Starter Project for Sentiment Analysis

Tristan Tran

12/14/2020

Introduction

Sentiment analysis and text mining allows Data Scientists to classify texts, novels, and other written forms of communication as data. Its practical applications can allow machine learning models and artificial intelligence to understand how humans talk and how they feel. Imagine a world where mental health could be diagnosed from a few text messages, consumers find the products they need, and computers speak more like humans. Although this project is merely an introductory level endeavor, it hopes to build a foundation to accomplish these feats in the future.

Purpose

The purpose of this project was practicing the basic techniques of data manipulation and working with text data. Text data often comes in forms that are difficult process and analyze. Text mining techniques and packages like tidyverse make this process a lot smoother.

The program is run by calling the following code.

```
source("../src/HorrorAnalysis.R")
```

The code reads through a csv "data/booklist.csv" and returns a word cloud for each book listed.

Important Packages

For this project several packages were used to make the data readable and to generate the graphics

dplyr

The dplyr package is used to manipulate and modify data frames. The included mutate() function was used to produce the line number and chapter columns in the tidy versions of the book.

gutenbergr

The gutenbergr package gives the user access to the thousands of books that have been uploaded to Project Gutenberg that aims to provide the world with free and easy to access classic literature so that our collective human culture and knowledge is not lost. This package was used to import several classical horror novels from the included csv file. Below is an example of how a single file is imported. By calling the head function, it is apparent that the formatting and white space make it difficult to perform any sort of analysis.

```
drjekyll<-gutenberg_download(42,meta_fields ="title")
head(drjekyll,12)
```

```
## # A tibble: 12 x 3
##    gutenberg_id text                                  title
##           <int> <chr>                                 <chr>
##  1           42 "                                  ST~ The Strange Case of Dr. Jek~
##  2           42 "                                  DR~ The Strange Case of Dr. Jek~
##  3           42 "                                    ~ The Strange Case of Dr. Jek~
##  4           42 ""                                   The Strange Case of Dr. Jek~
##  5           42 "                                  ~ The Strange Case of Dr. Jek~
##  6           42 "                                ROBER~ The Strange Case of Dr. Jek~
##  7           42 ""                                   The Strange Case of Dr. Jek~
##  8           42 ""                                   The Strange Case of Dr. Jek~
##  9           42 ""                                   The Strange Case of Dr. Jek~
## 10           42 ""                                   The Strange Case of Dr. Jek~
## 11           42 "1)"                                 The Strange Case of Dr. Jek~
## 12           42 ""                                   The Strange Case of Dr. Jek~
```

Tidy Things

The function located in the sentiment_analysis.R file

```
create_tidy_books
```

```
## function (gutenberg_collection)
## {
##     result <- gutenberg_collection %>% group_by(gutenberg_id) %>%
##         mutate(linenumber = row_number(), chapter = cumsum(str_detect(text,
##             regex("^chapter [\\divxlc]", ignore_case = TRUE)))) %>%
##         ungroup() %>% unnest_tokens(word, text)
##     return(result)
## }
```

The tidytext, stringr, and tidyr packages are all designed to improve the readability of the data. The tidytext website describes a tidy data format as one observation per row with one attribute per column. The function create_tidy_books utilizes the functions from these packages separate the words from their line numbers without losing any information, other than capitalization.

```
tidy_jekyll <- create_tidy_books(drjekyll)

head(tidy_jekyll,12)
```

```
## # A tibble: 12 x 5
##    gutenberg_id title                                linenumber chapter word
##           <int> <chr>                                    <int>   <int> <chr>
##  1           42 The Strange Case of Dr. Jekyll and ~          1       0 strange
##  2           42 The Strange Case of Dr. Jekyll and ~          1       0 case
##  3           42 The Strange Case of Dr. Jekyll and ~          1       0 of
##  4           42 The Strange Case of Dr. Jekyll and ~          2       0 dr
##  5           42 The Strange Case of Dr. Jekyll and ~          2       0 jekyll
```

```
##  6          42 The Strange Case of Dr. Jekyll and ~          2       0 and
##  7          42 The Strange Case of Dr. Jekyll and ~          3       0 mr
##  8          42 The Strange Case of Dr. Jekyll and ~          3       0 hyde
##  9          42 The Strange Case of Dr. Jekyll and ~          5       0 by
## 10          42 The Strange Case of Dr. Jekyll and ~          6       0 robert
## 11          42 The Strange Case of Dr. Jekyll and ~          6       0 louis
## 12          42 The Strange Case of Dr. Jekyll and ~          6       0 stevens~
```

This package also comes with the get_sentiments function that queries text lexicons useful for the analysis.

Graphics

This project created two main types of graphics. Below

Word Clouds

```
## function (id, title, tidy_books)
## {
##     filename = paste("../results/", title, ".jpeg", sep = "")
##     jpeg(file = filename)
##     tidy_books %>% filter(gutenberg_id == id) %>% anti_join(stop_words) %>%
##         count(word) %>% with(wordcloud(word, n, max.words = 100,
##         min.freq = 10, scale = c(4, 0.5)))
##     dev.off()
## }
## <bytecode: 0x000000001db72540>
```

The first type of data visualization involves getting a count of words for each novel and creating a word cloud of the 100 most common words. The names of the protagonists and supporting characters are over-represented. To address this, a file "ignore.csv" was created. The possessive and singular of each name must be included at the moment. There most likely exists a more elegant solution; however, this improvement is non-trivial. The word cloud will ignore anything in that file.

```
all_stop_words <- create_stop_words("../data/ignore.csv")
```

```
## Parsed with column specification:
## cols(
##   word = col_character()
## )
```

