This is one of the frequent questions I’ve heard from the first timer NLP / Text Analytics – programmers (or as the world likes it to be called “Data Scientists”).

**Prerequisite**

For simplicity, this post assumes that you already know how to install a package and so you’ve got tidytext installed on your R machine.

install.packages("tidytext")

**Loading the Library**

Let’s start with loading the tidytext library.

library(tidytext)

**Extracting App Reviews**

We’ll use the R-package itunesr for downloading iOS App Reviews on which we’ll perform Simple Text Analysis (unigrams, bigrams, n-grams). getReviews() funciton of itunesr helps us in extracting reviews of Medium iOS App.

library(itunesr)

library(tidyverse)

# Extracting Medium iOS App Reviews

medium <- getReviews("828256236","us",1)

**Overview of the extract App Reviews**

As usual, we’ll start with seeing what’s head of the dataframe.

head(medium)

## Title

## 1 Great source...

## 2 I love it!

## 3 Medium Provide wide variety of articles

## 4 A bargain at 50$

## 5 Awesome

## 6 Love Medium

## URL Name

## 1 <https://itunes.apple.com/us/reviews/id14871198> Helpful Program

## 2 <https://itunes.apple.com/us/reviews/id622727268> tacos are lit

## 3 <https://itunes.apple.com/us/reviews/id124091445> Anjan12344321

## 4 <https://itunes.apple.com/us/reviews/id105720950> Judster64

## 5 <https://itunes.apple.com/us/reviews/id39489978> jalton

## 6 <https://itunes.apple.com/us/reviews/id26999143> girlbakespies

## App\_Version Rating

## 1 3.89 5

## 2 3.89 5

## 3 3.89 5

## 4 3.89 5

## 5 3.89 4

## 6 3.88 5

## Review

## 1 Great source for top content and food for mind and soul.

## 2 ⠀⠀⠀⠀

## 3 I am feeling happy about Medium yearly subscription, Each penny os worth. Medium provides wide range of articles. I really like some of the authors! I am trying to start writing my own articles, this is the best forum to express your opinions and based on feedback you can improve your self.

## 4 The most interesting articles at your fingertips. No ads. Love it.

## 5 Just need to be able to bookmark without crashing the app and it’ll be 5 stars.

## 6 I am on my second month.I am getting back into writing again and Medium is a brilliant community of writers. I Highly recommend it for entertainment and an outanding information resource #READMORE

## Date

## 1 2019-08-04 15:09:50

## 2 2019-08-04 10:04:59

## 3 2019-08-03 03:10:22

## 4 2019-08-01 14:40:14

## 5 2019-07-31 23:56:41

## 6 2019-07-31 03:15:44

Now, we know that there are two Text Columns of our interest – Title and Review.

To make our n-grams analysis a bit more meaningful, we’ll extract only the positive reviews (5-star) to see what’s good people are writing about Medium iOS App. To make a better sense of the filter we’ve to use let’s see the split of Rating.

table(medium$Rating)

##

## 1 3 4 5

## 5 5 5 34

So, 5-star is the major component in the text reviews we extract and we’re good to go filtering only 5-star.We’ll pick Review from that and also we’ll specify it only for Rating == 5. Since we need a dataframe (or tibble) for tidytext to process it, we’ll put these 5-star reviews as a new column in a new dataframe.

reviews <- data.frame(txt = medium$Review[medium$Rating==5],

stringsAsFactors = FALSE)

**Tokens**

Tokenization in NLP is the process of splitting a text corpus based on some splitting factor – It could be Word Tokens or Sentence Tokens or based on some advanced alogrithm to split a conversation. In this process, we’ll just simply do word tokenization.

reviews %>%

unnest\_tokens(output = word, input = txt) %>%

head()

## word

## 1 great

## 1.1 source

## 1.2 for

## 1.3 top

## 1.4 content

## 1.5 and

As you can see above, unnest\_tokens() is the function that’ll help us in this tokenization process. Since it supports %>% pipe operator, the first argument of the function is a dataframe or tibble, the second argument output is the name of the output (new) column where the tokenized words are going to be put in. The third column input is where the input text is fed in.

