

Lecture 9: Sentiment Analysis

Isabel Casas
icasas@deusto.es

- This section is based in Chapter 20 of Galit et al (2019).
- We have worked with variables that were numeric or could be translated into numbers.
- Now we want the computer to read text.
- The representation of text into mathematical tools like a matrix is a bit different.

Sentiment analysis

- Sentiment analysis is a technique used in natural language processing and computational linguistics to figure out if a text, like a review or a social media post, has a positive, negative, or neutral tone.
- It does this by breaking down the text into individual words or phrases, and then checking if those words or phrases have a positive or negative meaning.
- This can be useful for businesses and researchers to understand how people feel about their products or services, or to monitor their online reputation.
- Sentiment analysis has many practical applications, such as social media monitoring, market research, customer feedback analysis, and brand reputation management.

A sentiment analysis consists of the following steps:

- 1 Get data and load it into R
- 2 Preprocess the data
- 3 Create a list of positive and negative words using a lexicon
- 4 Apply sentiment algorithm
- 5 Analyse the results (frequency distribution, mean, wordclouds, etc)

Sentiment analysis

We define something we call the *term-document matrix*. For example, consider the following tree sentences:

S1. One elephant

S2. Two elephants

S3. Three spiders

We can represent the words (called *terms*) in these three sentences (called *documents*) in a *term-document matrix*, where each row is a word and each column is a sentence.

Term-document matrix

- In R, we can convert a set of documents into a *term-document matrix* by using functions in library *tm*.
- First, we must create a **corpus** (a set of documents) and then use function *TermDocumentMatrix()*.

Term-document matrix

In the following chunk, we create a vector of sentences, then make a corpus and count each term.

```
library(tm)
text <- c("One elephant", "Two elephants", "Three spiders")
#convert sentences into a corpus
corp <- Corpus(VectorSource(text))
#compute term frequency
tdm <- TermDocumentMatrix(corp)
inspect(tdm)
```

```
## <<TermDocumentMatrix (terms: 6, documents: 3)>>
## Non-/sparse entries: 6/12
## Sparsity           : 67%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample            :
##
##      Docs
## Terms  1 2 3
## elephant 1 0 0
## elephants 0 1 0
## one      1 0 0
## spiders  0 0 1
## three    0 0 1
## two      0 1 0
```

This binary matrix could be used in clustering or other numerical methods. But we are not interested on this now. We are interested in sentiment analysis.

Section 1

Text Preprocessing

Preprocessing the text

Considered the following example of sentences that we want to understand:

S1. One pink elephant!

S2. One blue, big elephant :)

S3. Thee spiders

S4. The elephants

We see that there are extra spaces, non-alphabetic characters, mistaken capitalization and even a spelling mistake “Thee”.

The term-document matrix of these sentences is:

Preprocessing the text

```
text <- c("One pink elephant!", " One blue, big elephant :)",  
         "Thee blue spiders", "The elephants")  
corp <- Corpus(VectorSource(text))  
tdm <- TermDocumentMatrix(corp)  
inspect(tdm)
```

```
## <<TermDocumentMatrix (terms: 11, documents: 4)>>  
## Non-/sparse entries: 12/32  
## Sparsity           : 73%  
## Maximal term length: 9  
## Weighting          : term frequency (tf)  
## Sample            :  
##  
##      Docs  
## Terms    1 2 3 4  
## big      0 1 0 0  
## blue     0 0 1 0  
## blue,    0 1 0 0  
## elephant 0 1 0 0  
## elephant! 1 0 0 0  
## elephants 0 0 0 1  
## one      1 1 0 0  
## pink     1 0 0 0  
## spiders  0 0 1 0  
## thee     0 0 1 0
```

- Even in a simple example, we already have other characters apart from words: punctuation, mistakes, emojis, etc.
- To help with this, we divide the text into terms and remove the punctuation, stopwords, non-alphanumeric characters, etc.

Tokenization

- Tokenization is dividing the text into separate “tokens” or terms.
- For example, a word (separated by spaces) is a token.
- $2+3$ will be divided into 3 tokens (2, +, 3).
- 23 will stay as one token.
- Punctuation like “,” or “.” or “@” will stay as one token.

