HW 11

May 6, 2021

1 IST 387 HW 11

[1]:
```
# Enter your name here: Connor Hanan
```

1.0.1 Attribution statement: (choose only one and delete the rest)

[2]:
```
# 1. I did this homework by myself, with help from the book and the professor.
```

Text mining plays an important role in many industries because of the prevalence of text in the interactions between customers and company representatives. Even when the customer interaction is by speech, rather than by chat or email, speech to text algorithms have gotten so good that transcriptions of these spoken word interactions are often available. To an increasing extent, a data scientist needs to be able to wield tools that turn a body of text into actionable insights. In this homework, we explore a real **City of Syracuse dataset** using the **quanteda** and **quanteda.textplots** packages. Make sure to install the **quanteda** and **quanteda.textplots** packages before following the steps below:

1.1 Part 1: Load and visualize the data file

A. Take a look at this article: https://samedelstein.medium.com/snowplow-naming-contest-data-2dcd38272caf and write a comment in your R script, briefly describing what it is about.

[4]:
```
# it is an article about the naming contest syracuse held for snow plows, as␣
↪well as links to all the data of the entries
# mentions the process he used to obtain the data as well
```

B. Copy the data from the following URL to a dataframe called **df**: https://ist387.s3.us-east-2.amazonaws.com/data/snowplownames.csv

[5]:
```
#install.packages('quanteda')
#install.packages('quanteda.textplots')
```

[7]:
```
library(tidyverse)
library(quanteda)
library(quanteda.textplots)
```

[8]:
```
df <- read_csv("https://ist387.s3.us-east-2.amazonaws.com/data/snowplownames.
↪csv")
```

```
  Column specification

cols(
  submission_number = col_double(),
  submitter_name_anonymized = col_character(),
  snowplow_name = col_character(),
  meaning = col_character(),
  winning_name = col_logical()
)
```

C. Inspect the **df** dataframe – which column contains an explanation of the meaning of each submitted snowplow name? Transform that column into a **document-feature matrix**, using the **corpus()** and **dfm()** functions. Do not forget to **remove stop words**.

[9]:
```
str(df)
```

```
tibble [1,907 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ submission_number        : num [1:1907] 1 2 3 4 5 6 7 8 9 10 …
 $ submitter_name_anonymized: chr [1:1907] "kjlt9cua" "KXKaabXN" "kjlt9cua"
"Rv9sODqp" …
 $ snowplow_name            : chr [1:1907] "rudolph" "salt life" "blizzard"
"butter" …
 $ meaning                  : chr [1:1907] "The red nose cuts through any
storm." "We may not be near the ocean like everyone else with the stickers that
say Salt Life, but we have plenty of salt!" "This plow can handle any storm."
"It's amazing how the snow plows through snow like butter!" …
 $ winning_name             : logi [1:1907] FALSE FALSE FALSE FALSE FALSE FALSE
…
 - attr(*, "spec")=
  .. cols(
  ..   submission_number = col_double(),
  ..   submitter_name_anonymized = col_character(),
  ..   snowplow_name = col_character(),
  ..   meaning = col_character(),
  ..   winning_name = col_logical()
  .. )
```

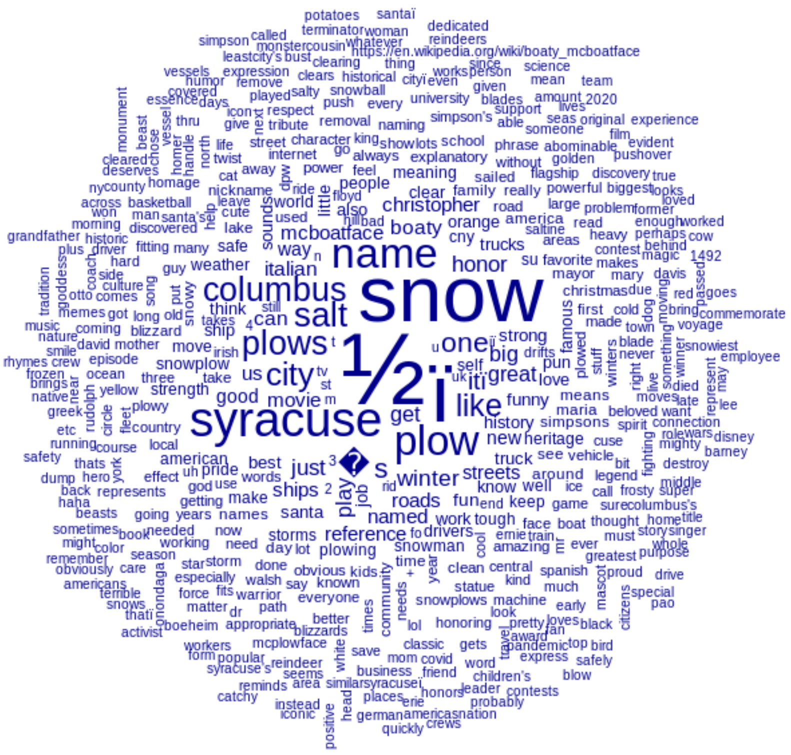
[11]:
```
corp <- corpus(df$meaning)
dfm <- dfm(corp, remove_punct=TRUE, remove=stopwords("english"))
```

```
Warning message:
"NA is replaced by empty string"
```

```
Warning message:
"'dfm.corpus()' is deprecated. Use 'tokens()' first."
Warning message:
"'…' should not be used for tokens() arguments; use 'tokens()' first."
Warning message:
"'remove' is deprecated; use dfm_remove() instead"
```

