# Big Data and Automated Content Analysis

## Week 4 – Thursday

## » Sentiment Analysis «

Anne Kroon

a.c.kroon@uva.nl
@annekroon

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

April 25, 2019

# Today

1. API assignment last week
2. Different types of analysis
   - What can we do?
   - Systematizing analytical approaches
3. Data analysis 1: Sentiment analysis
   - What is it?
   - Bag-of-words approaches
   - Advanced approaches
   - A sentiment analysis tailored to your needs!
   - Packages for sentiment analysis
   - A recipe
   - Machine Learning as alternative
4. Take-home message, next meetings, & exam

# Creating an URL to make an API request

```
1  import requests
2  import json
3
4  base_url = 'https://www.rijksmuseum.nl/api/pages/en/{}?key={}&format=
       json'
5  page_of_interest ='whats-on/exhibitions-past'
6  key = 'YOURKEY'
7
8  full_url = base_url.format(page_of_interest, key)
```

# Creating an URL to make an API request

```
1  import requests
2  import json
3
4  base_url = 'https://www.rijksmuseum.nl/api/pages/en/{}?key={}&format=
       json'
5  page_of_interest ='whats-on/exhibitions-past'
6  key = 'YOURKEY'
7
8  full_url = base_url.format(page_of_interest, key)
```

NB: the built-in `.format()` method returns a formatted value / representation. More specifically, it replaces the '{}' in a string by the argument specified between '()'

# Creating an URL to make an API request

```
1  import requests
2  import json
3
4  base_url = 'https://www.rijksmuseum.nl/api/pages/en/{}?key={}&format=
       json'
5  page_of_interest ='whats-on/exhibitions-past'
6  key = 'YOURKEY'
7
8  full_url = base_url.format(page_of_interest, key)
```

NB: the built-in `.format()` method returns a formatted value / representation. More specifically, it replaces the '{}' in a string by the argument specified between '()'

NB: assign your unique API-key requested to the variable `key`

# Making an API request

```
r = requests.get(full_url)
print('Response HTTP Status Code: {}'.format(r.status_code))

data = json.loads(r.content.decode('utf-8'))
```

# Making an API request

```
r = requests.get(full_url)
print('Response HTTP Status Code: {}'.format(r.status_code))

data = json.loads(r.content.decode('utf-8'))
```

NB: Response HTTP Status Code: 200 means success

# Making an API request

```
from collections import defaultdict
import dateutil.parser
counts = defaultdict(int)

for item in data['contentPage']['overviewItems']:
    end_date = ' '.join(item['text'].split(' ')[-3:])
    dt = dateutil.parser.parse(end_date)
    counts[dt.month] += 1
print(counts)
```

# Making an API request

```
from collections import defaultdict
import dateutil.parser
counts = defaultdict(int)

for item in data['contentPage']['overviewItems']:
    end_date = ' '.join(item['text'].split(' ')[-3:])
    dt = dateutil.parser.parse(end_date)
    counts[dt.month] += 1
print(counts)
```

NB: you can parse dates with `dateutil.parser`

# Now its your turn!

How to get started?
Search and consult online documentation on using APIs (e.g., on the organization's webpage, github page).
Consult Stackoverflow or other online sources for help.

# Reddit API

```
1  import praw
2  import pandas as pd
3  import datetime as dt
4
5  client_id = '****'
6  client_secret = '****'
7  user_agent = '****'
8  username = '****'
9  password = '****'
10
11 reddit = praw.Reddit(client_id=client_id, client_secret=client_secret,
       user_agent=user_agent, username=username, password=password)
```

# Reddit API

```
1  import praw
2  import pandas as pd
3  import datetime as dt
4
5  client_id = '****'
6  client_secret = '****'
7  user_agent = '****'
8  username = '****'
9  password = '****'
10
11 reddit = praw.Reddit(client_id=client_id, client_secret=client_secret,
       user_agent=user_agent, username=username, password=password)
```

This is very helpful:
http://www.storybench.org/how-to-scrape-reddit-with-python/

# Reddit API

```
1  import praw
2  import pandas as pd
3  import datetime as dt
4
5  client_id = '****'
6  client_secret = '****'
7  user_agent = '****'
8  username = '****'
9  password = '****'
10
11 reddit = praw.Reddit(client_id=client_id, client_secret=client_secret,
       user_agent=user_agent, username=username, password=password)
```

This is very helpful:
http://www.storybench.org/how-to-scrape-reddit-with-python/

and this: https://towardsdatascience.com/scrape-reddit-data-using-python-and-google-bigquery-44180b579892

# Reddit API

```
1  reddit_dict = { "title":[], "score":[], "id":[], "url":[], "comms_num":
       [], "created": [], "body":[]}
2
3  for submission in reddit.subreddit('learnprogramming').hot(limit=15):
4      reddit_dict["title"].append(submission.title)
5      reddit_dict["score"].append(submission.score)
6      reddit_dict["id"].append(submission.id)
7      reddit_dict["url"].append(submission.url)
8      reddit_dict["comms_num"].append(submission.num_comments)
9      reddit_dict["created"].append(submission.created)
10     reddit_dict["body"].append(submission.selftext)
```

# Reddit API

```
reddit_dict = { "title":[], "score":[], "id":[], "url":[], "comms_num":
    [], "created": [], "body":[]}

for submission in reddit.subreddit('learnprogramming').hot(limit=15):
    reddit_dict["title"].append(submission.title)
    reddit_dict["score"].append(submission.score)
    reddit_dict["id"].append(submission.id)
    reddit_dict["url"].append(submission.url)
    reddit_dict["comms_num"].append(submission.num_comments)
    reddit_dict["created"].append(submission.created)
    reddit_dict["body"].append(submission.selftext)
```

https://praw.readthedocs.io/en/latest/code_overview/models/subreddit.html

# What we already can do

## with regard to data collection:

- query a (JSON-based) API (GoogleBooks, Twitter)
- handle CSV files
- handle JSON files

# What we already can do

## with regard to data collection:

- query a (JSON-based) API (GoogleBooks, Twitter)
- handle CSV files
- handle JSON files

## with regard to analysis:

Not much. We counted some frequencies and calculated some averages.