Tokenization

These tokens are now row headers in the data matrix. In the chunk below, we use *tm_map* to remove blanks and punctuation.

```
corp <- tm_map(corp, stripWhitespace)
corp <- tm_map(corp, removePunctuation)
tdm <- TermDocumentMatrix(corp)
inspect(tdm)
```

```
## <<TermDocumentMatrix (terms: 9, documents: 4)>>
## Non-/sparse entries: 12/24
## Sparsity           : 67%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample            :
##               Docs
## Terms      1 2 3 4
## big        0 1 0 0
## blue       0 1 1 0
## elephant   1 1 0 0
## elephants  0 0 0 1
## one        1 1 0 0
## pink       1 0 0 0
## spiders    0 0 1 0
## the        0 0 0 1
## thee       0 0 1 0
```

- We have very long matrices of tokens from big sets of text or comments.
- Many of those tokens will not be informative and need to be removed to keep the important words.
- We define as **stopwords**, those words which occur very frequently in our text but are not informative. This include articles, adverbs, pronouns, etc.

Text reduction

A list of stopwords in library *tm* can be found with:

```
stopwords("english")
```

```
## [1] "i" "me" "my" "myself" "we"
## [6] "our" "ours" "ourselves" "you" "your"
## [11] "yours" "yourself" "yourselves" "he" "him"
## [16] "his" "himself" "she" "her" "hers"
## [21] "herself" "it" "its" "itself" "they"
## [26] "them" "their" "theirs" "themselves" "what"
## [31] "which" "who" "whom" "this" "that"
## [36] "these" "those" "am" "is" "are"
## [41] "was" "were" "be" "been" "being"
## [46] "have" "has" "had" "having" "do"
## [51] "does" "did" "doing" "would" "should"
## [56] "could" "ought" "i'm" "you're" "he's"
## [61] "she's" "it's" "we're" "they're" "i've"
## [66] "you've" "we've" "they've" "i'd" "you'd"
## [71] "he'd" "she'd" "we'd" "they'd" "i'll"
## [76] "you'll" "he'll" "she'll" "we'll" "they'll"
## [81] "isn't" "aren't" "wasn't" "weren't" "hasn't"
## [86] "haven't" "hadn't" "doesn't" "don't" "didn't"
## [91] "won't" "wouldn't" "shan't" "shouldn't" "can't"
## [96] "cannot" "couldn't" "mustn't" "let's" "that's"
## [101] "who's" "what's" "here's" "there's" "when's"
## [106] "where's" "why's" "how's" "a" "an"
## [111] "the" "and" "but" "if" "or"
## [116] "because" "as" "until" "while" "of"
## [121] "at" "by" "for" "with" "about"
## [126] "against" "between" "into" "through" "during"
## [131] "before" "after" "above" "below" "to"
## [136] "from" "up" "down" "in" "out"
```

Activity 1

Find the list of stopwords in Spanish

In general, we can reduce the volume of text using the following techniques:

- *Stemming*, reduces different variates of words to a common core (root of the word). For example, “eleph” is the root of “elephant” and “elephants”.
- Frequency filters are used to remove very frequent words or very rare
- Synonyms
- Letter case can be ignored
- A variety of specific terms in a category can be replaced by the category. For example, different e-mail addresses or different numbers may be replaced by “emailtoken” or “numbertoken”

Text reduction

```
library(SnowballC)
## remove English stopwords
corp <- tm_map(corp, removeWords, stopwords("english"))
## Use stemming to reduce the term-document matrix
corp <- tm_map(corp, stemDocument)
tdm <- TermDocumentMatrix(corp)
inspect(tdm)
```

```
## <<TermDocumentMatrix (terms: 8, documents: 4)>>
## Non-/sparse entries: 12/20
## Sparsity          : 62%
## Maximal term length: 6
## Weighting          : term frequency (tf)
## Sample            :
##           Docs
## Terms   1 2 3 4
## big     0 1 0 0
## blue    0 1 1 0
## eleph   1 1 0 1
## one     1 1 0 0
## pink    1 0 0 0
## spider  0 0 1 0
## the     0 0 0 1
## thee    0 0 1 0
```

Section 2

Meaning Extraction

- Now that the text have been pre-processed, we can analyse what is left.
- For example we could count frequencies or lack of words that were expected to be there.
- Also, we can calculate the sentiment score of different type of emotions and perceptions.

