Week 14 Text Mining

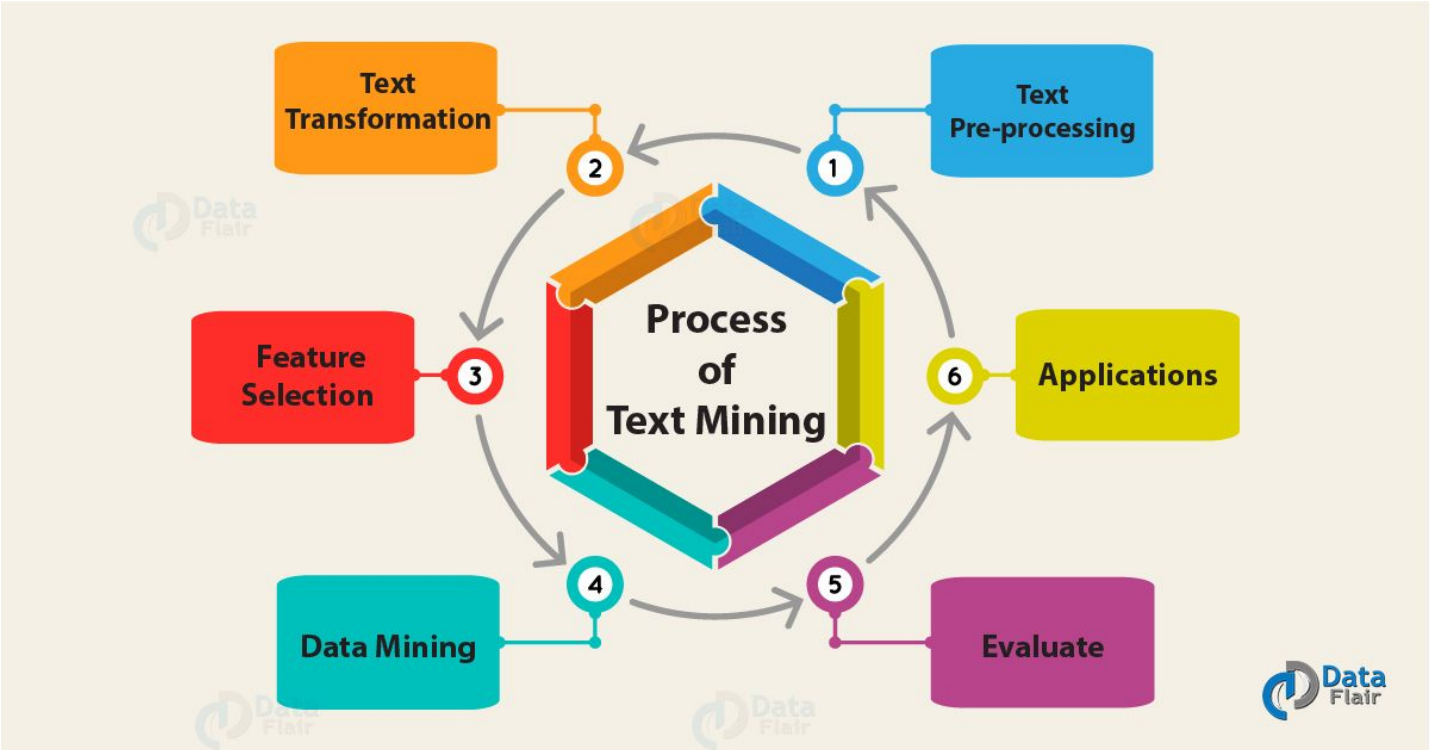
Theory and Practice

Ying Lin, Ph.D

Text Mining vs. Data Mining

- Text mining deals with unstructure data (text). Also known as text analytics.
- Goal: the process or practice of examining large collections of written resources in order to generate new and relevant information
- Conversion from text data to record dataset and then apply all the data mining techniques covered in the course

Text Mining Workflow



3/33

Natural Language Processing (NLP)

