is not significant. Accruals do not significantly relate to score.

When regressing filing period return on document score and other controls (same as in the previous paragraph), the score is always statistically significant. Hence text in the 10-Ks does correlate with the market's view of the firm after incorporating the information in the 10-K and from other sources. Finally, JW find a negative relation between tone and IPO underpricing, suggesting that term weights from one domain can be reliably used in a different domain.

When using company filings, it is often an important issue as to whether to use the entire text of the filing or not. Sharper conclusions may be possible from specific sections of the filing such as a 10-K. Loughran and McDonald [2011] examined whether the Management Discussion and Analysis (MD&A) section of the filing was better at providing tone (sentiment) then the entire 10-K. They found not. They also showed that using their six tailor-made word lists gave better re-

sults for detecting tone than did the Harvard Inquirer words. And as discussed earlier, proper word-weighting also improves tone detection. Their word lists also worked well in detecting tone for seasoned equity offerings and news articles, providing good correlation with returns.

In an interesting paper, Loughran and McDonald [2014] examine the readability of financial documents, by surveying at the text in 10-K filings. They compute the Fog index for these documents and compare this to post filing measures of the information environment such as volatility of returns, dispersion of analysts recommendations. When the text is readable, then there should be less dispersion in the information environment, i.e., lower volatility and lower dispersion of analysts expectations around the release of the 10-K. Whereas they find that the Fog index does not seem to correlate well with these measures of the information environment, the file size of the 10-K is a much better measure and is significantly related to return volatility, earnings forecast errors, and earnings forecast dispersion, after accounting for control variates such as size, book-to-market, lagged volatility, lagged return, and industry effects. Li [2008] also shows that 10-Ks with high Fog index and longer length have lower subsequent earnings. Thus managers with poor performance may try to hide this by increasing the complexity of their documents, mostly by increasing the size of their filings.

The readability of business documents has caught the attention of many researchers, and not unexpectedly, in the accounting area. De Franco et al. [2013] combine the Fog, Flesh-Kincaid, and Flesch scores to show that higher readability of analyst's reports is related to higher trading volume, suggesting that a better information environment induces people to trade more and not shy away from the market. Lehavy et al. [2011] show that a greater Fog index on 10-Ks is correlated with greater analyst following, more analyst dispersion, and lower accuracy of their forecasts. Most of the literature focuses on 10-Ks because these are deemed the most information to investors, but it would be interesting to see if readability is any different when looking at shorter documents such as 10-Qs. Whether the simple, dominant (albeit lan-

guage independent) measure of file size remains a strong indicator of readability remains to be seen in documents other than 10-Ks.

Another examination of 10-K text appears in Bodnaruk et al. [2013]. Here, the authors measure the percentage of negative words in 10-Ks to see if this is an indicator of financial constraints that improves on existing measures. There is low correlation of this measure with size, where bigger firms are widely posited to be less financially constrained. But, an increase in the percentage of negative words suggests an inflection point indicating the tendency of a firm to lapse into a state of financial constraint. Using control variables such as market capitalization, prior returns, and a negative earnings indicator, percentage negative words helps more in identifying which firm will be financially constrained than widely used constraint indexes. The negative word count is useful in that it is independent of the way in which the filing is written, and picks up cues from managers who tend to use more negative words. This further validates the negative word list in Loughran and McDonald [2011]. The number of negative words is useful in predicting liquidity events such as dividend cuts or omissions, downgrades, and asset growth. A one standard deviation increase in negative words increases the likelihood of a dividend omission by 8.9% and a debt downgrade by 10.8%. An obvious extension of this work would be to see whether default probability models may be enhanced by using the percentage of negative words as an explanatory variable.

6.4 Text Mining Public Data and Network Modeling

The revolution in text analytics is intertwined with the growth in Big Data analytics. The quantum of textual data being generated in the public domain is so vast that harvesting and analyzing this data needs modern computer science techniques in information extraction, and information integration.

In the previous section, we surveyed the literature that examined corporate filings, mostly 10-Ks. The universe of filings is much larger than one merely restricted to 10-Ks. Mining all the filings and collating this information with other public data has immense potential. This

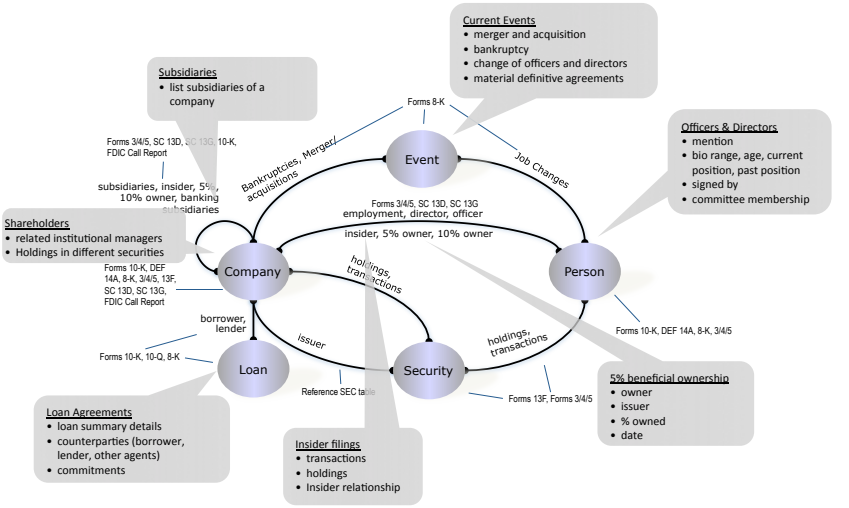

Figure 6.1: The data map in the IBM `Midas` system for the entities being populated with data from corporate filings.

potential is explored in Burdick et al. [2011], where a team from IBM Labs developed a technology for using data from public filings to create useful applications like bank personnel tracking and forensics, systemic risk from loan networks, and bank and subsidiary corporate structures. A system called `Midas` is developed to undertake all these tasks.

The Burdick et al. [2011] paper downloads large documents from SEC's servers, and then extracts relevant information from volumes of unstructured text therein. To do this efficiently a data entity model is used, where various entities are defined, such as Event, Company, Loan, Security, and Person. See Figure 6.1.

Large-scale text mining of this sort is best carried out in a flow manner once data entities are finalized. For example, if one is reading a 10-K, then the simple approach is as one reads to take any information that is relevant for a data entity and drop it into the data structure for that entity. A single filing may impact one or more data entities, and information will flow accordingly. This is similar to the way in which data broker firms collect your personal data and assign it to various

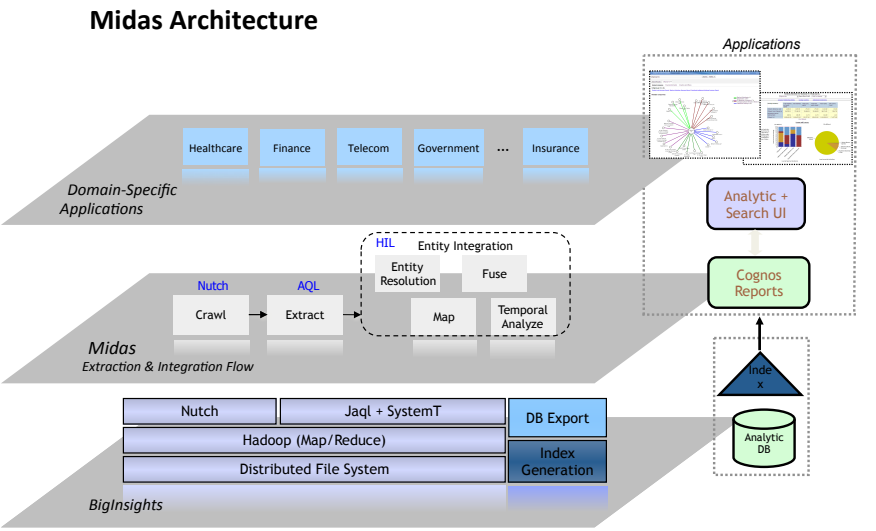

Figure 6.2: The `Midas` system architecture. See also Figure 8 in the paper by Burdick et al. [2011] for a reduced version of the same.

tags. Rules are usually written for this purpose, and a large system ends up with hundreds of rules. Entity resolution is also needed to ensure that mix-ups do not occur, such as the word "Morgan" which may refer to Bob Morgan, a Person-entity, or "J.P. Morgan", the Company-entity. Proper parsing resolves these conflicts and assures that data is tagged correctly to the correct entity. The architecture of this unstructured large-scale text-mining system is depicted in Figure 6.2.

