Sentiment Analysis

Max Callaghan

2022-11-10

Introduction and Objectives

Assignment 4

Assignment 4 is still live. If you have issues, or encounter difficulties, raise an issue on the Github repository, or write me an email!

Assignment 5

Assignment 5 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: What are they about?

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

Intro 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, sentiment analysis is a special case of the broader **classification** task, where we ask:

- Is a text of class A or not? or,
- Is a text of class A or B

With sentiment analysis, the classes are inherent features of how humans use language which *generalise* (to a certain extent) across contexts.

An overview of techniques to do sentiment analysis

Doing sentiment analysis usually involves **rule-based** or **statistical machine-learning** techniques

An overview of techniques to do sentiment analysis

Doing sentiment analysis usually involves **rule-based** or **statistical machine-learning** 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 machine-learning** 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 criticizing    -2
## 2 aggression     -2
## 3 lovable         3
## 4 abilities       2
## 5 friendly        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-165
  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
##  5 abduction     -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
```

Using Lexicons in Python

Doing this in Python is very similar. We can use the afinn package to access the afinn lexion

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from afinn import Afinn
import numpy as np

afinn = Afinn()
lex = pd.DataFrame(afinn._dict.items(),columns=["word","value"])

lex.head()
```

```
##          word  value
## 0     abandon     -2
## 1   abandoned     -2
## 2    abandons     -2
## 3    abducted     -2
## 4   abduction     -2
```

Using Lexicons in Python

Then, in the same way, we can put our dfm into a "tidy" form and merge with our lexicon

```
texts = [
    "Elon Musk is a champion of free speech",
    "It's a terrible shame to see mashed potato thrown at art"
]
vec = CountVectorizer()
dfmat = vec.fit_transform(texts)

def tidy_dfmat(dfmat, vec):
    nz = dfmat.nonzero()
    text_df = pd.DataFrame({
        "document": np.array(texts)[nz[0]],
        "term": vec.get_feature_names_out()[nz[1]],
        "count": dfmat[nz].A1
    })
    return text_df
text_df = tidy_dfmat(dfmat, vec)
text_tokens = text_df.merge(lex, left_on="term", right_on="word")
text_tokens["value"] *= text_tokens["count"]
text_tokens
```

```
##                                        document      term  ...      word value
## 0             Elon Musk is a champion of free speech  champion  ...  champion     2
## 1             Elon Musk is a champion of free speech      free  ...      free     1
## 2  It's a terrible shame to see mashed potato thr...  terrible  ...  terrible    -3
## 3  It's a terrible shame to see mashed potato thr...     shame  ...     shame    -2
```

Using Lexicons in Python

In the very same way, we can sum across documents

```
doc_sentiments = text_tokens.groupby("document")["value"].sum()
doc_sentiments
```

```
## document
## Elon Musk is a champion of free speech                      3
## It's a terrible shame to see mashed potato thrown at art   -5
## Name: value, dtype: int64
```

Although we can also get the sentiment from a text directly

```
scores = [afinn.score(t) for t in texts]
for text, score in zip(texts,scores):
  print(text, score)
```

```
## Elon Musk is a champion of free speech 3.0
## It's a terrible shame to see mashed potato thrown at art -5.0
```

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

Exercise: trying VADER out

Try out a few examples of texts with VADER. See if you can show how the rules it employs work, and see if you can "trick" the algorithm.

```
library(vader)
get_vader("This text is a test")
```

```
##       word_scores           compound                pos                neu
## "{0, 0, 0, 0, 0}"                "0"                "0"                "1"
##                neg         but_count
##                "0"                "0"
```

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
analyzer.polarity_scores("This text is a test")
```

```
## {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

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)
yes <-read_csv("../datasets/YesScotlandTweets_cleaned.csv")
yes$campaign <- "YesScotland"
no <- read_csv("../datasets/UkTogetherTweets_cleaned.csv")
no$campaign <- "UkTogether"
tweets <- rbind(yes, no)