- A component of text mining that performs a special kind of linguistic analysis that helps a machine text and decipher the ambiguities in human langauge
  - automatic summarization, Part-Of-Speech (POS) tagging, entity extraction, relations extraction
  - requires a consistent knowledge basis: detailed thesaurus, linguistic and grammatic rules, ontology and up-to-date entities
- Natural language vs. programming language
- Challenges of understanding natural languages (grammar complexity)
  - Type of words
  - Word play
  - Ambiguity
  - Voice recognition: accent, tone

Difference between Text Mining and NLP

- Goal
- Method
  - Text mining is shallow and does not consider the text structure (sequence) and context
  - Bag of words, n-grams, and stemming
  - NLP: consider the text structure and sequence
  - sentence splitting, part of speech tagging, parse tree construction

Applications

- Chatbot and dialogue systems
- Virtual assistant
- Topic modeling
- Text classifications
- Speech synthesis
- Voice/speech recognition

Bag-of-Word (BoW) Model (1)

- Representing text data and extracting features when modeling text with machine learning algorithms
    - A vocabulary of known words
    - A measure of the presence of known words
    - Order and structure of words is discarded. Alternative is semantics parsing.
- Process
    - Collect text data
    - Design the vocabulary
    - Create document vectors and score the words (Document-Term Matrix or DTM)

Bag-of-Word (BoW) Model (2)

- Tokenization: break down sentences into words and tokens
  - Curse of high dimensionality
  - Stop word removal: those words that occur in almost everywhere and don’t significantly differentiate among sentences, such as articles, pronoun, etc.
  - Stemming: replace with the root form of words
  - Casing: remove the case difference
  - n-gram: bigram and trigram
- Limitations of BoW
  - High dimensionality
  - Sparse representation
  - Context agnostic

8/33

TF-IDF Transformation for DTM

- TF (Term Frequency): frequency of the term (t) adjusted for the document (d) length

$$tf(t,d) = \frac{f_t}{f_d}$$

- IDF (Inversed Document Frequency): logarithmically scaled inverse fraction of the collection (D) of documents (d) that contain the term (t). N is the total number of documents in the corpus ($N = |D|$).

$$idf(t,D) = \log \frac{N}{|d \in D : t \in d|}$$

- TF-IDF

$$tf - idf(t,d,D) = tf(t,d) * idf(t,D)$$

Cosine Similarity/Distance

- Cosine similarity (orientation/direction/angle) vs. euclidean distance (magnitude)
  - Cosine similarity ignores the size of the text data, instead focus on semantics
  - Cosine removes the impact of magnitude
  - Normalize data will change euclidean distance, but not cosine.
- When to use which distance function?
  - Cosine is popular for text mining
  - Cosine fits those applications where data units have different length, such as texts or streaming data (video/audio)

$$CosSim(v_1, v_2) = Cosine(\theta) = \frac{v_1 \cdot v_2}{||v_1||\ ||v_2||}$$

Text Mining: Sentiment Analysis

- Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.
- Simple approach: add the sentiment scores for all the terms used in the documents
- Data science approach: convert text data into Document-Term Matrix (DTM) and treat each term as a feature/attribute to build machine learning model

–>

Load in Necessary Libraries

```
library(tidytext)
library(stringr)
library(dplyr)
library(tidyr)
library(wordcloud)
library(ggplot2)
```