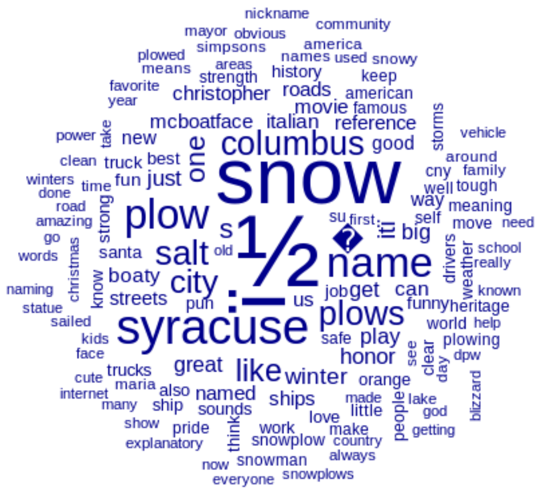
D. Plot a **word cloud** where a word is only represented if it appears **at least 2 times** in the corpus. **Hint:** use **textplot_wordcloud()**:

```
[12]: textplot_wordcloud(dfm, min_count = 2)
```



E. Next, **increase the minimum count to 10**. What happens to the word cloud? **Explain in a comment**.

[14]:
```
textplot_wordcloud(dfm, min_count = 10) #it shrinks as it excludes words with␣
↪frequencies under the new minimum
```



F. What are the top words in the word cloud? Explain in a brief comment.

[ ]:
```
#1/2, snow, plow, name, syracuse, salt, plows, etc.
```

1.2 Part 2: Create a sorted list of word counts from the reviews

G. Create a **named list of word counts by frequency**. **Hint**: use the functions as.matrix() and colSums()

[16]:
```
m <- as.matrix(dfm)
wordCounts <- colSums(m)
wordCounts <- sort(wordCounts, decreasing=TRUE)
```

H. Explain in a comment what you observed in the sorted list of word counts.

[19]:
```
head(wordCounts) #a decreasing list of the frequencies of non-stopwords in the␣
↪corpus
```

```
½        432 ï       336 snow       321 syracuse       174 name       143 plow       140
```

1.3 Part 3: Match the review words with positive and negative words

I. Read in the list of positive words (using the scan() function), as well as the negative words list: https://ist387.s3.us-east-2.amazonaws.com/data/positive-words.txt

https://ist387.s3.us-east-2.amazonaws.com/data/negative-words.txt

There should be 2006 positive words and 4783 negative words, so you may need to clean up these lists a bit.

[20]:
```
posWords <- scan("https://ist387.s3.us-east-2.amazonaws.com/data/positive-words.
↪txt", character(0), sep = "\n")
posWords <- posWords[-1:-34]
negWords <- scan("https://ist387.s3.us-east-2.amazonaws.com/data/negative-words.
↪txt", character(0), sep = "\n")
negWords <- negWords[-1:-34]
```

J. Here's a code example for matching the words from the name explanations (stored in **wordCounts**) to the list of positive words (stored in **posWords**):

[21]:
```
matchedP <- match(names(wordCounts), posWords, nomatch = 0)
```

Create a similar line of code to match the name explanations to the negative words.

[22]:
```
matchedN <- match(names(wordCounts), negWords, nomatch = 0)
```

K. Examine the contents of **matchedP**. What does each non-zero entry contain? How does that relate to **wordCounts** and **posWords**?