# Data analysis: Overview
# What can we do?

Data analysis: Overview
What can we do?

**What do you think? What are interesting methods to analyze large data sets (like, e.g., social media data? What questions can they answer?)**

# What else can we do?

## For example

- sentiment analysis
- automated coding with regular expressions
- natural language processing
- supervised and unsupervised machine learning
- network analysis

# What else can we do?

## Or ideally...

...a combination of these techniques.

**Overview**

Systematizing analytical approaches

# Systematizing analytical approaches

Taking the example of Twitter:

## Analyzing the *structure*

- Number of Tweets over time
- singleton/retweet ratio
- Distribution of number of Tweets per user
- Interaction networks

Bruns, A., & Stieglitz, S. (2013). Toward more systematic Twitter analysis: metrics for tweeting activities. *International Journal of Social Research Methodology*. doi:10.1080/13645579.2012.756095

# Systematizing analytical approaches

Taking the example of Twitter:

## Analyzing the *structure*

- Number of Tweets over time
- singleton/retweet ratio
- Distribution of number of Tweets per user
- Interaction networks

⇒ **Focus on the amount of content and on the question who interacts with whom, not on what is said**

Bruns, A., & Stieglitz, S. (2013). Toward more systematic Twitter analysis: metrics for tweeting activities. *International Journal of Social Research Methodology*. doi:10.1080/13645579.2012.756095

# Systematizing analytical approaches

Taking the example of Twitter:

## Analyzing the *content*

- Sentiment analysis
- Word frequencies, searchstrings
- Co-word analysis ($\Rightarrow$frames)

# Systematizing analytical approaches

Taking the example of Twitter:

## Analyzing the *content*

- Sentiment analysis
- Word frequencies, searchstrings
- Co-word analysis ($\Rightarrow$frames)

$\Rightarrow$ **Focus on what is said**

# Systematizing analytical approaches

$\Rightarrow$ **It depends on your research question which approach is more interesting!**

# Automated Content Analysis

| | **Methodological approach** | | |
|---|---|---|---|
| | *Counting and Dictionary* | *Supervised Machine Learning* | *Unsupervised Machine Learning* |
| **Typical research interests and content features** | visibility analysis<br>sentiment analysis<br>subjectivity analysis | frames<br>topics<br>gender bias | frames<br>topics |
| **Common statistical procedures** | string comparisons<br>counting | support vector machines<br>naive Bayes | principal component analysis<br>cluster analysis<br>latent dirichlet allocation<br>semantic network analysis |


Boumans, J.W., & Trilling, D. (2016). Taking stock of the toolkit: An overview of relevant automated content analysis approaches and techniques for digital journalism scholars. *Digital Journalism, 4*, 1. 8–23.

Data analysis 1:
**Sentiment analysis**

# What is sentiment analysis?

## Extracting subjective information from texts

# What is sentiment analysis?

## Extracting subjective information from texts

- the author's attitude towards the topic of the text

# What is sentiment analysis?

## Extracting subjective information from texts

- the author's attitude towards the topic of the text
- *polarity*: negative—positive

# What is sentiment analysis?

## Extracting subjective information from texts

- the author's attitude towards the topic of the text
- *polarity*: negative—positive
- *subjectivity*: neutral—subjective *

# What is sentiment analysis?

## Extracting subjective information from texts

- the author's attitude towards the topic of the text
- *polarity*: negative—positive
- *subjectivity*: neutral—subjective *
- advanced approaches: different emotions

# What is sentiment analysis?

## Extracting subjective information from texts

- the author's attitude towards the topic of the text
- *polarity*: negative—positive
- *subjectivity*: neutral—subjective *
- advanced approaches: different emotions

* Less sophisticated approaches do not see this as a seperate dimension but simply calculate $objectivity = 1 - (negativity + positivity)$

# Example

```
>>> sentiment("Great service by @NSHighspeed")
(0.8, 0.75)
>>> sentiment("Bad service by @NSHighspeed")
(-0.6166666666666667, 0.6666666666666666)
```

(polarity, subjectivity) with

$-1 \leq \textit{polarity} \leq +1$
$0 \leq \textit{subjectivity} \leq +1$ )

This is the module pattern.nl
De Smedt, T., & Daelemans W. (2012). Pattern for Python. *Journal of Machine Learning Research, 13*, 2063-2067.

# Applications

## Who uses it?

- Companies
- especially for Web Analytics
- Social Scientists
- applications in data journalism, politics, ...

Many references to examples in Mostafa (2013).

⇒ Cases in which you have a huge amount of data or real-time data and you want to get an idea of the tone.

Mostafa, M. M. (2013). More than words: Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications, 40*(10), 4241– 4251. doi:10.1016/j.eswa.2013.01.019

# Data analysis 1: Sentiment analysis
# Bag-of-words approaches

# Bag-of-words approaches

## How does it work?

- We take each word of a text and look if it's positive or negative.

# Bag-of-words approaches

## How does it work?

- We take each word of a text and look if it's positive or negative.
  - Most simple way: compare it with a list of negative words and with a list of positive words (That's what Mostafa (2013) did)

# Bag-of-words approaches

## How does it work?

- We take each word of a text and look if it's positive or negative.
  - Most simple way: compare it with a list of negative words and with a list of positive words (That's what Mostafa (2013) did)
  - More advanced: look up a subjectivity score from a table

# Bag-of-words approaches

## How does it work?

- We take each word of a text and look if it's positive or negative.
  - Most simple way: compare it with a list of negative words and with a list of positive words (That's what Mostafa (2013) did)
  - More advanced: look up a subjectivity score from a table
- e.g., add up the scores and average them.

# How to do this

If you were to run an analyis like the one by Mostafa (2013), how could you do this?