Once the data has been mined, data sets are constructed. One of the applications in `Midas` is assessing systemic risks from an interbank co-lending network. Interbank loans are usually syndicated and are for large amounts. Text mining is used to extract loan details (lenders, borrower, amounts, etc.) and then these loan flows are represented in a network, whose mathematical form is an adjacency matrix A_{ij}, where each element in the matrix represents the flow from bank i to bank j (if the network is directed), or simply the shared flow (if the network is undirected). When the network is undirected, $A_{ij} = A_{ji}$, across all n banks, i.e., $i, j = 1...n$. Diagonal elements of the matrix are zero. For

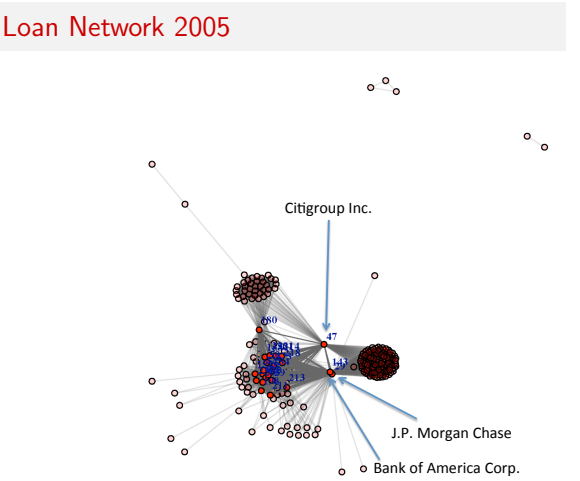

Figure 6.3: The 2005 loan network in the IBM `Midas` system for the entities being populated with data from corporate filings. All loans made in 2005 are included in this plot. From Figure 1 in the paper by Burdick et al. [2011].

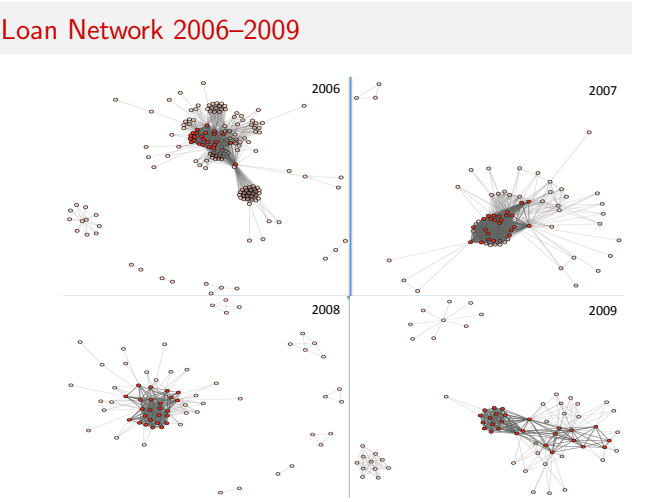

Figure 6.4: The 2006 to 2009 loan networks in the IBM `Midas` system for the entities being populated with data from corporate filings. From Figure 2 in the paper by Burdick et al. [2011].

(Year = 2005) Financial Institution	Normalized Centrality
J P Morgan Chase & Co.	1.000
Bank of America Corp.	0.926
Citigroup Inc.	0.639
Deutsche Bank Ag New York Branch	0.636
Wachovia Bank NA	0.617
The Bank of New York	0.573
Hsbc Bank USA	0.530
Barclays Bank Plc	0.530
Keycorp	0.524
The Royal Bank of Scotland Plc	0.523
Abn Amro Bank N.V.	0.448
Merrill Lynch Bank USA	0.374
PNC Financial Services Group Inc	0.372
Morgan Stanley	0.362
Bnp Paribas	0.337
Royal Bank of Canada	0.289
The Bank of Nova Scotia	0.289
U.S. Bank NA	0.284
Calyon New York Branch	0.273
Lehman Brothers Bank Fsb	0.270
Sumitomo Mitsui Banking	0.236
Suntrust Banks Inc	0.232
UBS Loan Finance Llc	0.221
State Street Corp	0.210
Wells Fargo Bank NA	0.198

Figure 6.5: Centrality computed from the 2005 loan network in the IBM `Midas` system for the entities being populated with data from corporate filings. From Table 1 in the paper by Burdick et al. [2011].

the year 2005, the extracted loan data yielded the network shown in Figure 1 of Burdick et al. [2011], reproduced here as Figure 6.3. Figure 6.4 shows the loan networks for four years, 2006–2009.

Burdick et al. [2011] computed the "eigenvector centrality" of each bank in the network. Centrality tells you which node in the network is the most influential, or in the case of this application, which bank contributes the most to the systemic risk of the lending system. The systemic risk contribution of each bank is stacked up in a n-vector denoted R made up of unknown values satisfying, for each bank i, the following equation:

$$R_i = \sum_{j=1} A_{ij} R_j, \qquad \text{for all } i.$$

The systemic risk of each bank is a function of the connections to other banks and vice-versa, in a circular manner, resulting in a series of interlocking equations. Stacking up equations for each i, we get a matrix system, $\mathbf{R}^\top \mathbf{AR}$. An eigenvalue decomposition of this system yields the

principal eigenvector, which is denoted as centrality (originally coined as such by Bonacich [1972]; Bonacich [1987]). The computed centrality is shown in Figure 6.5. We see that the top three systemically risky banks are J.P. Morgan Chase, Bank of America, and Citigroup. The centrality in the table is normalized so that the top value is 1 for J.P. Morgan Chase. Bank of America is 92.6% as risky as J.P. Morgan Chase, and Citigroup is relatively less risky, at 63.9% of the systemic risk of J.P. Morgan Chase.

Burdick et al. [2011] also provide analyses of filings that relate to corporate insiders. They analyze insider holdings, insider transactions, and employment history. These forensics produce timelines of insider activity that may be used forensically. See Figure 6.6 for some examples.

6.5 News Analytics

Processing news stories for trading signals has become an exhaustive business, in addition to the processing of web forums, microblogs, and corporate filings. Das [2011] provides an introduction to some of the approaches to text handling for news and other text. In a delightful and instructive book, Leinweber [2009] includes a large discussion on news analytics based trading rules. Wikipedia offers a definition of news analytics that embodies Leinweber's work, as follows:

> "News analysis refers to the measurement of the various qualitative and quantitative attributes of textual (unstructured data) news stories. Some of these attributes are: sentiment, relevance, and novelty. Expressing news stories as numbers and metadata permits the manipulation of everyday information in a mathematical and statistical way."

