2018년 6월 29일 5일차

R 을 이용한 텍스트 데이터 전처리

1) Regular Expression : 정규표현식

정규표현식 : 특정 조건을 만족하는 문자열을 찾기 위한 문자식이다.

^ x : x 로 시작하는 것
x $: x 로 끝나는 것
. x : x 앞에 임의의 문자가 존재하는 문자열. 만약 x 뒤에 문자열이 나타나도 선택한다.
x + : x 가 1회 이상 반복되는 것
x ? : x 가 존재하거나 존재하지 않는 것
x * : x 가 0번 이상 반복되는 것

이러한 정규 표현식을 R 에서 사용하기 위한 함수로는 gsub 이 존재한다. 다음의 예시로 확인해보자.

test = 'ZXY'
changedText = gsub ('.X', '변환값', test)
> test 문자열에 대해 .X 패턴을 찾아서 해당 패턴을 '변환값' 으로 바꿔준 다음, 결과를 저장한다.
> changedText 에는 '변환값Y' 라는 문자열이 들어간다.

정규표현식에서 사용되는 ?, {, } 등의 특수문자는 역슬래쉬 2번 \\ 을 이용하여 escape 할 수 있다.

2) Text Data Preprocessing : 텍스트 데이터 전처리

이전까지의 과정 :
text_all 변수에 벡터 형태로 모든 본문 기사를 저장한 상태이다.
gsub 함수 : gsub ('a' , 'b' , c) : c 데이터에 대하여 a 조건식을 만족하는 부분을 b 로 대치.

A. 기사에서 쓸데없는 문장 삭제하기

clz_news = gsub('^.+\\{\\}', ' ', text_all)
> clz_news 라는 새로운 변수에 text_all 을 특정 조건식으로 수정한 값을 저장한다.
> ^ . + \\ { \\ } 는 문장의 가장 처음부터 { } 문자가 나올 때 까지의 패턴을 찾는 것이다.
> 그러므로, 위의 함수로 기사의 처음부터 { } 문자까지를 선택하여 ' ' 로 대치한다.

B. 기사에서 특수문자 / 알파벳 제거하기

clz_news = gsub('[[:punct:]]', ' ', clz_news)
clz_news = gsub('[A-Za-z0-9]', ' ', clz_news)
> 앞에서 저장된 clz_news 를 가져와 다시 해당 변수에 저장하는 함수 2개이다.
> 첫 번째는 [[: punct :]] 조건으로, 모든 특수문자를 선택할 수 있는 조건식이다.
> 두 번째는 [A - Z a - z 0 - 9] 조건으로, 모든 알파벳 대, 소문자와 숫자를 선택할 수 있는 조건식이다.
> A - Z 와 a - z 사이에는 공백이 없어야 하며, 한글의 경우 [가 - 힣] 으로 모든 한글을 선택할 수 있다.

C. 기사에서 필요없는 정보 삭제하기

clz_news = gsub('[가-힣]{2,5}뉴스+', ' ', clz_news)
> [가 - 힣] { 2 , 5 } 뉴스 조건으로, 뉴스 라는 문자열에 2개에서 5개의 한글이 붙어있는 문자열을 찾는다.
> (가나다라마바사 연합뉴스 가나다라) 라는 문자열에서는 연합뉴스만 선택되게 된다.
> 이를 다양하게 활용해 자신이 원하지 않는 문자열을 삭제할 수 있다.

D. 공백 제거하기

clz_news = gsub('[[:space:]]+', ' ', clz_news)
> 위의 특수문자와 유사하게 [[: space :]] + 조건식을 이용해 모든 2개 이상의 space 문자를 선택한다.
> 이 때, \t \n 등의 탭문자, 개행문자도 다중 space 문자로 선택되어 삭제된다.

E. 짧은 기사만 선택하기

n_news = clz_news[nchar(clz_news) < 1000]
> nchar(clz_news) 은 각각의 기사에 대해 문자열의 길이를 반환하는 함수이다.
> 이를 nchar (clz_news) < 1000 과 같은 조건식으로 반환하면, 1000 보다 작은 문자열만 True 로 반환한다.
> 최종적으로 clz_news [nchar(clz_news) < 1000] 를 이용해 1000자 이하의 기사만 추출할 수 있다.

더 다양한 조건식을 알고싶다면, ?gsub 명령어를 실행시켜 확인할 수 있다.

