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  767 1080 1077 1049 847 1080 858 858 1077 1077 ...
##  $ Age                    : int  33 34 60 50 47 49 39 39 24 34 ...
##  $ Title                  : Factor w/ 13994 levels "","\"beach business\"",..: 1 1 11451 8055 4365 8
##  $ Review.Text            : Factor w/ 22635 levels "","- this really is lovely. the overall design fi
##  $ Rating                 : int  4 5 3 5 5 2 5 4 5 5 ...
##  $ Recommended.IND        : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ Positive.Feedback.Count: int  0 4 0 0 6 4 1 4 0 0 ...
##  $ Division.Name          : Factor w/ 4 levels "","General","General Petite",..: 4 2 2 3 2 2 3 3 2 2
##  $ Department.Name        : Factor w/ 7 levels "","Bottoms","Dresses",..: 4 3 3 2 6 3 6 6 3 3 ...
##  $ Class.Name             : Factor w/ 21 levels "","Blouses","Casual bottoms",..: 7 5 5 15 2 5 10 10
```

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))
head(data)
```

```
##   Clothing.ID
## 1         767
## 2        1080
## 3        1077
## 4        1049
## 5         847
## 6        1080
##
## 1
## 2
## 3 I had such high hopes for this dress and really wanted it to work for me. i initially ordered the p
## 4
## 5
## 6             I love tracy reese dresses, but this one is not for the very petite. i am just under 5
```

Tokenization

Split the data into sentences

```
data %>%
  unnest_tokens(output = "sentences", input = Review.Text, token = "sentences")%>%
  # Count sentences using the the clothing>ID column
count(Clothing.ID)
```

```
## # A tibble: 1,180 x 2
##    Clothing.ID     n
##          <int> <int>
##  1           1     4
##  2           2     2
##  3           3     1
##  4           4     2
##  5           5     1
##  6           7     1
##  7           8    11
##  8           9     1
##  9          10     1
## 10          11     1
## # ... with 1,170 more rows
```

Split the data into words

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

```
## # A tibble: 1,179 x 2
##    Clothing.ID     n
##          <int> <int>
##  1           1   111
##  2           2    71
##  3           3    13
##  4           4    85
##  5           5    40
##  6           7    41
##  7           8   467
##  8           9    28
##  9          10    19
## 10          11    50
## # ... with 1,169 more rows
```

Split the data using regular expressions

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

```
## # A tibble: 1,180 x 2
##    Clothing.ID     n
##          <int> <int>
##  1        1078  4155
##  2         862  3157
##  3        1094  3103
##  4        1081  2360
##  5        1110  2117
##  6         829  2108
##  7         872  2066
##  8         868  1775
##  9         895  1547
## 10         936  1545
## # ... with 1,170 more rows
```

Filter data to first 100 customer id and identify sentenses which mentions love regardless of capital letter

```
Filtered_data<-data %>%
  filter(Clothing.ID <100) %>%
```

```
  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
```

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
```

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 usually refers to a crude heuristic process that chops off the ends of words.",
     "Lemmatization usually refers to doing things properly with the use of a vocabulary and morphologi
     "If confronted with the token saw, stemming might return just s, whereas lemmatization would attem
    "The two may also differ in that stemming most commonly collapses derivationally related words, whe
    "Linguistic processing for stemming or lemmatization is often done by an additional plug-in compone

stem_strings(y)
```

```
## [1] "Stem usual refer to a crude heurist process that chop off the end of word."
## [2] "Lemmat usual refer to do thing properli with the us of a vocabulari and morpholog analysi of wo
## [3] "If confront with the token saw, stem might return just, wherea lemmat would attempt to return e
## [4] "The two mai also differ in that stem most commonli collaps derivation relat word, wherea lemmat
## [5] "Linguist process for stem or lemmat i often done by an addit plug - in compon to the index proc
```

```
lemmatize_strings(y)
```

```
## [1] "stem usually refer to a crude heuristic process that chop off the end of word."
## [2] "Lemmatization usually refer to do thing properly with the use of a vocabulary and morphological
## [3] "If confront with the token see, stem may return just s, whereas lemmatization would attempt to
## [4] "The two may also differ in that stem much commonly collapse derivationally relate word, whereas
## [5] "Linguistic process for stem or lemmatization be often do by a additional plug - in component to
```