

[R을 활용한 텍스트 마이닝 입문 - 서진수]

1. 현재 많이 사용되는 텍스트 분석 기법들

1) 핵심 키워드 추출 및 분석: 단어의 출현 빈도 카운트 및 워드클라우드
단어들이 발생한 빈도 후 워드클라우드로 시각화하여 핵심키워드를 도출

2) 단어간의 관계 분석: Co-occurrence Matrix

문서 내에서 동시에 빈번하게 출현하는 단어를 파악하여 단어간의 상관관계 계산

3) 감성 분석(감정 분석) : 감성사전과 단어 비교 대조하여 감성 점수산출

긍정, 부정감성사전이 만들어져 있다면 사전과 대조하여 긍정단어와 부정단어의 빈도를 추출한 후 긍정단어가 나타나면 +1, 부정단어가 나타나면 -1의 점수를 부여해 문서의 감성을 수치화함

4) 토픽 분석: 문서 내에 어떤 주제나 이슈가 존재하는지 파악할 때 사용

- LSA: 분절된 단어들에 벡터값을 부여하고 차원축소를 하여 축소된 차원에서 근접한 단어들을 주제로 묶음

- LDA: 확률을 바탕으로 단어가 특정 주제에 존재할 확률과 문서에 특정 주제가 존재할 확률을 결합확률로 추정하여 토픽추출

5) 맥락에 따른 단어 및 문서 분석: Bag of words, word2vec, doc2vec

- Bag of words: 텍스트 분석의 단위를 개별단어가 아닌 단어들의 모음으로 구분하는 방법(각 단어는 문서의 특징이 됨)

- word to vec: 문장(단어들의 모음)자체를 뉴럴넷에 입력하여 뉴럴넷이 문장 내에 있는 단어들에 벡터값을 부여하게 함

2. 먼저 알아야 할 텍스트 처리 기본 사항들

1) R에서 영어를 처리하는 일반적인 방법

```
> txt1 <- "Start R programming with R-LOVE book."
```

```
> txt1
```

```
[1] "Start R programming with R-LOVE book."
```

```
> strsplit(txt1, " ")
```

```
[[1]]
```

```
[1] "Start"      "R"          "programming" "with"       "R-LOVE"
```

```
[6] "book."
```

2) R에서 한글을 처리하는 방법

(1) 공백을 기준으로 단어 분리하기

```
> install.packages("KoNLP")
```

```
> library(KoNLP)
```

```
> install.packages("stringr")
```

```
> library(stringr)
```

```
> txt2 <- "R라뷰 책으로 R 프로그래밍을 시작하세요~!"
```

```
> txt2
```

```
[1] "R라뷰 책으로 R 프로그래밍을 시작하세요~!"
```

```
> strsplit(txt2, " ")
```

```
[[1]]
```

```
[1] "R라뷰"      "책으로"      "R"           "프로그래밍을"
```

```
[5] "시작하세요~!"
```

```
> extractNoun(txt2)
```

```
[1] "R라뷰"      "책"          "R"           "프로그래밍" "시작"
```

(2) 품사까지 자세하게 출력하기 - SimplePos09() / SimplePos22() 함수 사용

```
> SimplePos09(txt2)
```

```
$`R라뷰`
```

```
[1] "R라뷰/N"
```

```
$책으로
```

```
[1] "책/N+으로/J"
```

```
$R
```

```
[1] "R/F"
```

```
$프로그래밍을
```

```
[1] "프로그래밍/N+을/J"
```

```
$`시작하세요~`
```

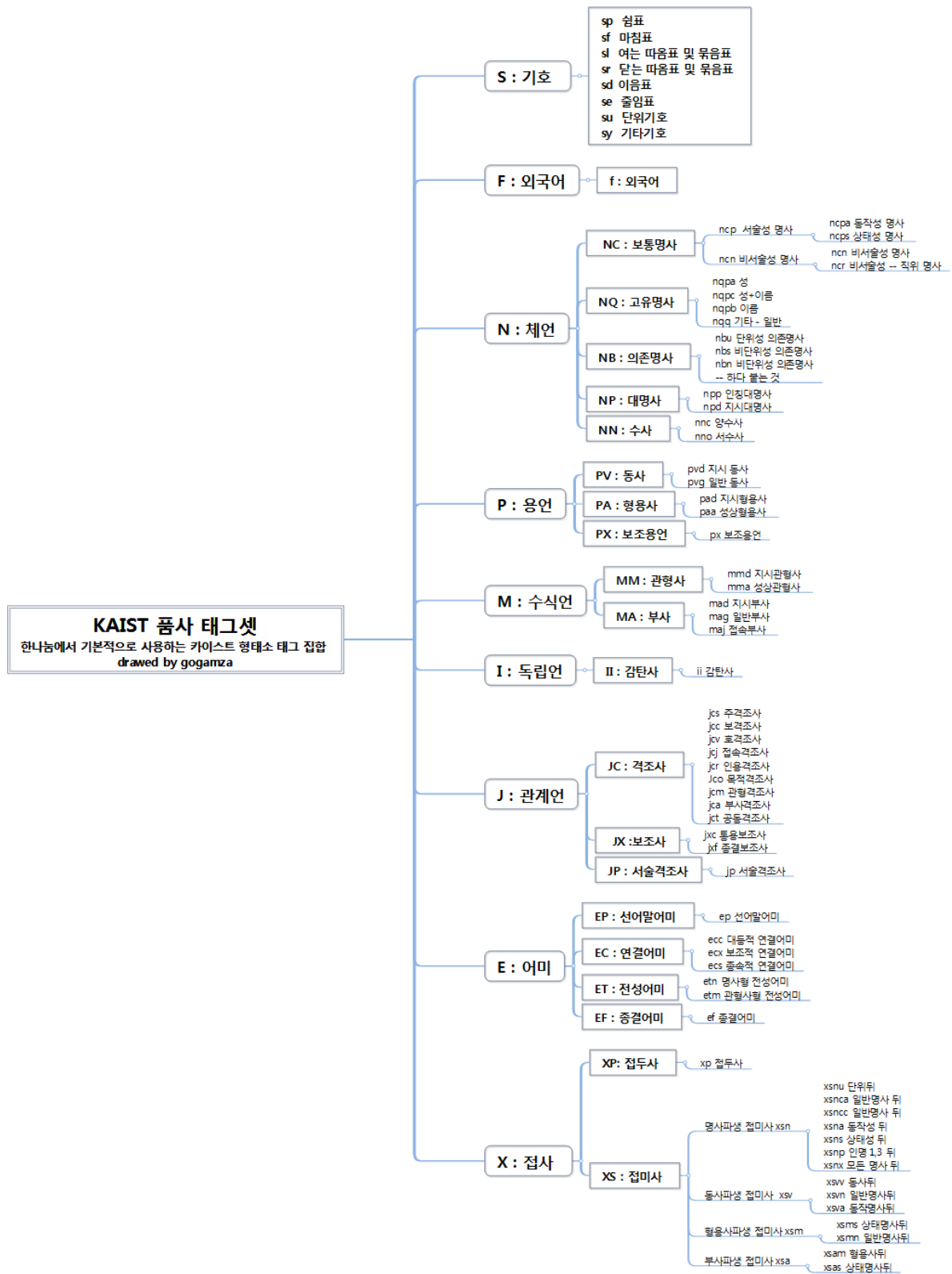
```
[1] "시작/N+하/X+세/E+요/J+~/S"
```

```
$`!`
```

```
[1] "!/S"
```

(3) 원하는 품사만 골라내기

(아래 그림은 전희원님께서 정리해 주신 표를 사용하였습니다.)



```
> txt3 <- "우리 모두 R라뷰 책으로 정말 재미있게 공부해요"
```

```
> txt4 <- SimplePos09(txt3)
```

```
> txt4
```

```
$`우리`
```

```
[1] "우리/N"
```

```
$모두
```

```
[1] "모두/M"
```

```
$R라뷰
```

```
[1] "R라뷰/N"
```

```
$책으로
```

```
[1] "책/N+으로/J"
```

```
$정말
```

```
[1] "정말/M"
```

```
$재미있게
```

```
[1] "재미있/P+게/E"
```

```
$공부해
```

```
[1] "공부해/N"
```

```
$요
```

```
[1] "요/M"
```

```
> txt_n <- str_match(txt4,'([A-Z가-힣]+)/N') # 명사 확인하기
```

```
> txt_n
```

	[1]	[2]
[1,]	"우리/N"	"우리"
[2,]	NA	NA
[3,]	"R라뷰/N"	"R라뷰"
[4,]	"책/N"	"책"
[5,]	NA	NA
[6,]	NA	NA
[7,]	"공부해/N"	"공부해"
[8,]	NA	NA

```
>
```

```

> txt_p <- str_match(txt4,'([A-Z가-힣]+)/P')
> txt_p
      [,1]      [,2]
[1,] NA        NA
[2,] NA        NA
[3,] NA        NA
[4,] NA        NA
[5,] NA        NA
[6,] "재미있/P" "재미있"
[7,] NA        NA
[8,] NA        NA
>
> txt_np <- str_match(txt4,'([A-Z가-힣]+)/[NP]')
> txt_np
      [,1]      [,2]
[1,] "우리/N"   "우리"
[2,] NA         NA
[3,] "R라뷰/N"  "R라뷰"
[4,] "책/N"     "책"
[5,] NA         NA
[6,] "재미있/P" "재미있"
[7,] "공부해/N" "공부해"
[8,] NA         NA
>
>

```

3. 한글 텍스트 분석하기

1) KoNLP() 패키지 기본 기능 익히기

이 패키지는 R 에서 한국어를 보다 쉽게 사용하기 위해 "전희원" 님께서 만드신 아주 유용한 패키지입니다. 한국어 관련 작업에서는 대부분 다 사용이 됩니다. 이 패키지에 관련된 자세한 내용은 구글에서 "KoNLP in r " 로 검색하시면 자세한 정보를 확인할 수 있습니다. 이 패키지에는 다양한 함수가 존재하는데 전체 내용을 다루기엔 지면 관계상 분량이 너무 많으므로 주로 사용되고 꼭 알아야 하는 함수에 대해 살펴보겠습니다.