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] "RT @dobbs_michael: Off to Edinburgh. Yesterday great campaigning in W Scotland, smiles, great support. Hope
## [1] 0.965
## [1] "A Yes means greater financial security for families - we can expand free childcare, safeguard free educatio
```

VADER in practice

Let's load a dataset of tweets from the VoteYes and UkTogether campaigns 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] ".@TogetherAlistairDarling - \"The nationalist threat to default on our debt is irresponsible and reckless.\
## [1] -0.889
## [1] "RT @AlexSalmond: The murder of David Haines shows a degree of brutality which defies description. Thoughts
## [1] -0.866
## [1] "Salmond warned that reckless threat to default on debt could lead to higher mortgage payments for Scots htt
## [1] -0.866
## [1] "RT @Kevin_Maguire: However much I hate banksters, hedgies, speculators et al I think Salmond threatening to
## [1] -0.862
```

Sentiment over time

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

```
library(tidyr)
tweet_sentiment$date <- as.Date(tweet_sentiment$created)
wide_sentiment <- tweet_sentiment %>%
  group_by(campaign, date) %>%
  summarise(score = mean(compound)) %>%
  pivot_wider(names_from=campaign, values_from=score)

days <- data.frame(date=seq(as.Date("2014-06-01"),as.Date("2014-09-18"),1))

daily_sentiment <- days %>% left_join(wide_sentiment) %>%
  pivot_longer(cols=-date, names_to="campaign", values_to="score") %>%
  group_by(campaign) %>% arrange(date) %>%
  mutate(score7 = data.table::frollmean(score, 7))

daily_sentiment %>% head()
```

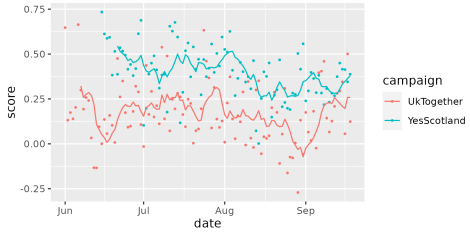
```
## # A tibble: 6 x 4
## # Groups:   campaign [2]
##   date       campaign     score score7
##   <date>     <chr>        <dbl>  <dbl>
## 1 2014-06-01 UkTogether   0.647     NA
```

Sentiment over time

Now that we have the data in the format we want, we can plug this into ggplot2

```
library(ggplot2)
ggplot(daily_sentiment, aes(date, colour=campaign)) +
  geom_point(aes(y=score),size=0.5) +
  geom_line(aes(y=score7))
```

```
ggsave("plots/sentiment_time.png", width=6, height=3)
```



VADER with Python

In python, we can use the vaderSentiment package

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

yes = pd.read_csv("../datasets/YesScotlandTweets_cleaned.csv")
yes["campaign"] = "YesScotland"
no = pd.read_csv("../datasets/UkTogetherTweets_cleaned.csv")
no["campaign"] = "UkTogether"
df = pd.concat([yes, no]).reset_index(drop=True)

analyzer = SentimentIntensityAnalyzer()

results = [analyzer.polarity_scores(x) for x in df["text"]]
sentiment = pd.DataFrame.from_dict(results)

sentiment_df = pd.concat([df,sentiment], axis=1)
sentiment_df.head()
```

```
##    x                                               text  ...    pos  compound
## 0  1  Thank you, everyone! #indyref http://t.co/1kTz...  ...  0.411    0.4199
## 1  2  As the polls close, total likes on the @YesSco...  ...  0.147    0.4754
## 2  3  RT @YESforScotland4: #voteyes YES what a fanta...  ...  0.297    0.8668
## 3  4  We can be proud of #indyref, which has seen a ...  ...  0.147    0.4767
## 4  5  RT @Crafty_Goddess: Txt from 25 year old niece...  ...  0.122    0.4497
##
## [5 rows x 22 columns]
```

VADER with Python

We get the data into the format where there is a row for every day, and the score for each day is in a column for each campaign

