Sentiment Analysis

Max Callaghan

2022-11-10

Introduction and Objectives

Assignment 2

Assignment 2 is still live. If you have issues, or encounter difficulties, raise an issue on the Github repository, or write me an email!

Assignment 3

Assignment 3 is approaching, and you should have a clear idea of what you want to do by the end of next week.

Feel free to ask for quick feedback on any ideas you have in the coming days

Objectives

By now we have spent a long time understanding how to **represent** texts in simple and more complex ways.

We've also started asking questions about texts. Viz. What is it about?

Today we will ask a new question about texts: what sentiment does it express?

Introduction to sentiment analysis

What is a sentiment?

The emotion embodied in a text. Often reduced to positive-negative, but can encompass a more complex range of emotions like joy, sadness, anger.

Sentiment analysis as classification

In some ways

An overview of techniques to do sentiment analysis

Doing sentiment analysis usually involves rule-based or statistical techniques

- Assessing sentiment based on counting words have a predefined sentiment

An overview of techniques to do sentiment analysis

Doing sentiment analysis usually involves rule-based or statistical techniques

- Assessing sentiment based on counting words have a predefined sentiment
- Using a classifier that has been trained to identify sentiment with text examples that have been labelled.

Lexicon-based sentiment analysis

Positive and negative words

We know about the “bag of words” model of representing texts.

We also know that some words are rather positive, whereas some are rather negative.

Consider the texts:

```
texts <- c(
  "Elon Musk is a champion of free speech",
  "It's a terrible shame to see mashed potato thrown at art"
)
```

Do they express positive or negative sentiment? How can we tell?

Using Lexicons in R

We can import a lexicon in R using tidytext. Each row, contains a word and its value

```
library(tidytext)
library(dplyr)
lex <- get_sentiments("afinn")
sample_n(lex, 5)
```

```
## # A tibble: 5 x 2
##   word        value
##   <chr>       <dbl>
## 1 incompetent    -2
## 2 honour          2
## 3 lawsuit        -2
## 4 whore          -4
## 5 flagship        2
```

Using Lexicons in R

Note that the Afinn lexicon is not the newest version. We can just read this in directly from the author's Github page.

```
library(readr)
lex <- read_tsv(
  "https://raw.githubusercontent.com/fnielsen/afinn/master/afinn/data/AFINN-en-16
  col_names=c("word","value")
)

lex
```

```
## # A tibble: 3,382 x 2
##    word        value
##    <chr>       <dbl>
##  1 abandon        -2
##  2 abandoned      -2
##  3 abandons       -2
##  4 abducted       -2
```

Using Lexicons in R

There are a few different lexicons, compiled by different authors, using different techniques involving amazon turk and author knowledge, which encode different types of emotions.

```
library(tidytext)
library(dplyr)
lex <- get_sentiments("nrc")
head(lex)
```

```
## # A tibble: 6 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
## 6 abandoned fear
```

Using Lexicons in R

We can also put our usual document feature matrix into a similar format

```
library(quanteda)
dfmat <- texts %>%
  tokens %>%
  dfm()

text_tokens <- tidy(dfmat)
head(text_tokens)
```

```
## # A tibble: 6 x 3
##   document term     count
##   <chr>    <chr>    <dbl>
## 1 text1    elon         1
## 2 text1    musk         1
## 3 text1    is           1
## 4 text1    a            1
## 5 text2    a            1
## 6 text1    champion     1
```

Tidy lexicons

Now we can join these to see which words in the texts have what sentiment

```
lex <- read_tsv("https://raw.githubusercontent.com/fnielsen/afinn/master/afinn/data/AFINN-en-165.txt", col_names=c(
dfmat <- texts %>%
  tokens %>%
  dfm()

text_tokens <- tidy(dfmat) %>%
  inner_join(lex, by=c("term" = "word"))

text_tokens
```

```
## # A tibble: 4 x 4
##   document term     count value
##   <chr>    <chr>    <dbl> <dbl>
## 1 text1    champion     1     2
## 2 text1    free         1     1
## 3 text2    terrible     1    -3
## 4 text2    shame        1    -2
```

Tidy lexicons

We can then just sum word scores for each document to get a sentiment score for that document

```
doc_sentiments <- tidy(dfmat) %>%
  inner_join(lex, by=c("term" = "word")) %>%
  mutate(value=value*count) %>%
  group_by(document) %>%
  summarise(value = sum(value))

doc_sentiments
```

```
## # A tibble: 2 x 2
##   document value
##   <chr>    <dbl>
## 1 text1        3
## 2 text2       -5
```

VADER

VADER represents just about the state of the art in lexicon-based sentiment analysis, and is especially suitable for social media texts.

It also incorporates rules that extend it beyond the bag-of-words model

5 Heuristics

The Vader paper identifies 5 heuristics that extend just counting words from a lexicon, and implements these in their algorithm.

- Punctuation (!) increases the magnitude of the sentiment: "Food here is good!!" > "Food here is good"

5 Heuristics

The Vader paper identifies 5 heuristics that extend just counting words from a lexicon, and implements these in their algorithm.

- Punctuation (!) increases the magnitude of the sentiment: ”Food here is good!!” $>$ ”Food here is good”
- CAPITALIZATION increaeses the magnitude of the sentiment: ”Food here is GREAT” $>$ ”Food here is great”

5 Heuristics

The Vader paper identifies 5 heuristics that extend just counting words from a lexicon, and implements these in their algorithm.

- Punctuation (!) increases the magnitude of the sentiment: "Food here is good!!" $>$ "Food here is good"
- CAPITALIZATION increaeses the magnitude of the sentiment: "Food here is GREAT" $>$ "Food here is great"
- Degree modifiers impact intensity $>$ or $<$. "Service is marginally good" $<$ "service is good" $<$ "service is extremely good".

