Lab 3: Basic Text Analysis

Srishti Saha (ss1078)

13 February, 2020

Question 1

Steps included (on the reviews dataset)

- Removal of stop words
- Remove intra and inter-word punctuations
- Removal of white spaces
- Lemmatization
- Converting word case to lower

Steps excluded

1. We will not be executing **removal of numbers** as they are present in all documents and refer to the index/ ID of the document. Moreover, numbers within the text might also carry important information. For instance, there are mentions of years- like 2002 and 10000 BC to refer to particular time lines. It is not wise to remove the same.

Question 2- Corpus

```r
# creating a corpus
movies_review_corpus <- Corpus(VectorSource(as.vector(movie_review$review)))
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

Question 3- Tidy Text

```r
# transforming into tidytext format
tidy_movie_reviews<- movie_review %>%
    select(id,review, sentiment) %>%
    unnest_tokens("word", review)
head(tidy_movie_reviews,10)
```

```
##         id sentiment   word
## 1   5814_8         1   with
## 1.1 5814_8         1    all
## 1.2 5814_8         1   this
## 1.3 5814_8         1  stuff
## 1.4 5814_8         1  going
## 1.5 5814_8         1   down
## 1.6 5814_8         1     at
## 1.7 5814_8         1    the
## 1.8 5814_8         1 moment
## 1.9 5814_8         1   with
```

```r
tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 42,652 x 2
##    word      n
##    <chr> <int>
##  1 the   68038
##  2 and   33657
##  3 a     33105
##  4 of    29512
##  5 to    27603
##  6 is    21658
##  7 br    20824
##  8 in    19107
##  9 it    15935
## 10 i     15465
## # ... with 42,642 more rows
```

Question 4- Pre processing on corpus format

Punctuation

```r
# removing punctuation
movies_review_corpus <- tm_map(movies_review_corpus,content_transformer(removePunctuation))
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

Stop words

```r
# create a concatenated dataset of stopwords to be removed
stopwords1<- c(stopwords("english"),c("anyway","always"))
# remove the stop words in the list above
movies_review_corpus <- tm_map(movies_review_corpus, removeWords, stopwords1)
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

White spaces

```r
# removing white spaces
movies_review_corpus <- tm_map(movies_review_corpus, content_transformer(stripWhitespace))
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

Lemmatization

```r
#Lemmatization
movies_review_corpus <- tm_map(movies_review_corpus, lemmatize_strings)
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

Converting to lower case

```
# lower case conersion
movies_review_corpus <- tm_map(movies_review_corpus,  content_transformer(tolower))
movies_review_corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 5000
```

Question 5- Pre processing on tidytext format

Punctuation

Although interword punctuation is removed in tidytext automatically, we will remove intraword punctuations separately.

```
# removing punctuation (intraword)
tidy_movie_reviews$word<-removePunctuation(tidy_movie_reviews$word,preserve_intra_word_contractions = FALSE,
                  preserve_intra_word_dashes = FALSE)

tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 41,779 x 2
##    word      n
##    <chr> <int>
##  1 the   68038
##  2 and   33657
##  3 a     33105
##  4 of    29512
##  5 to    27603
##  6 is    21659
##  7 br    20824
##  8 in    19107
##  9 it    15936
## 10 i     15465
## # ... with 41,769 more rows
```

Stop words

```
data("stop_words")

# remove the stop words in the list above
tidy_movie_reviews<-tidy_movie_reviews %>%
      anti_join(stop_words)
typeof(tidy_movie_reviews)
```

```
## [1] "list"
```

```r
tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 41,128 x 2
##    word       n
##    <chr>  <int>
##  1 br     20824
##  2 movie   8485
##  3 film    8025
##  4 time    2570
##  5 story   2353
##  6 people  1893
##  7 bad     1802
##  8 dont    1652
##  9 films   1634
## 10 movies  1618
## # ... with 41,118 more rows
```

It looks like 'br' from '' tags in html also feature as the top-most list of frequent words here. We will thus create an additional list and use the anti-join to remove those.