Here, we examine the literature on news analytics. In early work using the "Abreast of the Market" column in the Wall Street Journal, Tetlock [2007] finds that media pessimism exerts abnormal downward pressure on prices, which is subsequently corrected. He uses word counts across 77 categories in the Harvard Inquirer, and then extracts the principal

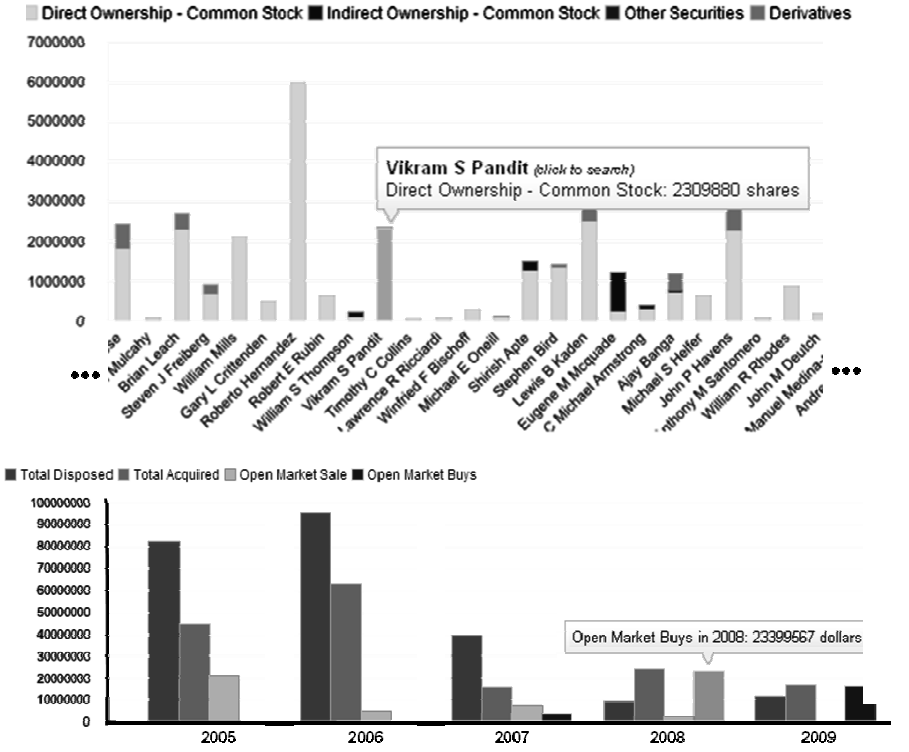

Figure 6.6: Insider time lines from the IBM `Midas` system for the entities being populated with data from corporate filings. The top chart shows insider holdings, and the bottom one shows insider transactions. From Figures 5 & 6 in the paper by Burdick et al. [2011]. The data is for insiders in Citigroup.

component across all categories. This principal component is denoted as a pessimism factor because it is found to be highly correlated with negative words. The pessimism factor is negatively related to one-day lagged Dow returns, and positively to five-day lagged return, confirming an overreaction to pessimistic media reports.

Trading volume is also related to media pessimism. It seems to play a direct role in forecasting volume, based on the idea that pessimism is a proxy for trading costs. The first lags of pessimistic words significant inverse predictors of volume (though this is a weak result, and vanishes with 1% outlier winsorization).

Leinweber and Sisk [2011] (hereafter denoted as LS) provide an interesting overview. They point out that the volume, breadth, depth, and frequency of news have all increased sharply in recent times. Whereas prior research in domains other than just news has shown clear linkages between the quantity of text and trading volume and volatility, it has been harder to show that the qualitative content of news contains information that is not in prices, thereby offering investors alpha, and delivering violations of the efficient market hypothesis.

Tetlock et al. [2008] is one of the first papers to use news content to generate trading signals, and assess whether text may be correlated to stock returns and accounting earnings. The three main results in the paper are that (i) the percentage of negative words forecasts reduced earnings; (ii) stock prices briefly under react to negative words; and (iii) stories that focus on fundamentals are more informative in that negative words in these stories elicit greater reaction. Earlier work by Li [2006] uncovered the interesting relationship between negative words in annual reports and earnings, by showing that greater occurrence of two words, "risk" and "uncertain", are indicative of risk and consequently lower earnings. Tetlock et al. [2008] pioneered the use of event study methodology in assessing news stories by examining the cumulative standardized unexpected earnings (CSUE) around the event date, i.e., news story. Figures 2 and 3 in their paper are reproduced here (see Figure 6.7). The figure clearly shows that sentiment scoring of news stories into positive and negative ones correlates with earnings and return outcomes. Information leakage starts to occur around ten

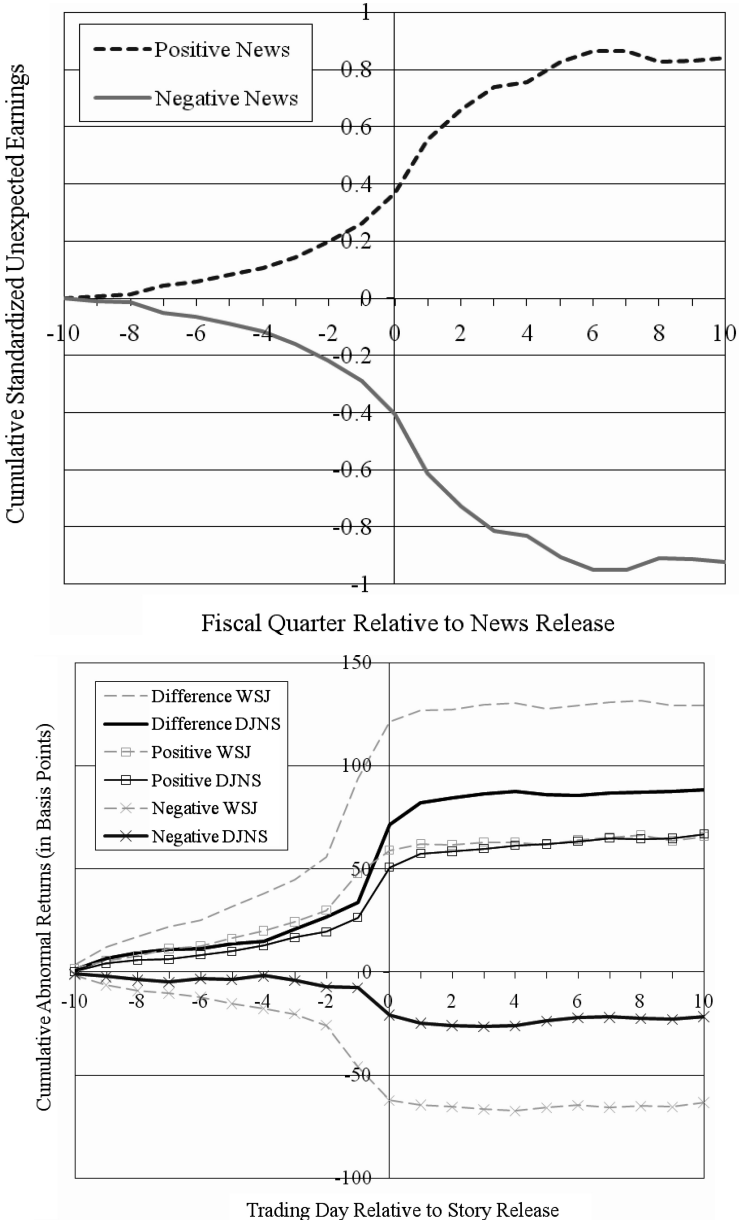

Figure 6.7: Reproduction from the *Journal of Finance* of Figures 2 and 3 from Tetlock et al. [2008]. The top figure shows the cumulative standardized unexpected earnings (CSUE) around news story date, separately for positive and negative news stories. The bottom plot shows the cumulative abnormal return (CAR) related to two news sources, the Wall Street Journal, and Dow Jones News.

days prior to the release of the news story. About half of the earnings and return surprise comes before news date, and the remaining in the ten days thereafter. Leinweber and Sisk [2011] warn that a substantial portion of the news effect comes from me-too stories, and does not represent additional trading signals.

