

# 텍스트 마이닝 분석

# Naver 기사 분석

[http://news.naver.com/main/list.nhn?  
mode=LS2D&mid=shm&sid2=264&sid1=100&date=20171023](http://news.naver.com/main/list.nhn?mode=LS2D&mid=shm&sid2=264&sid1=100&date=20171023)

# Naver 기사 분석

- url 분석

http://news.naver.com/main/list.nhn?  
mode=LS2D&mid=shm&sid2=264&sid1=100&date=20171023

- sid1: 카테고리
- sid2: 세부 카테고리
- date: 기사 날짜

# Naver 기사 분석

## 1. 범위 지정

1. 카테고리 선택
2. 세부 카테고리 선택
3. 검색할 날짜 지정

## 2. 기사 크롤링

1. “페이징 url 목록” 생성
2. 페이지 내에 있는 기사 추출

## 3. 단어 추출

1. 기사에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# Naver 기사 분석

- 주요 Library

```
if (!require(devtools)) {  
  install.packages("devtools")  
  require(devtools)  
}
```

```
if (!require(N2H4)) {  
  devtools::install_github("forkonlp/N2H4")  
  require(N2H4)  
}
```

- 주요 함수 설명

- getMainCategory()
- getSubCategory(sid1)
- getMaxPageNum(url)
- getUrlListByCategory(page)
- getContent(link)

# Naver 기사 분석

## 1. 범위 지정

1. 카테고리 선택
2. 세부 카테고리 선택
3. 검색할 날짜 지정

## 2. 기사 크롤링

1. “페이징 url 목록” 생성
2. 페이지 내에 있는 기사 추출

## 3. 단어 추출

1. 기사에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드로 출력

# Naver 기사 분석

# 카테고리 선택

```
cate <- getMainCategory()  
sid1 <- cate$sid1[1]
```

# 세부 카테고리 선택

```
sub_cate <- getSubCategory(sid1=sid1)  
sid2 <- sub_cate$sid2[select]
```

# 날짜 선택

```
start_date <- "20171021"  
end_date <- "20171021"
```

# Naver 기사 분석

## 1. 범위 지정

1. 카테고리 선택
2. 세부 카테고리 선택
3. 검색할 날짜 지정

## 2. 기사 크롤링

1. “페이징 url 목록” 생성
2. 페이지 내에 있는 기사 추출

## 3. 단어 추출

1. 기사에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력



# Naver 기사 분석


## NH농협손보 헤아림 봉사단 다문화가정 어린이 요리교실

NH농협손해보험은 자사 헤아림 봉사단이 '다문화가정 어린이와 함께하는 쌀사랑 밥사랑 요리교실' 행사를 실시했다 ...

파이낸셜뉴스 |  25면1단 | 2017-10-24 19:22



## 新DTI·DSR 적용되면 '주담대 2억·연봉 6천' 직장인 대출가능액?

이번에도 다주택자가 타깃이다. 정부는 24일 가계부채 증가세를 잡을 방안으로 신(新)DTI(총부채상환비율) 산정 방식 개선을 내 ... 국민일보 |  3면1단 | 2017-10-24 19:15

1

2

3

4

5

6

7

8

9

10

다음 >

10월24일(화) · 10월23일(월) · 10월22일(일) · 10월21일(토) · 10월20일(금)

# Naver 기사 분석

# 기사 목록 페이지의 마지막 페이지 수를 가져옵니다.

```
max <- getMaxPageNum(base_url)
```

# page\_list 생성

# page\_list - 기사 목록의 1페이지, 2페이지, ... 의 url을 저장할 벡터

```
for (page_num in 1:max) {
```

```
  page_list <- c(page_list, paste0(base_url,  
                                     "&page=", page_num))
```

```
  if (length(page_list) >= max_page) break
```

```
}
```

# Naver 기사 분석

# 뉴스 리스트 페이지 내의 개별 네이버 뉴스 url들을 가져옵니다.

```
news_list <- getUrlListByCategory(page)
```

# page\_list에 저장된 url에 접속하여 기사를 가져옵니다.

```
for (news_link in news_list$links) {
```

```
  # 기사 가져오기
```

```
  tem <- getContent(news_link)
```

```
  news_data <- rbind(news_data, tem)
```

```
}
```

# Naver 기사 분석

```
# 가져온 뉴스들을 csv 형태로 저장합니다.  
# (한 페이지에 20개의 뉴스가 있습니다.)  
dir.create("./data", showWarnings=F)  
filename <- paste0("./data/news_", sid1, "_", sid2, "_",  
                    date, "_", page_num, ".csv")  
write.csv(news_data, file=filename, row.names=F)
```

# Naver 기사 분석

# 파일 이름 가져오기

```
file.name <- list.files('data'); file.name
```

# 파일 하나씩 가져와서 하나의 데이터셋에 담기

```
for (i in 1:length(file.name)) {  
  temp_ds <- read.csv(paste('data/', file.name[i], sep=""),  
                      stringsAsFactor=FALSE, header=T)  
  ds_news <- rbind(ds_news, temp_ds)  
}
```