Sentiment analysis

- The word *sentiment* refers to “the underlying feeling, attitude, evaluation or emotion associated with an opinion” (Definition 2.4 in *Liu, 2015*).

	Type	Orientation	Example
1	Rational	Positive	This car is worth the price
2	Rational	Negative	The voice on this phone is not clear
3	Emotional	Positive	I love my new bag
4	Emotional	Negative	I am so angry with their service people
5		Neutral	The box contains a cable and a phone

Sentiment analysis

```
## This code generates the table below
tab1 <- data.frame (Type = c("Rational", "Rational", "Emotional", "Emotional", " "),
                    Orientation=c("Positive", "Negative", "Positive", "Negative", "Neutral"),
                    Example =c("This car is worth the price", "The voice on this phone is not clear",
                               "I love my new bag", "I am so angry with their service people",
                               "The box contains a cable and a phone"))

library("kableExtra")
kable(tab1) %>%kable_styling() %>%
  column_spec(1, bold = TRUE, width = "1.5cm", border_left =TRUE, border_right = T) %>%
  column_spec(2, bold = T, width = "3cm", border_right = T) %>%
  column_spec(3, bold = T, width = "15cm", border_right = T)
```

Type	Orientation	Example
Rational	Positive	This car is worth the price
Rational	Negative	The voice on this phone is not clear
Emotional	Positive	I love my new bag
Emotional	Negative	I am so angry with their service people
	Neutral	The box contains a cable and a phone

Sentiment analysis

- The sentiment orientation (polarity) can be *positive*, *negative* or *neutral* and it also can have different intensities.
- For example, *good* has a lower intensity than *wonderful*.
- There are also other words that intensify the meaning when placed in front of other words like *extremely*, *very*, etc.
- It is important to know that emotional sentiments are usually stronger than rational sentiments and are the source of study of most sentiment analysis projects.

- There are several lexicons of words with different sentiment orientations, which have been validated. For example, we find 4 different lists in library *tidytext*. Check the R help for details.

```
library(tidytext)
library(textdata)
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##   word      value
##   <chr>    <dbl>
## 1 abandon    -2
## 2 abandoned  -2
## 3 abandons   -2
## 4 abducted   -2
## 5 abduction  -2
## 6 abductions -2
## 7 abhor      -3
## 8 abhorred   -3
## 9 abhorrent  -3
## 10 abhors    -3
## # i 2,467 more rows
```

Lexicons

- Lexicon “NRC” is interesting because each word is classified into eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). So we can use it to define the type of feeling.

```
get_sentiments("nrc")
```

```
## # A tibble: 13,872 x 2
##   word      sentiment
##   <chr>    <chr>
## 1 abacus   trust
## 2 abandon  fear
## 3 abandon  negative
## 4 abandon  sadness
## 5 abandoned anger
## 6 abandoned fear
## 7 abandoned negative
## 8 abandoned sadness
## 9 abandonment anger
```


Section 3

Application to Amazon reviews data

Sentiment analysis of Amazon reviews

- Sentiment analysis on comments from Amazon reviews inside *Lab6_2023.Rdata*
- Recall we scraped reviews of the Amazon product *A song of Ice and Fire* in Lab6
- *output_list* in *Lab6_2023.Rdata* contains reviews from 50 pages, 10 reviews for each page scraped in 2023. It contains the fields *review_title*, *review_text*, *review_star* and *page*

Exploratory Data Analysis (EDA)

The analytical possibilities: wordclouds, n-gram analysis, sentiment analysis, network diagram, etc. The choice is yours and also depends on your problem.