[24]:
```
head(matchedP, 50) #the non-zero numbers stored in this vector are indices␣
↪which correlate to positions on the posWords vector
```

```
1. 0 2. 0 3. 0 4. 0 5. 0 6. 0 7. 0 8. 0 9. 0 10. 0 11. 0 12. 1088 13. 0 14. 0 15. 0 16. 0 17. 0 18. 920
19. 0 20. 857 21. 0 22. 0 23. 0 24. 0 25. 0 26. 0 27. 0 28. 0 29. 0 30. 0 31. 0 32. 0 33. 0 34. 0 35. 0
36. 0 37. 832 38. 778 39. 0 40. 1701 41. 0 42. 196 43. 0 44. 0 45. 0 46. 0 47. 0 48. 0 49. 0 50. 1987
```

L. Use R to print out the positive words in the name explanation variable in **df**.

[25]:
```
posWords[matchedP[matchedP != 0]]
```

1. ’like’ 2. ’honor’ 3. ’great’ 4. ’good’ 5. ’fun’ 6. ’strong’ 7. ’best’ 8. ’work’ 9. ’love’ 10. ’clear’ 11. ’famous’ 12. ’safe’ 13. ’pride’ 14. ’tough’ 15. ’well’ 16. ’favorite’ 17. ’clean’ 18. ’amazing’ 19. ’cute’ 20. ’beloved’ 21. ’right’ 22. ’better’ 23. ’honoring’ 24. ’powerful’ 25. ’respect’ 26. ’homage’ 27. ’cool’ 28. ’appropriate’ 29. ’golden’ 30. ’classic’ 31. ’pretty’ 32. ’greatest’ 33. ’support’ 34. ’mighty’ 35. ’loves’ 36. ’works’ 37. ’magic’ 38. ’enough’ 39. ’clears’ 40. ’proud’ 41. ’won’ 42. ’cleared’ 43. ’winner’ 44. ’super’ 45. ’hero’ 46. ’smile’ 47. ’award’ 48. ’popular’ 49. ’top’ 50. ’humor’ 51. ’loved’ 52. ’dedicated’ 53. ’win’ 54. ’important’ 55. ’positive’ 56. ’liked’ 57. ’catchy’ 58. ’excellent’ 59. ’awesome’ 60. ’celebrate’ 61. ’hilarious’ 62. ’worked’ 63. ’courage’ 64. ’happy’ 65. ’uplifting’ 66. ’safely’ 67. ’lead’ 68. ’saint’ 69. ’trust’ 70. ’nice’ 71. ’benefit’ 72. ’honored’ 73. ’freedom’ 74. ’rich’ 75. ’awesomeness’ 76. ’neat’ 77. ’wins’ 78. ’gold’ 79. ’autonomous’ 80. ’bright’ 81. ’easy’ 82. ’protection’ 83. ’noble’ 84. ’winning’ 85. ’holy’ 86. ’free’ 87. ’recovery’ 88. ’boom’ 89. ’warm’ 90. ’beautiful’ 91. ’ready’ 92. ’prosperity’ 93. ’trophy’ 94. ’perseverance’ 95. ’loving’ 96. ’gifted’ 97. ’accurate’ 98. ’loyal’ 99. ’modern’ 100. ’spirited’ 101. ’awards’ 102. ’fantastic’ 103. ’winners’ 104. ’prefer’ 105. ’friendly’ 106. ’logical’ 107. ’motivated’ 108. ’respectfully’ 109. ’glory’ 110. ’encourage’ 111. ’crisp’ 112. ’brighten’ 113. ’consistent’ 114. ’easier’ 115. ’humorous’ 116. ’elite’ 117. ’sensation’ 118. ’clearer’ 119. ’helped’ 120. ’dignity’ 121. ’honest’ 122. ’awesomely’ 123. ’heroine’ 124. ’thrilled’ 125. ’lovable’ 126. ’grateful’ 127. ’unforgettable’ 128. ’fresh’ 129. ’grace’ 130. ’incredible’ 131. ’worth’ 132. ’proactive’ 133. ’fidelity’ 134. ’sweet’ 135. ’coolest’ 136. ’darling’ 137. ’wow’ 138. ’beauty’ 139. ’respectful’ 140. ’innovation’ 141. ’savings’ 142. ’envy’ 143. ’unparalleled’ 144. ’excellence’ 145. ’trusting’ 146. ’savior’ 147. ’recover’ 148. ’abounds’ 149. ’capable’ 150. ’clever’ 151. ’achievements’ 152. ’blossom’ 153. ’likes’ 154. ’cheer’ 155. ’miracle’ 156. ’backbone’ 157. ’unlimited’ 158. ’fair’ 159. ’enjoy’ 160. ’legendary’ 161. ’smooth’ 162. ’tenacity’ 163. ’dawn’ 164. ’significant’ 165. ’achievement’ 166. ’fame’ 167. ’talented’ 168. ’inspiring’ 169. ’genius’ 170. ’sturdy’ 171. ’accomplish’ 172. ’shiny’ 173. ’satisfy’ 174. ’faith’ 175. ’glow’ 176. ’hail’ 177. ’hardy’ 178. ’correctly’ 179. ’finest’ 180. ’wonderful’ 181. ’fastest’ 182. ’strongest’ 183. ’everlasting’ 184. ’excited’ 185. ’instantly’ 186. ’steady’ 187. ’continuity’ 188. ’smiles’ 189. ’capability’ 190. ’jolly’ 191. ’lucky’ 192. ’pleasant’ 193. ’accolades’ 194. ’brave’ 195. ’magical’ 196. ’fav’ 197. ’angel’ 198. ’success’ 199. ’instrumental’ 200. ’helping’ 201. ’supported’ 202. ’persevere’ 203. ’accomplishments’ 204. ’patriot’ 205. ’decent’ 206. ’protect’ 207. ’appeal’ 208. ’merit’ 209. ’smart’ 210. ’courageous’ 211. ’freedoms’