# Naver 기사 분석

## 1. 범위 지정

1. 카테고리 선택
2. 세부 카테고리 선택
3. 검색할 날짜 지정

## 2. 기사 크롤링

1. “페이징 url 목록” 생성
2. 페이지 내에 있는 기사 추출

## 3. 단어 추출

1. 기사에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# 데이터 전처리 - 단어 추출

- 주요 Library

```
if (!require(rJava)) {  
  install.packages("rJava")  
  require(rJava)  
}
```

```
if (!require(KoNLP)) {  
  install.packages("KoNLP")  
  require(KoNLP)  
}
```

- 주요 함수 설명

- useSejongDic()
- extractNoun(text)
- unlist(list)
- text\_cleanup(text)
- sapply()
- Filter()

# 데이터 전처리 - 단어 추출

# 단어만 골라내기

```
word_data <- sapply(ds_news, extractNoun, USE.NAMES=F)
```

```
word_data <- unlist(word_data)
```

# 함수를 사용하여 불필요한 문자를 걸러낸다.

# source() 를 사용하면 다른 스크립트에 저장된 코드를 사용할 수 있다.

# 함수 코드는 text\_cleanup.R 문서에서 가져온다.

```
source("text_cleanup.R", encoding="UTF-8")
```

```
word_data <- text_cleanup(word_data)
```

# 두글자 이상 단어만 추출하기

```
results_nouns <- Filter(function (x) { nchar(x) >= 2 }, word_data)
```



# Naver 기사 분석

## 1. 범위 지정

1. 카테고리 선택
2. 세부 카테고리 선택
3. 검색할 날짜 지정

## 2. 기사 크롤링

1. “페이징 url 목록” 생성
2. 페이지 내에 있는 기사 추출

## 3. 단어 추출

1. 기사에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# Word Cloud

- 주요 Library

```
if (!require(wordcloud)) {  
  install.packages("wordcloud")  
  require(wordcloud)  
}
```

- 주요 함수 설명

- wordcloud()

# Word Cloud

# 가장 많이 나타난 단어 50개만 살펴보자.

```
results_wordcount <- table(results_nouns) # 문자 카운팅  
head(sort(results_wordcount, decreasing=T), 50)
```

```
pal <- brewer.pal(12, "Paired") # 컬러 세팅
```

# 워드 클라우드 출력

```
wordcloud(names(results_wordcount), freq=results_wordcount,  
          scale=c(10, .5), min.freq=2, max.words=Inf,  
          random.order=F, rot.per=.15, random.color=T,  
          colors=brewer.pal(9, "Dark2"))
```

# Twitter 트윗 분석

1. Twitter API 세팅
  1. app 생성
  2. api 연결
2. 트윗 크롤링
  1. 검색어 지정
  2. 트윗 추출
3. 단어 추출
  1. 트윗에서 단어 목록 추출
  2. 단어 데이터 정리
4. 출력
  1. 단어별 갯수 카운트
  2. 클라우드 출력

# Twitter 트윗 분석

- 주요 Library

```
if (!require(twitteR)) {  
  install.packages("twitteR")  
  require(twitteR)  
}
```

- 주요 함수 설명

- `setup_twitter_oauth(custom_key, custom_secret, access_token, access_token_secret)`
- `searchTwitter(keyword, n=1000)`
- `twListToDF(url)`

