Text Mining
Sentiment Analysis

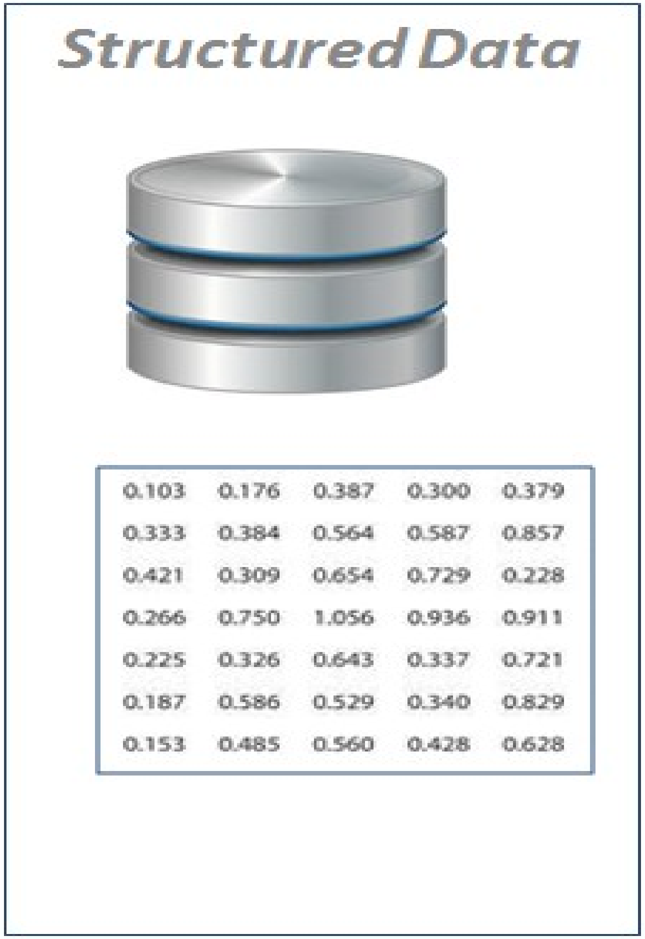
DATA SCIENCE
INSTITUTE

Text Mining

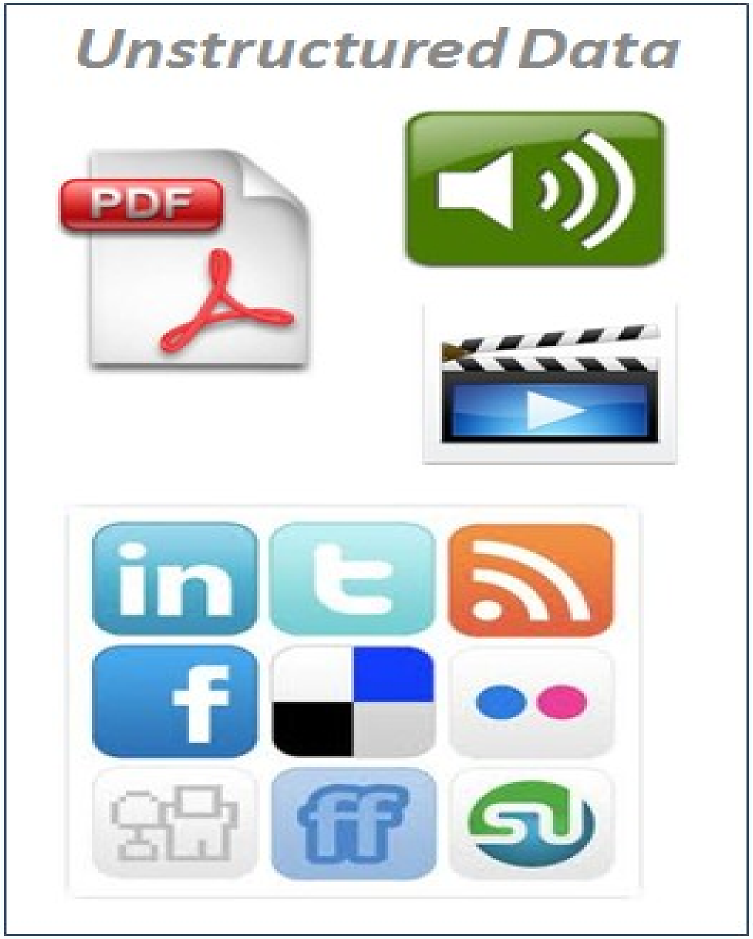
- Natural languages (English, French, Mandarin etc.) are different from programming languages. The semantic or the meaning of a statement depends on the context, tone and a lot of other factors. Unlike programming languages, natural languages are ambiguous.