A similar analysis is presented in Leinweber and Sisk [2011]. They use data from the Thomson Reuters NewsScope Sentiment Engine (RNSE) developed with Infonics/Lexalytics. This product is known as Thomson Reuters News Analytics (TRNA). It covers over 7,000 U.S. stocks from 2003–2008 for contemporaneous S&P 1500 stocks. This uses real time data and is on a much larger scale than the Tetlock et al. [2008] study. LS compute various linguistic scores for the news stories they analyze, such as sentiment, relevance, and novelty. Sentiment is defined as bullish, neutral, or bearish: $\{-1, 0, +1\}$, respectively. They keep track of the percentage of positive, negative, and neutral words. They also compute "relevance" - defined as applicability to a particular stock, and serves to weight articles. And "novelty" captures the inverse notion to the number of links to previous stories, thereby accounting for the me-too effect. They measure the volume of news (i.e., number of stories); the breadth of news, i.e.. number of tickers in the coverage; and depth, i.e., average number of news stories per ticker. The extracted aggregate sentiment appears to track market conditions very well, see a reproduction of Exhibit 7 in Leinweber and Sisk [2011], shown here in Figure 6.8. They also are able to replicate the event study results of Tetlock et al. [2008] by building a special tool, named Event Study Explorer, that generates a dashboard for replicating any event study seamlessly. They also find persistent evidence of news leakage.

One of the important findings in Leinweber and Sisk [2011] is that textual information is predictive, and this is demonstrated in their replicated event studies. Since, the plots show persistent positive abnormal returns for stocks with positive news stories, and negative abnormal returns for those with negative news, a trading strategy that goes long stocks with positive stories and short those with negative stories, is plausibly likely to produce alpha. Using a carefully constructed hold-out sample, nine months post model fitting, LS achieve an alpha of

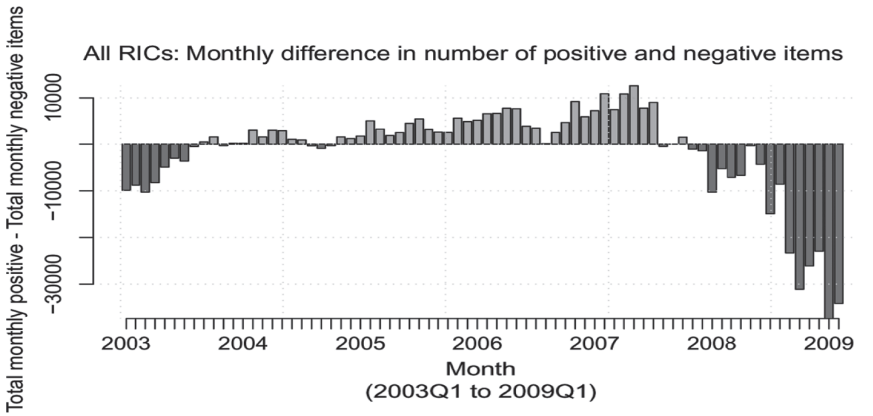

Figure 6.8: Reproduction from the *Journal of Portfolio Management* of Exhibit 7 from Leinweber and Sisk [2011], showing the net sentiment in aggregate from 2003–2009.

11.5%, but not without apparently risk returns. LS fail to provide risk adjusted measures such as the Sharpe ration, or just risk measures, especially higher-order moments of returns, which would enable a clearer assessment of whether these high returns are truly bankable in a reasonably safe manner. This corroborates the positive alpha produced in Tetlock et al. [2008] from a four-factor Fama-French model. However, these returns are possibly hard to capture, since they are wiped out if trading costs exceed 9 bps.

LS also ask an interesting follow-up question: For what kinds of stocks is mining news stories more likely to be profitable? They hypothesize and confirm that this is the case for smaller stocks, where the CAR plots reveal quite clearly higher abnormal returns for lower capitalization firms.

6.6 Commercial Vendors

Textual analysis of financial text has come of age. Begun as an academic interest, maturation is now signaled by the commercialization of text

sentiment by many firms. Here we briefly examine some of these services and the role they play.

Ravenpack is one of the leading names in the financial text mining business. Their web site positions them as[3] - "... a pioneer in financial news and sentiment analytics. We provide structured data derived in real-time from the unstructured text produced by reputable traditional and social media." They provide services in quantitative and algorithmic trading by generating timely signals and build custom models across all asset classes and trading frequencies. They also offer risk management services and market surveillance. They also construct company sentiment indicators (CSI) and macro sentiment indicators (MSI). These have proven to be quite successful for trading strategies. The indicators are used to enhance short-term (one week) strategies over all S&P500 stocks to exploit the short-term reversal anomaly with two news-based indicators, company event volume and sentiment. They find stronger reversal effects for past losers when unsupported by news. Strong reversal effects also exist for past winners when supported by news. Intuitively, reversal effects are stronger for past losers when company sentiment is positive, and for past winners when company sentiment is negative.[4]

Ravenpack focuses on news analytics. They find that news signals have the greatest impact on European markets. followed by US and Emerging markets.[5] The signals extracted from real-time content published by Dow Jones Newswires, The Wall Street Journal, and Barron's are used to construct long-only and long-short portfolios based on sector level 1-month expected returns. These portfolios consistently outperform the S&P 500 on both an absolute and risk-adjusted basis.[6] Their research is available on their web site. Mostly, they are a leading

[3] `www.ravenpack.com`.

[4] "Enhancing Short Term Reversal Strategies with News Analytics," 2014, Ravenpack White Paper.

[5] "Exploring Global Variations in News Impact on Equities," 2014, Ravenpack White Paper.

[6] "Tactical Equity Portfolio Formation Using News Analytics," 2013, Ravenpack White Paper.

RavenPack Regional Macro Sentiment Indicators, week ending Aug 24th

Region	1 month	1 month change	3 months	3 months change
Global	48,2	1,3%	48,4	0,1%
US	50,4	1,5%	49,1	0,0%
Eurozone	52,9	-1,0%	52,8	0,0%
UK	55,3	-1,4%	53,8	-0,7%
Japan	47,4	5,4%	46,2	-0,8%
Australia	51,0	-1,8%	52,7	0,5%
Canada	52,2	4,1%	49,7	0,4%
Switzerland	41,3	4,4%	50,1	-0,5%

Source: RavenPack, August 2014

Figure 6.9: Illustrative output of Ravenpack macro indicators. The one-month measure is an indicator of shorter-term macroeconomic sentiment and the three month measure a longer-term indicator. A score of 50 is neutral and the maximum range is 0 (most negative) to 100 (most positive).

player in the macro signals arena, see Figure 6.9 for an example of the signals.

In contrast to the commercial production of sentiment from news media, firms like iSentium extract sentiment from social media, in particular Twitter. iSentium touts itself as converting social media into alpha. They produce long and short horizon sentiment streams in real time, and are also accessible through Bloomberg terminals. They offer a product called iSense that is claimed to predict short term price movements. See Figure 6.10 for a depiction.

Finance related social media has also been commercialized in the retail market. StockTwits is a firm that provides sentiment indicators to the public at large and also has a mobile enabled web application. StockTwits mines Twitter for ticker based sentiment, which is calculated on a rolling seven-day basis. Standard output is of stock price, message volume and sentiment, see Figure 6.11.

