Text Preprocessing using Tidytext

Promothesh Chatterjee*

Alternative Representation of Text into Data

Previously, we have seen how document term matrix representation of text data. While we will use that representation for rest of the semester, there is another format which is fairly widely used, the tidy data representation. In the tidy data format each word is represented in a different row.

Though, please note most computational packages in R use the DTM/DFM approach, tidy text package is supported by the tidyverse view of R.

Representing Text in Tidy Data Format

Let's start with the simplest approach first based on r package tidytext (see the excellent book by Julia Silge and David Robinson– tidytextmining.com for details).

Tidytext package (author: Julia Silge) uses the tidyverse approach. Hadley Wickham, the architect of tidy approach (and lots of tidyverse packages) defines tidy data such that

Each variable is a column

Each observation is a row

Each type of observational unit is a table

The tidytext package command "unnest_tokens" does lots of preprocessing and converts text into tidy format by stripping punctuation, converting tokens (more about this when we talk about the second approach using quanteda and tm packages) to lowercase and retains the line numbers the words came from. Let's first create tidy text data. Note we are using read_csv function from tidyverse as opposed to readtext that we saw in the Quanteda notes.

```
library(tidyverse)# For loading bunch of packages such as dplyr, stringi etc.
library(tidytext) # for tidy data format text processing

# Load up the .CSV data.
hotel_raw <- read_csv("C:/Users/u0474728/Dropbox/Utah Department Stuff/Teaching/Text Analysis/Summer 20

set.seed(1245654)# for replicability
hotel_raw <- slice_sample(hotel_raw,n=5000)# take a small sample

hotel_raw_token<-hotel_raw %>% unnest_tokens(word, Description) # Description variable is where the rev

head(hotel_raw_token)
```

```
## # A tibble: 6 x 5
##   User_ID Browser_Used Device_Used Is_Response word
##   <chr>   <chr>        <chr>       <chr>       <chr>
## 1 id36745 Firefox      Desktop     not happy   i
## 2 id36745 Firefox      Desktop     not happy   had
## 3 id36745 Firefox      Desktop     not happy   some
```

```
## 4 id36745 Firefox      Desktop     not happy   reservations
## 5 id36745 Firefox      Desktop     not happy   about
## 6 id36745 Firefox      Desktop     not happy   staying
```

As can be seen from the output, unnest_tokens takes each word from each review and puts them in a column called 'word'. The variable User_ID refers to a particular user tweet and so on. Essentially the data in a long format. As you can see, the 'word' column has many words such as 'i', 'had' that have not much unique meanings. Often in text analysis, we will want to remove these stop words. We can remove stop words (kept in the tidytext dataset stop_words) with an anti_join().

```
data(stop_words) # comes from tidytext package
hotel_raw_token <- hotel_raw_token %>% anti_join(stop_words)
# anti_join takes the dataframe and eliminates the common words
```