Demo Dataset: Yelp Review

```
reviews <- read.csv("yelp_reviews.csv", stringsAsFactors = FALSE)
str(reviews)
```

```
## 'data.frame':    3000 obs. of  11 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_id     : chr  "rLtl8ZkDX5vH5nAx9C3q5Q" "0a2KyEL0d3Yb1V6aivbIuQ" "0hT2KtfLiobPvh6cDC8
##  $ review_id   : chr  "fWKvX83p0-ka4JS3dc6E5A" "IjZ33sJrzXqU-0X6U8NwyA" "IESLBzqUCLdSzSqm0eC
##  $ stars       : int  5 5 4 5 5 4 5 4 4 5 ...
##  $ date        : chr  "2011-01-26" "2011-07-27" "2012-06-14" "2010-05-27" ...
##  $ text        : chr  "My wife took me here on my birthday for breakfast and it was excellen
##  $ type        : chr  "review" "review" "review" "review" ...
##  $ business_id : chr  "9yKzy9PApeiPPOUJEtnvkg" "ZRJwVLyzEJq1VAihDhYiow" "6oRAC4uyJCsJl1X0WZp
##  $ votes.funny : int  0 0 0 0 0 1 4 0 0 0 ...
##  $ votes.useful: int  5 0 1 2 0 3 7 1 0 1 ...
##  $ votes.cool  : int  2 0 0 1 0 4 7 0 0 0 ...
```

13/33

Tokenization

```
review_words <- reviews %>%
  select(review_id, business_id, stars, text) %>%
  unnest_tokens(word, text, to_lower = TRUE)
head(review_words, 10)
```

```
##                  review_id            business_id stars      word
## 1   fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5        my
## 1.1 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5      wife
## 1.2 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5      took
## 1.3 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5        me
## 1.4 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5      here
## 1.5 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5        on
## 1.6 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5        my
## 1.7 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5  birthday
## 1.8 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5       for
## 1.9 fWKvX83p0-ka4JS3dc6E5A 9yKzy9PApeiPPOUJEtnvkg     5 breakfast
```

Stemming and Lemmatization

- Goal: reduce inflectional forms and derivationally related forms of a word to a common base form.
- Examples
  - car, cars, car's, cars' => car
  - am, are, is => be

```
library(SnowballC)
review_words$word <- wordStem(review_words$word)
```

Stop Words: Vocabulary

- A data frame in package with curated stop words (1149)

```
set.seed(368)
cat(stop_words$word[sample(x = 1:nrow(stop_words), size = 20)], sep = ", ")
```

```
## whose, is, though, anyone, us, actually, if, under, ours, despite, finds, p, followed, herek
```

- Source of stop words: 3 lexicons

```
unique(stop_words$lexicon)
```

```
## [1] "SMART"    "snowball" "onix"
```

16/33

Stop Words: Removal

```
nrow(review_words)
```

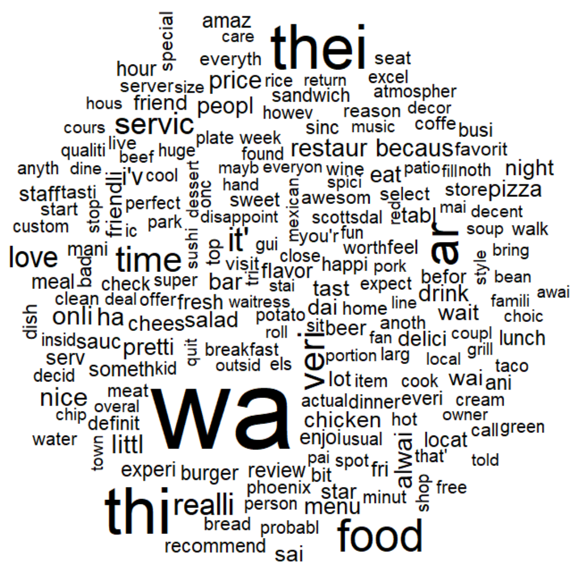
```
## [1] 386390
```

```
review_words <- review_words %>%
  filter(!word %in% stop_words$word,
         str_detect(word, "^[a-z']+$"))
nrow(review_words)
```

```
## [1] 164797
```

Frequency and Wordcloud

```
review_words %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 200))
```



Sentiment Lexicons

- data frame in package contains four lexicons

```
unique(sentiments$lexicon)
```

```
## [1] "nrc"      "bing"     "AFINN"    "loughran"
```

- Data frame contains the lexicon names, sentiment (categories), score (numerical values for the sentiments)

```
str(sentiments)
```