3) Export Data with CSV : CSV 파일의 생성

CSV 파일 : 컴마와 탭 등으로 구분되어진 텍스트 데이터 파일. 용량이 매우 작은게 장점이다.

getwd( ) / setwd(" ") : 현재 Working Directory 를 확인하고, 원하는 Working Directory 를 설정한다.

write.csv(n_news, 'n_news.csv')
> 앞에서 최종적으로 만들어진 n_news 변수의 텍스트 데이터를 csv 형태의 데이터 파일로 저장한다.
> 이 때, 앞에서 설정한 Working Directory 에 파일이 저장된다.

MySQL 을 이용한 데이터 분석

1) Change Encoding of CSV File

앞에서 저장한 CSV 파일을 사용하기 위해서는 인코딩을 UTF-8 으로 변경하여야 한다.

텍스트 에디터 / 메모장 등으로 UTF - 8 형식을 선택해 다시 저장하면 된다.

2) Create Schema & Table

원하는 계정에 접속하여 왼쪽 상단의 버튼으로 원하는 이름의 스키마를 생성한다.

스키마를 선택한 다음, 왼쪽 상단의 버튼으로 다음의 테이블을 생성한다.

no = int

text = varchar (20,000)
charset / collation 을 UTF-8 으로 지정.

3) CSV File to Table

왼쪽의 스키마 - 테이블 - 테이블 이름을 선택한 다음, 해당 테이블을 오른쪽 버튼으로 클릭한다.

Table Data Import Wizard 를 선택하여 CSV 파일을 테이블에 저장한다.

4) 데이터 분석

A) 특정 단어를 포함하는 기사의 검색

SELECT text FROM news WHERE text LIKE '%월드컵%'
> 월드컵 단어가 포함된 모든 기사 본문을 찾아낸다.

B) 기사의 일부분만 잘라내어 검색

SELECT SUBSTRING(text, 20, 40) RESULT
FROM news
WHERE no < 10
> text 필드의 20 ~ 40 번째 문자열을 추출하여 RESULT 라 이름붙인다.
> 최종적으로 10개의 데이터만 추출한다.

R 에서 사용하는 MySQL 쿼리

1) MySQL 비밀번호 확인 과정

기본 Shell / Terminal 창을 띄워서 mysql 에 접속한 다음, 아래의 명령어로 비밀번호를 초기화한다.

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234' ;

이를 통해 mysql 의 비밀번호가 1234 로 변경된다.

2) RMySQL 패키지의 설치

R Studio 에서 아래의 명령어로 MySQL 과 R 을 연동시킬 수 있는 RMySQL 패키지를 설치한다.

install.packages("RMySQL")

3) MySQL 연동

```
library(RMySQL)
con = dbConnect(      drv = MySQL( ), dbname = news,
                      user = 'root', password = '1234', host = localhost, port = 3306
                      client.flag = CLIENT_MULTI_RESULTS )
```
> con 이라는 설정 변수에 dbConnect 함수를 이용해 설정값을 저장한다.
> drv = 어떤 DBMS 를 사용할 것인지 설정
> dbname = 위에서 생성했던 mysql 스키마 이름 작성
> user / password / host / port = MySQL 첫 화면의 계정 정보에서 찾아서 작성
> client.flag = CLIENT_MULTI_RESULTS = 한 쿼리문에서 여러 테이블에 대한 쿼리를 작성하기 위한 설정

```
dbListTables(con)
dbListFields(con, 'news')
```
> 위에서 설정한 con 을 바탕으로, 해당 DB 의 테이블, 혹은 컬럼을 추출할 수 있다.

```
dbDisconnect(con)
```
> 사용이 끝나면 연결을 끊어줘야 한다.

4) MySQL 쿼리문 작성

```
db_news = dbGetQuery(con, 'SELECT * FROM news')
```
> 위의 설정에 따라 스키마를 선택하고, 쿼리문을 작성해 결과값을 db_news 변수에 저장한다.

5) R 인코딩 설정

한글 텍스트 데이터 활용에서는 인코딩이 매우 중요하다.
인코딩 문제를 해결하기 위한 R 의 3대 함수를 소개한다.

Encoding(db_news[1, 2]) : 해당 텍스트의 인코딩을 출력한다.

repair_encoding(db_news$text) : 인코딩을 자동을 감지하여 적절한 인코딩으로 출력한다.

iconv(db_news$text, from = 'UTF-8', to = 'UTF-8') : 특정 인코딩을 다른 인코딩으로 강제 변환한다.

데이터 마이닝 기초

1) 텍스트 데이터 분석

텍스트 마이닝 개념

> 단어의 빈도를 기반으로 분석
> 문장 이해는 불가능. 문장을 뽑아내 어떤 단어가 많이 쓰였는지를 분석
> 단어를 통해서 전반적인 추세 확인이 가능
> 문장 구조와 단어를 통해 원하는 것을 알아낼 수 있다.
> Automentic essay scoring : 학생의 에세이를 단어 기반으로 분석해 채점. 신뢰도 높음.

