Tokenization

CHINDU

Text preprocessing

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded.

Reading data and setting the correct structure

```
library(widyr)
library(tm)
```

```
## Loading required package: NLP
```

```
library(tidytext)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(SnowballC)
datax<- read.csv("Womens Clothing ECommerce Reviews.csv")
str(datax)
```

```
## 'data.frame':    23486 obs. of  10 variables:
##  $ Clothing.ID            : int  0 1 1 1 2 3 4 5 6 7 ...
##  $ Age                    : int  26 50 36 24 28 36 28 39 39 39 ...
##  $ Title                  : Factor w/ 13994 levels "","\"beach business\"",..: 1 7365 11540 6984 501
##  $ Review.Text            : Factor w/ 22635 levels "","- this really is lovely. the overall design fi
##  $ Rating                 : int  5 5 5 2 4 5 5 5 5 5 ...
##  $ Recommended.IND        : int  1 1 1 0 1 1 1 1 1 1 ...
##  $ Positive.Feedback.Count: int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Division.Name          : Factor w/ 4 levels "","General","General Petite",..: 2 4 4 4 2 2 2 2 2 2
##  $ Department.Name        : Factor w/ 7 levels "","Bottoms","Dresses",..: 5 4 4 4 6 6 6 6 6 5 ...
##  $ Class.Name             : Factor w/ 21 levels "","Blouses","Casual bottoms",..: 14 11 11 11 10 19 1
```

We need to change our text variable from a factor to a character for analysis

```
# Subset data to only the fields requiried
data<-datax %>% select(Clothing.ID,Review.Text)
data$Review.Text<-as.character((data$Review.Text))
data$Clothing.ID<-as.factor(data$Clothing.ID)
str(data)
```

```
## 'data.frame':    23486 obs. of  2 variables:
##  $ Clothing.ID: Factor w/ 1206 levels "0","1","2","3",..: 1 2 2 2 3 4 5 6 7 8 ...
##  $ Review.Text: chr  "" "Originally i bought this in black and white. recently purchased several more
```

Tokenization

Split the data into sentences (This is usually useful for long text documents)

```
data %>%
  unnest_tokens(output = "sentences", input = Review.Text, token = "sentences")%>%
count(Clothing.ID,sentences,sort=TRUE)
```

```
## # A tibble: 26,086 x 3
##    Clothing.ID sentences               n
##    <fct>       <chr>               <int>
##  1 1094        i love this dress!      9
##  2 1078        i love this dress!      8
##  3 1099        i love this dress!      7
##  4 829         love this top!          5
##  5 872         i love this top!        5
##  6 1078        love this dress!        5
##  7 835         check!                  4
##  8 862         i love this top!        4
##  9 862         love this top!          4
## 10 1059        i love these pants!     4
## # ... with 26,076 more rows
```

Split the data into words (This is a common approach)

```
data %>%
  unnest_tokens(output = "word", input = Review.Text, token = "words")%>%
  count(Clothing.ID,word,sort=TRUE)
```

```
## # A tibble: 272,873 x 3
##    Clothing.ID word      n
##    <fct>       <chr> <int>
##  1 1078        the    3658
##  2 1094        the    2817
##  3 1078        i      2683
```

```
##  4 862         the    2412
##  5 1078        and    2252
##  6 1078        it     2159
##  7 1081        the    2010
##  8 1094        i      1969
##  9 1110        the    1967
## 10 1078        a      1863
## # ... with 272,863 more rows
```

Split the data using regular expressions

```
data %>%
  unnest_tokens(output = "regexsplit", input = Review.Text,
                token = "regex", pattern = "\\.") %>%
  count(Clothing.ID,regexsplit,sort=TRUE)
```

```
## # A tibble: 93,975 x 3
##    Clothing.ID regexsplit                n
##    <fct>       <chr>                 <int>
##  1 862         "\ni love this top"       7
##  2 1078        "\ni love this dress"     7
##  3 1094        " "                       7
##  4 872         "\nlove this top"         4
##  5 1056        "\n\n2"                   4
##  6 1056        "\n\n3"                   4
##  7 1078        "\nlove this dress"       4
##  8 1081        "\nlove this dress"       4
##  9 1083        "\ni love this dress"     4
## 10 1094        " fits true to size"      4
## # ... with 93,965 more rows
```

Filter data identify sentenses which mentions love regardless of capital letter

```
Filtered_data<-data %>%
  unnest_tokens (output= "Love",Review.Text,
                 token="regex",pattern="(?i)love") %>%
  #to skip first token
  slice(2:n())
```

2. Normalization

Normalization generally reers to a series of related tasks meant to pull all text on a level playing field. Example converting all text to lower casse, removing punctuations, converting number to their word equivalents etc. Normalization puts all words on equal footing and allows processing to proceed uniformly.

```
# First tokenize by words
clothes <- data %>%
  unnest_tokens(word, Review.Text)
```

```
# Print the word frequencies
clothes %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 14,804 x 2
##    word      n
##    <chr> <int>
##  1 the   76114
##  2 i     59237
##  3 and   49007
##  4 a     43012
##  5 it    42800
##  6 is    30640
##  7 this  25751
##  8 to    24581
##  9 in    20721
## 10 but   16554
## # ... with 14,794 more rows
```

The top words are all stopwords and this is not going to be useful for the analysis.