```
import matplotlib.pyplot as plt
sentiment_df["date"] = pd.to_datetime(df.created).dt.date
daily_sentiment = (sentiment_df
                   .groupby(["date","campaign"])["compound"]
                   .mean()
                   .reset_index()
                   .pivot_table(columns="campaign", values="compound", index="date")
                   .reset_index()
                  )

daily_sentiment["date"] = pd.to_datetime(daily_sentiment["date"])
days = pd.date_range(start="2014-06-01",end="2014-09-18")
daily_sentiment = pd.DataFrame({"date": days}).merge(daily_sentiment, how="left")

daily_sentiment.head()
```

```
##         date  UkTogether  YesScotland
## 0 2014-06-01    0.647000          NaN
## 1 2014-06-02    0.228642          NaN
## 2 2014-06-03    0.174267          NaN
## 3 2014-06-04    0.209287          NaN
## 4 2014-06-05    0.202725          NaN
```

Sentiment over time

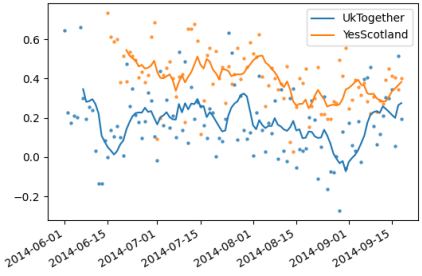
The easiest way to plot this is with base matplotlib

```
fig, ax = plt.subplots(figsize=(6,4))

for campaign in ["UkTogether", "YesScotland"]:
    ax.scatter(
      daily_sentiment.date, daily_sentiment[campaign],
      s=5, alpha=0.7
    )
    x = daily_sentiment[campaign].rolling(7).mean()
    ax.plot(daily_sentiment.date, x, label=campaign)

fig.autofmt_xdate()
ax.legend()