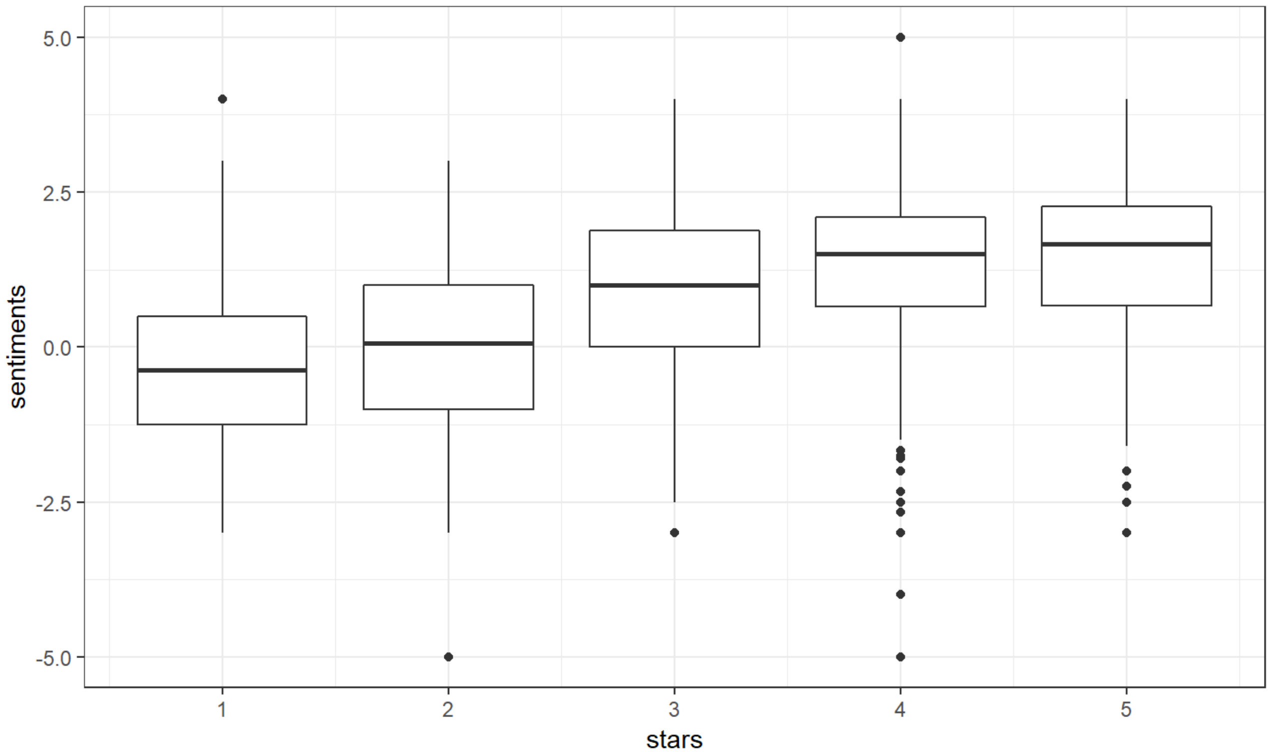
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    27314 obs. of  4 variables:
##  $ word     : chr  "abacus" "abandon" "abandon" "abandon" ...
##  $ sentiment: chr  "trust" "fear" "negative" "sadness" ...
##  $ lexicon  : chr  "nrc" "nrc" "nrc" "nrc" ...
##  $ score    : int  NA NA NA NA NA NA NA NA NA NA ...
```

Choose a Lexicon to Score Text Documents

```
AFINN <- sentiments %>%
  filter(lexicon == 'AFINN') %>%
  select(word, afinn_score = score)

reviews_sentiment <- review_words %>%
  inner_join(AFINN, by = "word") %>%
  group_by(review_id, stars) %>%
  summarise(sentiments = mean(afinn_score))
```

Correlation of Review Score and the Sentiment

