**Motivation**

After reading [The Life Changing Magic of Tidying Text](https://juliasilge.com/blog/life-changing-magic/) and [A tidy text analysis of Rick and Morty](http://tamaszilagyi.com/blog/a-tidy-text-analysis-of-rick-and-morty/) I wanted to do something similar for Rick and Morty and I did. Now I’m doing something similar for Bojack Horseman.

In this post I’ll focus on the Tidy Data principles. However, [here](https://github.com/pachamaltese/rick_and_morty_tidy_text) is the Github repo with the scripts to scrap the subtitles of Rick and Morty and other shows.

*Note: If some images appear too small on your screen you can open them in a new tab to show them in their original size.*

**Let’s scrap**

The subtools package returns a data frame after reading srt files. In addition to that resulting data frame I wanted to explicitly point the season and chapter of each line of the subtitles. To do that I had to scrap the subtitles and then use str\_replace\_all. To follow the steps clone the repo from Github:

git clone https://github.com/pachamaltese/rick\_and\_morty\_tidy\_text

**Bojack Horseman Can Be So Tidy**

After reading the tidy file I created after scraping the subtitles, I use unnest\_tokens to divide the subtitles in words. This function uses the tokenizers package to separate each line into words. The default tokenizing is for words, but other options include characters, sentences, lines, paragraphs, or separation around a regex pattern.

if (!require("pacman")) install.packages("pacman")

p\_load(

tidyverse,

tidytext,

igraph,

ggraph

)

p\_load\_gh("dgrtwo/widyr")

bojack\_horseman\_subs <- read\_csv("../../data/2017-10-13-rick-and-morty-tidy-data/bojack\_horseman\_subs.csv") %>%

mutate(text = iconv(text, to = "ASCII")) %>%

drop\_na()

bojack\_horseman\_subs\_tidy <- bojack\_horseman\_subs %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words)

The data is in one-word-per-row format, and we can manipulate it with tidy tools like dplyr. For example, in the last chunk I used an anti\_join to remove words such a “a”, “an” or “the”.

Then we can use count to find the most common words in all of Bojack Horseman episodes as a whole.

bojack\_horseman\_subs\_tidy %>%

count(word, sort = TRUE)

# A tibble: 10,845 x 2

word n

1 bojack 807

2 yeah 695

3 hey 567

4 gonna 480

5 time 446

6 uh 380

7 diane 345

8 todd 329

9 people 307

10 love 306

# … with 10,835 more rows

Sentiment analysis can be done as an inner join. There is one sentiment lexicon in the tidytext package. Let’s examine how sentiment changes changes during each season. Let’s count the number of positive and negative words in the chapters of each season.

bojack\_horseman\_sentiment <- bojack\_horseman\_subs\_tidy %>%

inner\_join(sentiments) %>%

count(episode\_name, index = linenumber %/% 50, sentiment) %>%

spread(sentiment, n, fill = 0) %>%

mutate(sentiment = positive - negative) %>%

left\_join(bojack\_horseman\_subs\_tidy[,c("episode\_name","season","episode")] %>% distinct()) %>%

arrange(season,episode) %>%

mutate(episode\_name = paste(season,episode,"-",episode\_name),

season = factor(season, labels = paste("Season", 1:4))) %>%

select(episode\_name, season, everything(), -episode)

bojack\_horseman\_sentiment

# A tibble: 537 x 6

episode\_name season index negative positive sentiment

1 S01 E01 - BoJack Horseman The … Seaso… 0 9 7 -2

2 S01 E01 - BoJack Horseman The … Seaso… 1 4 3 -1

3 S01 E01 - BoJack Horseman The … Seaso… 2 6 8 2

4 S01 E01 - BoJack Horseman The … Seaso… 3 6 2 -4

5 S01 E01 - BoJack Horseman The … Seaso… 4 12 8 -4

6 S01 E01 - BoJack Horseman The … Seaso… 5 15 5 -10

7 S01 E01 - BoJack Horseman The … Seaso… 6 13 5 -8

8 S01 E01 - BoJack Horseman The … Seaso… 7 7 8 1

9 S01 E01 - BoJack Horseman The … Seaso… 8 11 3 -8

10 S01 E01 - BoJack Horseman The … Seaso… 9 17 8 -9

# … with 527 more rows

Now we can plot these sentiment scores across the plot trajectory of each season.