(1) extractNoun () 함수 활용하기 - 한글의 명사 추출 함수

이 함수는 한글을 입력 받아서 명사를 추출 해 주는 역할을 합니다. 함수 이름 쓸 때 대소문자 특히 조심하세요. 이 함수는 Hannanum analyzer (한나눔 분석기)를 사용한다고 패키지의 저자가 밝히고 있습니다. 한나눔에 대해서 더 자세한 정보를 원하시는 분들은

(<http://kldp.net/projects/hannanum>) 를 참고하시기 바랍니다.

- 문 법 : **extractNoun**(분석할 문장이나 변수)

- 사용예 :

```
> txt1 <- readLines("좋아하는과일.txt")
```

```
> txt1
```

```
[1] "나는 사과와 바나나를 좋아합니다."
```

```
[2] "나는 바나나 바나나 바나나 바나나 바나나가 최고 좋아요!"
```

```
[3] "나는 복숭아와 사과를 좋아합니다."
```

```
[4] "나는 복숭아와 사과를 좋아합니다."
```

```
[5] "나는 사과와 포도를 좋아합니다."
```

```
[6] "나는 파인애플과 복숭아를 좋아합니다."
```

```
>
```

```
> txt2 <- extractNoun(txt1)
```

```
> txt2
```

```
[[1]]
```

```
[1] "나"      "사과"    "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" "바나나" "바나나" "바나나" "바나나" "최고"
```

[[3]]

[1] "나" "복숭아" "사과"

[[4]]

[1] "나" "복숭아" "사과"

[[5]]

[1] "나" "사과" "포도"

[[6]]

[1] "나" "파인애플" "복숭아"

명사를 잘 찾아 주는 것이 보이죠? 그런데 이 함수는 공백을 기준으로 단어를 판단합니다. 아래 예문처럼 띄어쓰기를 잘 못하면 명사를 잘 못 찾게 됩니다.

> v2 <- ("봄이지나 면여름이고 여름이지나면가을 입니다")

> extractNoun(v2)

[1] "면여름이고" "여름" "이지" "나면" "가을"

이상하게 분류되는 것이 확인 되시죠?

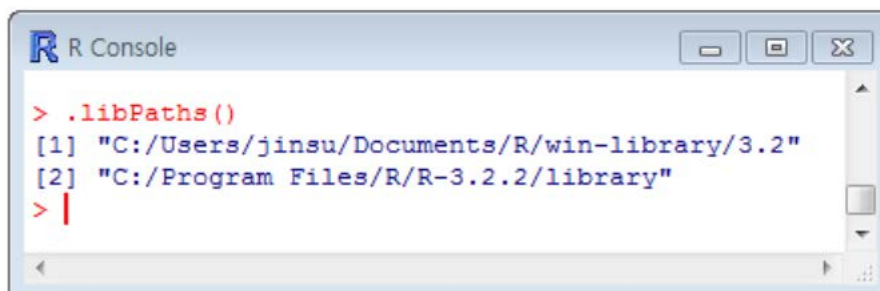
(2) 사전 활용하기

위에서 살펴 본 바와 같이 extractNoun() 함수는 공백 단위로 단어를 만들고 명사를 골라 냅니다. 그런데 이 함수는 과연 그 단어가 명사인지 아닌지는 어떻게 알까요?

바로 내부적으로 가지고 있는 사전과 비교를 하기 때문에 알 수 있는 것입니다.

먼저 사전 파일의 위치부터 확인해 보겠습니다.

사전 파일의 경로는 아래와 같이 확인할 수 있습니다.



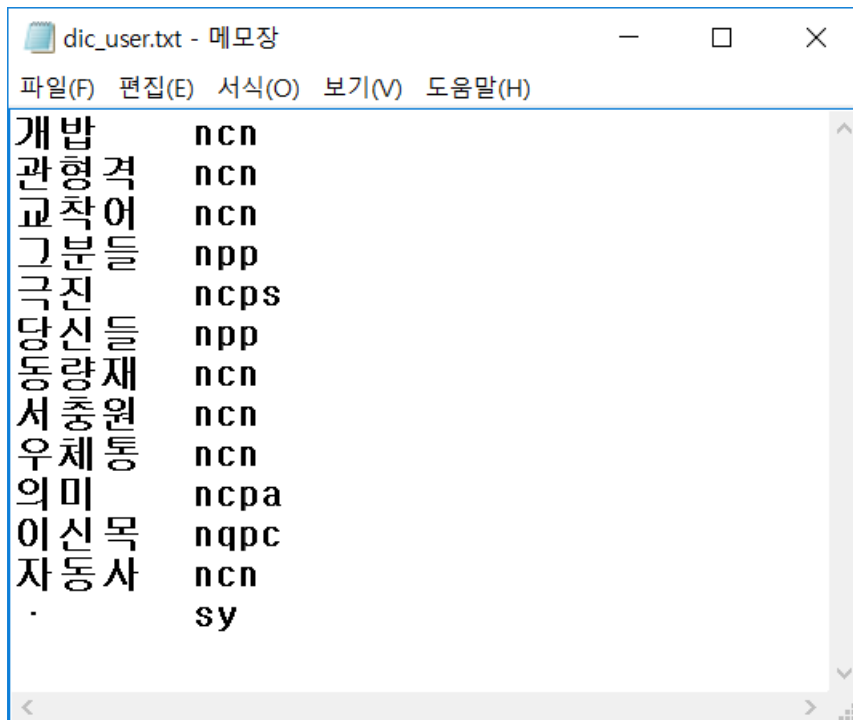
```
R Console
> .libPaths()
[1] "C:/Users/jinsu/Documents/R/win-library/3.2"
[2] "C:/Program Files/R/R-3.2.2/library"
> |
```

위 두 경로는 주요 패키지들이 설치되는 장소입니다.

그 안에 KoNLP_dic 라는 폴더가 있는데 그 안에 가면 backup 과 current 폴더가 있고

current 라는 폴더에 가면 dic_user.txt 라는 파일이 사전 파일입니다.

사전 파일은 아래와 같은 형태입니다.



위 사전의 내용은 KoNLP() 패키지를 최초 설치했을 때의 내용입니다.
단어가 몇 개 없죠?

이 사전을 사용하려면 코드에서 아래의 코드를 추가하세요.

```
> useSejongDic( )
```

```
Backup was just finished!
```

```
370957 words dictionary was built.
```

위와 같이 사전을 사용하라고 설정 한 후 한글 텍스트로 테스트 해 보겠습니다.

```
> txt_5 <- "우리는 유관순 의사와 안중근 의사가 독립투사임을 반드시 기억합니다"
```

```
> extractNoun(txt_5)
```

```
[1] "우리"      "유관"      "순"        "의사"      "안중"      "근"
```

```
[7] "의사"      "독립투사" "기억"      " "
```

결과가 약간 생각과 다르게 나오죠?

유관순, 안중근 이런 단어들이 분리되어서 나오는 이유는 사전에 등록이 안되었기 때문입니다.

위의 예에서 알 수 있듯이 정확한 한글 분석을 하기 위해서는 분석 작업을 하기 전에 해당 용어들을 사전에 아래와 같이 추가를 해야 합니다.

```
> mergeUserDic(data.frame(c('유관순','안중근'),c('ncn')))
```

```
2 words were added to dic_user.txt.
```


경고메시지(들):

'mergeUserDic' is deprecated.

Use 'buidDictionary()' instead.

See help("Deprecated")

사전에 단어를 추가한 후 다시 `extractNoun()` 으로 분석하면 아래와 같이 나옵니다.

> extractNoun(txt_5)

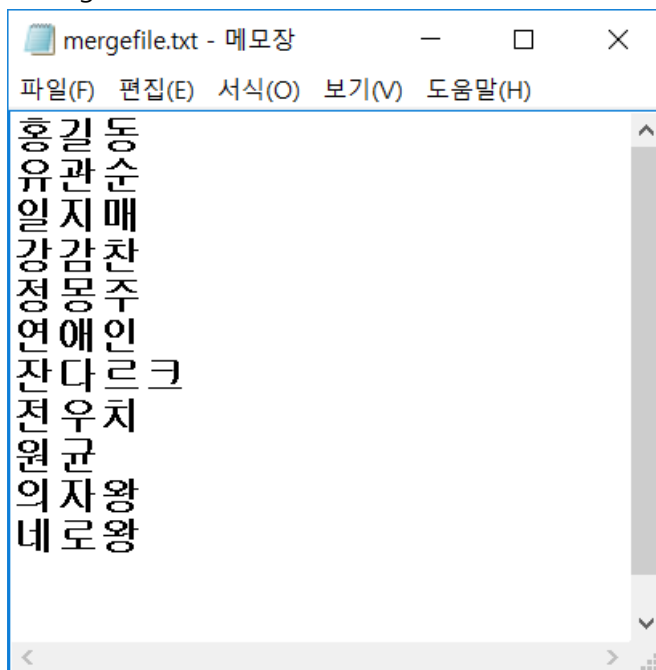
[1] "우리" "유관순" "의사" "안중근" "의사" "독립투사"

[7] "기억" "합"

참고로 만약 사전에 추가해야 할 내용이 많을 경우는 추가할 단어를 파일에 저장해 둔 후 아래와 같이 자동으로 불러서 추가할 수 있습니다.

> mergeUserDic(data.frame(readLines("mergefile.txt"), "ncn"))

mergefile.txt 의 내용입니다.



(3) 중복되는 값 제거하기

이번 예제를 위해서 위에서 사용했던 데이터를 다시 살펴 보겠습니다.

```
> txt1 <- readLines("좋아하는과일.txt")
```

```
> txt1
```

```
[1] "나는 사과와 바나나를 좋아합니다."
[2] "나는 바나나 바나나 바나나 바나나 바나나가 최고 좋아요!"
[3] "나는 복숭아와 사과를 좋아합니다."
[4] "나는 복숭아와 사과를 좋아합니다."
[5] "나는 사과와 포도를 좋아합니다."
[6] "나는 파인애플과 복숭아를 좋아합니다."
```

위 예에서 2번 줄을 보면 1 사람이 바나나를 5번 언급했습니다.

이 경우에는 5번으로 카운트 되면 안되고 1번으로 카운트 되도록 코드가 작성되어야 더 정확한 결과가 나올 것입니다.

그리고 3번줄과 4번줄은 완전 동일한 문장입니다.

즉 1 명이 2번 작성을 했을 확률이 아주 높기에 제거를 해야 합니다.