# How to do this

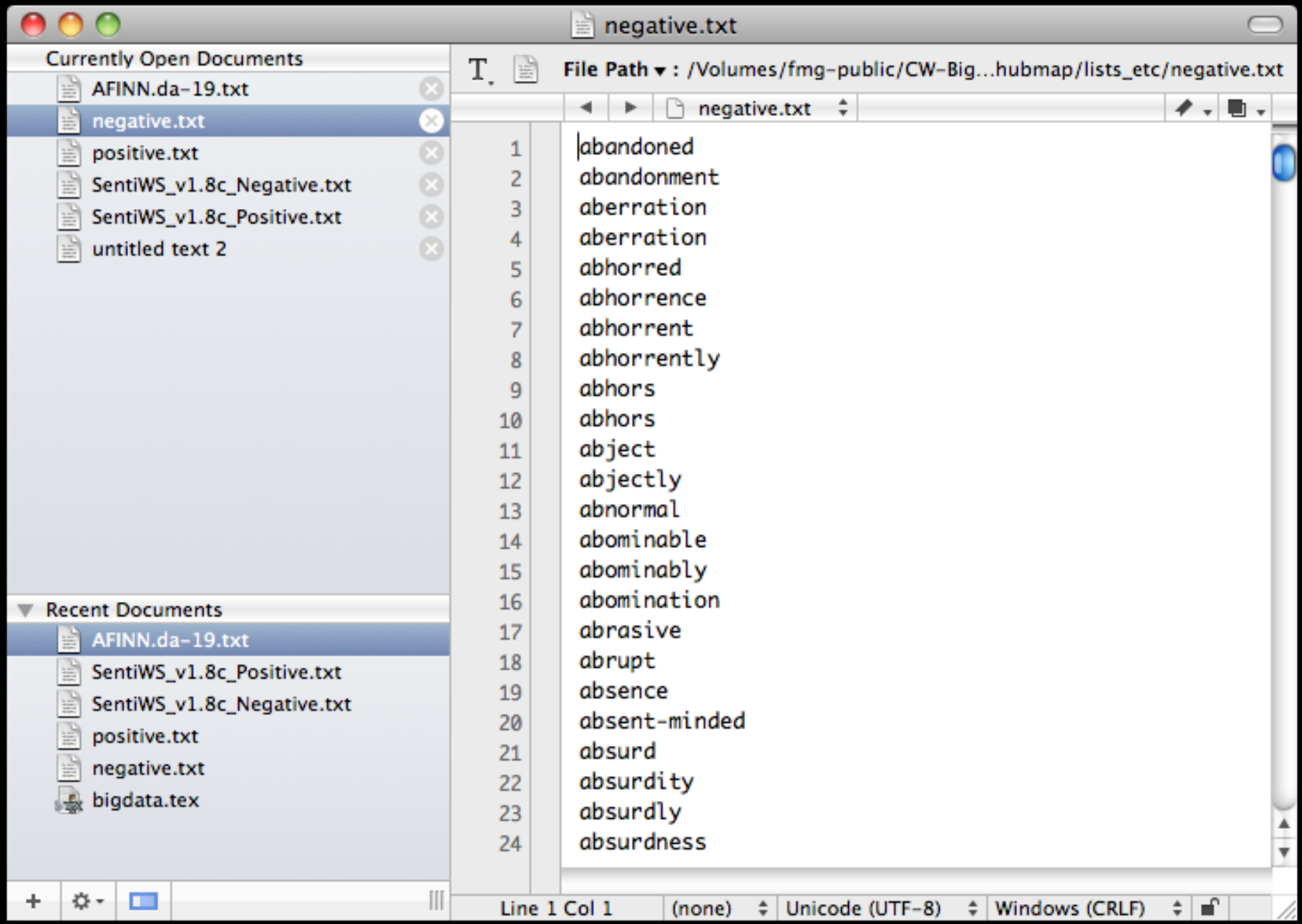
(given a *string* `tekst` that you want to analyze and two *lists* of strings with negative and positive words, `lijstpos=["great","fantastic",...,"perfect"]` and `lijstneg`)

```
sentiment=0
for woord in tekst.split():
    if woord in lijstpos:
        sentiment=sentiment+1  #same as sentiment+=1
    elif woord in lijstneg:
        sentiment=sentiment-1  #same as sentiment-=1
print (sentiment)
```

# Do we need to have the lists in our program itself?

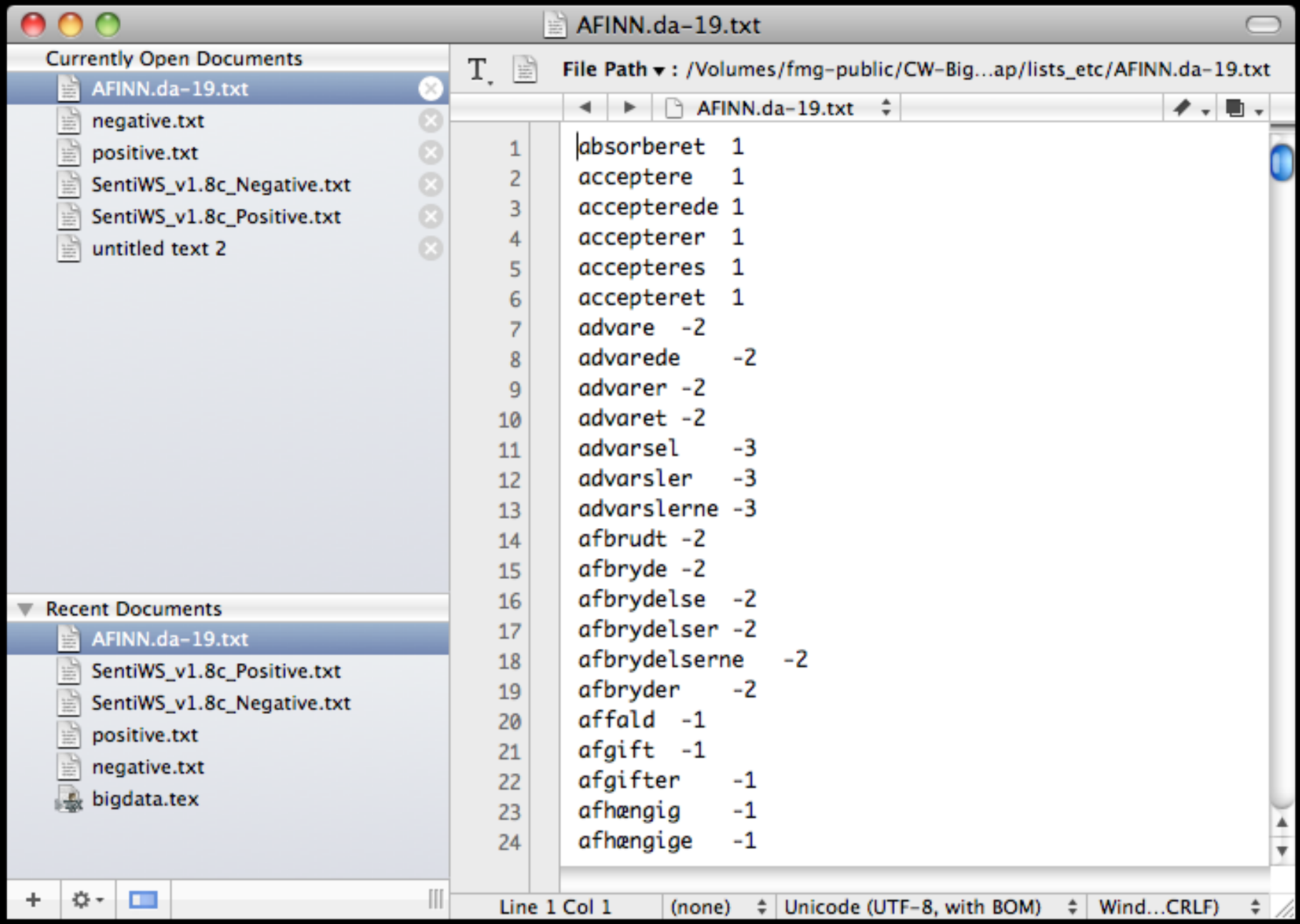
No.
You could have them in a separate text file, one per row, and then read that file directly to a list.

