- I recommend reading the summary for a quick overview, the notes for more detail!

4 Sentiment Analysis 1 & 2

Previous Lectures

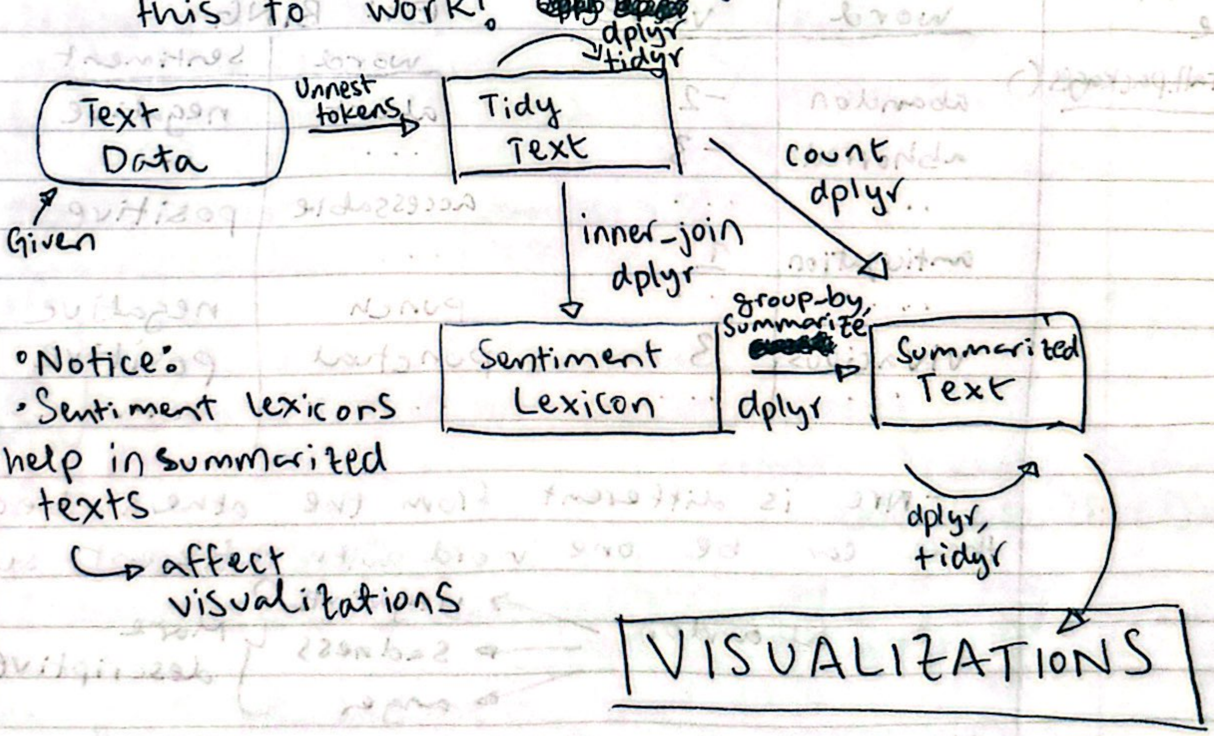
- Tidy Text Format → Approaches for Word Frequency

*NEW TOPICS:

- Option Mining
- Sentiment Analysis → THIS IS WHAT WE ARE DOING THIS WK

A Human Approach → Using emotional extent of words

→ Tools of text mining allows for this to work!



- Notice:
- Sentiment lexicons help in summarized texts
  - → affect visualizations

Sentiments Datasets

Lexicons

- AFINN
- bing
- nrc

} single words

→ given scores for positive/negative sentiment

- Note nrc lexicons are categorized in a binary fashion (yes/no) → among many categories

- Reminder: If you want to download the lexicons use install.packages()

```
install.packages("textdata")
library("textdata")
get_sentiments("afinn")
```

Ex: AFINN

word	value
abandon	-2
abhorred	-3
...	...
anticipation	1
...	...
vivacious	3
...	...

Ex: BING

word	sentiment
abort	negative
...	...
accessable	positive
...	
punch	negative
punctual	positive
...	...

- nrc is different from the others because there can be one word with different sentiments

abandon → negative, → sadness, → anger } More descriptive

HOW IS THIS DATA VALIDATED?

- Crowdsourcing!