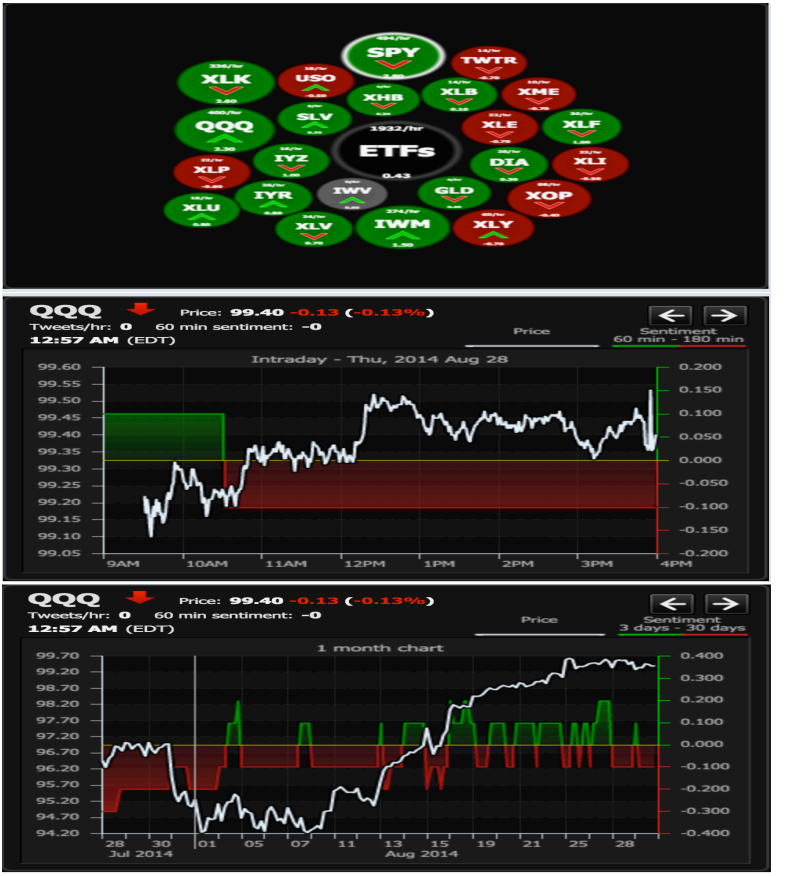

Figure 6.10: Illustrative output of iSentium's iSense product. This reaction indicator represents the live sentiment of a portfolio of financial assets. In the top plot each bubble displays information about the short- and long-term sentiment signal for the asset. Clicking on the bubble returns both the short term (middle plot) and long term (bottom plot) sentiment signals. The middle and bottom plots show asset the price in white overlaid onto sentiment signals at various time scales. Positive green and negative red regions indicate that recent sentiment is more bullish (respectively, bearish) than average. Sentiment is a leading indicator of price at both the intraday scale and longer term.

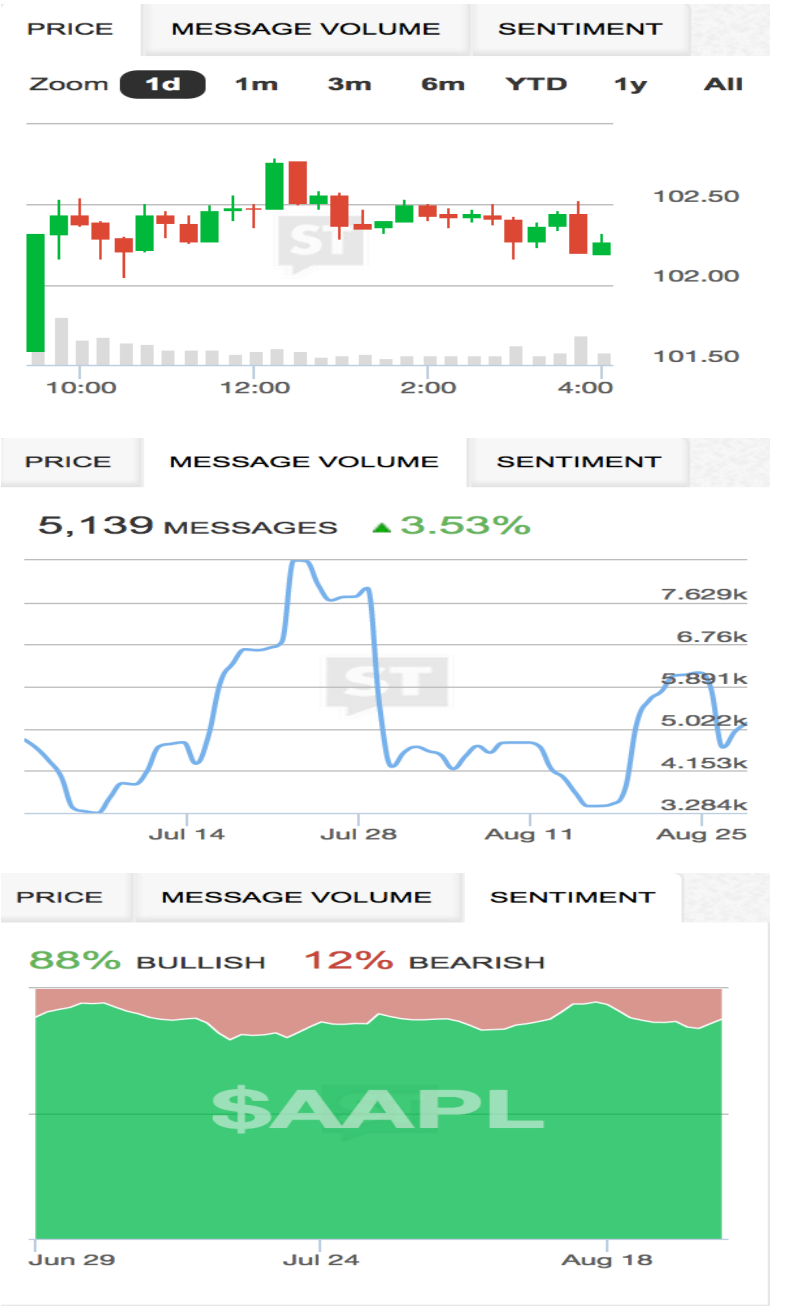

Figure 6.11: Illustrative output of StockTwits.

There are several other firms in the intersection space of finance and social media, such as SNTMNT[7], The Stock Sonar[8], SentDex[9], StockBuss[10] from BBVA (Banco Bilbao Vizcaya Argentaria), Downside Hedge[11], HedgeChatter[12]. These sentiment modeling firms complement the vast industry that offers sentiment based trading, or insights from social media. See for example firms such as SentimenTrader[13], Topsy[14], SocialMention[15], ViralHeat[16], SocialReport[17], Twitalyzer[18], WildFire[19].

[7] www.sntmnt.com.
[8] http://www.thestocksonar.com/.
[9] http://sentdex.com/.
[10] http://www.stockbuzz.es/.
[11] http://www.downsidehedge.com/twitter-stock-market-sentiment-download/.
[12] http://www.hedgechatter.com/.
[13] www.sentimentrader.com.
[14] http://topsy.com/.
[15] http://socialmention.com/.
[16] https://www.viralheat.com/.
[17] http://www.socialreport.com/.
[18] http://twitalyzer.com/.
[19] http://www.wildfireapp.com/.

7

Text Analytics – The Future

> Leonard: You'll never guess what just happened.
> Sheldon: You went out in the hallway, stumbled into an inter-dimensional portal, which brought you 5,000 years into the future, where you took advantage of the advanced technology to build a time machine, and now you're back, to bring us all with you to the year 7010, where we are transported to work at the think-a-torium by telepathically controlled flying dolphins?
> Leonard: No. Penny kissed me.
> Sheldon: Who would ever guess that?
>
> *"The Boyfriend Complexity"*
> *The Big Bang Theory, Season 4, Episode 9*

This monograph assessed the current ideas, technologies, paradigms, and empirical findings of the extant literature on the analytics of financial text. Suffice to say this is but the tip of the looming iceberg of progress in this area. The growing commercialization of fi-

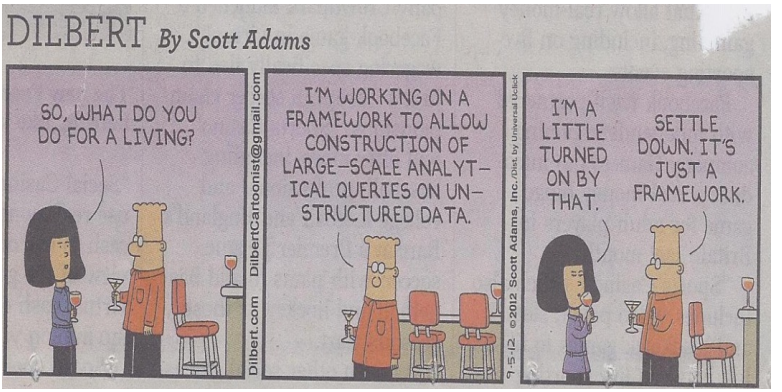

Figure 7.1: Dilbert on Text Mining.

nancial text analysis suggests that there is unexploited value that will drive future developments, in a modern technology-driven gold rush.