(이런 대표적인 예가 트위터 데이터의 리트윗 제거 입니다)

이번에는 이렇게 중복인 데이터가 있을 경우 어떻게 처리를 해야 하는지 간단하게 살펴 보겠습니다.

```
> txt1 <- readLines("좋아하는과일.txt")
```

```
> txt1
```

```
[1] "나는 사과와 바나나를 좋아합니다."
[2] "나는 바나나 바나나 바나나 바나나 바나나가 최고 좋아요!"
[3] "나는 복숭아와 사과를 좋아합니다."
[4] "나는 복숭아와 사과를 좋아합니다."
[5] "나는 사과와 포도를 좋아합니다."
[6] "나는 파인애플과 복숭아를 좋아합니다."
```

```
>
```

```
> txt2 <- Map(extractNoun,txt1)
```

```
> txt2
```

```
$`나는 사과와 바나나를 좋아합니다.`
```

```
[1] "나"      "사과"    "바나나"
```

```
$`나는 바나나 바나나 바나나 바나나 바나나가 최고 좋아요!`
```

```
[1] "나"      "바나나" "바나나" "바나나" "바나나" "바나나" "바나나" "최고"
```

```
$`나는 복숭아와 사과를 좋아합니다.`
```

```
[1] "나"      "복숭아" "사과"
```

```
$`나는 복숭아와 사과를 좋아합니다.`
```

```
[1] "나"      "복숭아" "사과"
```

```
$`나는 사과와 포도를 좋아합니다.`
```

```
[1] "나"      "사과" "포도"
```

```
$`나는 파인애플과 복숭아를 좋아합니다.`
```

```
[1] "나"      "파인애플" "복숭아"
```

```
> txt3 <- unique(txt2) # 중복되는 리스트를 제거합니다.
```

```
> txt3
```

```
[[1]]
```

```
[1] "나"      "사과"      "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" "바나나" "바나나" "바나나" "바나나" "최고"
```

```
[[3]]
```

```
[1] "나"      "복숭아" "사과"
```

```
[[4]]
```

```
[1] "나"      "사과" "포도"
```

```
[[5]]
```

```
[1] "나"      "파인애플" "복숭아"
```

```
> txt4 <- lapply(txt3,unique) # 각 리스트 안에서 중복되는 단어를 제거합니다
```

```
> txt4
```

```
[[1]]
```

```
[1] "나"      "사과"      "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" "최고" # 이 줄이 중복이 제거되었죠?
```

```
[[3]]
```

```
[1] "나"      "복숭아" "사과"
```

```
[[4]]
```

```
[1] "나"      "사과" "포도"
```

```
[[5]]
```

```
[1] "나"      "파인애플" "복숭아"
```

실전에서도 반드시 위의 방법으로 중복되는 데이터들을 정리해야 합니다!

(4) 필요 없는 단어 제거하기 – 특정 단어나 글자수로 제거하기

주어진 문장을 단어로 잘라 내고 중복된 단어나 리스트를 제거 한 후 결과를 보면 여전히 필요없어서 제거되어야 할 단어들이 있습니다. 위의 예에서는 “나”, “최고” 등의 글자들이죠? 이번에는 필요 없는 글자들을 어떻게 제거하는 지를 살펴 보겠습니다.

```
> txt4
```

```
[[1]]
```

```
[1] "나"      "사과"    "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" "최고"
```

```
[[3]]
```

```
[1] "나"      "복숭아" "사과"
```

```
[[4]]
```

```
[1] "나"      "사과" "포도"
```

```
[[5]]
```

```
[1] "나"      "파인애플" "복숭아"
```

먼저 특정 글자를 없애는 기능으로 `gsub()` 함수를 이용하겠습니다.

문법 : `gsub("변경전글자", "변경후글자", data)`

```
> txt5 <- rapply(txt4, function(x) gsub("최고", "", x), how = "replace")
```

```
> txt5
```

```
[[1]]
```

```
[1] "나"      "사과"    "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" ""      # <- 여기에 있던 "최고" 라는 글자 없어졌죠?
```

```
[[3]]
```

```
[1] "나"      "복숭아" "사과"
```

```
[[4]]
```

```
[1] "나"      "사과"    "포도"
```

```
[[5]]
```

```
[1] "나"      "파인애플" "복숭아"
```

만약 제거해야 할 글자가 많다면 위 코드를 여러 번 쓰는 것이 불편합니다.

그럴 경우 아래와 같이 제거하고 싶은 단어를 파일에 별도로 저장한 후 반복문에서 불러와서 제거하도록 코드를 작성하면 됩니다.

```
> txt <- readLines("gsub.txt")
> txt
> cnt_txt <- length(txt)
> cnt_txt
> for( i in 1:cnt_txt) {
+   txt5 <- rapply(txt4, function(x) gsub((txt[i]),"", x), how = "replace")
+   }
> txt
```

위 방법을 이용하면 특정 글자를 다른 것으로 변경할 수도 있습니다.

아래의 예를 보세요.

```
> txt6 <- rapply(txt5, function(x) gsub("포도", "청포도", x), how = "replace")
> txt6
```

```
[[1]]
```

```
[1] "나"      "사과"    "바나나"
```

```
[[2]]
```

```
[1] "나"      "바나나" ""
```

```
[[3]]
```

```
[1] "나"      "복숭아" "사과"
```

```
[[4]]
```

```
[1] "나"      "사과"    "청포도" # ← 포도가 청포도로 바꿨죠?
```

```
[[5]]
```

```
[1] "나"      "파인애플" "복숭아"
```

위 gsub() 함수에 정규식 을 사용할 수도 있습니다.

아래의 예를 보세요.

```
> data1 <- readLines("gsub연습_정규식.txt")
```

```
> data1
```

```
[1] "101 홍길동 ?"      "102 일지매 #"      "103 유관순 ."      "104 강감찬 /"
```

```
[5] "105 을지문덕 ₩₩" "106 김유신 1000"
```

```
> data2 <- Map(extractNoun,data1)
```

```
> data2
```

```
$`101 홍길동 ?`
```

```
[1] "101"      "홍길동"
```

```
$`102 일지매 #`
```

```
[1] "102" "지"  "#"
```

```
$`103 유관순 .`
```

```
[1] "103" "유관" "순"
```

```
$`104 강감찬 /`
```

```
[1] "104"      "강감찬"
```

```
$`105 을지문덕 ₩₩`
```

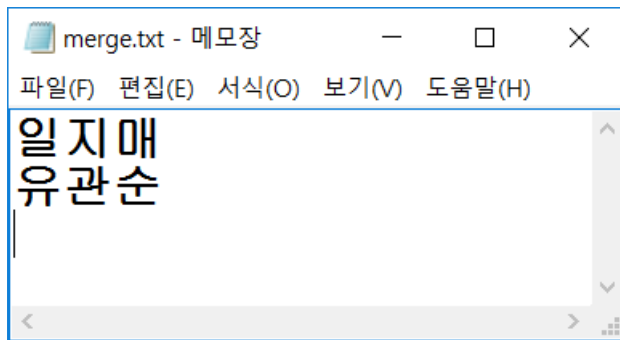
```
[1] "105"      "을지문덕" "₩₩"
```

```
$`106 김유신 1000`
```

```
[1] "106"      "김유신" "1000"
```

위 결과를 보니 "일지매" 와 "유관순" 이 이상하게 출력된 것 보이죠?

사전 파일에 위 단어를 등록한 후 다시 불러 보겠습니다.



```
> mergeUserDic(data.frame(readLines("merge.txt"), "ncn"))
```

2 words were added to dic_user.txt.

```
>
```

```
> data1 <- readLines("gsub연습_정규식.txt")
```

```
> data1
```

```
[1] "101 홍길동 ?"      "102 일지매 #"      "103 유관순 ."      "104 강감찬 /"
```

```
[5] "105 을지문덕 WW" "106 김유신 1000"
```

```
>
```

```
> data2 <- Map(extractNoun,data1)
```

```
> data2
```

```
$`101 홍길동 ?`
```

```
[1] "101"      "홍길동"
```

```
$`102 일지매 #`
```

```
[1] "102"      "일지매" "#"      #<-- 일지매 잘 보이죠?
```

```
$`103 유관순 .`
```

```
[1] "103"      "유관순"  #<-- 유관순 잘 보이죠?
```

```
$`104 강감찬 /`
```

```
[1] "104"      "강감찬"
```

```
$`105 을지문덕 WW`
```

```
[1] "105"      "을지문덕" "WW"
```

```
$`106 김유신 1000`
```

```
[1] "106"      "김유신" "1000"
```

위 결과에서 모든 숫자를 한번에 제거해 보겠습니다.

정규식에서 숫자는 wwd 였던거 기억하시죠??

```
> data3 <- rapply(data2, function(x) gsub("wwd+", "", x), how = "replace")
```

```
> data3
```

```
$`101 홍길동 ?`
```

```
[1] ""      "홍길동"
```

```
$`102 일지매 #`
```

```
[1] ""      "일지매" "#"
```

```
$`103 유관순 `
```

```
[1] ""      "유관순"
```

```
$`104 강감찬 /`
```

```
[1] ""      "강감찬"
```

```
$`105 을지문덕 ww`
```

```
[1] ""      "을지문덕" "ww"
```

```
$`106 김유신 1000`
```

```
[1] ""      "김유신" ""
```

위 결과를 보면 모든 숫자가 다 제거 된 것이 확인됩니다.

여기서는 숫자만 제거했지만 다른 정규식도 마찬가지로 사용할 수 있습니다.

한가지 주의 사항은 메타 캐릭터는 w (탈출문자)를 하나 더 쓰고 제거해야 합니다.

이번에는 글자수로 제거하는 방법을 보여드리겠습니다.

아래 예는 txt6 에 담겨있는 값 중에서 1글자 보다 크고 6글자 이하인 것만 남기고 모두 삭제하는 코드입니다.

```
> txt7 <- sapply(txt6, function(x) {Filter(function(y) {nchar(y) <= 6 && nchar(y) > 1 },x)} )
```

```
> txt7
```

```
[[1]]
```

```
[1] "사과"  "바나나"
```

```
[[2]]
```

```
[1] "바나나"
```

```
[[3]]
```



```
[1] "복숭아" "사과"
```

```
[[4]]
```

```
[1] "사과" "청포도"
```

```
[[5]]
```

```
[1] "파인애플" "복숭아"
```

만약 아래와 같이 쓴다면 1글자 보다 크고 3글자 이하인 글자들만 남기라는 뜻이겠죠?

```
> txt8 <- sapply(txt7, function(x) {Filter(function(y) {nchar(y) <= 3 && nchar(y) > 1 },x)} )  
> txt8
```

```
[[1]]
```

```
[1] "사과" "바나나"
```

```
[[2]]
```

```
[1] "바나나"
```

```
[[3]]
```

```
[1] "복숭아" "사과"
```

```
[[4]]
```

```
[1] "사과" "청포도"
```

```
[[5]]
```

```
[1] "복숭아" # ← 여기 있던 파인애플이 삭제되었죠??
```

여기까지 한글 텍스트 분석을 위한 패키지인 KoNLP() 패키지와 여러 가지 함수들을 사용하여 글자를 바꾸거나 제거하는 방법들을 살펴 보았습니다.