- Text mining deals with helping computers understand the “meaning” of the text. Some of the common text mining applications include sentiment analysis e.g if a Tweet about a movie says something positive or not, text classification e.g classifying the mails you get as spam or ham etc.

Structured Vs. Unstructured Data



0.103	0.176	0.387	0.300	0.379
0.333	0.384	0.564	0.587	0.857
0.421	0.309	0.654	0.729	0.228
0.266	0.750	1.056	0.936	0.911
0.225	0.326	0.643	0.337	0.721
0.187	0.586	0.529	0.340	0.829
0.153	0.485	0.560	0.428	0.628



Examples of Unstructured Data

- Examples include
  - Personal messaging – email, instant messages, tweets, chat
  - Business documents – business reports, presentations, survey responses
  - Web content – web pages, blogs, wikis, audio files, photos, videos
  - Sensor output – satellite imagery, geolocation data, scanner transactions

What is Text Mining?

- Text Mining is also known as Text Data Mining (TDM) and Knowledge Discovery in Textual Database (KDT)

- A process of identifying novel information from a collection of texts (Also known as a ‘Corpus’)

- Corpus is a collection of ‘documents’ containing natural language text. Here, documents, generally, are sentences. Each document is represented as a separate line.

Text Mining – Example

- Playing It My Way is the autobiography of former Indian cricketer Sachin Tendulkar

- It was launched on 5 November 2014 in Mumbai

- The book summarises Tendulkar's early days, his 24 years of international career and aspects of his life that have not been shared publicly

- Soon after it’s launch, Twitter was flooded with emotions, comments and thoughts by the readers around the world

- The following data was retrieved from Twitter

Data Snapshot (.txt File)

```
no mention of match fixing. Silent on Controversial  issues
good to read.
not a great book.
Only sachin's fan can read
Match fixing not touched
He could have written more about his colleagues
I will prefer to watch his old matches.
Silent on match fixing.
Good to read about his early days.
Good book.
Not for avid readers.
Only for Sachin fans.
Had high hopes about revealing match fixing.
Excellent book
The book is worth reading
```

Text Mining in R

As a result of Text mining of the above document we will get to know:

1) Most frequently appearing word in the text in graphical (bar chart),pictorial (word cloud) and tabular format.

2) Sentiment score for each respondent (Negative, Positive, Neutral etc.)

Text Mining in R

```
# Import  text file with one text record in one row
data<-readLines(file.choose())
head(data)
```

class(data) is character

```
[1] "no mention of match fixing. Silent on Controversial  issues"
[2] "good to read."
[3] "not a great book."
[4] "Only sachin's fan can read"
[5] "Match fixing not touched"
[6] "He could have written more about his colleagues"
```

```
# Convert data into corpus
library(tm)
corp <- Corpus(VectorSource(data))
```

A ‘vector source’ interprets each element of the vector as a document.

```
class(corp)
[1] "SimpleCorpus" "Corpus"
```

Text Mining in R

```
# Clean the corpus for further analysis

# Convert to lower case and remove punctuation
corp <- tm_map(corp, tolower)
corp <- tm_map(corp, removePunctuation)

# Remove numbers
corp <- tm_map(corp, removeNumbers)

# Remove  stop words like: and, the, is, etc.
corp <- tm_map(corp,removeWords, stopwords("english"))

# Display  a particular document from corpus
writeLines(as.character(corp[[1]]))
# Inspect corpus
inspect(corp[1:3])
```

tm_map()
applies transformation functions to a corpus

If you wish to remove specific words from the corpus use tm_map(corp, removeWords, “word”)

