Text Mining

Satish Kumar Sahu

15th July 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

Write a code to extract the keywords (like Inheritance, encapsulation, multithreading) from the document shared in the link http://bit.ly/epo_keyword_extraction_document

1.2 Data

Flowing is the preview of the dataset.we have been provided a pdf. using text mining tools we have to extract the frequency of the word used in the pdf.

Java Basics

Topics in this section include:

What makes Java programs portable, secure, and robust

The structure of Java applets and applications

How Java applications are executed

How applets are invoked and executed

The Java Language, Part I

Comments

Declarations

Expressions

Statements

Garbage collection

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. For our data we apply pre processing techniques that we necessary.
Our pre processing involves following removal of:
1.Case folding

2.stopwords

3.Punctuation

4.white spaces

5.lemmatization

2.1.1 Case folding

The writing systems that distinguish between the upper and lower case have two parallel sets of letters, with each letter in one set usually having an equivalent in the other set. The two case variants are alternative representations of the same letter: they have the same name and pronunciation and will be treated identically when sorting in alphabetical order.

Due to this,we have as a preprocessing step case folding is performed, in which all letters are reduced to lower case.

2.1.2 Stopwords

In computing, **stop words** are words which are filtered out before or after processing of natural language data (text).[1] Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search.

Stopwords reduces the efficiency of the code.therefore we should remove stopwords before applying model on the dataset.

2.1.3 Punctuation

Punctuation (formerly sometimes called **pointing**) is the use of spacing, conventional signs, and certain typographical devices as aids to the understanding and correct reading of handwritten and printed text, whether read silently or aloud.Removal of punctuation is done to reduce the dimensionality of the dataset.

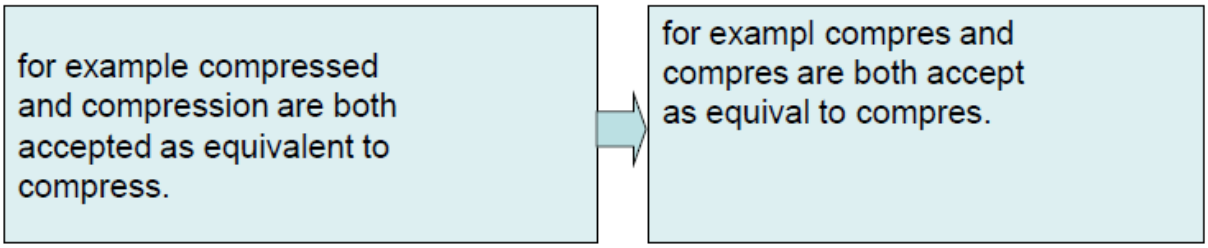
2.1.4 Lemmatization

Lemmatisation (or **lemmatization**) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.[1]

In computational linguistics, lemmatisation is the algorithmic process of determining the lemma of a word based on its intended meaning. Unlike stemming, lemmatisation depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.

2.1.5 Stemming

It reduce tokens to “root” form of words to recognize morphological variation.
–“computer”, “computational”, “computation” all reduced to same token “compute”
Stemming correct morphological analysis is language specific and can be complex.Stemming “blindly” strips off known affixes (prefixes and suffixes) in an iterative fashion.



3.1Term-Document Matrix

A collection of n documents can be represented in the vector space model by a term-document matrix. An entry in the matrix corresponds to the “weight” of a term in the document; zero means the term has no significance in the document or it simply doesn’t exist in the document.

$$\begin{array}{cccccc} & T_1 & T_2 & \ldots & T_t \\ D_1 & w_{11} & w_{21} & \ldots & w_{t1} \\ D_2 & w_{12} & w_{22} & \ldots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \ldots & w_{tn} \end{array}$$

3.1.1 TF-IDF Weighting

A typical weighting is tf-idfweighting:

$$W = tf * idf$$

A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
Experimentally, tf-idfhas been found to work well. It was also theoretically proved to work well (Papineni, NAACL 2001)

Term Weights: Term Frequency (TF)
More frequent terms in a document are more important, i.e. more indicative of the topic.
May want to normalize term frequency(tf) across the entire corpus:
TF = (Number of times term t appears in a document) / (Total number of terms in the document)

Term Weights: Inverse Document Frequency (IDF)
Terms that appear in many different documents are less indicative of overall topic.

IDF = log10(Total number of documents / Number of documents with term t in it).

4.Word Cloud

Word clouds are normally used to display the frequency of appearance of words in a particular document or speech
More frequently used words appear larger in the word cloud.
The frequency is assumed to reflect the importance of the term in the context of the document.

5.Conclusion

5.1Word Frequency

An excel sheet is obtained s output of the problem statement. Excel sheet contains the frequency of the words used in the pdf.

Appendix A

R code

```
rm(list=ls())


#Set working directory

setwd("E:/1ST SEM/eng/edwisor_assignments/text_minning")


#load libraries

library(stringr)

library(tm)

library(wordcloud)

#library(slam)



# here is a pdf for mining
```

```
url <- "https://drive.google.com/file/d/1gZCnlhwVMBIE0SugUUxDIgQrfVz-cDQR/view"

dest <- tempfile(fileext = ".pdf")

download.file(url, dest, mode = "wb")


# set path to pdftotxt.exe and convert pdf to text

exe <- "C:/Program Files/xpdf-tools-win-4.00/xpdf-tools-win-4.00/bin32/pdftotext.exe"

system(paste("\"", exe, "\" \"", dest, "\"", sep = ""), wait = F)


# get txt-file name and open it

filetxt <- sub(".pdf", ".txt", dest)

system(filetxt)



txt <- readLines(filetxt) # don't mind warning..


df=txt


#case folding

txt <- tolower(txt)


#remove stopwords

txt <- removeWords(txt, c("\\f", stopwords()))
```

```
corpus <- Corpus(VectorSource(txt))

#remove puctuation marks
corpus <- tm_map(corpus, removePunctuation)

#convert to text document matrix
tdm <- TermDocumentMatrix(corpus)



#Convert term document matrix into dataframe
TDM_data = as.data.frame(t(as.matrix(tdm)))

library(slam)
##calculate the terms frequency
words_freq = rollup(tdm, 2, na.rm=TRUE, FUN = sum)

#Convert into matrix
words_freq = as.matrix(words_freq)

#Convert to proper dataframe
words_freq = data.frame(words_freq)
```

```
#Convert row.names into index
words_freq$words = row.names(words_freq)
row.names(words_freq) = NULL
words_freq = words_freq[,c(2,1)]
names(words_freq) = c("Words", "Frequency")



words_freq <- words_freq[order(words_freq$Words),]
rownames(words_freq) <- 1:nrow(words_freq)
words_freq=words_freq[-(1:15), , drop = FALSE]
rownames(words_freq) <- 1:nrow(words_freq)

#storing the result in an excel sheet
library(xlsx) #load the package
write.xlsx(x =words_freq , file = "test.excelfile.xlsx",
        sheetName = "TestSheet", row.names = FALSE)



#Most frequent terms which appears in atleast 700 times
findFreqTerms(tdm, 10)



##wordcloud
```

```
postCorpus_WC = corpus


library(RColorBrewer)

library(wordcloud)

pal2 = brewer.pal(8,"Dark2")

png("wordcloud_v22.png", width = 12, height = 8, units = 'in', res = 300)

wordcloud(postCorpus_WC, scale = c(5,.2), min.freq = 5, max.words = 150,
random.order = FALSE, rot.per = .15, colors = pal2)

dev.off()----
```