Our `output_list` is a list of documents, so we have to bind all the documents together first. We do this only with the “review_text” field.

```
load("Lab6_2023.RData")
library(tidytext)
library(wordcloud)
library(tidyverse)
class (output_list)
```

```
## [1] "list"
```

```
comments <- output_list%>%bind_rows() #binding all together
class(comments)
```

```
## [1] "tbl_df"      "tbl"        "data.frame"
```

Exploratory Data Analysis (EDA)

- The objective is to create a wordcloud of the main words in the *review_text*.
- Tokenise words first, creating a row for each token of *review_text* that goes into column *words* in the chunk below (you can choose any other name)

```
tokenized <- unnest_tokens(comments, output = "word",  
                           input = "review_text", token = "words")  
token.number <- count(tokenized, word, sort = TRUE)  
head(token.number)
```

```
## # A tibble: 6 x 2  
##   word      n  
##   <chr> <int>  
## 1 the    1535  
## 2 and     706  
## 3 i       638  
## 4 to      609  
## 5 a       606  
## 6 of      530
```

Frequency analysis

Let us choose the words related to “joy” from our comments, using the NRC lexicon. For example, we can compare the frequency of the top 10 words related to joy and to fear and then see the results graphically.

```
#joy
nrc_joy <- get_sentiments("nrc") %>% filter(sentiment == "joy")
word.joy <- token.number %>%inner_join(nrc_joy)
#fear
nrc_fear <- get_sentiments("nrc")%>%filter(sentiment == "fear")
word.fear <- token.number %>%inner_join(nrc_fear)
```

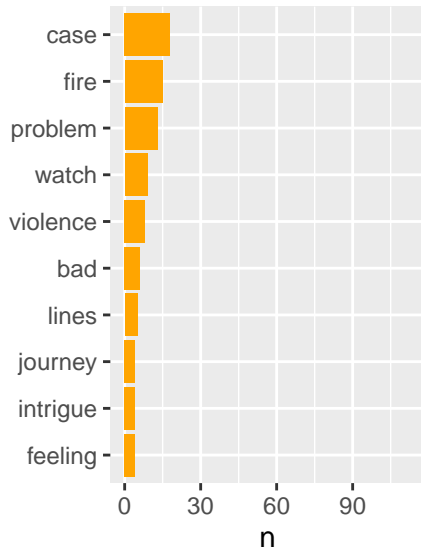
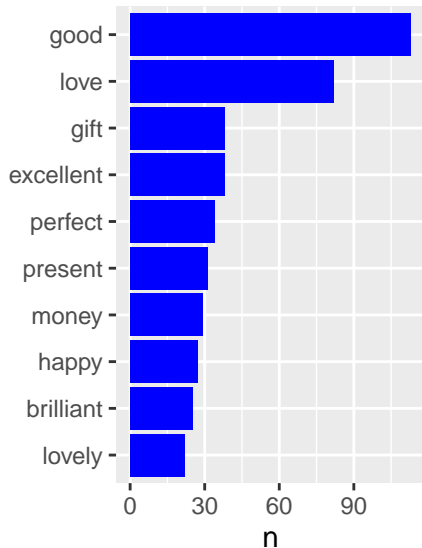
Now we will show these result in bar-plots.

Frequency analysis joy vs fear

```
library(ggplot2)
library(gridExtra)
plot1 <-word.joy[1:10,] %>%
  ggplot(aes(x=reorder(word, n), y=n)) + ylim(0, max(word.joy$n, word.fear$n))+
  geom_bar(stat='identity', fill="blue") + coord_flip() + theme(legend.position = "none") +
  labs(x="")

plot2 <-word.fear[1:10,] %>%
  ggplot(aes(x=reorder(word, n), y=n)) + ylim(0, max(word.joy$n, word.fear$n))+
  geom_bar(stat='identity', fill="orange") + coord_flip() + theme(legend.position = "none") +
  labs(x="")
grid.arrange(plot1, plot2, ncol=2)
```

Frequency analysis joy vs fear



Frequency analysis positive vs negative

- If we do the same with positive and negative, then we get the plots below.
- We see that words *good*, *love* and *recommend* are very common and even some of the negative words can have a positive meaning like *hooked*.
- Also some people found it *violent* and were *dissapointed*.
- In general, the overall sentiment for this sequel seems positive.