5 Heuristics

The Vader paper identifies 5 heuristics that extend just counting words from a lexicon, and implements these in their algorithm.

- Punctuation (!) increases the magnitude of the sentiment: ”Food here is good!!” $>$ ”Food here is good”
- CAPITALIZATION increaeses the magnitude of the sentiment: ”Food here is GREAT” $>$ ”Food here is great”
- Degree modifiers impact intensity $>$ or $<$. ”Service is marginally good” $<$ ”service is good” $<$ ”service is extremely good”.
- ”But” signals shift in sentiment, and that second clause is stronger: ”Food here is good, but the service is bad” -$>$ Overall more negative than positive

5 Heuristics

The Vader paper identifies 5 heuristics that extend just counting words from a lexicon, and implements these in their algorithm.

- Punctuation (!) increases the magnitude of the sentiment: "Food here is good!!" > "Food here is good"
- CAPITALIZATION increaeses the magnitude of the sentiment: "Food here is GREAT" > "Food here is great"
- Degree modifiers impact intensity > or <. "Service is marginally good" < "service is good" < "service is extremely good".
- "But" signals shift in sentiment, and that second clause is stronger: "Food here is good, but the service is bad" -> Overall more negative than positive
- Negations in a tri-gram preceeding a sentiment-laden feature flip the polarity

VADER in practice

Let's load a dataset of tweets from the VoteYes campaign from the Scottish independence referendum. We can calculate sentiment for each tweet using `vader_df()`.

Let's look at the most positive tweets

```
library(vader)
tweets <-read_delim("../datasets/YesScotlandTweets_cleaned.csv", delim=",", escape_double=TRUE)
sentiments <- vader_df(tweets$text)

tweet_sentiment <- cbind(tweets, select(sentiments,-text))

pos <- tweet_sentiment %>% arrange(desc(compound)) %>%
  head()

for( i in rownames(pos) ) {
  print(pos[i, "text"])
  print(pos[i, "compound"])
}
```

```
## [1] "A Yes means greater financial security for families - we can expand free childcare, safeguard free educatio
## [1] 0.96
## [1] "RT @mstewart_23: #indyref is about the country we want to live in &amp; how best to create that. YES gives
## [1] 0.952
## [1] "With Yes, we can build on Scotland's successes in delivering for older people, such as free personal care a
## [1] 0.944
## [1] "With a Yes, we can make Scotland's wealth work better for our families - with better jobs and increased fre
```

VADER in practice

Let's load a dataset of tweets from the VoteYes campaign from the Scottish independence referendum. We can calculate sentiment for each tweet using `vader_df()`.

Let's look at the most negative tweets

```
neg <- tweet_sentiment %>% arrange(compound) %>%
  head()

for( i in rownames(neg) ) {
  print(neg[i, "text"])
  print(neg[i, "compound"])
}
```

```
## [1] "A statement: There is ABSOLUTELY no place for attacks - be they abuse, graffiti, vandalism or physical assa
## [1] -0.934
## [1] "Westminster wants to waste our resources on renewing obscene and dangerous weapons of mass destruction. Sco
## [1] -0.925
## [1] "RT @AlexSalmond: The murder of David Haines shows a degree of brutality which defies description. Thoughts
## [1] -0.866
## [1] "Damaging Westminster cuts are threatening Scotland\u008a\u0097Ès public services. #indyref #VoteYes http://
## [1] -0.836
## [1] "RT @martin_compston: More ridiculous scare stories this regarding losing BBC shows I'm in a hotel in Irelan
## [1] -0.835
## [1] "RT @StephenNoon: Hearing that a truly desperate &amp; shameful scare story is coming @DHgovuk about to thre
## [1] -0.813
```

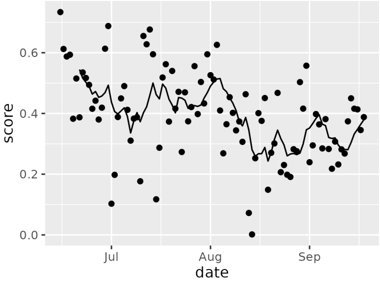
Sentiment over time

We can also look at how sentiment changed over time by taking the mean compound score in each time period. Given the regular week-weekend variation, it also makes sense to show the 7 day rolling mean

```
tweet_sentiment$date <- as.Date(tweet_sentiment$created)
daily_sentiment <- tweet_sentiment %>% group_by(date) %>%
  summarise(score = mean(compound)) %>%
  mutate(score7 = data.table::frollmean(score, 7))

library(ggplot2)
ggplot(daily_sentiment, aes(date)) +
  geom_point(aes(y=score)) +
  geom_line(aes(y=score7))
```

```
ggsave("plots/sentiment_time.png", width=4, height=3)
```



Comparing sentiment analysis with wordshift

Why is one corpus more positive/negative than another?

Fancy sentiment analysis

Fancy sentiment analysis

Fancy NLP does not apply rules that we give it. It *learns* rules from training data.

Complex models, which encode text in complex ways, have outperformed lexicon-based sentiment analysis *on the main benchmarked tasks for which they are often optimized*.

Sentiment datasets are often comprised of movie or product reviews.

Fancy sentiment analysis

We will learn more about how training such models work in the next sessions, but you can access one of many such models here

Sentiment analysis validation

Validation

Almost all methods for sentiment analysis are validated, but almost none are validated on your dataset. Unless your dataset is very similar to the validation dataset, you should validate yourself.

This means selecting a random sample of your texts, labelling the sentiment of these texts by hand, then comparing the label you gave with the score given by your method.

If your method gives the same label as you in 100% of cases, then you have an accuracy of 100%

Sentiment analysis in the wild

Paper 1