```
theme_set(theme_bw())
ggplot(reviews_sentiment, aes(stars, sentiments, group = stars)) +
  geom_boxplot()
```

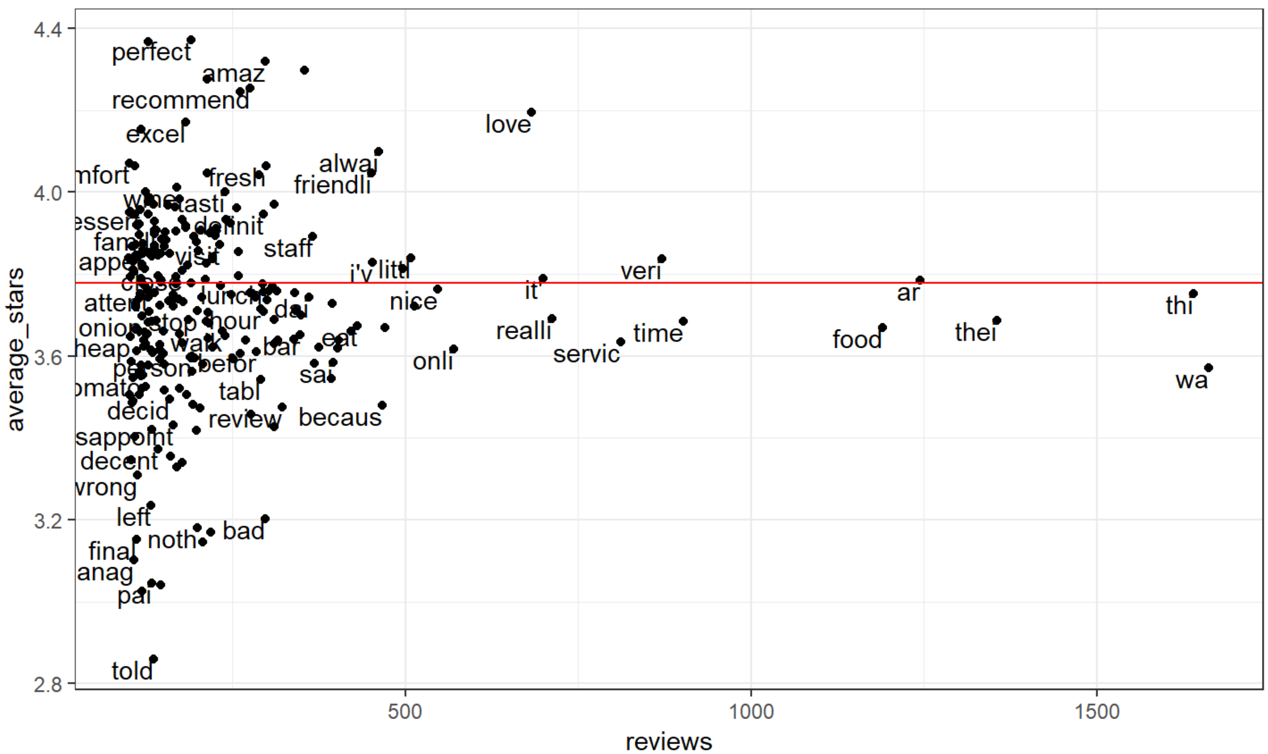


```r
word_summaries <- review_words %>%
  count(review_id, business_id, stars, word) %>%
  group_by(word) %>%
  summarise(businesses = n_distinct(business_id),
            reviews = n(),
            uses = sum(n),
            average_stars = mean(stars)) %>%
  ungroup()

word_summaries_filtered <- word_summaries %>%
  filter(reviews >= 100, businesses >= 5) %>%
  arrange(desc(average_stars))

head(word_summaries_filtered)
```

```
## # A tibble: 6 x 5
##   word      businesses reviews  uses average_stars
##   <chr>          <int>   <int> <int>         <dbl>
## 1 perfect          171     191   211          4.37
## 2 fantast          118     128   141          4.37
## 3 amaz             263     298   345          4.32
## 4 delici           311     354   420          4.30
## 5 awesom           201     214   242          4.28
## 6 recommend        262     276   306          4.25
```

Visualize the Sentiments of Words in Reviews