```r
# creating an additional list of stop words to be removed
stop_words2= as.data.frame("br")
names(stop_words2)[1]<- "word"

# use the same anti join method to remove the additional stop words
tidy_movie_reviews<-tidy_movie_reviews %>%
      anti_join(stop_words2)


tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 41,127 x 2
##    word           n
##    <chr>      <int>
##  1 movie       8485
##  2 film        8025
##  3 time        2570
##  4 story       2353
##  5 people      1893
##  6 bad         1802
##  7 dont        1652
##  8 films       1634
##  9 movies      1618
## 10 characters  1510
## # ... with 41,117 more rows
```

White Spaces

```r
# removing white spaces
tidy_movie_reviews$word <- gsub("\\s+","",tidy_movie_reviews$word)
tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 41,127 x 2
##    word           n
##    <chr>      <int>
##  1 movie       8485
##  2 film        8025
##  3 time        2570
##  4 story       2353
##  5 people      1893
##  6 bad         1802
##  7 dont        1652
##  8 films       1634
##  9 movies      1618
## 10 characters  1510
## # ... with 41,117 more rows
```

Lemmatization

```
#Lemmatization
tidy_movie_reviews<-tidy_movie_reviews %>%
  filter(word %in% tidy_movie_reviews$word) %>%
  distinct() %>%
  mutate(word = textstem::lemmatize_words(word))


tidy_movie_reviews %>%
  count(word) %>%
    arrange(desc(n))
```

```
## # A tibble: 32,263 x 2
##    word          n
##    <chr>     <int>
##  1 movie      4143
##  2 film       4059
##  3 time       2361
##  4 watch      2258
##  5 character  2038
##  6 bad        1895
##  7 story      1676
##  8 act        1472
##  9 scene      1462
## 10 play       1404
## # ... with 32,253 more rows
```

Converting to lower case

This step happens automatically in the tidytext format

Question 6- Document Term matrix from corpus

```
## DTM from corpus
reviews_DTM <- DocumentTermMatrix(movies_review_corpus, control = list(wordLengths = c(2, Inf)))

inspect(reviews_DTM[1:5,3:9])
```

```
## <<DocumentTermMatrix (documents: 5, terms: 7)>>
## Non-/sparse entries: 9/26
## Sparsity           : 74%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs alone also another attention away bad because
##    1     1    2       1         1    1   3       1
##    2     0    0       0         0    0   0       0
##    3     0    0       0         0    0   0       0
##    4     0    1       2         0    0   0       0
##    5     0    0       0         0    0   0       0
```

Question 7- Document Term matrix from tidytext

```
## DTM from tidytext
tidy_reviews_DTM<-
  tidy_movie_reviews %>%
  count(id, word) %>%
  cast_dtm(id, word, n)

inspect(tidy_reviews_DTM[1:5,3:9])
```

```
## <<DocumentTermMatrix (documents: 5, terms: 7)>>
## Non-/sparse entries: 7/28
## Sparsity           : 80%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##          Terms
## Docs      bathroom bet blaze bolt bracelet brook building
##   10000_8        1   2     1    2        1     1        1
##   10001_4        0   0     0    0        0     0        0
##   10004_3        0   0     0    0        0     0        0
##   10004_8        0   0     0    0        0     0        0
##   10006_4        0   0     0    0        0     0        0
```

Question 8- Difference between corpus and tidytext

The biggest difference is in the **structure of the corpus and the tidytext format**. The *corpus also retains every document in its entirety and does not split it into individual terms* or words. The tidytext format *breaks each document into individual terms and assign each term and its frequency to a row.* As a result, we can not visualize the entire document as whole. We see that the corpus document term matrix matches the document-term pairs to IDs defined by the row titles (or indices). However, the document term matrix obtained from the tidytext format uses review ID in the original dataset. This would make it difficult to compare the two results.

Furthermore, if we look at the **dimensions of the data** in the corpus format, it is of the size of the number of tweets (5000) while the dimensions of the tidytext is of the order of ~380k rows. This difference arises because in the corpus method, the (document) reviews are kept intact and every record corresponds to a document. The tidytext method, however, breaks the document into multiple records with each record corresponding to a word (term) in the document. Thus, the dimension of this dataset is also higher.

Moreover, the document-term matrix from the tidytext format would be convenient to **map the results back to the original dataset** as the IDs for the document are retained from the original dataset. However, for documents with large number of words or with data that has a lot of documents, the size of the dataset will be huge.