Removing stop words

```
# Remove stop words, using `stop_words` from tidytext
clothes<-clothes %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
clothes %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 14,143 x 2
##    word           n
##    <chr>      <int>
##  1 dress      10553
##  2 love        8948
##  3 size        8768
##  4 top         7405
##  5 fit         7318
##  6 wear        6439
##  7 fabric      4790
##  8 color       4605
##  9 perfect     3772
## 10 flattering  3517
## # ... with 14,133 more rows
```

##Custom stop words (to remove words that you feel is not useful in this analysis) lets remove the word "online"

```
custom<- add_row(stop_words,word= "online",lexicon="custom")
Custom_clothes<- clothes %>%
  anti_join(custom)
```

```
## Joining, by = "word"
```

Stemming

Stemming is the process of eliminating affixes from a word in order to obtain a word stem.

```
# Perform stemming
stemmed_clothes <- Custom_clothes %>%
  mutate(word = wordStem(word))

# Print the old word frequencies
Custom_clothes %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 14,142 x 2
##    word           n
##    <chr>      <int>
##  1 dress      10553
##  2 love        8948
##  3 size        8768
##  4 top         7405
##  5 fit         7318
##  6 wear        6439
##  7 fabric      4790
##  8 color       4605
##  9 perfect     3772
## 10 flattering  3517
## # ... with 14,132 more rows
```

```
# Print the stemmed word frequencies
stemmed_clothes %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 10,183 x 2
##    word        n
##    <chr>   <int>
##  1 dress   12173
##  2 fit     11504
##  3 love    11391
##  4 size    10716
##  5 top      8360
##  6 wear     8075
##  7 color    7299
##  8 fabric   4885
##  9 perfect  3852
## 10 nice     3819
## # ... with 10,173 more rows
```

Notice how the top words changed when we applied stemming

Lemmatization

```
library(textstem)
```

```
## Loading required package: koRpus.lang.en
```

```
## Loading required package: koRpus
```

```
## Loading required package: sylly
```

```
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
```

```
lemmatize_words(Custom_clothes)%>%
  count(word, sort = TRUE)
```

```
## # A tibble: 14,142 x 2
##    word           n
##    <chr>      <int>
##  1 dress      10553
##  2 love        8948
##  3 size        8768
##  4 top         7405
##  5 fit         7318
##  6 wear        6439
##  7 fabric      4790
##  8 color       4605
##  9 perfect     3772
## 10 flattering  3517
## # ... with 14,132 more rows
```

In this data, lemmatization did not make any changes to the data.

Lets explore the difference between lemmatization and stemming

```
dw <- c('driver', 'drive', 'drove', 'driven', 'drives', 'driving')

stem_words(dw)
```

```
## [1] "driver" "drive"  "drove"  "driven" "drive"  "drive"
```

```
lemmatize_words(dw)
```

```
## [1] "driver" "drive"  "drive"  "drive"  "drive"  "drive"
```

```
bw <- c('are', 'am', 'being', 'been', 'be')

stem_words(bw)
```

```
## [1] "ar"   "am"   "be"   "been" "be"
```

```
lemmatize_words(bw)
```

```
## [1] "be" "be" "be" "be" "be"
```

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma . If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Linguistic processing for stemming or lemmatization is often done by an additional plug-in component to the indexing process, and a number of such components exist, both commercial and open-source.

here is another example

```
y <- c(
    "Stemming refers to a crude heuristic process that chops off the ends of words.",
     "Lemmatization refers to doing things properly with the use of a vocabulary and morphological anal
)

stem_strings(y)
```

```
## [1] "Stem refer to a crude heurist process that chop off the end of word."
## [2] "Lemmat refer to do thing properli with the us of a vocabulari and morpholog analysi of word"
```

```
lemmatize_strings(y)
```

```
## [1] "stem refer to a crude heuristic process that chop off the end of word."
## [2] "Lemmatization refer to do thing properly with the use of a vocabulary and morphological analysis
```