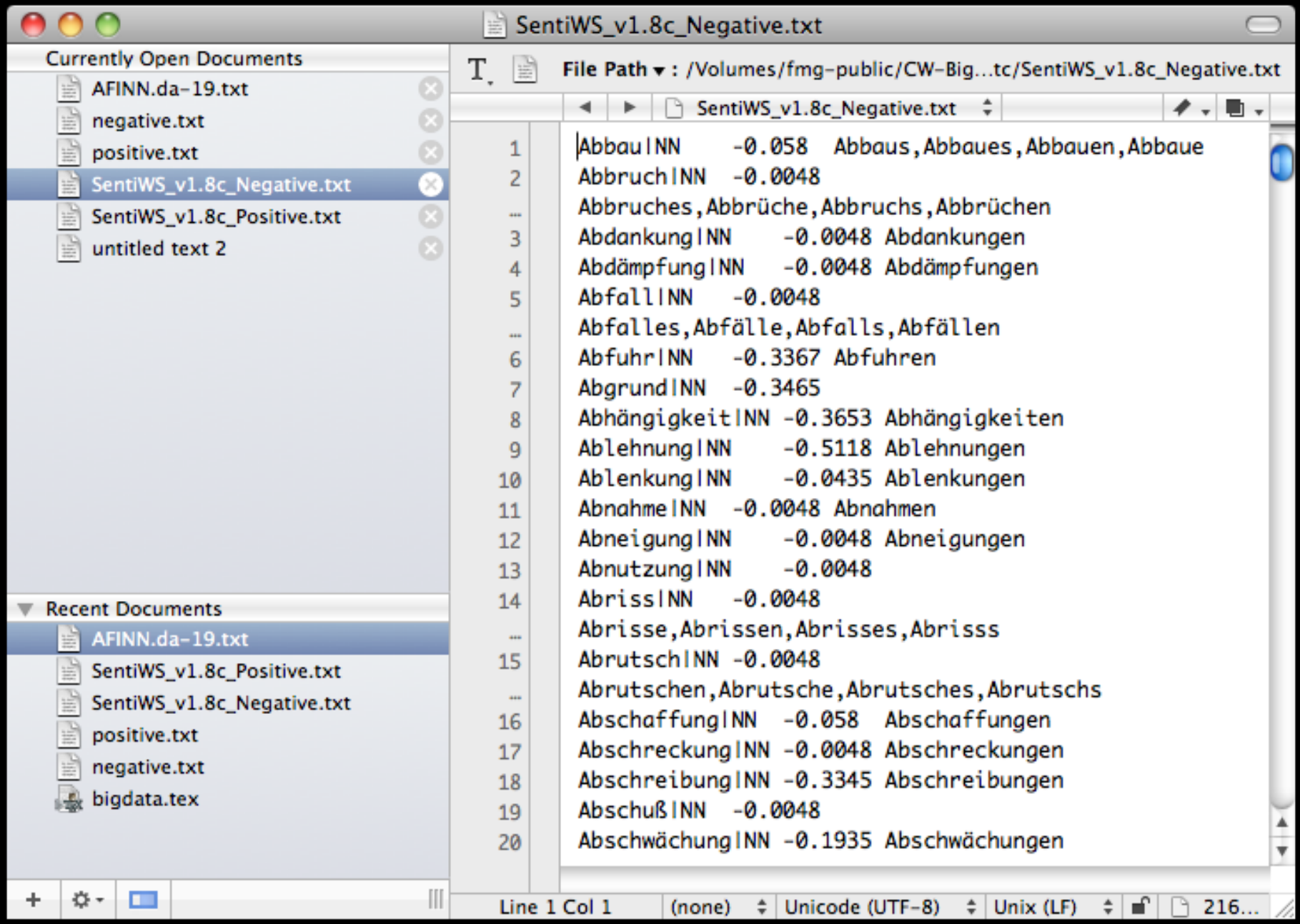
```
1 poslijst=open("filewithonepositivewordperline.txt").read().splitlines()
2 neglijst=open("filewithonenegativewordperline.txt").read().splitlines()
```


negative.txt
Currently Open Documents
AFINN.da-19.txt
negative.txt
positive.txt
SentiWS_v1.8c_Negative.txt
SentiWS_v1.8c_Positive.txt
untitled text 2
Recent Documents
AFINN.da-19.txt
SentiWS_v1.8c_Positive.txt
SentiWS_v1.8c_Negative.txt
positive.txt
negative.txt
bigdata.tex
File Path : /Volumes/fmg-public/CW-Big...hubmap/lists_etc/negative.txt
negative.txt
1 abandoned
2 abandonment
3 aberration
4 aberration
5 abhorred
6 abhorrence
7 abhorrent
8 abhorrently
9 abhors
10 abhors
11 abject
12 abjectly
13 abnormal
14 abominable
15 abominably
16 abomination
17 abrasive
18 abrupt
19 absence
20 absent-minded
21 absurd
22 absurdity
23 absurdly
24 absurdness
Line 1 Col 1
(none)
Unicode (UTF-8)
Windows (CRLF)

# More advanced versions

- CSV files or similar tables with weights
- Or some kind of dict?