Now, this is what unigrams are for this Medium iOS App Reviews. As with many other data science projects, Data like this is not useful unless it’s visualized in a way to look at insights.

reviews %>%

unnest\_tokens(output = word, input = txt) %>%

count(word, sort = TRUE)

## # A tibble: 444 x 2

## word n

##

## 1 the 45

## 2 i 35

## 3 and 34

## 4 of 27

## 5 to 27

## 6 a 18

## 7 it 14

## 8 medium 14

## 9 this 13

## 10 articles 12

## # … with 434 more rows

Roughly, looking at the most frequently appeared unigram we end up with the,i,and and this is one of those places where we need to *remove stopwords*

**Stopword Removal**

Fortunately, tidytext helps us in removing stopwords by having a dataframe of stopwords from multiple lexicons. With that, we can use anti\_join for picking the words (that are present in the left df (reviews) but not present in the right df (stop\_words)).

reviews %>%

unnest\_tokens(output = word, input = txt) %>%

anti\_join(stop\_words) %>%

count(word, sort = TRUE)

## Joining, by = "word"

## # A tibble: 280 x 2

## word n

##

## 1 medium 14

## 2 articles 12

## 3 app 9

## 4 reading 9

## 5 content 6

## 6 love 5

## 7 read 5

## 8 article 4

## 9 enjoy 4

## 10 i’ve 4

## # … with 270 more rows

With that stop word removal, now we can see better represenation of most frequently appearing unigrams in the reviews.

**unigram Visualziation**

We’ve got our data in the shape that we want so, let’s go ahead and visualize it. To keep the pipeline intact, I’m not creating any temporary object to store the previous output and just simply continue using the same. Also too many bars (words) wouldn’t make any sense (except resulting in a shabby plot), We’ll filter taking the top 10 words

reviews %>%

unnest\_tokens(output = word, input = txt) %>%

anti\_join(stop\_words) %>%

count(word, sort = TRUE) %>%

slice(1:10) %>%

ggplot() + geom\_bar(aes(word, n), stat = "identity", fill = "#de5833") +

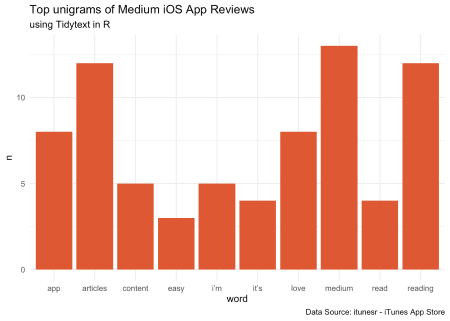
theme\_minimal() +

labs(title = "Top unigrams of Medium iOS App Reviews",

subtitle = "using Tidytext in R",

caption = "Data Source: itunesr - iTunes App Store")

## Joining, by = "word"



**Bigrams & N-grams**

Now that we’ve got the core code for unigram visualization set up. We can slightly modify the same – just by adding a new argument n=2 and token="ngrams" to the tokenization process to extract n-gram. 2 for bigram and 3 trigram – or n of your interest. But remember, large n-values may not useful as the smaller values.

Doing this naively also has a catch and the catch is – the stop-word removal process we used above was using anti\_join which wouldn’t be supported in this process since we’ve a bigram (two-word combination separated by a space). So, we’ll separate the word by space and then filter out the stop words in both word1 and word2 and then unite them back – which gives us the bigram after stop-word removal. This is the process that you might have to carry out when you are dealing with n-grams.

reviews %>%

unnest\_tokens(word, txt, token = "ngrams", n = 2) %>%

separate(word, c("word1", "word2"), sep = " ") %>%

filter(!word1 %in% stop\_words$word) %>%

filter(!word2 %in% stop\_words$word) %>%

unite(word,word1, word2, sep = " ") %>%

count(word, sort = TRUE) %>%

slice(1:10) %>%

ggplot() + geom\_bar(aes(word, n), stat = "identity", fill = "#de5833") +

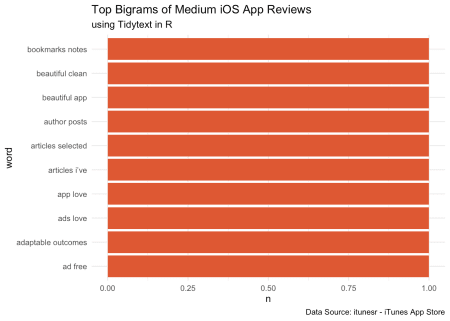
theme\_minimal() +

coord\_flip() +

labs(title = "Top Bigrams of Medium iOS App Reviews",

subtitle = "using Tidytext in R",

caption = "Data Source: itunesr - iTunes App Store")



**Summary**

This particular assignment that may not reveal some meaningful insights as we started with less data, but this is really useful when you have a decent amount of text corpus and this simple analysis of unigram, bigram (n-gram analysis) can reveal something business-worthy (let’s say in Customer Service, App Development or in multiple other use-cases).