Frequency analysis positive vs negative

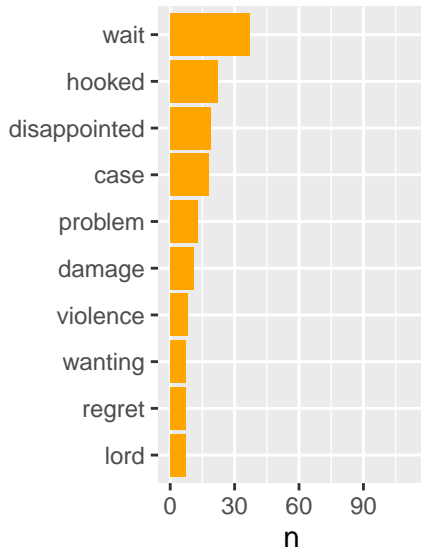
```
#positive
nrc_pos <- get_sentiments("nrc") %>% filter(sentiment == "positive")
word_pos <- token.number %>%inner_join(nrc_pos)

#negative
nrc_neg <- get_sentiments("nrc")%>%filter(sentiment == "negative")
word_neg <- token.number %>%inner_join(nrc_neg)

library(ggplot2)
library(gridExtra)
plot1 <-word_pos[1:10,] %>%
  ggplot(aes(x=reorder(word, n), y=n)) + ylim(0, max(word_pos$n, word_neg$n))+
  geom_bar(stat='identity', fill="blue") + coord_flip() + theme(legend.position = "none") +
  labs(x="")

plot2 <-word_neg[1:10,] %>%
  ggplot(aes(x=reorder(word, n), y=n)) + ylim(0, max(word_pos$n, word_neg$n))+
  geom_bar(stat='identity', fill="orange") + coord_flip() + theme(legend.position = "none") +
  labs(x="")
grid.arrange(plot1, plot2, ncol=2)
```

Frequency analysis positive vs negative



- A wordcloud is a very descriptive and aesthetical way of showing what comments are about.
- For an informative word cloud, we remove numbers, stopwords and in our example, the words “book” and “books” which are not informative in this case.
- First we plot the wordcloud without cleaning

EDA: Wordcloud without cleansing

```
word_tb1 <- token.number %>%anti_join(tidytext::stop_words, by = "word")
wordcloud::wordcloud(words = word_tb1$word, freq = word_tb1$n, max.words = 100,
  random.color = TRUE, colors= brewer.pal(6,"Dark2"), scale=c(2.5,.5),
  main = "Before cleaning")
```

EDA: Wordcloud without cleansing



EDA: Wordcloud removing some words

```
#without words book and books and stopwords
word_tb2 <- token.number %>%filter(!(word %in% c("book","books")))%>%
  filter(!grepl("^\\d+$", word)) %>% # remove numbers
  anti_join(tidytext::stop_words, by = "word")
wordcloud::wordcloud(words = word_tb2$word, freq = word_tb2$n, max.words = 100,
  random.color = TRUE, colors=brewer.pal(9,"PuOr"), scale=c(2.5,.5))
```

EDA: Wordcloud removing some words



Sentiment score

We have a positive score of 88 more positive words than negative which is a 44% more positive words than negative of the total 498 words with sentiment scale.

```
bind_rows(word.pos, word.neg)%>% count(sentiment) %>%  
  spread(sentiment, n) %>% mutate(score1=positive-negative,  
                                   score2 = positive/negative)
```

```
## # A tibble: 1 x 4  
##   negative positive score1 score2  
##   <int>    <int> <int> <dbl>  
## 1      200     288   88  1.44
```


Homework

- Go through the slides and make sure that the code works for you.
- Use it with other dataset of your interest.

References

Galit Shmueli, Peter C. Bruce, Inbal Yahav, Nitin R. Patel, Kenneth C. Lichtendahl Jr. (2019). "Data Mining for Business Analytics. Concepts, techniques and applications". Wiley.

Bing Liu (2015). "Sentiment Analysis. Mining opinions, sentiments, and emotions". Cambridge University Press.

Martin Chan Blog. "Vignette: Scraping Amazon Reviews in R". Series [URL](#)