Further Free Response Exploration

Jason Cairns

May 3, 2019

1 Background

Adding to the previous exploration, there were subtleties felt worth exploring further; namely, the base bigram count, and the `textrank_keywords` function, as well as their relation.

2 Bigram Count

Let's look at the most common bigrams in the Cancer Society data

```
library(tidyverse)
library(readxl)
library(tidytext)

cancersoc <- read_excel("../data/raw/Cancer Soc.xlsx")

cancersoc %>%
  mutate(id = row_number()) %>%
  unnest_tokens(bigram, sq6a, token = "ngrams", n = 2) %>%
  count(bigram, sort = TRUE) %>%
  head
```

bigram	n
nil	589
to help	284
to give	69
a good	67
help others	65
good cause	61

The previous attempt at assessing bigrams used the following code:

```
cancer_soc_bigrams <-
  cancersoc %>%
  select(sq6a) %>%
  mutate(id = row_number()) %>%
  unnest_tokens(bigram, sq6a, token = "ngrams", n = 2) %>%
  na.omit()
```

```r

bigrams_separated <- cancer_soc_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")

# new bigram counts:
bigram_counts <- bigrams_united %>%
  count(bigram, sort = TRUE)

head(bigram_counts)
```

bigram	n
salvation army	19
raise money	9
donated money	5
red cross	5
spare time	5
cancer society	4

These results are completely different. On closer inspection, it seems that stopword removal got rid of a lot of bigrams that would otherwise be useful. This raises the question of whether stopword removal is so essential when ngrams are needed?

3 Further exploration of `textrank_keywords`

The results of `textrank_keywords` appeared extremely useful, but the lingering question was how they were generated. The documentation of `textrank_keywords` was helpful in this respect, showing that

> In order to find relevant keywords, the textrank algorithm constructs a word network. This network is constructed by looking which words follow one another. A link is set up between two words if they follow one another, the link gets a higher weight if these 2 words occur more frequenctly next to each other in the text. On top of the resulting network the 'Pagerank' algorithm is applied to get the importance of each word. The top 1/3 of all these words are kept and are considered relevant. After this, a keywords table is constructed by combining the relevant words together if they appear following one another in the text.