AFINN.da-19.txt
Currently Open Documents
AFINN.da-19.txt
negative.txt
positive.txt
SentiWS_v1.8c_Negative.txt
SentiWS_v1.8c_Positive.txt
untitled text 2
Recent Documents
AFINN.da-19.txt
SentiWS_v1.8c_Positive.txt
SentiWS_v1.8c_Negative.txt
positive.txt
negative.txt
bigdata.tex
File Path : /Volumes/fmg-public/CW-Big...ap/lists_etc/AFINN.da-19.txt
AFINN.da-19.txt
absorberet 1
acceptere 1
accepterede 1
accepterer 1
accepteres 1
accepteret 1
advare -2
advarede -2
advarer -2
advaret -2
advarsel -3
advarsler -3
advarslerne -3
afbrudt -2
afbryde -2
afbrydelse -2
afbrydelser -2
afbrydelserne -2
afbryder -2
affald -1
afgift -1
afgifter -1
afhængig -1
afhængige -1
Line 1 Col 1
(none)
Unicode (UTF-8, with BOM)
Wind...CRLF)


SentiWS_v1.8c_Negative.txt
Currently Open Documents
AFINN.da-19.txt
negative.txt
positive.txt
SentiWS_v1.8c_Negative.txt
SentiWS_v1.8c_Positive.txt
untitled text 2
Recent Documents
AFINN.da-19.txt
SentiWS_v1.8c_Positive.txt
SentiWS_v1.8c_Negative.txt
positive.txt
negative.txt
bigdata.tex
File Path : /Volumes/fmg-public/CW-Big...tc/SentiWS_v1.8c_Negative.txt
SentiWS_v1.8c_Negative.txt
1 Abbau|NN -0.058 Abbaus,Abbaues,Abbauen,Abbaue
2 Abbruch|NN -0.0048
... Abbruches,Abbrüche,Abbruchs,Abbrüchen
3 Abdankung|NN -0.0048 Abdankungen
4 Abdämpfung|NN -0.0048 Abdämpfungen
5 Abfall|NN -0.0048
... Abfalles,Abfälle,Abfalls,Abfällen
6 Abfuhr|NN -0.3367 Abfuhren
7 Abgrund|NN -0.3465
8 Abhängigkeit|NN -0.3653 Abhängigkeiten
9 Ablehnung|NN -0.5118 Ablehnungen
10 Ablenkung|NN -0.0435 Ablenkungen
11 Abnahme|NN -0.0048 Abnahmen
12 Abneigung|NN -0.0048 Abneigungen
13 Abnutzung|NN -0.0048
14 Abriss|NN -0.0048
... Abrisse,Abrissen,Abrisses,Abrisss
15 Abrutsch|NN -0.0048
... Abrutschen,Abrutsche,Abrutsches,Abrutschs
16 Abschaffung|NN -0.058 Abschaffungen
17 Abschreckung|NN -0.0048 Abschreckungen
18 Abschreibung|NN -0.3345 Abschreibungen
19 Abschuß|NN -0.0048
20 Abschwächung|NN -0.1935 Abschwächungen
Line 1 Col 1
(none)
Unicode (UTF-8)
Unix (LF)
216...

# Mustafa 2013: Interpreting the output

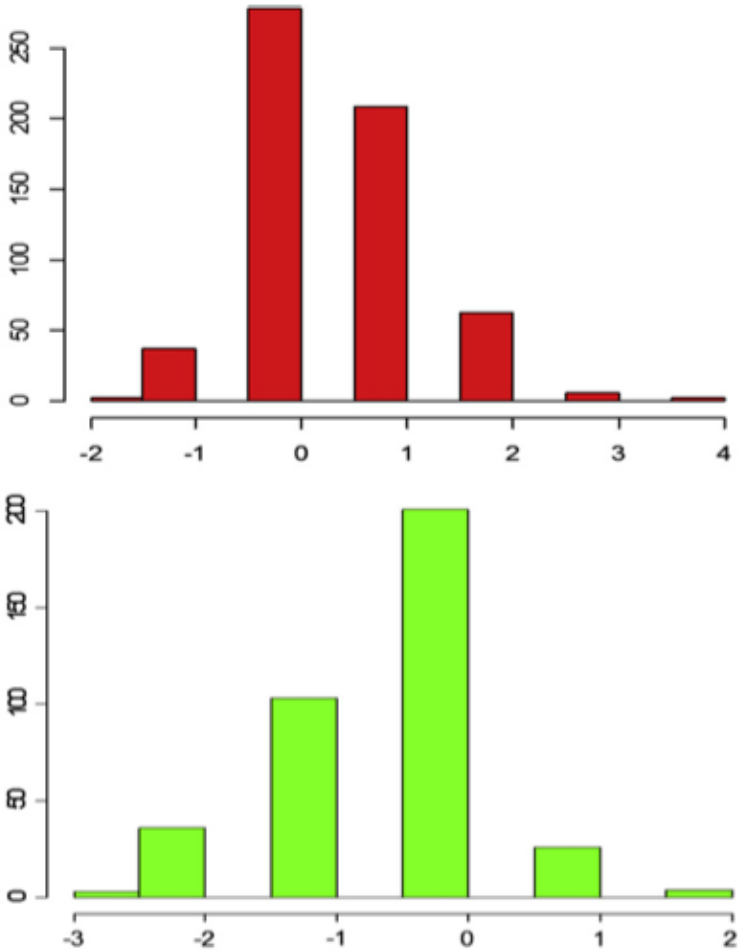



**Fig. 5.** Sentiment scores for Nokia (top) and Pfizer (bottom). *X*-axis represents score distributions, *Y*-axis represents count/frequencies.

