[R] R을 통한 텍스트마이닝에서 워드클라우드 까지 — 나무늘보의 개발 블로그

노트북: blog
만든 날짜: 2020-10-06 오후 4:44
URL: https://continuous-development.tistory.com/53?category=793392

R

[R] R을 통한 텍스트마이닝에서 워드클라우드 까지

2020. 8. 5. 23:55 수정 삭제 공개

#텍스트 마이닝

시작전에 자바 jdk를 깔고 환경변수를 설정해줘야 한다. jdk를 통해 KoNLP 통해 돌리기 때문이다. 이부분이 선행되야 나머지가 된다.

```
9
10 install.packages(c("hash", "tau", "Sejong",
11                    "RSQLite", "devtools", "bit",
12                    "rex", "lazyeval", "htmlwidgets",
13                    "crosstalk", "promises", "later",
14                    "sessioninfo", "xopen", "bit64",
15                    "blob", "DBI", "memoise", "plogr",
16                    "covr", "DT", "rcmdcheck", "rversions"),
17                  type = "binary")
18
19 # github에서 패키지 설치
20 install.packages("remotes")
21 # 64bit 에서만 동작합니다.
22 remotes::install_github('haven-jeon/KoNLP',
23                         upgrade = "never",
24                         INSTALL_opts=c("--no-multiarch")) #konlp는 직접인스톨하면 에러가 나서 이런 방식으로 인스톨해야 된다.
25
26 devtools::install_github("r-pkgs/usethis")
```

KoNLP는 일반적인 인스톨을 해서는 안되고 위와같이 github라는 것을 통해서 받아야 한다.

```
#기본 라이브러리 로딩
Sys.setenv(JAVA_HOME='C:\\Program Files\\JAVA\\jdk1.8.0_121')
library(rJava)
library(KoNLP)

install.packages("tm")
install.packages("wordcloud")
library(tm) # 전처리 용도 라이브러리
library(wordcloud)
library(RColorBrewer) # 팔레트 역할 (색깔)
```

없는부분은 install하자 java home의 위치나 버전은 사람마다 다를 수도 있으니 해당 경로를 맞춰주자.

```
# 명사 , 형용사등을 추출한 사전
?useSejongDic()

useSejongDic(text)

text <-"최근 이슈가 되고 있는 빅데이터에 대한 이해와 활용을 위해 데이터 과학(Data Science)의 측면에서 접근한다.
빅데이터는 통계학을 비롯한 경영, IT 등의 다양한 분야들이 서로 결합되어 있고 그 정의가 다양하지만, 본 강의는 데이터 분석을
기반으로 하는 과학적 의사결정의 관점에서 바라보고자 한다. 빅데이터에 대한 이해를 위해 실제 사례들을 살펴보고,
데이터를 통해 의사결정에 유용한 정보 및 지식을 찾는 과정을 이해한다. 나아가 빅데이터 분석에서 필수적으로 언급되고 있는
R 통계프로그램을 소개하고 이를 분석에 활용할 수 있게 한다."

nouns ← extractNoun(text)
```

test라는 변수에 문자열을 넣어준다.

extractNoun을 통해 명사를 추출해낸다.

```
nouns ← extractNoun(text) #명사를 뽑아내는 기능
```

```
> nouns ← extractNoun(text)
> nouns
 [1] "이슈"      "빅데이터에" "이해"      "활용"      "데이터"    "과학(Data" "Science"   "측면"      "접근"      "빅데이터는" "통계학"    "비롯"      "한"        "경영"
[15] "IT"        "등"         "다양"      "한"        "분야"      "들이"      "결합"      "정의"      "다양"      "강의"       "데이터"    "분석"      "기반"      "과학"
[29] "적"        "의사결정"   "관점"      "빅데이터에" "이해"      "실제"      "사례"      "들"        "데이터"    "의사결정"   "유용"      "한"        "정보"      "지식"
[43] "과정"      "이해"       "빅데이터"  "분석"      "필수"      "적"        "언급"      "R"         "통계"      "프로그램"   "소개"      "이"        "분석"      "활용"
[57] "수"
```

이제 뽑아낸 명사를 전처리 하는 과정을 거친다.