```
ggplot(word_summaries_filtered, aes(reviews, average_stars)) +
  geom_point() +
  geom_text(aes(label = word), check_overlap = T, vjust = 1, hjust = 1) +
  geom_hline(yintercept = mean(reviews$stars), color = "red")
```



Construct a Document-Term Matrix (DTM)

```
review_words_dtm <- review_words %>%
  count(review_id, stars, word) %>%
  bind_tf_idf(word, review_id, n) %>%
  select(review_id, stars, word, tf_idf) %>%
  spread(key = word, value = tf_idf)
```

Load Python Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import nltk
```

Load Yelp Dataset and Vectorization

```
yelp = pd.read_csv('https://raw.githubusercontent.com/sinanuozdemir/sfdat22/master/data/yelp.cs
reviews = yelp['text']
vectorizer = CountVectorizer(encoding='utf-8')
review_words = vectorizer.fit_transform(reviews)
review_words[:3]
```

```
## <3x29185 sparse matrix of type '<class 'numpy.int64'>'
##  with 252 stored elements in Compressed Sparse Row format>
```

```
print(len(vectorizer.vocabulary_))
```

```
## 29185
```

```
print(list(vectorizer.vocabulary_.items()))
```

```
## [('my', 17130), ('wife', 28506), ('took', 26448), ('me', 16151), ('here', 12364), ('on', 179
```

Stop Words Removal and Stemming/Lemmatization

```
vectorizer_stopwords = CountVectorizer(encoding='utf-8', stop_words='english', lowercase=True)
review_words_stopwords = vectorizer_stopwords.fit_transform(reviews)

from nltk.stem.snowball import SnowballStemmer
import re
sno = nltk.stem.SnowballStemmer('english')
def stemming_tokenizer(str_input):
  words = re.sub(r"[^A-Za-z0-9\-]", " ", str_input).lower().split()
  words = [sno.stem(word) for word in words]
  return words

vectorizer_stem = CountVectorizer(encoding='utf-8', stop_words='english', lowercase=True, toker
review_words_stem = vectorizer_stem.fit_transform(reviews)
```

```
## C:\Users\ylin65\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\feature_ext
##   'stop_words.' % sorted(inconsistent))
```

```
print(len(vectorizer_stem.vocabulary_))
```

```
## 23699
```

Wordcloud Visualization

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('english'))
word_tokens = []
for i in reviews:
  word_tokens.append(word_tokenize(i))
filtered_sentence = []
for w in word_tokens:
  for i in w:
    if i not in stop_words:
      filtered_sentence.append(i)
text = pd.Series(filtered_sentence).str.cat(sep=' ')
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800, max_font_size=200).generate(text)
plt.figure(figsize=(12, 10))
plt.imshow(wordcloud, interpolation='bilinear')
```

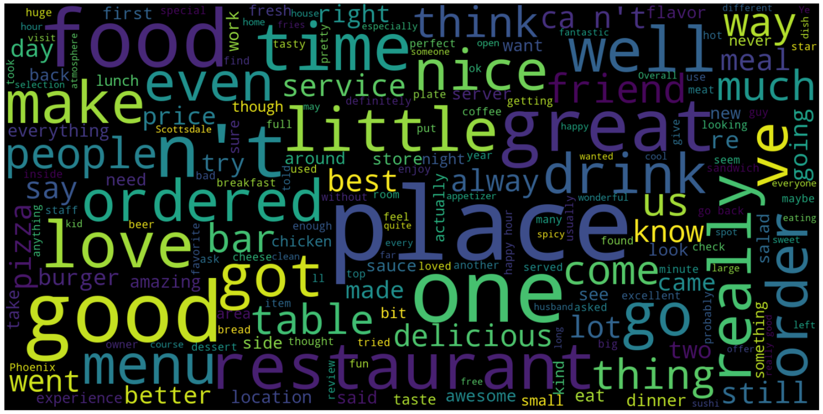
```
## <matplotlib.image.AxesImage object at 0x0000000081C2BE08>
```