Text mining encompasses several technical processes such as information retrieval, information extraction, clustering, categorization, visualization, database technologies, machine learning, and data mining. These techniques overlap and complement each other. Broadly, we will see better ways of extracting and storing text, followed by knowledge distillation therefrom.

Big Data is inextricably wound up with Text Analytics, simply because there is so much text. There is huge appeal in mining all this extensive structured and unstructured text, see Figure 7.1. One may envisage the co-mingled field known as "Big Text", offering a utopian solution to our growing oppression from information overload. This suggests that new summarization technologies (briefly described in Section 5.8) for finance will make a huge difference in the future. As a consequence, document readability (Loughran and McDonald [2014]) will matter a lot and may even be legislated as a requirement.

Text serves to enhance Data, adding reasoning and stories to bland numbers. Future financial text algorithms will parse the news and explain to us why stocks moved in the way they did or offer automated analyst reports. Whereas there is some evidence that text offers statistically significant predictions of price, volatility, and volume, one may

imagine that as text mining algorithms improve, we will reach an elevated form of information efficiency in markets, the algorithms robbing themselves of their ability to remain predictive. Text analytics will always thus remain vulnerable to be hoist on its own petard.

One may conjecture that the future of text mining in finance will be statistical and minimally language dependent, reflecting the well accepted abdication of natural language processing (NLP) in favor of statistical language processing (SLP), subsumed under the rubric of machine learning. In finance, this is more crucial given the need for good data before decisions may be taken, unlike the case with consumer marketing text applications where the signal from noise is not as much an issue. Therefore, much needs to be done to resolve several important issues in text mining for finance such as data cleaning, data security and integrity, handling multilingual and correlated textual data, entity resolution, managing jargon-laden text, domain specificity, and the fact that bigger data means more noise, making sentiment extraction a search for a needle in a ever-growing haystack. On the flip side, this augurs well for an exciting time ahead for researchers in this field. Add to this the fact that numerical data and text will eventually be augmented with other mediums such as voice, video, temperature, and many other physical phenomena, the complexity of the ecosystem for sentiment extraction is extensive. The exciting thing is, it's happening now!

Financial firms are only now realizing that data science and text mining offers serious added value, something that marketing firms realized earlier on. An illustrative list of *applications* for finance firms is as follows:

1. Monitoring corporate buzz.
2. Analyzing textual data to detect, analyze, and understand the more profitable customers or products.
3. Targeting new clients.
4. Customer retention, which is a huge issue. Text mining complaints to prioritize customer remedial action makes a huge difference, especially in the insurance business.

5. Lending activity - automated management of profiling information for lending screening.
6. Market prediction and trading.
7. Risk management.
8. Automated financial analysts.
9. Financial forensics to prevent rogue employees from inflicting large losses.
10. Fraud detection.
11. Detecting market manipulation.
12. Social network analysis of clients.
13. Measuring institutional risk from systemic risk.

This list is by no means exhaustive. There are major *benefits* to automated textual analysis. First, handling of big data is getting easier and more critical as well. Big Text has the same three Vs of Big Data: Volume (amount of data), Velocity (speed/rate of the stream), Variability (forms and variety such as blogs, news, posts, filings, emails, mobile app information) are critical for the competitiveness of a large financial institution. (A fourth commonly asserted V is Volatility.)

Second, automated decisions based on Big Text are less judgmental. Even when human interaction is required, text mining creates a new culture of good judgment based on data. Third, the decision process from automated text mining is repeatable and replicable. Fourth, text analytics lead to new business opportunities through uncovering unexpected correlations in the data. Fifth, the scope now extends to system wide risk and return, and generates useful data aggregates. Sixth, Big Text data mining models lead to real time decision-making from massive data, infeasible for human decision makers.

Of course, Big Text comes with pitfalls. Garbage in, garbage out (GIGO), which now occurs with greater propensity. The infrastructure is expensive, and huge effort may be expended on collecting too much

data and not using it correctly; this is extremely wasteful. Big data leads to bigger errors if misused. For example, a common problem is confusing text correlation with financial variates as causality or predictability. Stricter statistical foundations need to be imposed on text analytics. And of course, there is a tendency for excessive misdirected automation leading to poor client service.

Final issues with text are efficiency, privacy and accessibility. Should financial firms be allowed to access our activity on Twitter and Facebook and determine the pricing of financial products using such information? Maybe the question is moot, as this is already happening, as data brokers actively farm personal data on open social media sites. How soon this will enter models for pricing medical insurance, auto insurance, mortgage rates is a good question, though one would speculate it is but a matter of short time. Maybe this will make markets more efficient at the cost of privacy? Where does one draw the line? As we provide more and more text for commercial mining, the question arises, should we be paid for voluntarily offering up our data, as argued by Lanier [2013][1]. Or is our personal data fair payment for the myriad free web services we receive? This debate will impact the commercialization of text analytics in finance.

Publishers also present another stumbling block to financial text mining. The Association for European Research Libraries (LIBER[2]) maintains that the right to read confers the right to text mine. This is contested by publishers, who assume that this is an infringement of terms of use, as they do not get paid for the results of use of the text. Moreover, summarization may kill their market for the original text. (And as a side issue, if everyone started reading the same distillations and summaries, differences and richness of opinion would be hugely impacted.) LIBER advocates argue that much public economic good comes from the applications of financial text mining, such as managing systemic risk, improving the employment market, predicting and clarifying political issues.

[1] See the New York Times article by Joe Nocera titled "Will Digital Networks Ruin Us?" - Jan 6, 2014.

[2] `http://libereurope.eu/`

We live in interesting times. Text analytics in finance is where talk and money intersect. So far the success of this new paradigm has been limited, as surveyed in preceding sections. Is this all hype? Or is there real potential here, as claimed by companies like Recorded Future[3] who claim that text analytics makes it possible to "know the future." Only time, money, and lots of talk will tell.

[3] `https://www.recordedfuture.com/`.

8

Appendix

Sample text from Bloomberg for summarization

Summarization is one of the major implementations in Big Text applications. When faced with Big Text, there are three important stages through which analytics may proceed: (a) Indexation, (b) Summarization, and (c) Inference. Automatic summarization[1] is a program that reduces text while keeping mostly the salient points, accounting for variables such as length, writing style, and syntax. There are two approaches: (i) Extractive methods selecting a subset of existing words, phrases, or sentences in the original text to form the summary. (ii) Abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original.

The following news article was used to demonstrate text summarization for the application in Section 3.4.

[1] `http://en.wikipedia.org/wiki/Automatic_summarization`.

Bloomberg

Wall Street Bond Dealers Whipsawed on Bearish Treasuries Bet

By Lisa Abramowicz and Daniel Kruger - Apr 21, 2014

Betting against U.S. government debt this year is turning out to be a fool's errand. Just ask Wall Street's biggest bond dealers.

While the losses that their economists predicted have yet to materialize, JPMorgan Chase & Co. (JPM), Citigroup Inc. (C) and the 20 other firms that trade with the Federal Reserve began wagering on a Treasuries selloff last month for the first time since 2011. The strategy was upended as Fed Chair Janet Yellen signaled she wasn't in a rush to lift interest rates, two weeks after suggesting the opposite at the bank's March 19 meeting.