(5) extractNoun() 함수 사용시 UTF-8 에러 처리하기

가끔 extractNoun 함수를 사용하여 문장을 단어로 분리할 때 아래와 같은 에러가 발생하는 경우가 있습니다.

```
> tran1 <- Map(extractNoun, data1)
```

```
Error in `Encoding<-`(*tmp*, value = "UTF-8") :
```

```
문자형 벡터 인자가 와야 합니다
```

```
추가정보: 50건 이상의 경고들을 발견되었습니다 (이들 중 처음 50건을 확인하기 위해서는...
```

```
## 위 에러는 띄어쓰기가 없이 너무 긴 문장이 들어갈 경우 발생하는 에러입니다.
```

```
## 아래의 nchar 뒤에 들어가는 글자수는 작업할 때 마다 적절하게 조절해서 사용하면 됩니다
```

```
data11 <- Filter(function(x) {nchar(x) <= 200} ,data1)
```

```
data11
```

```
tran1 <- Map(extractNoun, data11)
```

```
tran1
```

```
( 이하 생략
```

```
자주 나오는 에러 이므로 이 방법을 꼭 기억해 두세요~
```

2) wordcloud() 패키지 배우기

이 패키지는 Ian Fellows 님이 워드 클라우드를 만들기 위해 제작한 패키지 입니다.

이 패키지 역시 다양한 함수를 가지고 있지만 이번 실습에서 사용된 wordcloud 함수만 살펴보겠습니다. 이 패키지에 대한 전체 설명을 보시고 싶다면 구글에서 " wordcloud in r " 로 검색하면 자세한 정보가 나옵니다.

- 문법 :

```
wordcloud(words,freq,scale=c(4,.5),min.freq=3,max.words=Inf,random.order=TRUE,  
random.color=FALSE,rot.per=.1,colors="black",ordered.colors=FALSE,  
use.r.layout=FALSE,fixed.asp=TRUE, ...)
```

- 주요 옵션 설명 :

- words : 출력할 단어들
- freq : 언급된 빈도수
- scale : 글자크기
- min.freq : 최소언급횟수지정 - 이 값 이상 언급된 단어만 출력합니다.
- max.words : 최대언급횟수지정. 이 값 이상 언급되면 삭제됩니다.
- random.order : 출력되는 순서를 임의로 지정합니다
- random.color : 글자 색상을 임의로 지정합니다.
- rot.per : 단어배치를 90 도 각도로 출력합니다.
- colors : 출력될 단어들의 색상을 지정합니다.
- ordered.colors : 이 값을 true 로 지정할 경우 각 글자별로 색상을 순서대로 지정할 수 있습니다.
- use.r.layout : 이 값을 false 로 할 경우 R 에서 c++ 코드를 사용할 수 있습니다.

이 패키지에 대한 자세한 내용은 실제 텍스트 분석을 하면서 살펴 보겠습니다.

[참고] wordcloud2 패키지도 있습니다.

```
install.packages("wordcloud2")  
library(wordcloud2)  
wordcount2 <- head(sort(wordcount, decreasing=T),100)  
wordcloud2(wordcount2,gridSize=1,size=0.5,shape="diamond")
```

shape = "diamond" , "star" , "circle" 등 다양한 옵션 사용 가능함.

예제 1. 영화 리뷰를 분석하여 워드 클라우드 작성하기

```
#####  
## 영화의 리뷰를 분석하여 워드 클라우드 그리기  
## 영화 밀정 워드 클라우드 그리기  
#####  
setwd("c:\\Wa_temp")  
install.packages("KoNLP")  
install.packages("wordcloud")  
install.packages("stringr")  
library(KoNLP)  
library(wordcloud)  
library(stringr)
```

#Step 1. 분석 파일을 불러 옵니다

```
data1 <- readLines("영화_밀정.txt")  
data1
```

#Step 2. 문장을 단어로 분리합니다.

```
#####  
##> tran1 <- Map(extractNoun, data1)  
## Error in `Encoding<-`(`*tmp*`, value = "UTF-8") :  
## 문자형 벡터 인자가 와야 합니다  
## 위 에러가 나오면 아래의 글자수로 걸러주는 작업을 합니다.  
## 위 에러는 띄어쓰기가 없이 너무 긴 문장이 들어갈 경우 발생하는 에러입니다.  
## 아래의 nchar 뒤에 들어가는 글자수는 작업할 때 마다 적절하게 조절해서 사용하면 됩니다
```

```
data2 <- Filter(function(x) {nchar(x) <= 200} ,data1)  
data2
```

```
tran1 <- Map(extractNoun, data2)  
tran1  
tran11 <- unique(tran1)  
tran2 <- sapply(tran11, unique)  
tran2
```

```
tran3 <- rapply(tran2, function(x) gsub("리뷰", "", x), how = "replace")  
tran3 <- rapply(tran3, function(x) gsub("영화", "", x), how = "replace")
```

```

tran3 <- rapply(tran3, function(x) gsub("평점", "", x), how = "replace")
tran3 <- rapply(tran3, function(x) gsub("내용", "", x), how = "replace")
tran3 <- rapply(tran3, function(x) gsub("제외", "", x), how = "replace")
tran3 <- rapply(tran3, function(x) gsub("ㅋㅋㅋ", "", x), how = "replace")
tran3 <- rapply(tran3, function(x) gsub("ㄱㄱㄱ", "", x), how = "replace")
tran3

tran4 <- sapply(tran3, function(x) {Filter(function(y) {nchar(y) <= 6 && nchar(y) > 1},x)} )
tran4

```

#Step 3. 공백을 제거하기 위해 저장 후 다시 불러 옵니다.

```

write(unlist(tran4),"밀정_2.txt")
data4 <- read.table("밀정_2.txt")
data4
nrow(data4)

```

#Step 4. 각 단어별로 집계하여 출현 빈도를 계산합니다 (1차 확인 단계)

```

wordcount <- table(data4)
wordcount
wordcount <- Filter(function(x) {nchar(x) <= 10},wordcount)
head(sort(wordcount, decreasing=T),100)

```

#Step 5. 필요없는 단어를 제거한 후 공백을 제거하고 다시 집계를 합니다.

이 과정을 여러 번 반복하여 필요 없는 단어들은 모두 제거해야 합니다.

```

txt <- readLines("영화gsub.txt")
txt
cnt_txt <- length(txt)
cnt_txt
for( i in 1:cnt_txt) {
  tran3 <- rapply(tran3, function(x) gsub((txt[i]),"", x), how = "replace")
}
tran3

```

```

data3 <- sapply(tran3, function(x) {Filter(function(y) { nchar(y) >=2 },x)} )
write(unlist(data3),"밀_2.txt")
data4 <- read.table("밀_2.txt")
data4
nrow(data4)

```

```
wordcount <- table(data4)
wordcount
head(sort(wordcount, decreasing=T),100)
```

#Step 6. 전처리 작업이 모두 완료되면 워드 클라우드를 그립니다.

```
library(RColorBrewer)
palette <- brewer.pal(7,"Set2")
wordcloud(names(wordcount),freq=wordcount,scale=c(5,1),rot.per=0.25,min.freq=5,
random.order=F,random.color=T,colors=palette)
legend(0.3,1,"영화 댓글 분석 - 밀정",cex=1.2,fill=NA,border=NA,bg="white",
text.col="red",text.font=2,box.col="red")

savePlot("영화_밀정.png",type="png")
```

영화 댓글 분석 - 밀정



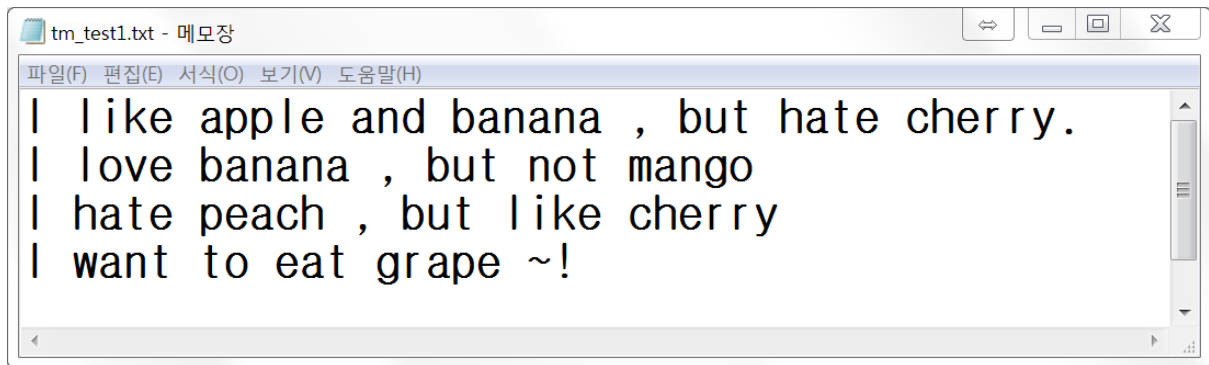
연습문제 : 주어진 다산콜센터상담내역_연습.txt파일을 분석하여 워드 클라우드를 생성하세요. (맥북을 쓰시는 분들은 다산콜센터상담내역_연습_mac.txt 파일을 이용하세요)

4. 영어 텍스트 분석하기 – tm() 패키지 사용하기

tm 패키지 실습에 사용될 테스트용 파일은 아래와 같습니다.

이건 어렵지 않으니까 직접 만드셔도 되겠죠?

메모장 열어서 아래와 같이 입력하신 후 작업 디렉토리에 tm_test1.txt 으로 저장해주세요.



위와 같이 4 줄로 되어 있는 tm_test1.txt 파일을 불러와서 tm 패키지를 활용해서 분석하는 방법을 보여 드리겠습니다.

아래 설명에서 편의상 입력하는 명령어는 빨간색으로, 출력 결과는 검은색으로 표시합니다.