plt.savefig("plots/sentiment_time_py.png")
```



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

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%.

We will explore validation metrics in the next session on supervised learning.

Exercise - validating on your own data

- In a spreadsheet, write 5 texts that express your feelings about recent political events, with each text in a new row of the same column
- Ask your neighbour to rate the sentiment of your texts in the neighbouring column. The value should be between -1 (completely negative) and 1 (completely positive).
- Apply one of the techniques we have learnt today to your texts and compare with the human annotation
- What is the average (absolute) difference between your human label and the automated technique?
- For what text is the divergence greatest?

Examples in research

Tones from a Narrowing Race

In Tones from a Narrowing Race: Polling and Online Political Communication during the 2014 Scottish Referendum Campaign, Evelyne Brie and Yannick Dufresne set out to investigate the dynamics of how political campaigns use negative communication.

They use the data that we've just seen: tweets sent by the pro-independence and pro-union campaigns in the Scottish independence referendum of 2014.

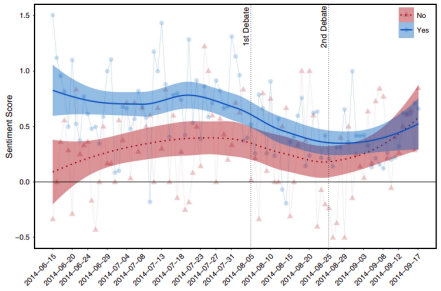
Tones from a Narrowing Race: Obectives

The authors want to provide evidence on negative campaigning *in practice* that complements political theory about how and why actors use negative campaigning.

They are interested in how real organisations use twitter, meaning their findings are not subject to the same external validity concerns generated by the unrepresentativeness of twitter.

Tones from a Narrowing Race: Results

The authors' analysis, which uses the Opinion Observer lexicon (I think available with the "bing" option in tidytext's `get_sentiments()`) finds a similar pattern of sentiment scores in the different campaigns over time



Tones from a Narrowing Race: Results

They find a significant relationship between the no campaign leading in the polls and the negativity of their tweets. The more NO leads, the more negatively they tweet.

In a simpler model, there is a significant relationship between the number of days to the election and the negativity of the YES campaign. That is, the decline in positivity as the election is statistically significant.

There is also evidence that the YES campaign tweeted more negatively the day after each debate (which they were said to have lost)

TABLE 1 *Testing the effect of public opinion and timing on sentiment score*

	Sentiment Score (Daily Average)					
	YES Campaign			NO Campaign		
Poll Lead	0.003		−0.01	−0.04*		−0.05**
	(0.01)		(0.01)	(0.01)		(0.02)
Momentum		−0.001	−0.001		−0.01	0.01
		(0.01)	(0.01)		(0.01)	(0.01)
Post-debate 1			−0.31*			0.08
			(0.15)			(0.19)
Post-debate 2			−0.38*			−0.20
			(0.18)			(0.24)
Poll Day	0.14	0.15*	0.15*	0.03	0.09	0.03
	(0.07)	(0.07)	(0.07)	(0.09)	(0.10)	(0.10)
Sentiment $Score_{t-1}$	0.20*	0.20*	0.13	0.005	0.08	−0.05
	(0.10)	(0.10)	(0.10)	(0.10)	(0.10)	(0.11)
Days to Referendum	0.003**	0.003*	−0.002	0.0002	0.001	−0.002
	(0.001)	(0.001)	(0.003)	(0.002)	(0.002)	(0.004)
Constant	0.27***	0.27**	0.74**	0.29*	0.23*	0.45
	(0.08)	(0.08)	(0.22)	(0.10)	(0.10)	(0.28)
N	94	94	94	94	94	94
R^2	0.18	0.18	0.23	0.11	0.03	0.15

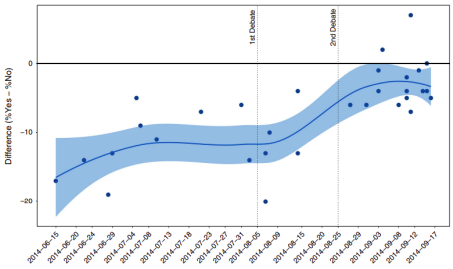
*p < 0.05; **p < 0.01; ***p <0.001.
Source: Official Twitter accounts of *Better Together* (N = 1,230) and of *Yes Scotland* (N = 3,078), and What Scotland Thinks (16 June to 17 September 2014).

Tones from a Narrowing Race: Results

They find a significant relationship between the no campaign leading in the polls and the negativity of their tweets. The more NO leads, the more negatively they tweet.

In a simpler model, there is a significant relationship between the number of days to the election and the negativity of the YES campaign. That is, the decline in positivity as the election is statistically significant.

There is also evidence that the YES campaign tweeted more negatively the day after each debate (which they were said to have lost)



Discusion

What does this tell us about the scottish election campagin?

What does this tell us about negative communication in political campaigns in general?

What potential confounding factors might explain the results?

Other examples

- Baylis and Obradovich et al. explore how weather affects sentiment
- Schwalbach looks at how political debate varies in sentiment through the legislative period, depending on their involvement in the government.
- Wang et al. explore how the COVID-19 pandemic affected how people expressed sentiment
- Arratia et al. describe how sentiment analysis is used to provide information about financial markets

Wrapup and Outlook

Wrapup

So far in this course, we have covered:

- How to get texts from tricky places
- How to process them and manipulate them
- How to represent them in different ways
- How to find the similarity between texts
- Hot to find what texts are about

Now we know how to find out what emotions are present in texts

Outlook

We said sentiment analysis is a special case of classification. We will explore this in detail next week when we cover **supervised text classification**.

We'll be training machine learning classifiers to assign documents to predefined classes, and learning how to evaluate how well these work.

This is the last major technique we will learn, before the penultimate session where we will find about how to apply a range of the techniques we have seen using the fanciest most up-to-date methods.

Homework assignment (optional)

Take a protocol from the Bundestag and process it as you did in Assignment 1

Using a German sentiment lexicon, calculate a sentiment score for each speech.

Show the average sentiment score for each party.

Show the 5 speeches with the most positive sentiment, and the 5 speeches with the most negative sentiment.