The surprising resilience of Treasuries has investors re-calibrating forecasts for higher borrowing costs as lackluster job growth and emerging-market turmoil push yields toward 2014 lows. That's also made the business of trading bonds, once more predictable for dealers when the Fed was buying trillions of dollars of debt to spur the economy, less profitable as new rules limit the risks they can take with their own money.

"You have an uncertain Fed, an uncertain direction of the economy and you've got rates moving," Mark MacQueen, a partner at Sage Advisory Services Ltd., which oversees $10 billion, said by telephone from Austin, Texas. In the past, "calling the direction of the market and what you should be doing in it was a lot easier than it is today, particularly for the dealers."

Treasuries (USGG10YR) have confounded economists who predicted 10-year yields would approach 3.4 percent by year-end as a strengthening economy prompts the Fed to pare its unprecedented bond buying.

Caught Short

After surging to a 29-month high of 3.05 percent at the start of the year, yields on the 10-year note have declined and were at 2.72 percent at 7:42 a.m. in New York.

One reason yields have fallen is the U.S. labor market, which has yet to show consistent improvement.

The world's largest economy added fewer jobs on average in the first three months of the year than in the same period in the prior two years, data compiled by Bloomberg show. At the same time, a slowdown in China and tensions between Russia and Ukraine boosted demand for the safest assets.

Wall Street firms known as primary dealers are getting caught short betting against Treasuries.

They collectively amassed $5.2 billion of wagers in March that would profit if Treasuries fell, the first time they had net short positions on government debt since September 2011, data compiled by the Fed show.

'Some Time'

The practice is allowed under the Volcker Rule that limits the types of trades that banks can make with their own money. The wagers may include market-making, which is the business of using the firm's capital to buy and sell securities with customers while profiting on the spread and movement in prices.

While the bets initially paid off after Yellen said on March 19 that the Fed may lift its benchmark rate six months after it stops buying bonds, Treasuries have since rallied as her subsequent comments strengthened the view that policy makers will keep borrowing costs low to support growth.

On March 31, Yellen highlighted inconsistencies in job data and said "considerable slack" in labor markets showed the Fed's accommodative policies will be needed for "some time."

Then, in her first major speech on her policy framework as Fed chair on April 16, Yellen said it will take at least two years for the U.S. economy to meet the Fed's goals, which determine how quickly the central bank raises rates.

After declining as much as 0.6 percent following Yellen's March 19 comments, Treasuries have recouped all their losses, index data compiled by Bank of America Merrill Lynch show.

Yield Forecasts

"We had that big selloff and the dealers got short then, and then we turned around and the Fed says, 'Whoa, whoa, whoa: it's lower for longer again,'" MacQueen said in an April 15 telephone interview. "The dealers are really worried here. You get really punished if you take a lot of risk."

Economists and strategists around Wall Street are still anticipating that Treasuries will underperform as yields increase, data compiled by Bloomberg show.

While they've ratcheted down their forecasts this year, they predict 10-year yields will increase to 3.36 percent by the end of December. That's more than 0.6 percentage point higher than where yields are

today.

“My forecast is 4 percent,” said Joseph LaVorgna, chief U.S. economist at Deutsche Bank AG, a primary dealer. “It may seem like it’s really aggressive but it’s really not.”

LaVorgna, who has the highest estimate among the 66 responses in a Bloomberg survey, said stronger economic data will likely cause investors to sell Treasuries as they anticipate a rate increase from the Fed.

History Lesson

The U.S. economy will expand 2.7 percent this year from 1.9 percent in 2013, estimates compiled by Bloomberg show. Growth will accelerate 3 percent next year, which would be the fastest in a decade, based on those forecasts.

Dealers used to rely on Treasuries to act as a hedge against their holdings of other types of debt, such as corporate bonds and mortgages. That changed after the credit crisis caused the failure of Lehman Brothers Holdings Inc. in 2008.

They slashed corporate-debt inventories by 76 percent from the 2007 peak through last March as they sought to comply with higher capital requirements from the Basel Committee on Banking Supervision and stockpiled Treasuries instead.

“Being a dealer has changed over the years, and not least because you also have new balance-sheet constraints that you didn’t have before,” Ira Jersey, an interest-rate strategist at primary dealer Credit Suisse Group AG (CSGN), said in a telephone interview on April 14.

Almost Guaranteed

While the Fed’s decision to inundate the U.S. economy with more than $3 trillion of cheap money since 2008 by buying Treasuries and mortgaged-backed bonds bolstered profits as all fixed-income assets rallied, yields are now so low that banks are struggling to make money trading government bonds.

Yields on 10-year notes have remained below 3 percent since January, data compiled by Bloomberg show. In two decades before the credit crisis, average yields topped 6 percent.

Average daily trading has also dropped to $551.3 billion in March from an average $570.2 billion in 2007, even as the outstanding amount of Treasuries has more than doubled since the financial crisis, according data from the Securities Industry and Financial Markets Association.

"During the crisis, the Fed went to great pains to save primary dealers," Christopher Whalen, banker and author of "Inflated: How Money and Debt Built the American Dream," said in a telephone interview. "Now, because of quantitative easing and other dynamics in the market, it's not just treacherous, it's almost a guaranteed loss."

Trading Revenue

The biggest dealers are seeing their earnings suffer. In the first quarter, five of the six biggest Wall Street firms reported declines in fixed-income trading revenue.

JPMorgan, the biggest U.S. bond underwriter, had a 21 percent decrease from its fixed-income trading business, more than estimates from Moshe Orenbuch, an analyst at Credit Suisse, and Matt Burnell of Wells Fargo & Co.

Citigroup, whose bond-trading results marred the New York-based bank's two prior quarterly earnings, reported a 18 percent decrease in revenue from that business. Credit Suisse, the second-largest Swiss bank, had a 25 percent drop as income from rates and emerging-markets businesses fell. Declines in debt-trading last year prompted the Zurich-based firm to cut more than 100 fixed-income jobs in London and New York.

Bank Squeeze

Chief Financial Officer David Mathers said in a Feb. 6 call that Credit Suisse has "reduced the capital in this business materially and we're obviously increasing our electronic trading operations in this area." Jamie Dimon, chief executive officer at JPMorgan, also emphasized the decreased role of humans in the rates-trading business on an April 11 call as the New York-based bank seeks to cut costs.

About 49 percent of U.S. government-debt trading was executed electronically last year, from 31 percent in 2012, a Greenwich Associates survey of institutional money managers showed. That may ultimately lead banks to combine their rates businesses or scale back their roles as primary dealers as firms get squeezed, said Krishna Memani, the New York-based chief investment officer of OppenheimerFunds Inc., which oversees $79.1 billion in fixed-income assets.

"If capital requirements were not as onerous as they are now, maybe they could have found a way of making it work, but they aren't as such," he said in a telephone interview.

To contact the reporters on this story: Lisa Abramowicz in New York at labramowicz@bloomberg.net; Daniel Kruger in New York at dkruger1@bloomberg.net

To contact the editors responsible for this story: Dave Liedtka at dliedtka@bloomberg.net Michael

Acknowledgements

I am especially grateful to my colleagues, and to collaborators and co-authors on several text analytics projects for the learning and wisdom gained from many interactions: Douglas Burdick, Mike Chen, Mauricio A. Hernandez, Howard Ho, Georgia Koutrika, Rajasekar Krishnamurthy, Asis Martinez-Jerez, Lucian Popa, Priya Raghubir, Jose Silva, Jacob Sisk, Ioana Stanoi, Peter Tufano, Shivakumar Vaithyanathan. Thanks to Sophie Raza for bringing calm to this frenetic endeavor. I am also grateful for excellent comments from an anonymous referee.