# Mustafa 2013: Interpreting the output

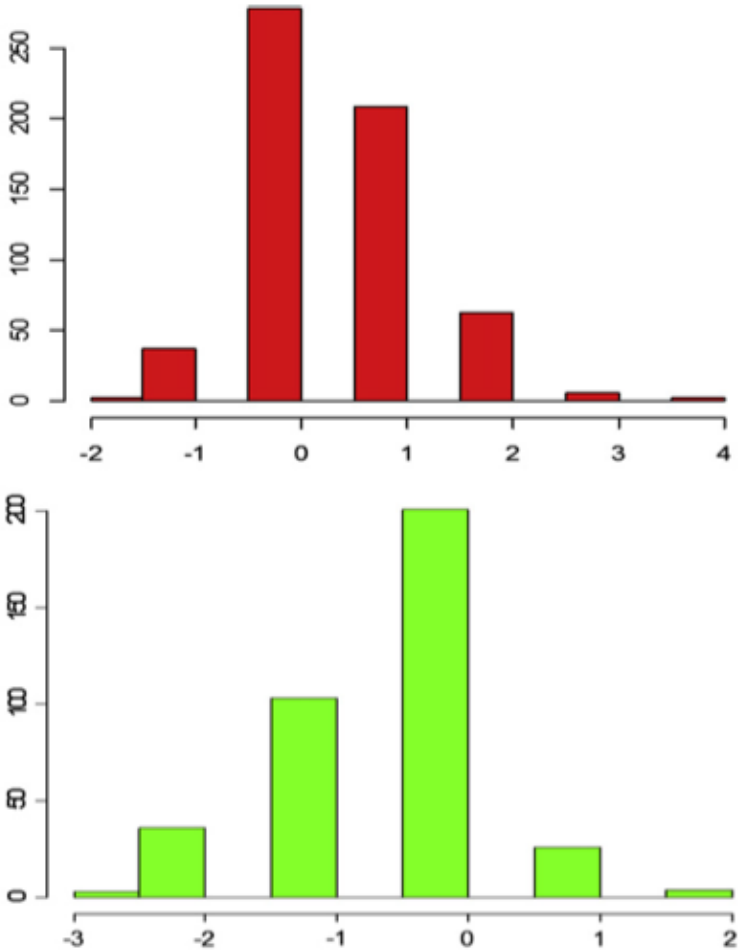



**Fig. 5.** Sentiment scores for Nokia (top) and Pfizer (bottom). *X*-axis represents score distributions, *Y*-axis represents count/frequencies.

Your thoughts?

# Mustafa 2013: Interpreting the output

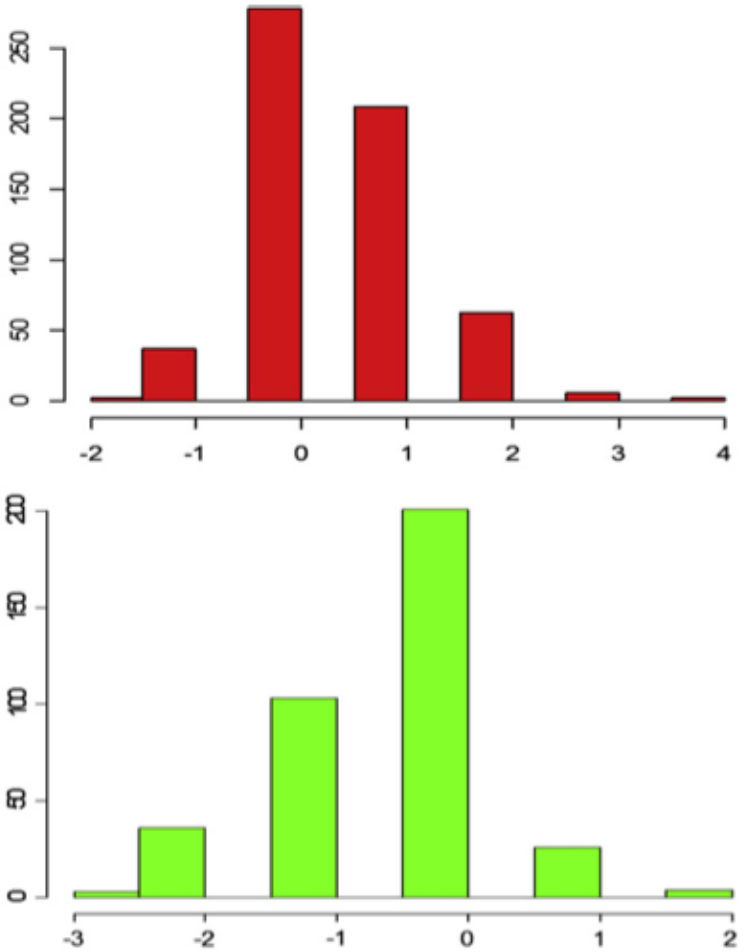



**Fig. 5.** Sentiment scores for Nokia (top) and Pfizer (bottom). *X*-axis represents score distributions, *Y*-axis represents count/frequencies.

Your thoughts?

- each word counts equally (1)
- many tweets contain no words from the list. What does this mean?
- Ways to improve BOW approaches?

# Bag-of-words approaches

e.g., Schut, L. (2013). Verenigde Staten vs. Verenigd Koningrijk: Een automatische inhoudsanalyse naar verklarende factoren voor het gebruik van positive campaigning en negative campaigning door vooraanstaande politici en politieke partijen op Twitter. *Bachelor Thesis*, Universiteit van Amsterdam.

# Bag-of-words approaches

## pro

- easy to implement
- easy to modify:
  - add or remove words
  - make new lists for other languages, other categories (than positive/negative), ...
- easy to understand (transparency, reproducability)

e.g., Schut, L. (2013). Verenigde Staten vs. Verenigd Koningrijk: Een automatische inhoudsanalyse naar verklarende factoren voor het gebruik van positive campaigning en negative campaigning door vooraanstaande politici en politieke partijen op Twitter. *Bachelor Thesis*, Universiteit van Amsterdam.