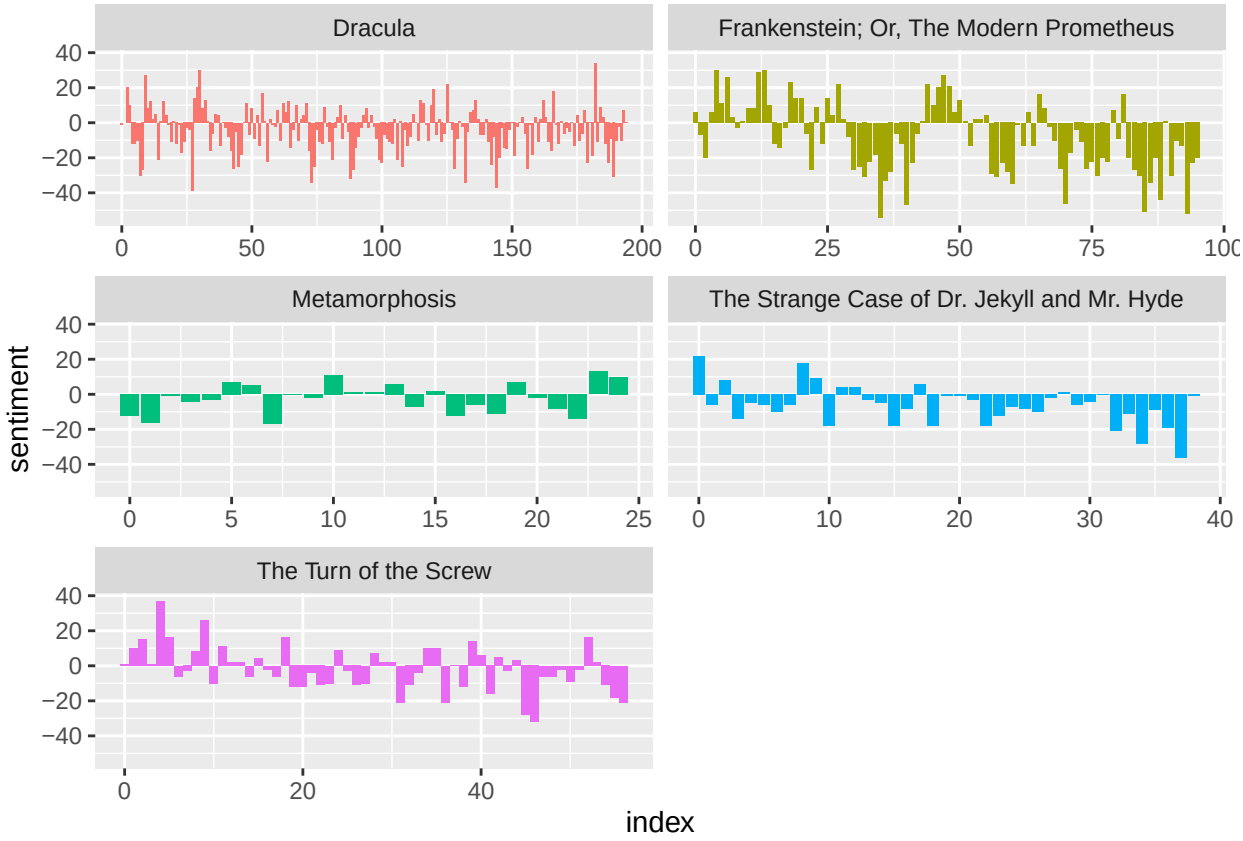
```
tidy_jekyll_no_stop <- tidy_jekyll%>%
  anti_join(all_stop_words,by="word")

tibble(
  total_words = nrow(tidy_jekyll),
  after_cleanup= nrow(tidy_jekyll_no_stop)
)
```

```
## # A tibble: 1 x 2
##   total_words after_cleanup
##         <int>         <int>
## 1       25899          8122
```

By cleaning up the data and removing stop words, the dimensionality of the data and make analysis easier can be reduced. The change in the word cloud is presented below.

Plot Progression and Trajectory

Figure: Trajectory Graph of Raw Text



The bing lexicon provides a negative or positive designation for English words. These scores were obtained in 2004. The sentiments are relevant for that time period and can be anachronistic compared to select classic literature. For instance, English speakers in 2004 might view the word "miss" as a negative word meaning "fail to hit or notice"; however, in older settings and literature this can mean "young woman of nobility". These words are homonyms whose popularity of use has reversed in over time. A similar case can be seen with the word "Nimrod" which has changed meaning following the televised adventures of Bugs Bunny. To address these anachronisms and homonyms, they will be similarly added to the ignore.csv file and processed in the same way. Below is the result of plot trajectory when the stop words library is applied

Figure: Trajectory Graph of Cleaned Text

The result is closer to what one would expect from a horror novel. Further testing and processing is necessary to determine if this is appropriate for analysis or if it causes a worse interpretation of storylines.

Difficulties and Opportunities

Although the program is much better than its first iteration, there is still room for improvement. The lexicon with anachronistic words removed is still not as good as mining lexicons from a group of scholars proficient in the English language at the time each book was written. Unfortunately, machine learning is not good at interpreting out-of-sample data. The code is most likely better applied to modern literature or social media data. Another challenge that would improve the reading is the inclusion of bigrams. Bigrams allow one to analyze a series of words and measure their net meaning. While "happy" connotes joy, "not happy" connotes sadness. When measuring sentiment in a word by word algorithm, the net measurement might be zero; however, if they were analyzed as a bigram, the program could interpret the net effect as a negative emotion.

References

1. In press. Text Mining with R: A Tidy Approach, year = 2020, url = https://www.tidytextmining.com/index.html, urldate = 2020-11-10.

2. In press. Sentiment analysis using tidytext.