#전처리

명사를 추출한 데이터에서 nchar()를 통해 두글자 이상의 글자만 가져온다. 한글자로 만된 것중에 등 한 이런것들을 빼기 위해서이다.

```
54
55  # nchar() - 문자열 길이 함수
56  nouns ← nouns[nchar(nouns) ⩾ 2]
57  nouns
58
```

table 함수를 사용해서 각각의 데이터가 몇번씩 나왔는지 확인한다.

```
> # 빈도표
> WFreq ← table(nouns)
> WFreq
nouns
       IT   Science        강의        결합        경영        과정        과학 과학(Data        관점        기반
        1         1         1         1         1         1         1         1         1         1
      다양      데이터       들이        분석        분야        비롯  빅데이터  빅데이터에        사례        소개
        2         3         1         3         1         1         4         2         1         1
      실제        언급        유용  의사결정        이슈        이해        접근        정보        정의        지식
        1         1         1         2         1         3         1         1         1         1
      측면        통계      통계학  프로그램        필수        활용
        1         1         1         1         1         2
```

빅데이터~ 는 명사로 뽑으면 빅데이터 단어가 끝이기 때문에 전처리가 제대로 안된 부분에 대해서는 이런식으로 전처리를 해준다.

```
58
59  #gsub은 바꾸는 기능을 한다. .은 한글자 그리고 *를 쓰면 무한대를 말한다. 그래서 빅데이터에/는 이런 것들을 빅데이터  로 바꿔준다.
60  nouns ← gsub("빅데이터.*","빅데이터",nouns) #
61  nouns
62
63  wFreq ← table(nouns)
64  wFreq
```

```
nouns
       IT   Science        강의        결합        경영        과정        과학 과학(Data        관점        기반
        1         1         1         1         1         1         1         1         1         1
      다양      데이터       들이        분석        분야        비롯  빅데이터        사례        소개        실제
        2         3         1         3         1         1         4         1         1         1
      언급        유용  의사결정        이슈        이해        접근        정보        정의        지식        측면
        1         1         2         1         3         1         1         1         1         1
      통계      통계학  프로그램        필수        활용
        1         1         1         1         2
```

테이블 데이터를 WFreq로 확인한다.

이제 테이블 데이터를 가지고 워드클라우드를 만든다.

```r
names(wFreq)

pal <- brewer.pal(6,"Accent") #6가지 Accent 를 나타낸다.

wordcloud(words = names(wFreq), # 빈도를 가지고 있는 속성값들을 넣어준다.
          freq = wFreq,     #빈도를 가지고 출력할 데이터를 넣는다.
          min.freq = 1,     #최소 빈도수를 정한다. 여기서 1이면 1개 이상은 모두 출력된다.
          random.order = F, #임의의 순서로 작성하는데 F일때는 감소 빈도이다
          colors=pal)    #색상을 넣는다.
?wordcloud
```

예제

기본적인 텍스트 마이닝 단계는 text -> corpus(말뭉치) -> TDM(단어에 따른 매트릭스) -> 매트릭스로 형변환 -> 워드클라우드
로 나뉜다. 전체적인 로직은 이렇다.

```r
#데이터 읽어오기
data ← readLines(file.choose(),encoding = 'UTF-8')
data


#Vcorpus(휘발성의 corpus를 생성 )를 써서 VectorSource(백터소스) 를 집합으로 만들어서 넣어준다. 말모음 뭉치로 하는 작업
corpus ← VCorpus(VectorSource(data))
corpus


# tm_map(TDM,function) - 전처리를 도와주는데 공백/숫자/특수문자 를 제거하는 함수를 호출하면 처리가 이루어진다.

# stopword(불용어) 처리 방법
stopwords('en')
stopwords2 ← stopwords('en')
stopwords2 ← c(stopwords('en'),"and","not","but") #더 추가하고 싶을때

corpus_map ← tm_map(corpus,removeWords,stopwords2) #tm_map (데이터 , 함수(단어를 지우겠다.) , 지울 단어들 (stopword2를 통해 만든 불용어 단어들))

corpus_map ← tm_map(corpus_map,stripWhitespace) #여러개의 공백을 하나의 공백으로 만들어준다.
corpus_map ← tm_map(corpus_map,removeNumbers) #숫자제거
corpus_map ← tm_map(corpus_map,removePunctuation) #특수문자 제거
corpus_map


#빈도 체크
TDM ← TermDocumentMatrix(corpus_map)

findFreqTerms(TDM,2)

#빈도표
matrix ←as.matrix(TDM)
rownames(matrix)

rowSums(matrix)

wFreq ←sort(rowSums(matrix),decreasing =T)


pal ← brewer.pal(6,"Accent")

wordcloud(words = names(wFreq),
          freq = wFreq,
          min.freq = 1,
          random.order = F,
          colors=pal)
```