```
plt.axis("off")
```

```
## (-0.5, 1599.5, 799.5, -0.5)
```

```
## <Figure size 1200x1000 with 0 Axes>
```

```
## (-0.5, 1599.5, 799.5, -0.5)
```



29/33

Sentiment Analysis: A Naive Approach

```
from afinn import Afinn
afinn = Afinn()
yelp.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 10000 entries, 0 to 9999
## Data columns (total 10 columns):
## business_id    10000 non-null object
## date           10000 non-null object
## review_id      10000 non-null object
## stars          10000 non-null int64
## text           10000 non-null object
## type           10000 non-null object
## user_id        10000 non-null object
## cool           10000 non-null int64
## useful         10000 non-null int64
## funny          10000 non-null int64
## dtypes: int64(4), object(6)
## memory usage: 781.4+ KB
```

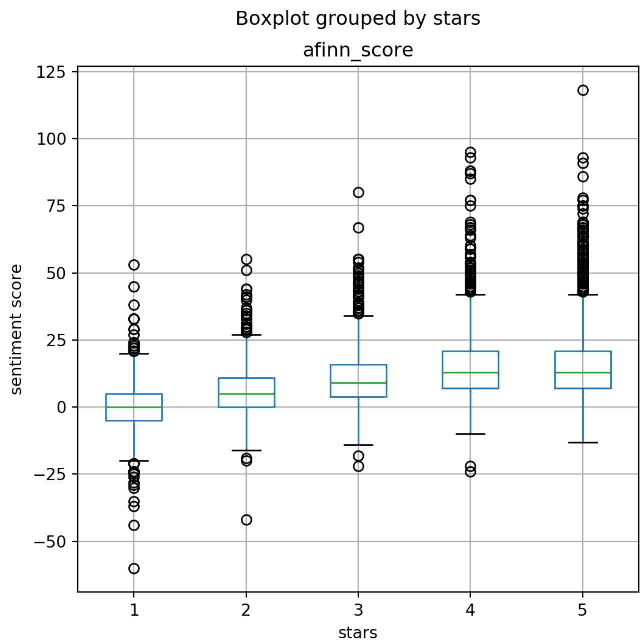
```
yelp['afinn_score'] = yelp['text'].apply(afinn.score)
```



```
reviews_sentiment = yelp.groupby(['review_id', 'stars'])[['afinn_score']].sum()
```

Correlation between Sentiment Analysis and Rating

```
fig, ax = plt.subplots(figsize=(6, 6))
plt.suptitle('Relation between Review Score and Sentiments')
plt.ylabel('sentiment score')
reviews_sentiment.boxplot(column=['afinn_score'], by='stars', ax=ax)
```



Naive Bayes Modeling

```
X = yelp['text'].values
Y = yelp['stars'].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=8)
X_train_vec_nb = vectorizer_stopwords.fit_transform(X_train)
X_test_vec_nb = vectorizer_stopwords.transform(X_test)

from sklearn.naive_bayes import MultinomialNB
nb_clf = MultinomialNB()
nb_clf.fit(X_train_vec_nb, y_train)
```

```
## MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
from sklearn.metrics import confusion_matrix, classification_report
y_pred_mnb = nb_clf.predict(X_test_vec_nb)
confusion_matrix(y_test, y_pred_mnb, labels=[1, 2, 3, 4, 5])
```

```
## array([[ 65,  31,  31,  69,  26],
##        [ 19,  26,  47, 158,  25],
##        [  9,  11,  48, 330,  44],
##        [  5,   7,  16, 771, 260],
##        [  7,   1,   9, 454, 531]], dtype=int64)
```