M. Use R to print out the total number of positive words in the name explanation variable in **df**.

[26]:
```
sum(wordCounts[matchedP != 0])
```

866

N. Repeat that process for the negative words you matched. Which negative words were in the name explanation variable, and what is their total number?

[27]:
```
negWords[matchedN[matchedN != 0]]
sum(wordCounts[matchedN != 0])
```

1. ’funny’ 2. ’cold’ 3. ’twist’ 4. ’hard’ 5. ’problem’ 6. ’abominable’ 7. ’bad’ 8. ’died’ 9. ’destroy’ 10. ’bust’ 11. ’terrible’ 12. ’frozen’ 13. ’monster’ 14. ’dump’ 15. ’busts’ 16. ’crush’ 17. ’blow’ 18. ’silly’ 19. ’messes’ 20. ’chilly’ 21. ’loud’ 22. ’evil’ 23. ’miser’ 24. ’joke’ 25. ’rough’ 26. ’crazy’ 27. ’killed’ 28. ’slow’ 29. ’dirt’ 30. ’bash’ 31. ’fear’ 32. ’stuck’ 33. ’challenging’ 34. ’erase’ 35. ’difficult’ 36. ’protest’ 37. ’defiance’ 38. ’avalanche’ 39. ’unfriendly’ 40. ’delay’ 41. ’critical’ 42. ’die’ 43. ’complaining’ 44. ’suffer’ 45. ’difficulty’ 46. ’upset’ 47. ’inexorable’ 48. ’object’ 49. ’inadequacy’ 50. ’scrap’ 51. ’pander’ 52. ’poorly’ 53. ’dope’ 54. ’naive’ 55. ’zombie’ 56. ’kills’ 57. ’undesirable’

58. ’dreary’ 59. ’worst’ 60. ’ironic’ 61. ’outbreak’ 62. ’fierce’ 63. ’sour’ 64. ’misfit’ 65. ’horrible’ 66. ’dead’ 67. ’loot’ 68. ’evasion’ 69. ’adversary’ 70. ’virus’ 71. ’infamous’ 72. ’nasty’ 73. ’bleeds’ 74. ’dumb’ 75. ’misery’ 76. ’limit’ 77. ’fall’ 78. ’unfortunately’ 79. ’fatally’ 80. ’disorder’ 81. ’ignorance’ 82. ’disabled’ 83. ’suffering’ 84. ’rust’ 85. ’ruining’ 86. ’worse’ 87. ’manipulate’ 88. ’mobster’ 89. ’crisis’ 90. ’pig’ 91. ’drunk’ 92. ’lone’ 93. ’freezing’ 94. ’enemies’ 95. ’hates’ 96. ’assault’ 97. ’mar’ 98. ’cancer’ 99. ’toughness’ 100. ’frost’ 101. ’apocalypse’ 102. ’damage’ 103. ’self-interest’ 104. ’ruins’ 105. ’cloudy’ 106. ’monstrous’ 107. ’disdain’ 108. ’tense’ 109. ’unknown’ 110. ’disrespect’ 111. ’hardships’ 112. ’treacherous’ 113. ’damn’ 114. ’restriction’ 115. ’rhetoric’ 116. ’puppet’ 117. ’doubt’ 118. ’deadly’ 119. ’moody’ 120. ’miss’ 121. ’dumps’ 122. ’cloud’ 123. ’death’ 124. ’inclement’ 125. ’destroyer’ 126. ’myth’ 127. ’trickery’ 128. ’rivalry’ 129. ’unusually’ 130. ’badly’ 131. ’abused’ 132. ’rival’ 133. ’intimidating’ 134. ’comical’ 135. ’forged’ 136. ’fallen’ 137. ’sarcasm’ 138. ’unpleasant’ 139. ’alarm’ 140. ’standstill’ 141. ’worry’ 142. ’scare’ 143. ’limited’ 144. ’dark’ 145. ’pity’ 146. ’harsh’ 147. ’despise’ 148. ’excuse’

255

O. Write a comment describing what you found after matching positive and negative words. Which group is more common in this dataset?

```
[28]: #there are many more (over 3x as many) positive words than negative words -␣
      ↪this might have been expected since it is a contest
```