이걸 하나씩 풀어서 설명해보자

```r
# 텍스트 마이닝 단계
# text → corpus(말뭉치) → TDM(term documnet matrix) → TM 분석 → Matrix(DF) → wordcloud
# service_data_I_love_mom.txt

#데이터 읽어오가
data ← readLines(file.choose(),encoding = 'UTF-8')
data
```

```
> data
[1] "Mommy I love you And I want to say thank you           " "I am happy everyday for the love you give to me       " "Sometimes I am sad Sometimes I do cry                 "
[4] "Then I just remember mommy how much you love me        " "Mommy you take care of me you help me to be strong   " "You help my dreams to come true with everything you do "
[7] "Your love always makes me smile I want to say thank you" "I really, really love you                             " "Mommy I love you ~~~~~!^^$ I'm                       "
>
```

#VCorpus - 받은 text를 말뭉치로 만드는 함수

받은 데이터를 말뭉치로 만드는 작업이다. 이렇게 만들어 놔야 전처리를 하기 쉽다.

```r

#Vcorpus(휘발성의 corpus를 생성 )를 써서 VectorSource(백터소스) 를 집합으로 만들어서 넣어준다. 말모음 뭉치로 하는 작업
corpus ← VCorpus(VectorSource(data))
corpus
```

#TermDocumentMatrix - 행이 단어, 열이 문서 형태로 된 행렬을 만드는 함수

이 말뭉치들을 TermDocumentMatrix를 통해 데이터를 확인한다.

```
> # TermDocumentMatrix : 행이 단어 , 열 문서 형태로 된 행렬
> TDM ← TermDocumentMatrix(corpus)
> TDM
<<TermDocumentMatrix (terms: 37, documents: 9)>>
Non-/sparse entries: 56/277
Sparsity           : 83%
Maximal term length: 10
Weighting          : term frequency (tf)
```

9개의 도큐먼트중에 37개의 단어를 를 찾았다.
행열로 봤을때 0 으로 매칭되지 않는 값이 56개가 있다는 뜻이다 . 즉 한번이라도 노출된 개수가 56개이다.
Maximal term Length 는 가장 긴 단어가 10글자이다.
Sparsity는 희소행렬이다.

#tm_map - 전처리를 도와주는 함수

```
> # tm_map(TDM,function) - 전처리를 도와주는데 공백/숫자/특수문자 를 제거하는 함수를 호출하면 처리가 이루어진다.
> corpus_map ← tm_map(corpus,stripWhitespace) #여러개의 공백을 하나의 공백으로 만들어준다.
> corpus_map ← tm_map(corpus,removeNumbers) #숫자제거
> corpus_map ← tm_map(corpus,removePunctuation) #특수문자 제거
> corpus_map
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 9
> corpus_map[[9]]$content
[1] "Mommy I love you  Im        "
```

이제 데이터에서 불용어(쓰이지 않는 단어)를 처리한다.

```
176  # tm_map(TDM,function) - 전처리를 도와주는데 공백/숫자/특수문자 를 제거하는 함수를 호출하면 처리가 이루어진다.
177
178  # stopword(불용어) 처리 방법
179  stopwords('en')
180  stopwords2 <- stopwords('en')
181  stopwords2 <- c(stopwords('en'),"and","not","but") #더 추가하고 싶을때
182
183  corpus_map <- tm_map(corpus,removeWords,stopwords2) #tm_map (데이터 , 함수(단어를 지우겠다.) , 지울 단어들 (stopword2를 통해 만든 불용어 단어들))
184
185  corpus_map <- tm_map(corpus_map,stripWhitespace) #여러개의 공백을 하나의 공백으로 만들어준다.
186  corpus_map <- tm_map(corpus_map,removeNumbers) #숫자제거
187  corpus_map <- tm_map(corpus_map,removePunctuation) #특수문자 제거
188  corpus_map
```