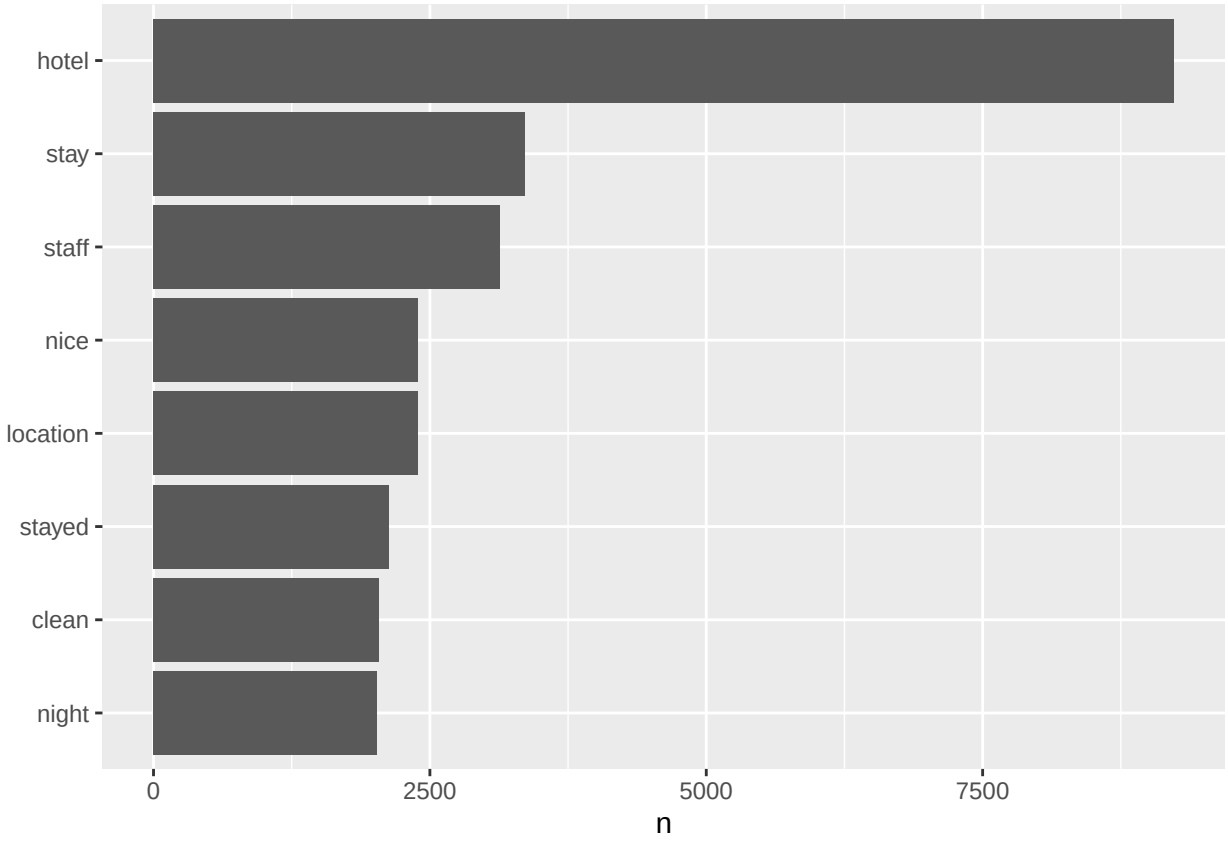
We can also use dplyr's count() to find the most common words.

```
hotel_raw_token %>% count(word, sort = TRUE)%>% filter(n > 2000)
```

```
## # A tibble: 8 x 2
##   word         n
##   <chr>    <int>
## 1 hotel     9234
## 2 stay      3359
## 3 staff     3132
## 4 nice      2394
## 5 location  2392
## 6 stayed    2129
## 7 clean     2036
## 8 night     2023
```

Let's plot these using ggplot2.

```
hotel_raw_token %>% count(word, sort = TRUE) %>%
    filter(n > 2000) %>% mutate(word = reorder(word, n)) %>%
    ggplot(aes(word, n)) + geom_col() + xlab(NULL) + coord_flip()
```




```
#we plotted only those words that appeared more than 2000 times
```

Thus, we can do much exploratory and advanced data analysis with tidytext format (including sentiment analysis, topic modeling etc. that we do in later chapters). The problem is that many text packages used for other types of analysis do not use the tidy text format. However tidytext package has some neat commands (such as cast_dtm() for converting to a Document Term Matrix object from tm package, and cast_dfm() for converting to a dfm object from quanteda package) to turn the tidy text format to other formats that these other packages use. We will come back to this later.

Some General Resources for Text Analytics in R

Text analytics requires a lot of data exploration, data pre-processing and data wrangling. There are many packages in the R ecosystem for performing text analytics. Some of the popular ones are:tm, quanteda, tidytext. We will mostly use the quanteda and tidytext packages but use others also as and when needed.

For a good representation of processes in text analytics see: https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk

Additional in-depth details about text pre-processing can be found in Chapter 2 of Jurafsky's book: http://web.stanford.edu/~jurafsky/slp3/2.pdf

For a good overview of R packages in Text Analytics, see: http://www.bnosac.be/index.php/blog/87-an-overview-of-the-nlp-ecosystem-in-r-nlproc-textasdata

For a wonderful overview of text analysis in R, see: http://eprints.lse.ac.uk/86659/1/Benoit_Text%20analysis%20in%20R_2018.pdf