# Bag-of-words approaches

## con

- simplistic assumptions
- e.g., intensifiers cannot be interpreted ("really" in "really good" or "really bad")
- or, even more important, negations.

# Data analysis 1: Sentiment analysis
# Advanced approaches

# Improving the BOW approach

## Example: The Sentistrenght algorithm

- $-5 \ldots -1$ and $+1 \ldots +5$
- spelling correction
- "booster word list" for strengthening/weakening the effect of the following word
- interpreting repeated letters ("baaaaaad"), CAPITALS and !!!
- idioms
- negation
- ...

Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Sentiment strength detection for the social Web. *Journal of the American Society for Information Science and Technology, 63*(1), 163-173.

```
abandon*        -2      liwc uness specified otherwise
abate   -2      General Inquirer Feb 2010
abdicate*       -2      General Inquirer Feb 2010
abhor*  -4      General Inquirer Feb 2010
abject  -2      General Inquirer Feb 2010
abnormal*       -2      General Inquirer Feb 2010
abolish*        -2      General Inquirer Feb 2010
abomina*        -3      General Inquirer Feb 2010
abrasive*       -2      General Inquirer Feb 2010
abrupt  -2      General Inquirer Feb 2010
abscond*        -2      General Inquirer Feb 2010
absence -2      General Inquirer Feb 2010
absent* -2      General Inquirer Feb 2010
absurd* -2      Feb-11
abuse*  -4
abusi*  -4      removed accept  1       accepta*        2       accepted        2       accepting       2       accepts 2
abyss   -2      General Inquirer Feb 2010
accident*       -2      General Inquirer Feb 2010
accomplish*     2       Hannes GI add
accost* -2      General Inquirer Feb 2010
accursed        -2      General Inquirer Feb 2010
accus*  -2      General Inquirer Feb 2010
accusation*     -2      General Inquirer Feb 2010
ache*   -2
achen*  1       kev
acher*  1       kev
acheson 1       kev
acheta  1       kev
aching  -2      removed active* 2
acrimon*        -2      General Inquirer Feb 2010
addict* -2      General Inquirer Feb 2010
```

25 lines (25 sloc) | 191 Bytes

```
aren't
arent
can't
cannot
cant
couldn't
couldnt
didn't
didnt
doesn't
doesnt
don't
dont
hasn't
hasnt
isn't
isnt
never
not
shouldn't
shouldnt
won't
wont
wouldn't
wouldnt
```

# Advanced approaches

## Take the structure of a text into account

- Try to apply linguistics concepts to identify sentence structure
- can identify negations
- can interpret intensifiers

# Example

```
from pattern.nl import sentiment
>>> sentiment("Great service by @NSHighspeed")
(0.8, 0.75)
>>> sentiment("Really")
(0.0, 1.0)
>>> sentiment("Really Great service by @NSHighspeed")
(1.0, 1.0)
```

(polarity, subjectivity) with
$-1 \leq \textit{polarity} \leq +1$
$0 \leq \textit{subjectivity} \leq +1$ )

Unlike in pure bag-of-words approaches, here, the overall sentiment is not just the sum or the average of its parts!

De Smedt, T., & Daelemans W. (2012). Pattern for Python. *Journal of Machine Learning Research, 13*, 2063-2067.

# Advanced approaches

# Advanced approaches

## pro

- understand intensifiers or negation
- thus: higher accuracy

# Advanced approaches

## pro

- understand intensifiers or negation
- thus: higher accuracy

## con

- Black box? Or do we understand the algorithm?
- Difficult to adapt to own needs
- *really* much better results?

# Data analysis 1: Sentiment analysis
# A sentiment analysis tailored to your needs!

# A sentiment analysis tailored to your needs!

## Identifying suicidal texts

- Bag-of-words-approach with very specific dictionary
- added negation
- added regular expression search for key phrases
- Very specific design requirements: False positives are OK, false negatives not!

Huang, Y.-P., Goh, T., & Liew, C.L. (2007). Hunting suicide notes in web 2.0 – preliminary findings. *Ninth IEEE International Symposium on Multimedia*. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4476021

Already this still relatively simple approach seems to work satisfactory, but if 106 scientists from 24 competing teams (!) work on it, they can

Pestian, J.P.; Matykiewicz, P., Linn-Gust, M., South, B., Uzuner, O., Wiebe, J., Cohen, K.B., Hurdle, J., & Brew, C. (2012). Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights, 5*(1), p. 3-16. Retrieved from http://europepmc.org/articles/PMC3299408?pdf=render

Already this still relatively simple approach seems to work satisfactory, but if 106 scientists from 24 competing teams (!) work on it, they can

## group suicide notes by these characteristics:

- swear
- family
- friend
- positive emotion
- negative emotion
- anxiety
- anger
- sad
- cognitive process
- biology
- sexual
- ingestion
- religion
- death

Pestian, J.P.; Matykiewicz, P., Linn-Gust, M., South, B., Uzuner, O., Wiebe, J., Cohen, K.B., Hurdle, J., & Brew, C. (2012). Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights, 5*(1), p. 3-16. Retrieved from http://europepmc.org/articles/PMC3299408?pdf=render

Packages for sentiment analysis

# Which packages are easy to use?

vader pro: in NLTK module, con: English only

# Which packages are easy to use?

vader pro: in NLTK module, con: English only

pattern pro: multiple languages (including Dutch)