눈으로만 보고 넘어가지 마시고 꼭 직접 타이핑 하시면서 따라 해 보세요~

#Step 1. 작업 디렉토리 지정해야겠죠?

```
> setwd("c:\\Wr_temp")
```

#Step 2. 필요한 패키지를 설치합니다. 여기서는 tm 패키지와 wordcloud 패키지를 설치합니다.

```
> install.packages("wordcloud")
```

```
> install.packages("tm")
```

```
> library("wordcloud")
```

```
> library("tm")
```

#Step 3. 데이터를 불러 옵니다.

```
> data1 <- readLines("tm_test1.txt")
```

```
> data1
```

```
[1] "I like apple and banana , but hate cherry."
```

```
[2] "I love banana , but not mango"
```

```
[3] "I hate peach , but like cherry"
```

```
[4] "I want to eat grape ~!"
```

class(data1) # data1 의 유형이 벡터 형태입니다.

[1] "character"

#만약 mac 을 사용할 경우 로딩한 글씨가 깨질 경우 아래 코드에서 주석을 제거해서

#utf-8로 변환을 해야 글씨가 깨지지 않고 잘 보입니다.

#data1 <- iconv(data1,"WINDOWS-1252","UTF-8")

#Step 4. 위 4 줄을 tm 패키지가 처리할 수 있는 형태인 Corpus (말뭉치) 형태로 변환합니다.

> corp1 <- Corpus(VectorSource(data1)) # 벡터이므로 **VectorSource()** 함수 사용함

> corp1 # Dataframe 의 경우 **DataframeSource()** 함수 씀.

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: **documents: 4**

#위 corp1 명령의 결과에서 documents : 4 부분이 중요합니다.

document 란 tm 패키지가 작업할 수 있는 특별한 형태를 의미하며 일반적으로는

1 줄이 1개의 document 가 됩니다. 위의 경우 원본 파일이 총 4 줄이라 documents : 4 입니다.

> inspect(corp1) # corpus 안의 내용 살펴보기

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: documents: 4

[[1]] # 첫번째 줄을 corpus 로 변환한 내용입니다.

<<PlainTextDocument>>

Metadata: 7

Content: chars: 42 # 1 번째 줄이 42 글자라는 뜻입니다. 세어 보세요~^^

[[2]] # 두번째 줄을 corpus 로 변환한 내용입니다.

<<PlainTextDocument>>

Metadata: 7

Content: chars: 29

[[3]] # 세번째 줄을 corpus 로 변환한 내용입니다.

<<PlainTextDocument>>

Metadata: 7

Content: chars: 30


```
[[4]] # 네번째 줄을 corpus 로 변환한 내용입니다.
```

```
<<PlainTextDocument>>
```

```
Metadata: 7
```

```
Content: chars: 22
```

```
# tm 패키지가 분석 할 수 있는 Term-Document 형식의 Matrix 로 변환해야 합니다.
```

```
> tdm <- TermDocumentMatrix(corp1)
```

```
> tdm
```

```
<<TermDocumentMatrix (terms: 15, documents: 4)>>
```

```
Non-/sparse entries: 20/40
```

```
Sparsity : 67%
```

```
Maximal term length: 7
```

```
Weighting : term frequency (tf)
```

```
#위 결과에서 초록색 줄을 유심히 보세요.
```

```
# terms : 15 는 총 15 개의 단어를 골랐다는 뜻이고 documents :4 는 소스가 4 개의 문장이라는
```

```
# 뜻입니다. sparsity 가 67% 는 tdm 안에 0 인 원소가 67% 라는 의미입니다.
```

```
# Term-Document Matrix 는 tm 패키지만 볼 수 있으므로 일반적으로 사용되는 Matrix 로 변환함
```

```
# 그래야 사람이 내용을 확인하기 쉽습니다.
```

```
> m <- as.matrix(tdm)
```

```
> m
```

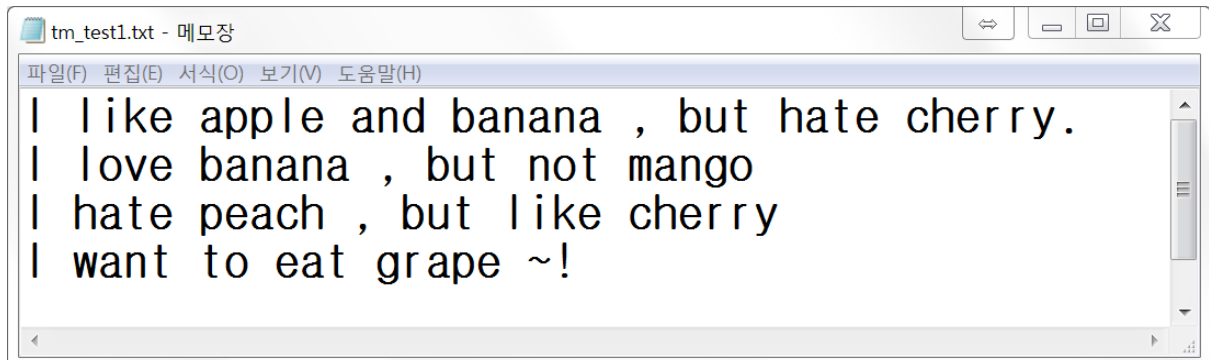
	Docs			
Terms	1	2	3	4
and	1	0	0	0
apple	1	0	0	0
banana	1	1	0	0
but	1	1	1	0
cherry	0	0	1	0
cherry	1	0	0	0
eat	0	0	0	1
grape	0	0	0	1
hate	1	0	1	0
like	1	0	1	0
love	0	1	0	0
mango	0	1	0	0
not	0	1	0	0

```

peach    0  0  1  0
want     0  0  0  1

```

위와 같이 결과가 조회됩니다.
 # 위 결과를 해석해 보면 첫번째 Terms 인 and 는 1번 문장에는 나오고(1) , 2,3,4 번은 모두
 # 0 이므로 나오지 않는다는 의미입니다.
 # 앞의 원본 파일을 한번 더 볼까요?



정말 첫 번째 문장에만 and 가 있고 나머지 줄에는 없죠?
 # 다른 Terms 도 그렇게 해석하면 됩니다.
 # 그런데 추출된 Terms 들을 잘 보면 별 의미 없는 단어들이 보입니다.
 # 예를 들어 and , but , not 같은 전치사 , 접속사 같은 단어들 말이죠.
 # 이런 단어들을 불용어라고 부르고 이런 단어들을 미리 정의 해 놓고 사용하면 제거됩니다.
 # corpus 안에 있는 불용어나 제거 하고 싶은 단어를 제거하는 방법은 tm_map() 함수를
 # 이용하면 됩니다. 아래의 예를 보세요.

```

> corp2 <- tm_map(corp1,stripWhitespace) # 여러 개의 공백을 하나의 공백으로 변환합니다
> corp2 <- tm_map(corp2,tolower) # 대문자가 있을 경우 소문자로 변환합니다
> corp2 <- tm_map(corp2,removeNumbers) # 숫자를 제거합니다
> corp2 <- tm_map(corp2,removePunctuation) # 마침표,coma,세미콜론,콜론 등의 문자 제거
> corp2 <- tm_map(corp2,PlainTextDocument)
> sword2 <- c(stopwords('en'),"and","but","not") # 기본 불용어 외 불용어로 쓸 단어 추가하기
> corp2 <- tm_map(corp2,removeWords,sword2) # 불용어 제거하기 (전치사 , 관사 등)

```

불용어나 공백등이 제거 된 후 다시 Term-Document Matrix 를 생성해서 볼까요?

```

> tdm2 <- TermDocumentMatrix(corp2)
> tdm2
<<TermDocumentMatrix (terms: 11, documents: 4)>> # terms 가 4 개 줄었죠?
Non-/sparse entries: 15/29

```

Sparsity : 66%

Maximal term length: 6

Weighting : term frequency (tf)

Matrix 로 변환해서 내용을 확인해 보겠습니다.

```
> m2 <- as.matrix(tdm2)
```

```
> m2
```

	Docs			
Terms	character(0)	character(0)	character(0)	character(0)
apple	1	0	0	0
banana	1	1	0	0
cherry	1	0	1	0
eat	0	0	0	1
grape	0	0	0	1
hate	1	0	1	0
like	1	0	1	0
love	0	1	0	0
mango	0	1	0	0
peach	0	0	1	0
want	0	0	0	1

```
> class(m2)
```

```
[1] "matrix"
```

위 결과를 보면 tm_map() 으로 제거한 and , but , not 이 사라진 것이 확인됩니다.

그런데 Docs 의 이름이 character(0) 로 변환되어 어떤 문서인지 구분이 안됩니다.

그래서 Matrix 명령어 중 colnames() 로 Docs 이름을 변경하겠습니다.

```
> colnames(m2) <- c(1:4)
```

```
> m2
```

	Docs			
Terms	1	2	3	4
apple	1	0	0	0
banana	1	1	0	0
cherry	1	0	1	0
eat	0	0	0	1
grape	0	0	0	1
hate	1	0	1	0
like	1	0	1	0
love	0	1	0	0

```
mango 0 1 0 0
peach 0 0 1 0
want 0 0 0 1
```

위와 같이 단어들이 추출되었습니다. 보기가 불편하죠?

그래서 위 결과를 단어 별로 집계를 해 보겠습니다.

```
> freq1 <- sort(rowSums(m2),decreasing=T)
```

```
> head(freq1,20)
```

```
banana cherry hate like apple eat grape love mango peach want
      2      2      2      2      1      1      1      1      1      1
```

위 결과처럼 행의 합계를 구할 때는 rowSums() 함수를 사용하면 됩니다.

만약 컬럼별로 집계를 하고 싶다면 아래와 같이 colSums() 함수를 사용하세요.

```
> freq2 <- sort(colSums(m2),decreasing=T)
```

```
> head(freq2,20)
```

```
1 3 2 4
```

```
5 4 3 3
```

만약 Term Document Matrix 에서 특정 회수 이상 언급된 것들만 출력하고 싶을 경우

아래와 같이 findFreqTerms() 사용 하세요.

```
> findFreqTerms(tdm2,2)
```

```
[1] "banana" "cherry" "hate" "like"
```

만약 특정 단어와 상관 관계를 찾고 싶을 경우 아래와 같이 findAssocs() 사용 하세요.