References

A. Admati and P. Pfleiderer. Noisytalk.com: Broadcasting opinions in a noisy environment. Working paper, Stanford University, 2001.

W. Antweiler and M. Frank. Is all that talk just noise? the information content of internet stock message boards. *Journal of Finance*, v59(3):1259–1295, 2004.

W. Antweiler and M. Frank. The market impact of corporate news stories. Working paper, University of British Columbia, 2005.

Mark Bagnoli, M. Beneish, and S. Watts. Whisper forecasts of quarterly earnings per share. *Journal of Accounting and Economics*, 28(1):27–50, 1999.

R. Bar-Haim, E. Dinur, R. Feldman, Fresko M, and G. Goldstein. Identifying and following experts in stock microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1310–1319. Edinburgh, UK, 2011.

A. Bodnaruk, T.Loughran, and B. McDonald. Using 10-k text to gauge financial constraints. Working paper, University of Notre Dame, 2013.

J. Bollen, H. Mao, and X-J. Zeng. Twitter mood predicts the stock market. *arXiv:1010.3003v1*, 2010.

P. Bonacich. Technique for analyzing overlapping memberships. *Sociological Methodology*, 4:176–185, 1972.

P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

J. Boudoukh, R. Feldman, S. Kogan, and M. Richardson. Which news moves stock prices? a textual analysis. Working paper, University of Texas, Austin, 2012.

M. Bradley and P. Lang. Affective norms for english words (anew): Stimuli, instruction manual and affective ratings. Technical report C-1, *The Center for Research in Psychophysiology*, University of Florida, 1999.

E. D. Brown. Will twitter make you a better investor? a look at sentiment, user reputation and their effect on the stock market. In *Proceedings of the Southern Association for Information Systems Conference*. Atlanta, GA, USA, March 23rd–24th 2012.

D. Burdick, S. Das, M. A. Hernandez, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, and S. Vaithyanathan. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Engineering Bulletin*, 34(3):60–67, 2011.

M. Coleman and T. L. Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60:283–284, 1975.

D. Conway and J. M. White. *Machine Learning for Hackers*. O'Reilly Press, Sebastopol, CA, 2012.

S. Das and M. Chen. Yahoo for amazon! sentiment extraction from small talk on the web. *Management Science*, 53:1375–1388, 2007.

S. Das and J. Sisk. Financial communities. *Journal of Portfolio Management*, 31(4):112–123, 2005.

S. Das, A. Martinez-Jerez, and P. Tufano. einformation: A clinical study of investor discussion and sentiment. *Financial Management*, 34(5):103–137, 2005.

S. R. Das. News analytics: Framework, techniques and metrics. In *The Handbook of News Analytics in Finance*. John Wiley & Sons, U.K, 2011.

G. De Franco, O.-K. Hope, D. Vyas, and Y. Zhou. Analyst report readability. *Contemporary Accounting Research*, forthcoming, 2013.

P. DeMarzo, D. Vayanos, and J. Zwiebel. Persuasion bias, social influence, and uni-dimensional opinions. *Quarterly Journal of Economics*, 118:909–968, 2003.

R. Feldman, B. Rosenfeld, R. Bar-Haim, and M. Fresko. The stock sonar sentiment analysis of stocks based on a hybrid approach. *Proceedings of the Twenty-Third Innovative Applications of Artificial Intelligence Conference*, pages 1642–1647, 2011.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

D. Godes, D. Mayzlin, Y. Chen, S. Das, C. Dellarocas, B. Pfeieffer, B. Libai, S. Sen, M. Shi, and P. Verlegh. The firm's management of social interactions. *Marketing Letters*, v16:415–428, 2005.

R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, 1952.

Y. Hochberg, A. Ljungqvist, and Y. Lu. Whom you know matters: Venture capital networks and investment performance. *Journal of Finance*, 62(1): 251–301, 2007.

P. Jaccard. Etude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

N. Jegadeesh and D. Wu. Word power: A new approach for content analysis. *Journal of Financial Economics*, 110(3):712–729, 2013.

R. Jordan. *Academic Writing Course*. Longman, London, 1999.

J. Lanier. *Who Owns the Future?* Simon and Schuster, New York, 2013.

R. Lehavy, F. Li, and K. Merkley. The effect of annual report readability on analyst following and the properties of their earnings forecasts. *Accounting Review*, 86:1087–1115, 2011.

D. Leinweber. *Nerds on Wall Street*. John Wiley and Sons, New Jersey, 2009.

D. Leinweber and J. Sisk. *Relating News Analytics to Stock Returns*. mimeo, Leinweber & Co, 2010.

D. Leinweber and J. Sisk. Event-driven trading and the "new news". *Journal of Portfolio Management*, Summer, 1–15 2011.

F. Li. Do stock market investors understand the risksentiment of corporate annual reports? Working paper, University of Michigan, 2006.

F. Li. Annual report readability, current earnings, and earnings persistence. *Journal of Accounting and Economics*, 45:221–247, 2008.

A. Logunov. A tweet in time: Can twitter sentiment analysis improve economic indicator estimation and predict market returns? Undergraduate Honors Thesis, University of New South Wales, 2011.

T. Loughran and W. McDonald. When is a liability not a liability. *Journal of Finance*, 66:35–65, 2011.

T. Loughran and W. McDonald. Measuring readability in financial disclosures. *Journal of Finance*, 69:1643–1671, 2014.

H.-M. Lu, H. Chen, T.-J. Chen, M.-W. Hung, and S.-H. Li. Financial text mining: Supporting decision making using web 2.0 content. *IEEE Intelligent Systems*, pages 78–82, 2010.

C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

W. J. Mayew and M. Venkatachalam. Speech analysis in financial markets. *Foundations and Trends in Accounting*, 7(2):73–130, 2012.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

D. McNair, J. P. Heuchert, and E. Shilony. *Profile of Mood States. Bibliography 1964–2002*. Multi-Health Systems, 2003.

G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

L. Mitra, G. Mitra, and D. diBartolomeo. Equity portfolio risk (volatility) estimation using market information and sentiment. Working paper, Brunel University, 2008.

N. Pervin, F. Fang, A. Datta, and K. Dutta. Fast, scalable, and context-sensitive detection of trending topics in microblog post streams. *ACM Transactions on Management Information Systems*, 3(4):Article 19, 2013.

M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

T. Rao and S. Srivastava. Twitter sentiment analysis: How to hedge your bets in the stock markets. Working paper, Indian Institute of Technology, Delhi, 2012.

R. Roll. R-squared. *Journal of Finance*, 43:541–566, 1988.

T. Sprenger. Tweettrader.net: Leveraging crowd wisdom in a stock micro blogging forum. *Association for the Advancement of Artificial Intelligence*, 2011.

T. Sprenger and I. M. Welpe. Tweets and trades: The information content of stock microblogs. Working paper, Technische Universität München, 2010.

J. Surowiecki. *The Wisdom of Crowds*. Anchor Books, New York, 2004.

P. Tetlock. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168, 2007.

P. Tetlock, P. M. Saar-Tsechansky, and S. Macskassay. More than words: Quantifying language to measure firm's fundamentals. *Journal of Finance*, 63(3):1437–1467, 2008.

R. Tumarkin and R.Whitelaw. News or noise? internet postings and stock prices. *Financial Analysts Journal*, 57(3):41–51, 2001.

R. Van Noorden. Trouble at the text mine. *Nature*, 483:134–135, 2012.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

V. Vapnik and Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, v16(2):264–280, 1964.

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, v24, 1963.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. Opinionfinder: A system for subjectivity analysis. *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 34–35, 2005.

P. Wysocki. Cheap talk on the web: The determinants of postings on stock message boards. Working Paper, November, University of Michigan, 1999.

W. Zhang and S. Skiena. Trading strategies to exploit blog and news sentiment. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 375–378, 2010.