형태소 분석

> 문장에서 핵심이 되는 부분만 가져와 분석해도 충분하다.

KoNLP : 한국어 자연어처리 패키지

> 한나눔 형태소 분석기 사용 : Java 기반

2) KoNLP 의 단어사전 설치

품사 태그셋 : simplePos09 / 22
> 한글을 9가지 / 22가지 기준으로 추출하는 것. 보통 09 를 활용하면 충분히 분석이 가능하다.

extractNoun('아버지 어머니 나 동생')
> 명사만 추출해내는 함수이다.
> 사전 형태로 만들어져 있기 때문에, 명사뿐만 아니라 자신이 원하는 태그대로 추출하는 함수를
> 직접 만들 수 있다. 예를 들어 대명사만을 추출하는 함수 등이 있다.

```
install.package('KoNLP')
library(KoNLP)
useNIADic()
```
> NIADic 이라는 최신 버전의 단어 사전을 사용하는 것이다.
> 기존의 9만 단어에서 현재 28만 단어까지 업데이트 된 상태이다.
> NIADic ( ) 의 단어 사전은 R 패키지의 KoNLP_Dic 폴더의 current에 존재한다.

3) Text Mining (TM) 패키지를 이용한 매트릭스 생성

2차원 매트릭스의 형태로, 가로는 각각의 신문 기사, 세로는 다양한 단어들로 이루어진다.
매트릭스의 Value 에는 각 단어의 빈도 수가 들어갈 것이다.
이러한 매트릭스를 활용하여 선형 회귀분석과 같은 통계 분석이 가능하다.

즉, KoNLP 에서 한국어처리가 이루어진 다음, 이를 TM으로 수치화 하는 것이다.

```
install.packages('tm')
library(tm)
ko_word = function(input){
  text = as.character(input)
  extractNoun(text)
}
```

> ko_word 라는 이름의 함수를 정의한다. input 입력을 받아와 해당 입력을 전부 char 형태로 변환한다.
> 여기서는 그럴 일이 없지만, 만약 숫자로만 이루어진 데이터가 있다면 문제가 생기기 때문이다.

tip)
> 함수를 저장하면 오른쪽의 Environment 창에 함수가 생성된다.
> 함수를 드래그하고, alt + enter 로 함수를 확인할 수 있다.
> 이는 패키지 함수도 마찬가지이므로, 함수 분석이 필요할 때 유용하게 활용할 수 있다.

```
cps = VCorpus(VectorSource(db_news$text))
ko_tdm <- TermDocumentMatrix(cps,
                             control = list(
                               tokenize = ko_word,
                               wordLengths = c(2, 7)
                             ))
```
> 위에서 저장된 db_news 의 text 컬럼의 내용들을 cps 에 VCorpus 함수를 이용해 저장한다.
> 이를 다시 Term - Doc 형태의 매트릭스로 변환한다. 만약 Doc - Term 이라면 행렬의 내용이 바뀐다.
> wordLengths 를 이용해 최소 2개, 최대 7개 문자가 연결된 단어를 찾아낸다.
> 기본 값이 3, inf 로, 2글자 단어를 찾아내지 못해 설정하지 않으면 적은 단어가 추출된다.
> 이는 영미권 단어가 최소 3글자부터 시작하는 이유이다.

ko_tdm 변수 결과값의 분석
> terms : 찾아낸 단어의 개수
> documnets : 분석한 문자열의 개수
> sparse : 값이 들어가지 않은 셀
> sparsity : 값이 있는 셀 / 값이 없는 셀 을 통한

```
ko_mat <- as.matrix(ko_tdm)
```
> 매트릭스 형태로 ko_tdm 을 변환해주는 과정

```
word_freq <- rowSums(ko_mat)
```
> 문자별 전체 빈도를 측정하여 word_freq 에 저장한다.

```
str(word_freq)
```
> word_freq 라는 변수에 대한 정보를 나타내는 함수이다.
> 몇 개의 데이터가 어떻게 구성되는지를 보여준다.