먼저 apple 단어와 상관계수가 0.5 이상인 값들만 출력하는 예입니다.

```
> findAssocs(tdm2,"apple",0.5)
```

```
$apple
```

```
banana cherry hate like
      0.58 0.58 0.58 0.58
```

```
> findAssocs(tdm2,"apple",0.6)
```

```
$apple
```

numeric(0) # 없을 경우 왼쪽과 같이 0 으로 나옵니다.

위 2 가지 함수를 사용해서 Term Document Matrix 내의 데이터를 쉽게 조회할 수 있어요.

위와 같이 집계된 내용을 워드 클라우드로 표현하겠습니다.

R라뷰 / 서진수 저 / 도서출판 더알음

```

> library(RColorBrewer)
> palette <- brewer.pal(7,"Set3")
> wordcloud(names(freq1),freq=freq1,scale=c(5,1),min.freq=1,colors=palette,random.order=F,
+ random.color=T)
# 워드 클라우드에 제목을 표시합니다
> legend(0.3,1 ,"tm Package Test #1",cex=1,fill=NA,border=NA,bg="white" ,
+ text.col="red",text.font=2,box.col="red")

```

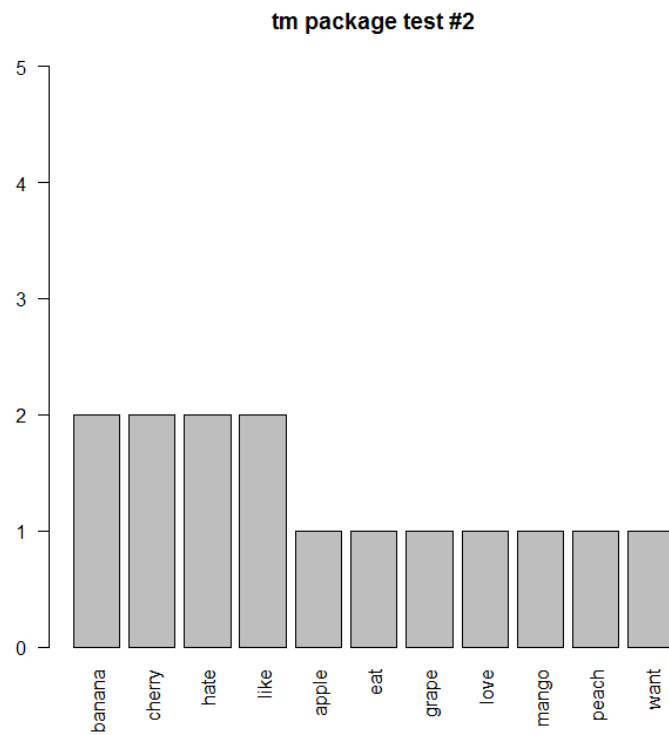
tm Package Test #1



```

# 이번에는 위 결과를 Bar chart 로 출력하겠습니다.
> barplot(freq1,main="tm package test #2",las=2,ylim=c(0,5))

```



위 예에서는 워드 클라우드와 bar chart 만 출력했으나 당연히 pie chart 나 line chart 등
다른 차트로도 얼마든지 표현할 수 있습니다. 꼭 해보세요.

```
# 이번 예제는 앞에서 배웠던 내용을 활용해 보는 시간입니다.
# 앞에서 이야기한대로 스티브 잡스님이 스탠포드 대학교에 가서 연설한 연설문을
# tm() 패키지를 사용하여 분석해 보겠습니다.
# 앞에서 공부할 때 자세하게 설명해서 여기서는 간략하게 설명하겠습니다.
# 분석용 파일은 http://cafe.naver.com/theareum/196 파일을 다운 받으세요~
```

```
> setwd("c:\\Wr_temp")
```

```
> data1 <- readLines("steve.txt")
```

```
> data1
```

```
[1] "'You've got to find what you love,' Jobs says"
```

```
[2] ""
```

```
[3] "This is the text of the Commencement address by Steve Jobs, CEO of Apple Computer and  
of Pixar Animation Studios, delivered on June 12, 2005."
```

```
[4] ""
```

```
[5] " "
```

```
[6] "I am honored to be with you today at your commencement from one of the finest  
universities in the world. I never graduated from college. Truth be told, this is the closest I've ever  
gotten to a college graduation. Today I want to tell you three stories from my life. That's it. No  
big deal. Just three stories. "
```

```
[7] ""
```

```
[8] "The first story is about connecting the dots. "
```

```
[9] "I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in  
for another 18 months or so before I really quit. So why did I drop out? "
```

```
( 지면 관계상 이하 내용은 생략하겠습니다 )
```

```
#만약 mac 을 사용할 경우 아래 코드에서 주석을 제거해서
```

```
#utf-8로 변환을 해야 합니다.
```

```
#data1 <- iconv(data1,"WINDOWS-1252","UTF-8")
```

```
> corp1 <- Corpus(VectorSource(data1)) # 말뭉치로 변환하기
```

```
> corp1
```

```
<<VCorpus>>
```

```
Metadata: corpus specific: 0, document level (indexed): 0
```

```
Content: documents: 59
```

```
> inspect(corp1) # corpus 안의 내용 살펴보기
```

```
<<VCorpus>>
```

```
Metadata: corpus specific: 0, document level (indexed): 0
```

```
Content: documents: 59
```

```
[[1]]
```

```
<<PlainTextDocument>>
```

```
Metadata: 7
```

```
Content: chars: 45
```

```
[[2]]
```

```
<<PlainTextDocument>>
```

```
Metadata: 7
```

```
Content: chars: 0
```

(지면 관계상 이하 내용은 생략합니다)

```
# tm_map() 함수는 corpus 안에 있는 문서의 내용을 지정된 형태로 변경해주는 역할을 합니다.
```

```
> corp2 <- tm_map(corp1,stripWhitespace) # 여러개의 공백을 하나의 공백으로 변환합니다
```

```
> corp2 <- tm_map(corp2,tolower) # 대문자가 있을 경우 소문자로 변환합니다
```

```
> corp2 <- tm_map(corp2,removeNumbers) # 숫자를 제거합니다
```

```
> corp2 <- tm_map(corp2,removePunctuation) # 마침표,콤마,세미콜론,콜론 등 문자 제거합니다
```

```
> corp2 <- tm_map(corp2,PlainTextDocument)
```

```
> stopwords2 <- c(stopwords('en'),"and","but") # 기본 불용어 외에 불용어로 쓸 단어 추가하기
```

```
> corp2 <- tm_map(corp2,removeWords,stopword2) # 불용어 제거하기 (전치사 , 관사 등)
```

```
# 아래 코드에서 wordLengths 옵션은 몇 글자 이상 되는 단어들만 가져와서 Term Document
```

```
# Matrix 를 생성하는 명령입니다.
```

```
# 단어가 많을 때 특정 글자수 이상의 단어만 골라와서 사용할 때 아주 많이 사용됩니다.
```

```
> corp3 <- TermDocumentMatrix(corp2,control=list(wordLengths=c(1,Inf)))
```

```
> corp3
```

```
<<TermDocumentMatrix (terms: 611, documents: 59)>>
```

```
Non-/sparse entries: 924/35125
```

```
Sparsity : 97%
```

```
Maximal term length: 14
```

```
Weighting : term frequency (tf)
```

```
> findFreqTerms(corp3,10) # TermDocumentMatrix 에서 10 회 이상 언급된 단어 출력하기
```

```
[1] "college" "life"
```

```
> findAssocs(corp3,"apple",0.5) # apple 단어와 0.5 이상의 상관관계 있는 단어 출력하기
```

```
$apple
```


company	fired	adult	billion	board	creation
0.87	0.78	0.73	0.73	0.73	0.73
devastating	directors	diverge	earlier	employees	eventually
0.73	0.73	0.73	0.73	0.73	0.73
falling	focus	garage	gone	grew	grown
0.73	0.73	0.73	0.73	0.73	0.73
hard	hired	lucky	publicly	released	sided
0.73	0.73	0.73	0.73	0.73	0.73
talented	two	us	visions	well	worked
0.73	0.73	0.73	0.73	0.73	0.73
woz	started	went	began	year	just
0.73	0.71	0.68	0.67	0.64	0.62
loved	thought	pixar			
0.58	0.53	0.52			

```
> corp4 <- as.matrix(corp3)
```

```
> corp4
```

(출력 결과가 너무 길어 생략하겠습니다)

```
> freq1 <- sort(rowSums(corp4),decreasing=T)
```

```
> freq2 <- sort(colSums(corp4),decreasing=T)
```

```
> head(freq2,20)
```

character(0)	character(0)	character(0)	character(0)	character(0)
77	71	68	65	63
character(0)	character(0)	character(0)	character(0)	character(0)
62	55	52	49	49
character(0)	character(0)	character(0)	character(0)	character(0)
48	47	45	43	37
character(0)	character(0)	character(0)	character(0)	character(0)
37	37	30	28	24

#위와 같이 컬럼 이름이 character(0) 으로 되면 안된다고 했죠?

corp4 에 총 몇 개의 컬럼이 있는지 확인 후 숫자로 변경하겠습니다.

```
> dim(corp4)
```

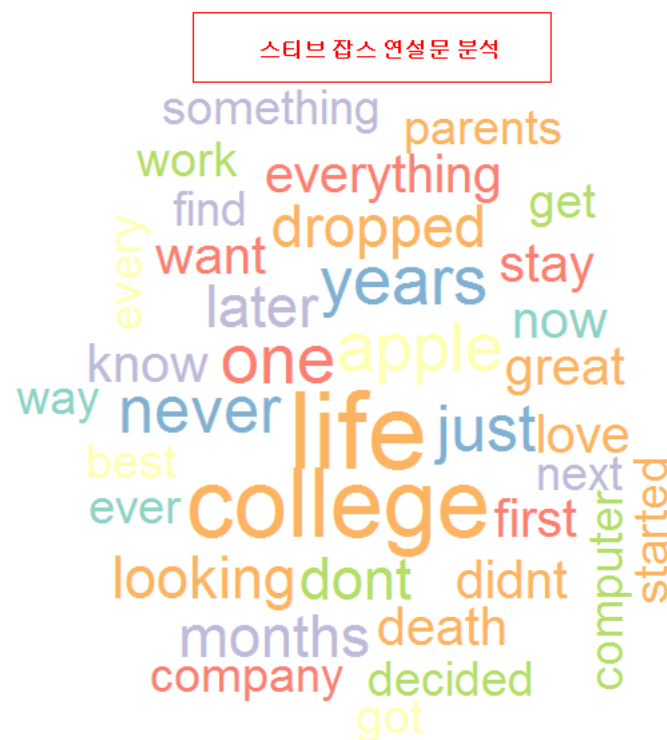
```
[1] 611 59 # 행 수: 611 행 , 컬럼 수 : 59 개 라는 의미 입니다.
```

```

> colnames(corp4) <- c(1:59)
> freq2 <- sort(colSums(corp4),decreasing=T)

# 위 결과에서 5회 이상 언급된 단어들만 모아서 워드 클라우드로 표현하기
> library(RColorBrewer)
> palette <- brewer.pal(7,"Set3")
> wordcloud(names(freq1),freq=freq1,scale=c(5,1),min.freq=5,colors=palette,random.order=F,
+ random.color=T)
> legend(0.3,1,"스티브 잡스 연설문 분석",cex=1,fill=NA,border=NA,bg="white",
+ text.col="red",text.font=2,box.col="red")