→ Even though using sentiment lexicons perhaps might be less accurate ... measuring the content of the words shared by lexicon is enough

$$\text{Total Sentiment} = \sum \text{Individual sentiment scores for each word}$$

Inner Join

data in tidy format → Sentiment analysis

↑ inner join

- antijoin → removing stop words
- innerjoin → sentiment analysis

Example in *EMMA*

1. First we need data in tidy format

```
library(janeaustenr)
library(dplyr)
library(stringr)

tidy_books <- austen_books()%>%
   group_by(book)%>%
   mutate(
      linenumber = row_number(),
      chapter = cumsum(str_detect(text,
                              regex("^chapter [\\divxlc]
                                 ignore_case=TRUE))))%>

   ungroup()%>%
   unnest_tokens(word,text)

nrc_joy <- get_sentiments("nrc")%>%
   filter(sentiment=="joy")
tidy_books %>%
  filter(book=="Emma")%>%
  inner_join(nrc_joy)%>%
  count(word,sort=TRUE)
```

method for sentiments data frame 'nrc'

2. Get a Sentiment lexicon dictionary

3. USE inner_join to produce new data frame applying sentiment

our tidy data

- The output is a data frame of a word and the corresponding count of each word (by the sentiment)

%/%
↳integer division (Ex: 5%/%2 = 2)

Sentiment: joy → picked lines from text → output: # of occurrences by sentiment

↳ Most common 'joy' words in EMMA → good, young, friend

- Note: depending on texts, things to consider:
  - How long lines are
  - How many lines to consider in analysis

→ pivot_wider() to separate positive and negative sentiment

Comparing the Sentiment Dictionaries

How about difference in lexicons?
- Different IMMEDIATE results
- Overall relatively same trajectories

- Be very careful on lexicon positive-to-negative word ratios, can lead to biased analysis

Most common Sentiment words

- analyze word counts that contribute to each sentiment

Ex: Bing Word Counts

```
bing_word_counts <- tidy_books %>%
  inner_join(get_sentiments("bing") %>%
   count(word, sentiment, sort = TRUE) %>%
    ungroup()
```

→ it allows us to spot anomalies in words used in our sentiment analysis

Wordclouds

- cloud of words
- words are sized by column (in our case we can use sentiment frequency)

Ex:

```
library(wordcloud)

tidy_books %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

- We can also use comparison.cloud() to make a wordcloud comparison

*Might require a matrix
↳ to convert dataframe to matrix
- USE acast()

Ex:

```
library(reshape2)

tidy_books %>%
   inner_join(get_sentiments("bing")) %>%
   count(word, sentiment, sort=TRUE) %>%
   acast(word ~ sentiment, value.var = "n", fill=0) %>%
   comparison.cloud(colors = c("gray20", "gray80"),
                    max.words = 100)
```

Different Units

- Some algorithms try to understand the sentiment of a sentence as a whole

Example of packages:
- coreNLP
- cleanNLP
- sentimentr

} Sentiment Analysis Algorithms

Example:

{Split tokens into Regex Pattern (this is bettern than using "sentence" token)}

Regex pattern by chapter →

```
austen_chapters <- austen_books() %>%
    group_by(book) %>%
    unnest_tokens(chapter, text, token = "regex",
                  pattern = "Chapter|CHAPTER [\\dIVXLC]") %>%
    ungroup()

austen_chapters %>%
    group_by(book) %>%
    summarise(chapters = n())
```

Q: What are the most negative chapters in each of Jane Austen's novels?

Step 1: List of Negative words (from Bing lexicon)

```
~~bing~~ bingnegative <- get_sentiments("bing") %>%
    filter(sentiment == "negative")
```

Step 2: Find number of negative words in each chapter

```
wordcounts <- tidybooks %>%
    group_by(book, chapter) %>%
    summarize(words = n())
```

Step 3:
Divide by Total words in each Chapter

```
tidy_books %>%
  semi_join(bingnegative) %>%
   group_by(book, chapter) %>%
   summerize(negative words = n()) %>%
   left_join(wordcounts, by = c("book", "chapter")) %>%
    mutate(ratio = negativeWords/ words) %>%
    filter(chapter != 0) %>%
     slice_max(ratio, n=1) %>%
     ungroup()
```

→ most sad words in each book, normalized for number of words in the chapter

Summary

Unit 8: Regular Expressions → Sentiment Analysis ← Unit 9: Text Mining I

Sentiment Analysis

- So far, we learned about converting our raw text into tidy data

  ↳ We use tidy data to perform sentiment analysis (categorizing words as emotions or giving a positive/negative value

*If you need to download the lexicons, refer to notes *

Sentiment Lexicon

- AFINN
- bing
- nrc

} we can use the sentiment library "library(tidytext)"

↳ get_sentiments()

- This is yet another analysis tool and we can analyze the trajectory of sentiment a text has

↳ Based on the scope of lines or classification we use, we may get different results so it is important to choose logical scopes

↳ From output tables (joined via sentiment tables) we can make wordclouds using the library(wordcloud) □