# Which packages are easy to use?

vader pro: in NLTK module, con: English only

pattern pro: multiple languages (including Dutch)

sentistrength pro: multiple languages, widely used, con: needs Python wrapper, license

vader: Chapter 6.3; pattern: Chapter 6.5; sentistrength: Chapter 6.4

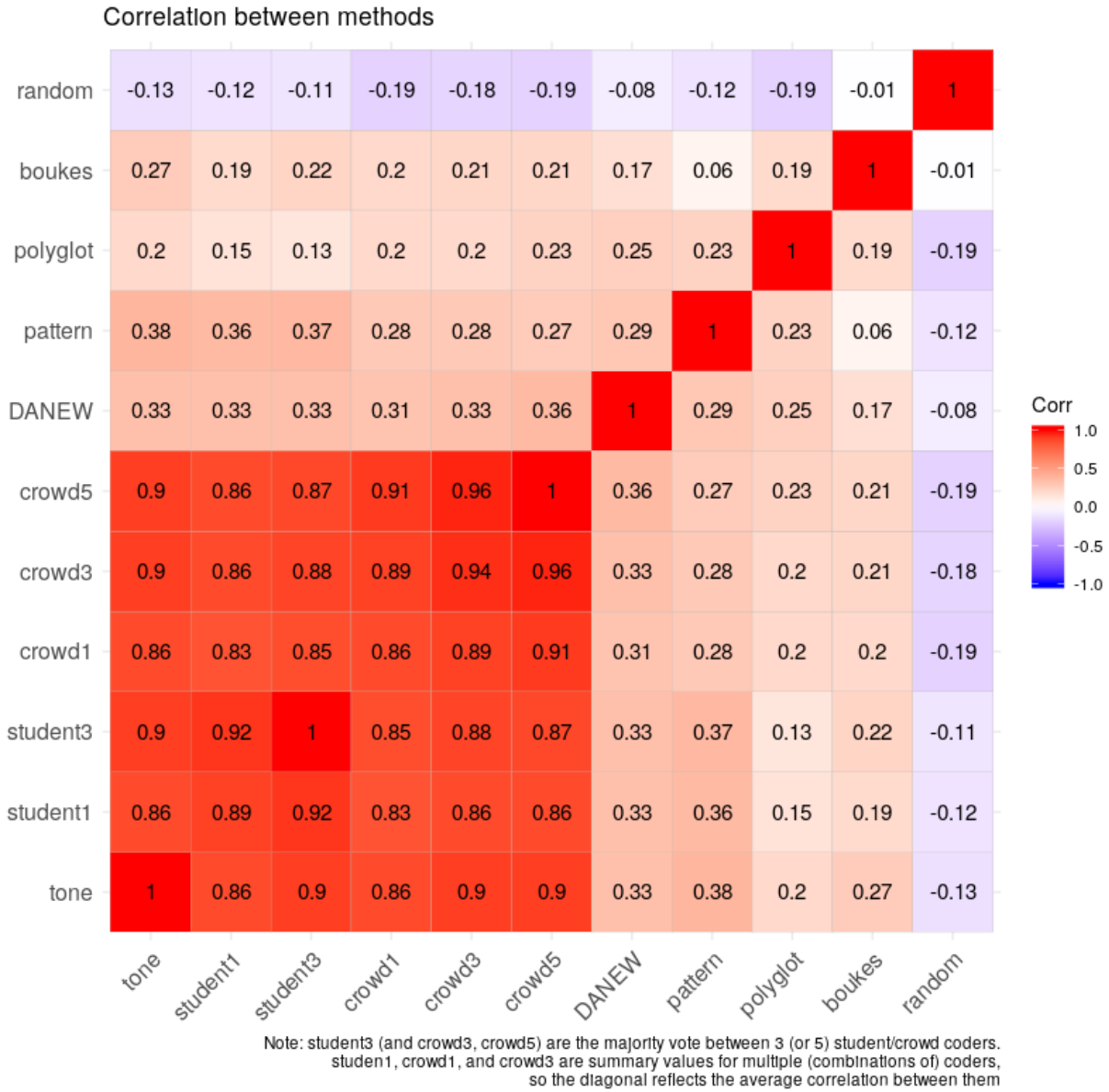
# Which packages are easy to use?

| | |
|---|---|
| vader | pro: in NLTK module, con: English only |
| pattern | pro: multiple languages (including Dutch) |
| sentistrength | pro: multiple languages, widely used, con: needs Python wrapper, license |

vader: Chapter 6.3; pattern: Chapter 6.5; sentistrength: Chapter 6.4

**BUT: Keep in mind that the results of *any* off-the-shelf-package might be biased and/or noisy in *your* domain!**

## Packages for sentiment analysis




Boukes, M., van der Velde, R.N., & Vliegenthart, R. (2018). The good and bad in economic news: Comparing (automatic) measurements of sentiment in Dutch economic news. *International Communication Association*

# A possible recipe for doing your sentiment analysis

1. Construct a list `data` of strings with your input data
2. Create an empty list `sent` for storing the results
3. For each text `t` in `data`, estimate the sentiment of `t` and append the result to `sent`[1]
4. Confirm that `len(data) == len(sent)`
5. use `zip()` and a `csv.writer` to write input and output next to each other to a csv file.

[1] use multiple lists instead if you estimate for instance subjectivity *and* polarity

# Superivsed ML ($\Rightarrow$ week 7)

An alternative state-of-the-art approach:

## Use supervised machine learning

- Instead of defining rules, hand-code ("annotate") the sentiment of some tweets manually and let the computer find out which words or characters ("features") predict sentiment
- Then use this model to predict sentiment for other tweets
- Essentially the same like what you know since the second year of your Bachelor: regression analysis (but now with DV sentiment and IV's word occurrences)

Gonzalez-Bailon, S., & Paltoglou, G. (2015). Signals of public opinion in online communication: A comparison of methods and data sources. *The ANNALS of the American Academy of Political and Social Science, 659*(1), 95-–107.

Take-home message
Mid-term take-home exam
Next meetings

# Take-home messages

## What you should be familiar with:

- You should have *completely* understood last week's exercise. Re-read it if neccessary.
- Approaches to the analysis (e.g., structure vs. content)
- Types of sentiment analysis, application areas, pros and cons

# Mid-term take home exam

## Week 5: Thursday, 2 May, to Tuesday, 6 May

- You get the exam on Thursday at the end of the meeting
- Answers have to be handed in no later than Tuesday evening, 23.59
- 30% of final grade
- 3 questions:
  1. Literature question: E.g., different methods ("Explain how... is done") and/or epistemological or theoretical implications ("What does this mean for social-scientific research?")
  2. Empirical question (conceptual)
  3. Empirical question (actual programming task)

If you *fully* understood all exercises until now, it shouldn't be difficult and won't take too long. But give yourself *a lot* of buffer time!!!

# Next meetings

## Monday, 29–4

Task for during the meeting: Conduct an own sentiment analysis! You can bring your own data (you will probably learn more!), but then already think about how to write some script to read the data (as we did last week and as described in Chapter 5).