4) WordCloud

```
word_name = names(word_freq)
```
> 벡터의 이름만 빼내 저장한다.

```
df_wc = data.frame(word = word_name, freq = word_freq)
```
> 단어와 단어에 해당하는 빈도수의 dataframe 을 생성한다

```
install.packages('wordcloud2')
install.packages('htmlwidgets')
```
> Word Cloud 를 만들기 위한 패키지 2개를 설치한다.

tip)
> 물음표 2개 ?? 를 이용하면 해당 내용에 대한 설명이 나타난다.
> ?? htmlwidgets 로 기능을 알아볼 수 있다.

```
library(wordcloud2)
library(htmlwidgets)
wc <- wordcloud(df_wc, color = 'random-dark')
```
> 위에서 만든 df_wc 를 이용하여 wordcloud 를 생성한다.

> 이 때, 색상은 자동적으로 어두운 색상으로 지정한다.

wordcloud 1 에서는 이미지 형식으로 만들어졌지만, 2 부터는 반응형 html 파일로 생성된다.
그러므로 이 html 파일을 생성하기 위해 htmlwidget 의 함수를 사용해야 한다.

savewidget(wc, 'wc.html', selfcontained = F)
> Working Directory 에 wc.html 이라는 파일로 저장된다.
> 이 파일을 실행시키면 생성된 wordcloud 를 볼 수 있다.

단어 연관성 분석

1) 단어 연관성 분석의 기초

각 Doc 문자열에 바나나 / 사과 단어가 들어간 횟수를 행렬로 만들면 다음과 같다.

	Doc1	Doc2
바나나	(1,	0)
사과	(1,	1)

먼저 이를 전치한 행렬을 만든다

	바나나	사과
Doc1	(1,	1)
Doc2	(0,	1)

두 행렬을 곱하여 새로운 행렬을 만든다.

	바나나	사과
바나나	(1	1)
사과	(1	2)

대각선 축을 기준으로 대칭되게 된다.
대각선의 값은 전체 문서에서 나타난 전체 횟수를 의미한다. (1, 2)
(바나나, 사과) 값은 바나나가 나타났을 때, 사과 1회 발생
(사과 , 바나나) 값은 사과가 나타났을 때, 바나나 1회 발생을 의미한다.

이를 다시 노드와 엣지로 그래프 형태의 표현이 가능하다

2) 연관성 그래프 생성

word_order = order(word_freq, decreasing = T) [1 : 50]
> word_freq 의 단어 빈도값을 내림차순으로 표현한다. 이 때 출력은 단어의 인덱스값이다.
> 그중에서도 상위 50개 값만 뽑아내어 word_order 에 저장하도록 한다.

occ_mat = ko_mat[word_order,]
> word_order 의 인덱스에 대해 모든 열을 가져와 occ_mat 에 저장한다.

occ_mat2 = occ_mat %*% t(occ_mat)
> %*% 는 행렬 곱을 연산하는 연산자이다.
> t(occ_mat) 함수는 occ_mat 의 전치 행렬을 만들어준다.

install.package(qgraph)
library(qgraph)
> 데이터 시각화를 위한 qgraph 패키지의 설치 및 사용

qgraph(occ_mat2, layout = 'spring', 'circle', color = 'blue', vsize = log(diag(occ_mat2)),

label.color = ‘white’, labels = colnames(occ_mat2), diag = F)

> layout 은 대표적으로 spring / circle 등이 존재한다.
> spring : 거리행렬의 값이 클 수록 가깝게, 작을수록 멀게 연결하는 것
> circle : 원탁 형태에 데이터를 나열하고, 관계를 선으로 연결하는 것

> color 는 노드의 색을 설정하는 것이다.
> vsize 는 노드의 크기를 설정하는 것이다. diag(occ_mat2) 함수로 실제 빈도에 따라 크기를 출력한다.
> 이 때, log 값을 씌워주어 각 값들의 격차를 줄여주는 것이 보기 편리하다.

> lable.color 는 각 노드의 이름 색상을 설정하는 것이다.
> labels 는 각 노드의 이름을 설정하는 것으로, colnames 혹은 rownames 를 이용해 각 이름에 따라 설정
> diag = F 는 자기 자신으로 들어가는 값을 출력하지 않도록 하는 것이다.

3) 연관성 그래프 이미지 출력

png(‘associ.png’, width = 500, height = 500)
> 500 x 500 짜리 png 파일로 그림 그리기를 시작한다.

dev.off()
> 그림 그리기를 종료한다.

즉, png 함수는 시작 - 끝 이 하나의 묶음이며, 그 사이에 qgraph 함수가 들어가면 된다.
이를 코드로 표현하면 다음과 같다.

```
png(‘associ.png’, width = 500, height = 500 )
qgraph(occ_mat2, layout = ‘spring’, ‘circle’, color = ‘blue’, vsize = log(diag(occ_mat2)),
        label.color = ‘white’, labels = colnames(occ_mat2), diag = F )
dev.off()
```