Text Mining in R

```
tdm <- TermDocumentMatrix(corp)
```

Term Document Matrix gives number of times each word (term) appears in each document.
Try DocumentTermMatrix()

```
findFreqTerms(tdm)
# Find terms with frequency of at least 10
findFreqTerms(tdm,10)
```

```
[1] "fixing" "match"  "good"   "read"   "book“
```

```
# Find words having high association with 'fixing'
findAssocs(tdm, 'fixing', 0.30)
```

```
$fixing
 match    silent    issues
  0.92     0.54      0.33
```

Word Cloud

```
library(wordcloud)
# Convert to a matrix
m <- as.matrix(tdm)
m
```

```
            Docs
Terms              1  2  3  4  5  6  7  8  9  10 …….. 62
  controversial    1  0  0  0  0  0  0  0  0   0
  fixing    1  0  0  0  1  0  0  1  0   0
  issues    1  0  0  0  0  0  0  0  0   0
  match            1  0  0  0  1  0  0  1  0   0
  mention          1  0  0  0  0  0  0  0  0   0
  silent    1  0  0  0  0  0  0  1  0   0
  good             0  1  0  0  0  0  0  0  1   1
  read      0  1  0  1  0  0  0  0  1   0
  book      0  0  1  0  0  0  0  0  0   1
  ……
```

Word Cloud

```
# Calculate the total frequency of words
v <- sort(rowSums(m), decreasing=TRUE)
myNames <- names(v)
d <- data.frame(word=myNames, freq=v)
```

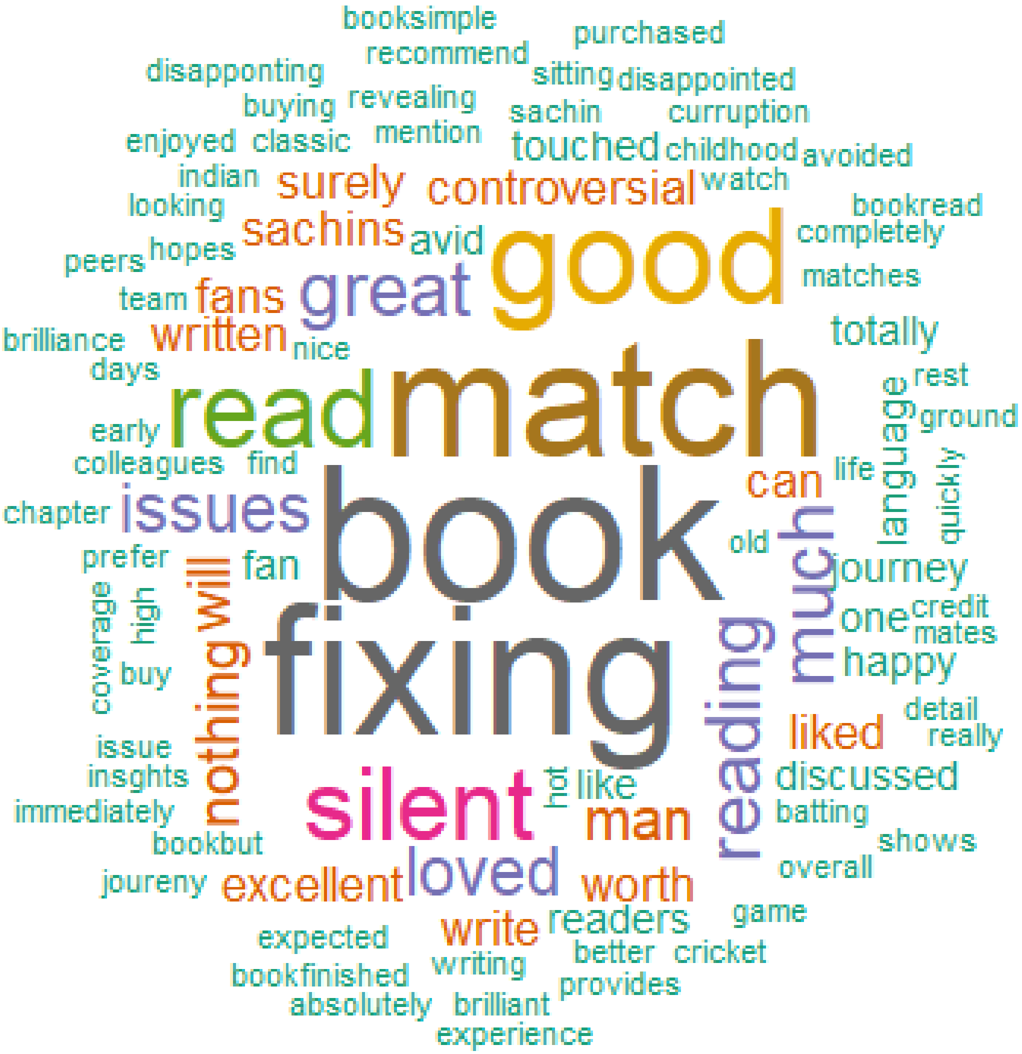
```
# Create colour palette
pal2 <- brewer.pal(8,"Dark2")
```

