Starter Project for Sentiment Analysis

Tristan Tran

12/14/2020

Introduction

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

Purpose

The purpose of this project was to practice the basic techniques of data manipulation and working with text data. Text data often comes in forms that are difficult to work with and analyze. Text mining techniques and packages such as tidy text 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 does not get lost. This package was used to import several classical horror novels from the included csv file. Below is an example of how one would import a single file. By calling

the head function, it is apparent that the formatting and whitespace 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 make the data more readable. 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: 0x0000000019057890>
```

The first type of data visualization involves getting a a word for each novel and creating a wordcloud of the 100 most common words. The names of the protagonists and supporting cast are over-represented. To address this, a file "ignore.csv" was created. The word cloud will ignore anything in that file

Plot Progression and Trajectory

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 some 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". To address these anachronisms, they will be similarly added to the ignore.csv file and processed in the same way.

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.