# Twitter 트윗 분석

## 1. Twitter API 세팅

1. app 생성
2. api 연결

## 2. 트윗 크롤링

1. 검색어 지정
2. 트윗 추출

## 3. 단어 추출

1. 트윗에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# Twitter API 연결

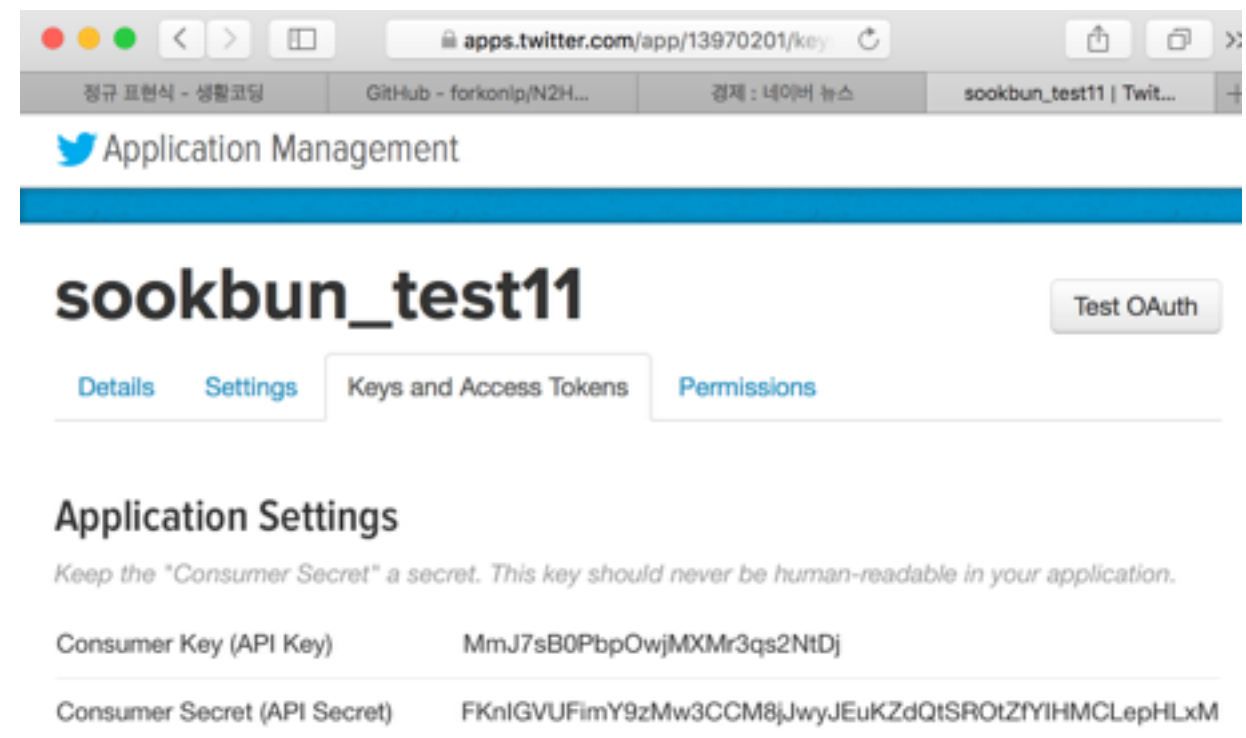
<http://apps.twitter.com>

customer key

customer secret

access token

access token secret



# Twitter API 연결

# token 설정

# 파일에는 순서대로 값이 한줄에 하나씩 있어야 한다.

```
tokens <- read.csv(file='twitter_token.txt', header=FALSE,  
                   colClasses='character', sep=',')
```

# tokens

```
consumer_key <- tokens[1,]
```

```
consumer_secret <- tokens[2,]
```

```
access_token <- tokens[3,]
```

```
access_token_secret <- tokens[4,]
```



# Twitter API 연결

# Create Twitter Connection

```
setup_twitter_oauth(consumer_key, consumer_secret,  
                    access_token, access_token_secret)
```

# Twitter 트윗 분석

## 1. Twitter API 세팅

1. app 생성
2. api 연결

## 2. 트윗 크롤링

1. 검색어 지정
2. 트윗 추출

## 3. 단어 추출

1. 트윗에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# Twitter API 연결

```
keyword <- '심심해'
```

```
## 영문일 때
```

```
results <- searchTwitter(keyword, n=100, lang="en")
```

```
## 한글일 때
```

```
results <- searchTwitter(enc2utf8(keyword), n=1000, lang='ko')
```

```
# twitter 검색 결과를 data frame으로 변환한다.
```

```
results.df <- twListToDF(results)
```

```
# 여러 컬럼 중 text 컬럼만 별도로 저장한다.
```

```
results.text <- results.df$text
```

# Twitter 트윗 분석

## 1. Twitter API 세팅

1. app 생성
2. api 연결

## 2. 트윗 크롤링

1. 검색어 지정
2. 트윗 추출

## 3. 단어 추출

1. 트윗에서 단어 목록 추출
2. 단어 데이터 정리

## 4. 출력

1. 단어별 갯수 카운트
2. 클라우드 출력

# 긍정/부정 반응 분석

# 긍정 단어 부정 단어를 가진 다음 파일이 같은 폴더에 있어야 한다.  
# 이 파일에 있는 단어를 기준으로 긍정/부정 점수를 부여한다.  
# 필요하면 이 단어 사전에 원하는 단어를 추가/ 삭제/ 수정 한다.  
# ANSI 인코딩일 때는 encoding="UTF-8 을 빼고 처리한다.

```
pos.word <- scan("positive-words-ko-v3.txt", what="character",  
                comment.char=";", fileEncoding="UTF-8")  
neg.word <- scan("negative-words-ko-v3.txt", what="character",  
                comment.char=";", fileEncoding="UTF-8")
```

# 긍정/부정 반응 분석

```
# 긍정 사전, 부정 사전에서 일치하는 단어가 있는지 찾아 낸다.  
# 있으면 각 1점을 부여한다.  
pos.matches <- match(results_nouns, pos.word); pos.matches  
neg.matches <- match(results_nouns, neg.word); neg.matches  
pos.matches <- !is.na(pos.matches); pos.matches  
neg.matches <- !is.na(neg.matches); neg.matches  
  
# 합을 구하고 긍정 - 부정 값을 구한다.  
sum(pos.matches); sum(neg.matches)  
score <- sum(pos.matches) - sum(neg.matches)
```

# Reference

- Introduction to KoNLP API

<https://github.com/haven-jeon/KoNLP/blob/master/etc/KoNLP-API.md>

- N2H4 - 네이버 뉴스 크롤링을 위한 도구

<https://github.com/forkonlp/N2H4>