brewer.pal () was developed by Cynthia Brewer. It makes the color palettes from Color Brewer available as R palettes.
Arguments:
Number of colors included in the palette: 8
Palette Name: Dark 2

```
# Word cloud
wordcloud(d$word, d$freq,random.order = FALSE , min.freq=1, colors=pal2)
```

random.order=FALSE plots words in decreasing frequency.

Word Cloud


booksimple
purchased
recommend
disapponting
sitting
disappointed
buying
revealing
sachin
curruption
enjoyed
classic
mention
touched
childhood
avoided
indian
surely
controversial
watch
looking
bookread
sachins
avid
good
completely
peers
hopes
matches
team
fans
great
brilliance
written
nice
totally
days
rest
read
match
language
ground
early
quickly
colleagues
find
can
life
chapter
issues
book
old
much
prefer
will
fan
journey
coverage
high
one
credit
mates
buy
fixing
reading
happy
nothing
detail
really
liked
issue
insghts
hot
like
discussed
immediately
silent
man
batting
bookbut
shows
overall
joureny
excellent
loved
worth
game
expected
write
readers
bookfinished
writing
better
cricket
provides
absolutely
brilliant
experience

Using ggplot

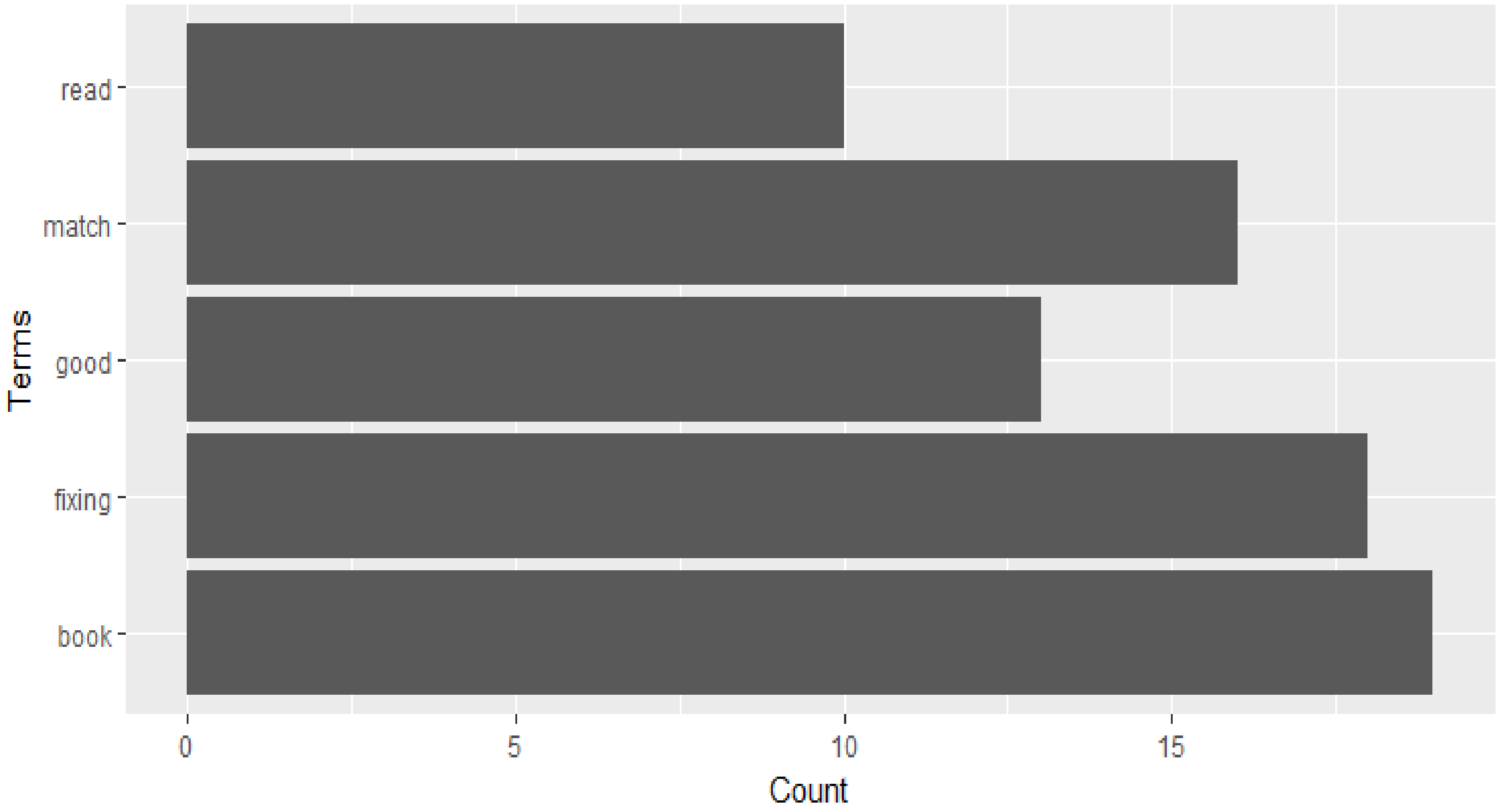
```
# Using ggplot
term.freq <- rowSums(m)
term.freq <- subset(term.freq, term.freq >= 10)
```

Object term.freq is of class numeric

```
# Transform as a dataframe
df <- data.frame(term = names(term.freq), freq = term.freq)

library(ggplot2)
ggplot(df, aes(x = term, y = freq))+
  geom_bar(stat = "identity") +
  xlab("Terms") + ylab("Count") + coord_flip()
```

Bar Chart Using ggplot()



What is Sentiment Analysis?

- Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral.

- It determines the emotional tone behind words, typically categorizing sentiments as positive, negative, or neutral.

- This analysis helps businesses understand customer opinions, monitor brand reputation, and improve customer experiences by analyzing reviews, social media posts, and other textual data sources.