ggplot(bojack\_horseman\_sentiment, aes(index, sentiment, fill = season)) +

geom\_bar(stat = "identity", show.legend = FALSE) +

facet\_wrap(~season, nrow = 3, scales = "free\_x", dir = "v") +

theme\_minimal(base\_size = 13) +

labs(title = "Sentiment in Bojack Horseman",

y = "Sentiment") +

scale\_fill\_viridis(end = 0.75, discrete = TRUE) +

scale\_x\_discrete(expand = c(0.02,0)) +

theme(strip.text = element\_text(hjust = 0)) +

theme(strip.text = element\_text(face = "italic")) +

theme(axis.title.x = element\_blank()) +

theme(axis.ticks.x = element\_blank()) +

theme(axis.text.x = element\_blank())

**Looking at Units Beyond Words**

Lots of useful work can be done by tokenizing at the word level, but sometimes it is useful or necessary to look at different units of text. For example, some sentiment analysis algorithms look beyond only unigrams (i.e. single words) to try to understand the sentiment of a sentence as a whole. These algorithms try to understand that *I am not having a good day* is a negative sentence, not a positive one, because of negation.

bojack\_horseman\_sentences <- bojack\_horseman\_subs %>%

group\_by(season) %>%

unnest\_tokens(sentence, text, token = "sentences") %>%

ungroup()

Let’s look at just one.

bojack\_horseman\_sentences$sentence[1778]

[1] "hey, boys. what is this, a crossover episode?"

We can use tidy text analysis to ask questions such as: What are the most negative episodes in each of Bojack Horseman’s seasons? First, let’s get the list of negative words from the lexicon. Second, let’s make a dataframe of how many words are in each chapter so we can normalize for the length of chapters. Then, let’s find the number of negative words in each chapter and divide by the total words in each chapter. Which chapter has the highest proportion of negative words?

sentiment\_negative <- sentiments %>%

filter(sentiment == "negative")

wordcounts <- bojack\_horseman\_subs\_tidy %>%

group\_by(season, episode) %>%

summarize(words = n())

bojack\_horseman\_subs\_tidy %>%

semi\_join(sentiment\_negative) %>%

group\_by(season, episode) %>%

summarize(negativewords = n()) %>%

left\_join(wordcounts, by = c("season", "episode")) %>%

mutate(ratio = negativewords/words) %>%

top\_n(1)

# A tibble: 4 x 5

# Groups: season [4]

season episode negativewords words ratio

1 S01 E06 133 1277 0.104

2 S02 E07 136 1273 0.107

3 S03 E07 131 1218 0.108

4 S04 E06 160 1392 0.115

**Networks of Words**

Another function in widyr is pairwise\_count, which counts pairs of items that occur together within a group. Let’s count the words that occur together in the lines of the first season.

bojack\_horseman\_words <- bojack\_horseman\_subs\_tidy %>%

filter(season == "S01")

word\_cooccurences <- bojack\_horseman\_words %>%

pairwise\_count(word, linenumber, sort = TRUE)

word\_cooccurences

# A tibble: 295,932 x 3

item1 item2 n

1 yeah bojack 57

2 bojack yeah 57

3 bojack hey 47

4 hey bojack 47

5 horseman bojack 44

6 bojack horseman 44

7 diane bojack 36

8 bojack diane 36

9 gonna bojack 35

10 bojack gonna 35

# … with 295,922 more rows

This can be useful, for example, to plot a network of co-occuring words with the igraph and ggraph packages.

set.seed(1717)

word\_cooccurences %>%

filter(n >= 25) %>%

graph\_from\_data\_frame() %>%

ggraph(layout = "fr") +

geom\_edge\_link(aes(edge\_alpha = n, edge\_width = n), edge\_colour = "#a8a8a8") +

geom\_node\_point(color = "darkslategray4", size = 8) +

geom\_node\_text(aes(label = name), vjust = 2.2) +

ggtitle(expression(paste("Word Network in Bojack Horseman's ",

italic("Season One")))) +

theme\_void()