```



[연습문제 : 미국 대통령인 트럼프의 연설문을 분석해서 워드 클라우드로 출력하세요]
 (제공되는 트럼프연설문.txt 파일을 사용하세요)

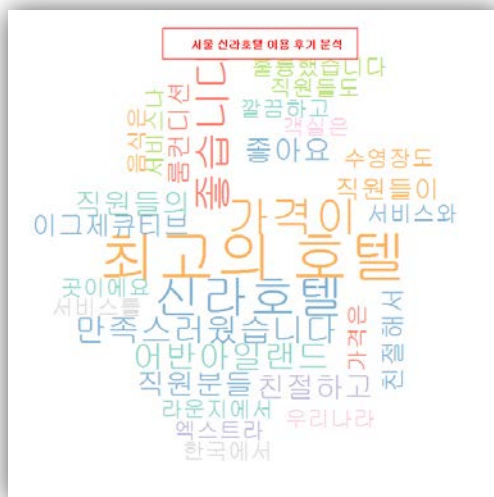
5. KoNLP() 패키지와 tm() 패키지 비교하기

이번 예제는 텍스트 분석에서 많이 사용되는 tm 패키지와 KoNLP 패키지를 사용해서 나온 결과를 비교해서 보여 드리겠습니다. 예제로 사용할 파일은 국내 유명 호텔 중 2 곳을 이용한 고객들의 이용 후기를 모은 파일입니다. 이미 앞에서 KoNLP 패키지와 tm 패키지 설명은 다 했기 때문에 내용이 중복되어 여기서는 결과 화면과 소스 코드만 보여 드리겠습니다.

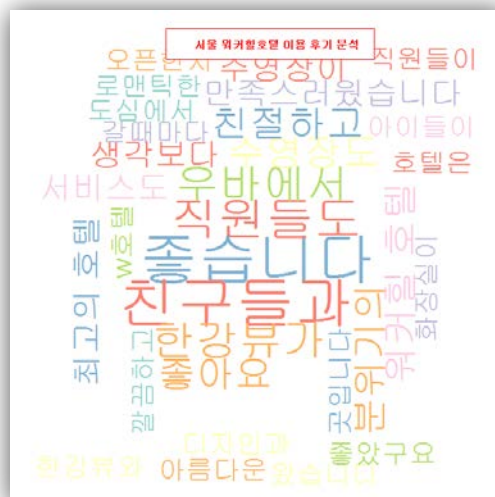
분석용 파일은 <http://cafe.naver.com/theareum/340> 번 자료를 다운로드 받으세요.

1) tm 패키지를 사용한 이용후기 분석 후 시각화 하기

[신라호텔 이용 후기 분석 화면]

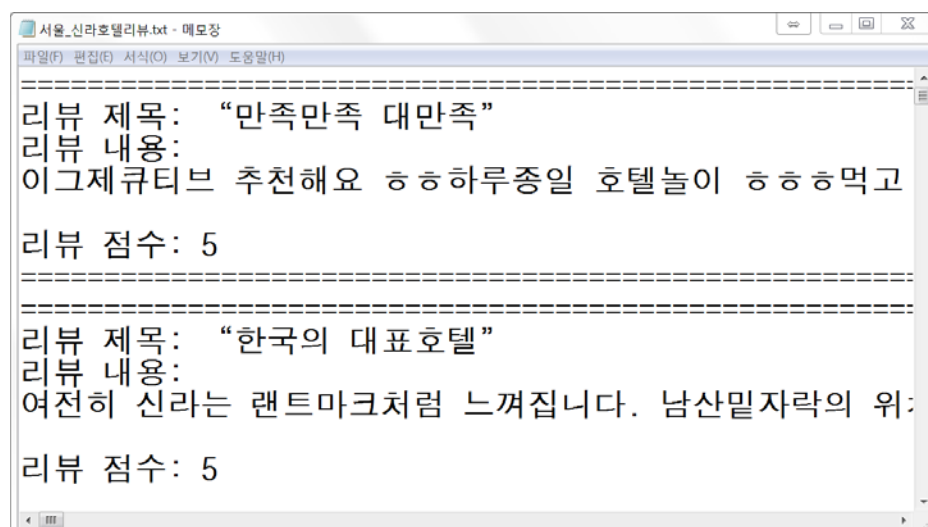


[워커히 호텔 이용 후기 분석 화면]



[원본 파일 일부]-

국내 호텔 이용 후기 사이트에서 위 두 호텔 이용 후기 약 100 여건을 크롤링 함



[소스 코드 - tm 패키지로 신라호텔 이용후기 분석하기]

```
> setwd("c:\\Wa_temp")
> install.packages("tm")
> library("tm")
> install.packages("wordcloud")
> library("wordcloud")

> shin1 <- readLines("서울_신라호텔리뷰.txt")
> shin5 <- Corpus(VectorSource(shin1))
> inspect(shin5)

> shin6 <- tm_map(shin5,stripWhitespace) # 여러개의 공백을 하나의 공백으로 변환합니다
> shin6 <- tm_map(shin6,tolower) # 대문자가 있을 경우 소문자로 변환합니다
> shin6 <- tm_map(shin6,removePunctuation) # 마침표,콤마,세미콜론,콜론 등의 문자 제거
> stopword2 <- c(stopwords('english'),"수 있는","같습니다","싶습니다","있습니다","서울에서",
+               "하고","은","서울에서","이용했습니다","곳이에요") # 기본 불용어 외에 불용어로 쓸
                                   # 단어 추가하기
> shin6 <- tm_map(shin6,removeWords,stopword2) # 불용어 제거하기 (전치사 , 관사 등)
> shin6 <- tm_map(shin6,PlainTextDocument)
> inspect(shin6)
> shin7 <- TermDocumentMatrix(shin6)
> inspect(shin7)
> findFreqTerms(shin7,5) # TermDocumentMatrix 안에서 지정된 빈도 수 이상 언급된 단어
> findAssocs(shin7,"신라호텔",0.5)
> shin8 <- as.matrix(shin7) # 일반 Matrix 형태로 변환합니다.
> shin8
> nrow(shin8) # 찾은 단어 개수 , 매트릭스에서 행 수가 됩니다.

> freq1 <- sort(rowSums(shin8),decreasing=T)
> head(freq1,20)

> library(RColorBrewer)
> palette <- brewer.pal(9,"Set3")
> wordcloud(names(freq1),freq=freq1,scale=c(5,1),rot.per=0.25,min.freq=3,
+   random.order=F,random.color=T,colors=palette)
> legend(0.3,1,"서울 신라호텔 이용 후기 분석",cex=0.8,fill=NA,border=NA,bg="white",
+   text.col="red",text.font=2,box.col="red")
```

tm 패키지를 활용한 서울 워커힐 호텔 이용후기 분석하기

```
> setwd("c:\\Wa_temp")
> install.packages("tm")
> install.packages("wordcloud")
> library("tm")
> library("wordcloud")
> wh1 <- readLines("서울_워커힐호텔리뷰.txt")
> wh2 <- Corpus(VectorSource(wh1))
> inspect(wh2)

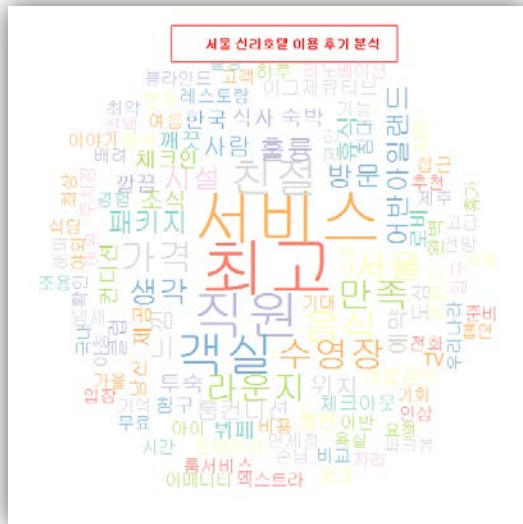
> wh3 <- tm_map(wh2,stripWhitespace) # 여러개의 공백을 하나의 공백으로 변환합니다
> wh3 <- tm_map(wh3,tolower) # 대문자가 있을 경우 소문자로 변환합니다
> wh3 <- tm_map(wh3,removePunctuation) # 마침표,코마,세미콜론,콜론 등의 문자 제거
> stopword2 <- c(stopwords('english'),"수 있는","같습니다","싶습니다","있습니다","서울에서",
+               "하고","은","곳이에요","서울에서","이용했습니다") # 기본 불용어 외에 불용어로 쓸
                                # 단어 추가하기
> wh3 <- tm_map(wh3,removeWords,stopword2) # 불용어 제거하기 (전치사 , 관사 등)
> wh3 <- tm_map(wh3,PlainTextDocument)
> wh4 <-TermDocumentMatrix(wh3)
> inspect(wh4)
> findFreqTerms(wh4,5) # TermDocumentMatrix 안에서 지정된 빈도 수 이상 언급된 단어 찾기

> wh5 <- as.matrix(wh4) # 일반 Matrix 형태로 변환합니다.
> nrow(wh5) # 찾은 단어 개수 , 매트릭스에서 행 수가 됩니다.
> freq1 <- sort(rowSums(wh5),decreasing=T)

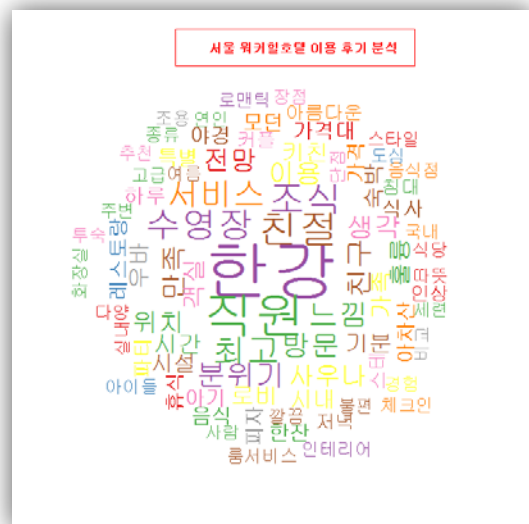
> library(RColorBrewer)
> palete <- brewer.pal(8,"Set3")
> wordcloud(names(freq1),freq=freq1,scale=c(5,1),rot.per=0.25,min.freq=3,
+   random.order=F,random.color=T,colors=palete)
> legend(0.3,1 ,"서울 워커힐호텔 이용 후기 분석  ",cex=0.8,fill=NA,border=NA,bg="white" ,
+       text.col="red",text.font=2,box.col="red")
```