Sentiment Analysis in R

```
# Package for sentimental analysis
install.packages("RSentiment")  # You may need to install package rJava
library(RSentiment)
# Import text file with one text record in one row
data<-readLines(file.choose())
head(data)

# Calculate sentiment score of each document
calculate_score(data)
```

```
....
 [1]  0  1 -1  1 -1  1  1  1  1  1 -1  1  1  1  2  0  1 -1  2 -1  2  0
[23]  0  2  2  1  3  0  2  0  1  2  1  2  2  0 -1  0  1 -1  1  1 -1  3
[45]  0  3  1 -2 -1 -1  3  1  0  2  3 -1  3  1  0  1  4  2
```

Scores:

- 0 indicates neutral sentiment
- positive value indicates positive sentiment
- negative value indicates negative sentiment
- 99 indicates sarcasm

Sentiment Analysis in R…

```
# Display the summary of sentiment score of all the documents
calculate_total_presence_sentiment(data)
```

```
     [,1]            [,2]          [,3]           [,4]
[1,] "Sarcasm"    "Neutral"    "Negative"    "Positive"
[2,] "0"          ”11"         ”11"           ”21"
     [,5]                     [,6]
[1,] "Very Negative"    "Very Positive"
[2,] "1"                 ”18"
```

Sentiment Analysis in R – Using Package "syuzhet"

- Syuzhet is an R package for the extraction of sentiment and sentiment-based plot arcs from text.

- The name “Syuzhet” comes from the Russian Formalists Victor Shklovsky and Vladimir Propp who divided narrative into two components, the “fabula” and the “syuzhet.” Syuzhet refers to the “device” or technique of a narrative whereas fabula is the chronological order of events. Syuzhet, therefore, is concerned with the manner in which the elements of the story (fabula) are organized (syuzhet)

- The package is more suitable for analysis of sentiment trajectory across a text document.

Sentiment Analysis in R – Using Package "syuzhet"...

- Syuzhet incorporates four sentiment lexicons:
  - “syuzhet” (Default) : Developed in the Nebraska Literary Lab under the direction of Matthew L. Jockers
  - “afinn” : Developed by Finn Arup Nielsen as the AFINN WORD DATABASE
  - “bing” : Developed by Minqing Hu and Bing Liu as the OPINION LEXICON
  - “nrc” : Developed by Mohammad, Saif M. and Turney, Peter D. as the NRC EMOTION LEXICON.
- These lexicons contain words tagged with sentiment values, which are typically positive, negative, or neutral.
- get_nrc_sentiment() is a useful function for analysing sentences in terms of emotions and sentiments. **NRC Emotion Lexicon**: Tags words with various emotions and sentiment categories.

- get_nrc_sentiment is a function from the syuzhet package in R, which is used to perform sentiment analysis using the NRC Emotion Lexicon. This lexicon assigns words to various emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) and sentiment categories (positive, negative).

Sentiment Analysis in R – Using Package "syuzhet"...

```
# Display emotions and sentiment scores for each document
library(syuzhet)
nrcsentiment <- get_nrc_sentiment(data)
head(nrcsentiment)
```

```
  anger anticipation disgust fear joy sadness surprise trust negative positive
1     1            0       0    0   0       0        0     0        1        0
2     0            1       0    0   1       0        1     1        0        1
3     0            0       0    0   0       0        0     0        0        0
4     0            0       0    0   0       0        0     0        0        0
5     0            0       0    0   0       0        0     0        1        0
6     0            0       0    0   0       0        0     0        0        0
```

After obtaining the sentiment scores using the get_nrc_sentiment function, there are several actions you can take to further analyze and utilize the sentiment information.

Summary Statistics and Aggregation

Compute summary statistics to get an overall sense of the sentiment in your dataset. This could include:

Average sentiment scores

Distribution of sentiment categories

Proportion of positive vs. negative sentiment

Remark:

Text mining can be applied to any language, although the complexity and effectiveness of the process may vary depending on the language and the available tools and resources.

Many text mining libraries and tools support multiple languages. Here are some popular ones:

- **NLTK (Python)**: Offers support for multiple languages, including tokenization, stop words, and stemming.
- **SpaCy (Python)**: Provides language models for various languages, including tokenization, POS tagging, named entity recognition (NER), etc.
- **TextBlob (Python)**: Supports basic text processing tasks for several languages.
- **Tidytext (R)**: Works with various languages through tokenization and text mining functions.
- **Syuzhet (R)**: Supports sentiment analysis in multiple languages using different lexicons.

THANK YOU!!