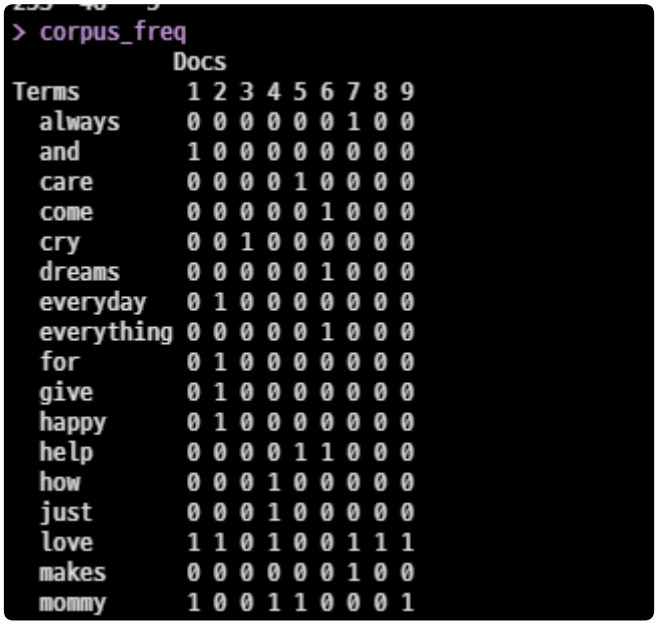
stopwords('en')에는 영어에서 불용어처리 하는 단어들이 들어있다. 그데이터에 추가적으로 걸리지 않는 and 나 not 등을 추가로 해 stopwords2를 만들어준다.
그후 데이터를 가공하는 작업을 진행한다.
이런 방식으로 한번에 하는 것 도 가능하다.

```
100
101  #이런 방식으로도 가능하다.
102  corpus_tm2 ← TermDocumentMatrix(corpus,
103                                  control = list(stripWhitespace = T,
104                                                 removeNumbers = T,
105                                                 removePunctuation = T))
```

#확인


```
121  #빈도 체크
122  TDM2 ← TermDocumentMatrix(corpus_map)
123  corpus_freq ← as.matrix(TDM2)
```

TermDocumentMatrix를 통해 해당문서에서 단어 발생유무를 0,1로 나타낸다.
그 후 그 값을 매트릭스로 형변환 시킨다.
왼쪽행에 값에는 단어가 들어있고 위에 Docs는 이 단어가 발생된 위치의 순서값에 0 / 1 로 값이 존재하는지를 나타낸다.



다시 전체로 보면

```r
#데이터 읽어오기
data <- readLines(file.choose(),encoding = 'UTF-8')
data


#Vcorpus(휘발성의 corpus를 생성 )를 써서 VectorSource(백터소스) 를 집합으로 만들어서 넣어준다. 말모음 뭉치로 하는 작업
corpus <- VCorpus(VectorSource(data))
corpus


# tm_map(TDM,function) - 전처리를 도와주는데 공백/숫자/특수문자 를 제거하는 함수를 호출하면 처리가 이루어진다.

# stopword(불용어) 처리 방법
stopwords('en')
stopwords2 <- stopwords('en')
stopwords2 <- c(stopwords('en'),"and","not","but") #더 추가하고 싶을때

corpus_map <- tm_map(corpus,removeWords,stopwords2) #tm_map (데이터 , 함수(단어를 지우겠다.) , 지울 단어들 (stopword2를 통해 만든 불용어 단어들))

corpus_map <- tm_map(corpus_map,stripWhitespace) #여러개의 공백을 하나의 공백으로 만들어준다.
corpus_map <- tm_map(corpus_map,removeNumbers) #숫자제거
corpus_map <- tm_map(corpus_map,removePunctuation) #특수문자 제거
corpus_map


#빈도 체크
TDM <- TermDocumentMatrix(corpus_map)

findFreqTerms(TDM,2)

#빈도표
matrix <-as.matrix(TDM)
rownames(matrix)

rowSums(matrix)

wFreq <-sort(rowSums(matrix),decreasing =T)


pal <- brewer.pal(6,"Accent")

wordcloud(words = names(wFreq),
          freq = wFreq,
          min.freq = 1,
          random.order = F,
          colors=pal)

```

이런식으로 결과가 나온다.

'R' 카테고리의 다른 글□

[R] R에서 교차검증을 위한 데이터 셋 분리방법 3가지□

[R] R을 활용한 상관분석과 회귀분석 - 1□

[R] R을 통한 텍스트마이닝에서 워드클라우드 까지□

[R] R로 하는 비정형 데이터 처리 (facebook 데이터를 통한 긍정/부정 나누기)□

[R] R에서 Database 사용하기 / DB 기본적인 구문 사용하기□

[R] 예제를 통한 데이터 전처리 작업□

Data Mining r data mining R wordcloud wordcloud 데이터마이닝 워드클라우드

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.