2) KoNLP 패키지로 이용후기 분석 후 시각화 하기

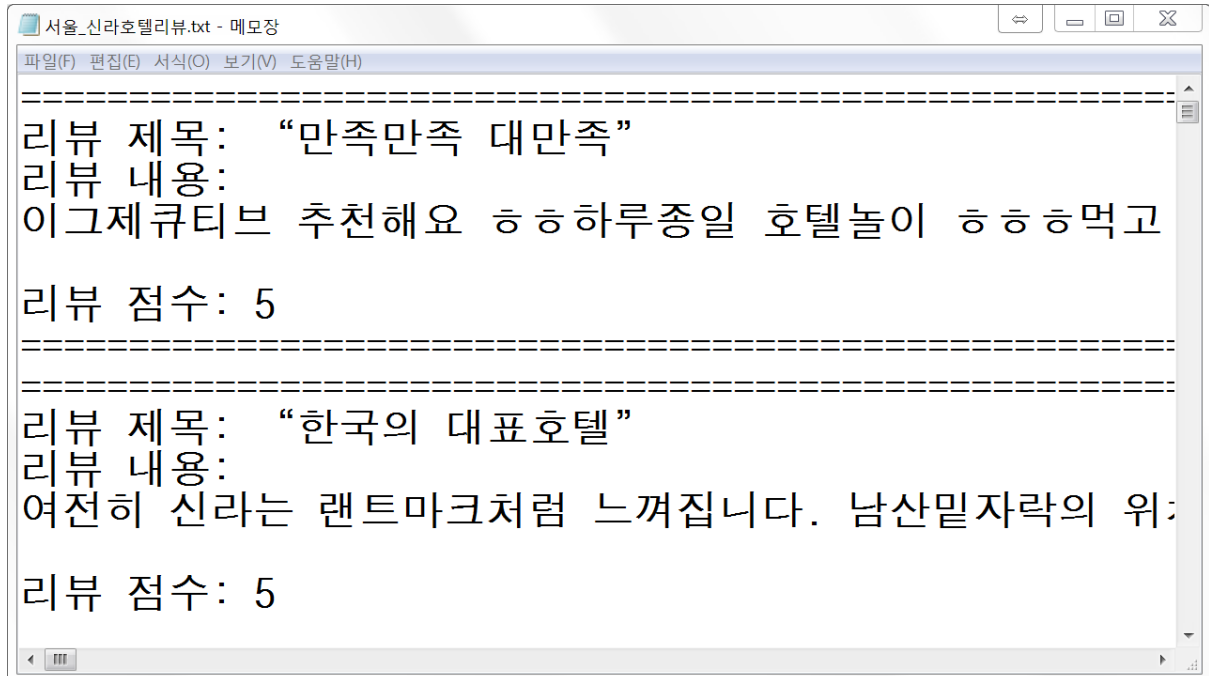
[서울 신라호텔 이용 후기 분석 화면]



[서울 워커힐 호텔 이용 후기 분석 화면]



[원본 파일 일부]



[소스 코드 - KoNLP 패키지로 신라호텔 이용후기 분석하기]

```
> setwd("c:\\Wa_temp")
> install.packages("KoNLP")
> install.packages("wordcloud")
> install.packages("stringr")
```

```
> library(KoNLP)
> library(wordcloud)
> library(stringr)
> useSejongDic()
```

아래 과정이 리뷰들을 R로 불러오는 과정입니다.

```
> data1 <- readLines("서울_신라호텔리뷰.txt")
> data1 <- gsub(" ","-",data1)
> data1 <- str_split(data1,"-")
> data1 <- str_replace_all(unlist(data1),"^[[:alpha:][:blank:]]","")
```

#아래 과정이 불러온 리뷰 문장을 단어로 분리하는 과정입니다.

```
> data2 <- sapply(data1,extractNoun,USE.NAMES=F)
> data2
> head(unlist(data2), 30)
> data3 <- unlist(data2)
```

아래 과정이 필요 없는 단어들이나 기호를 제거하는 과정입니다.

```
> data3 <- Filter(function(x) {nchar(x) <= 10} ,data3)
> head(unlist(data3), 30)
> data3 <- gsub("WW"," ",data3)
> data3 <- gsub(" ","",data3)
> data3 <- gsub("WW"," ",data3)
> data3 <- gsub("제목"," ",data3)
> data3 <- gsub("리뷰"," ",data3)
> data3 <- gsub("내용"," ",data3)
> data3 <- gsub("ㅋㅋㅋㅋ"," ",data3)
> data3 <- gsub("이용"," ",data3)
> data3 <- gsub("호텔"," ",data3)
> data3 <- gsub("신라호텔"," ",data3)
> data3 <- gsub("점수"," ",data3)
```

```

> write(unlist(data3),"신라호텔_2.txt")
> data4 <- read.table("신라호텔_2.txt")
> nrow(data4)
> wordcount <- table(data4)
> wordcount
> wordcount <- Filter(function(x) {nchar(x) >= 2} ,wordcount)
> head(sort(wordcount, decreasing=T),50)

```

```

> txt <- readLines("호텔gsub.txt")
> cnt_txt <- length(txt)
> cnt_txt
> for( i in 1:cnt_txt) {
+   data3 <- gsub((txt[i]),"",data3)
+ }
> data3 <- Filter(function(x) {nchar(x) >= 2} ,data3)
> write(unlist(data3),"신라호텔_3.txt")
> data4 <- read.table("신라호텔_3.txt")
> nrow(data4)

```

#아래 과정이 필터링이 완료된 단어들을 언급 빈도수로 집계하는 과정입니다.

```

> wordcount <- table(data4)
> wordcount
> head(sort(wordcount, decreasing=T),100)

> library(RColorBrewer)
> palette <- brewer.pal(9,"Set3")
> wordcloud(names(wordcount),freq=wordcount,scale=c(5,1),rot.per=0.25,min.freq=4,
+ random.order=F,random.color=T,colors=palette)
> legend(0.3,1 ,"서울 신라호텔 이용 후기 분석 ",cex=0.8,fill=NA,border=NA,bg="white" ,
+ text.col="red",text.font=2,box.col="red")

```


KoNLP 패키지를 활용한 서울 워커히 호텔 이용후기 분석 소스

```
> dev.new( )
> setwd("c:\wwa_temp")
> install.packages("KoNLP")
> install.packages("wordcloud")
> library(KoNLP)
> library(wordcloud)
> useSejongDic()
```

아래 과정이 리뷰들을 R로 불러오는 과정입니다.

```
> data1 <- readLines("서울_워커히호텔리뷰.txt")
> data1
> data1 <- gsub(" ", "-", data1)
> data1 <- str_split(data1, "-")
> data1 <- str_replace_all(unlist(data1), "[^[:alpha:][:blank:]]", "")
```

아래 과정이 불러온 리뷰 문장을 단어로 분리하는 과정입니다.

```
> data2 <- sapply(data1, extractNoun, USE.NAMES=F)
> data2
> head(unlist(data2), 30)
> data3 <- unlist(data2)
```

아래 과정이 필요 없는 단어들이나 기호를 제거하는 과정입니다.

```
> data3 <- Filter(function(x) {nchar(x) <= 10}, data3)
> head(unlist(data3), 30)
> data3 <- gsub("Wwd+", "", data3) ## <--- 모든 숫자 없애기
> data3 <- gsub("WW.", "", data3)
> data3 <- gsub(" ", "", data3)
> data3 <- gsub("WW", "", data3)
> data3 <- gsub("리뷰", "", data3)
> data3 <- gsub("내용", "", data3)
> data3 <- gsub("ㅋㅋㅋㅋ", "", data3)
> data3 <- gsub("서울", "", data3)
> data3 <- gsub("리버뷰", "한강", data3)
> data3 <- gsub("한강뷰", "한강", data3)
> data3 <- gsub("한강전망", "한강", data3)

> write(unlist(data3), "워커히호텔_2.txt")
```

```
> data4 <- read.table("워커힐호텔_2.txt")
> data4
> nrow(data4)
> wordcount <- table(data4)
> wordcount
> wordcount <- Filter(function(x) {nchar(x) >= 2} ,wordcount)
```

```
> txt <- readLines("호텔gsub.txt")
> cnt_txt <- length(txt)
> for( i in 1:cnt_txt) {
+     data3 <- gsub((txt[i]),"",data3)
+ }
```

```
> data3 <- Filter(function(x) {nchar(x) >= 2} ,data3)
> write(unlist(data3),"워커힐호텔_3.txt")
> data4 <- read.table("워커힐호텔_3.txt")
> nrow(data4)
```

#아래 과정이 필터링이 완료된 단어들을 언급 빈도수로 집계하는 과정입니다.

```
> wordcount <- table(data4)
> wordcount
> head(sort(wordcount, decreasing=T),100)

> library(RColorBrewer)
> palette <- brewer.pal(9,"Set1")
> wordcloud(names(wordcount),freq=wordcount,scale=c(5,1),rot.per=0.25,min.freq=5,
+ random.order=F,random.color=T,colors=palette)
> legend(0.3,1 ,"서울 워커힐호텔 이용 후기 분석  ",cex=0.8,fill=NA,border=NA,bg="white" ,
+ text.col="red",text.font=2,box.col="red")
```

이상으로 R 을 활용하여 다양한 텍스트를 분석하는 기본을 살펴 보았습니다.

다음으로 이런 기본을 활용하여 다양한 분석을 하는 기법들을 살펴 보겠습니다.