

텍스트 마이닝(1)

숙명여자대학교 경영학부 오중산

텍스트 마이닝 소개

- 텍스트 마이닝 정의

- ❓ 문자로 된 데이터에서 가치 있는 정보를 얻어 내는 분석 기법

- ❓ 개인 온라인 활동이 확산됨에 따라 중요해진 분석 방법

- 텍스트 마이닝 단계별 구분

- ❓ 단어 빈도 분석 / 형태소 분석기를 이용한 단어 빈도 분석 / 비교 분석

- ❓ 감정 분석 / 의미망 분석 / 토픽 모델링

텍스트 전처리

- 텍스트 전처리란?

- ❓ 텍스트에서 분석하는 데 불필요한 요소 제거

- ❓ 텍스트를 다루기 쉬운 형태로 만드는 과정

- 텍스트 마이닝에서 활용할 자료

- ❓ 문재인 대선 출마 선언문 (text_moon.txt)

```
raw_moon <- readLines("speech_moon.txt", encoding = "UTF-8")  
head(raw_moon)
```

텍스트 전처리

- 불필요한 문자 제거하기

[?] 한글을 제외하고 모두 제거하기

```
txt <- "치킨은!! 맛있다. xyz 정말 맛있다!@#"
```

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

```
library(stringr)
```

```
str_replace_all(string = txt, pattern = "[^가-힣]", replacement = " ")
```

string : 처리할 텍스트

pattern : 규칙 [^가-힣] : 한글이 아닌 모든 문자

replacement : 바꿀 문자

텍스트 전처리

- raw_moon에서 불필요한 문자 제거하기

```
moon <- raw_moon %>%  
  str_replace_all("[^가-힣]", " ")
```

- moon에서 연속된 공백 제거하기

```
moon <- moon %>%  
  str_squish()
```

- 데이터 구조를 tibble로 바꾸기

❓ 문장이 길면 보기 힘들기 때문에 문자열 벡터를 tibble 형태로 변경

```
library(dplyr)  
moon <- as_tibble(moon)
```

텍스트 전처리

- 데이터 프레임과 비교한 tibble의 특성

- ❑ tibble 형태 데이터를 실행하면 console 창에서 행과 열의 개수를 제시

- ❑ 변수의 척도도 보여줌

- ❑ 이후 텍스트를 토큰화하려면, tibble 형태로 저장해야 함

- 지금까지의 전처리 과정을 한줄 코드로 실현하기

```
moon <- raw_moon %>%  
  str_replace_all("[^가-힣]", " ") %>% # 한글만 남기기  
  str_squish() %>% # 연속된 공백 제거  
  as_tibble() # tibble로 변환
```



토큰화하기

- 토큰(token)이란?

- ❓ 텍스트를 나눈 다양한 단위: 문장, 구절, 단어, 형태소 등

- ❓ 토큰화란 텍스트를 토큰 형태로 만드는 것

- unnest_tokens 함수를 이용한 토큰화한 tibble 데이터 형성

- ❓ tidytext 패키지에 있는 unnest_tokens 활용

- ❓ tidytext 패키지는 dplyr, ggplot2와 함께 사용됨

- ❓ input: tibble 형태 데이터에 있는 토큰화 대상 변수

- ❓ output: 출력 변수명

- ❓ token: 토큰 형태(sentences, words, characters)

```
word_space <- moon %>%  
  unnest_tokens(input = value,  
                output = word,  
                token = "words")
```

↳ 단어중심 토큰화

단어 빈도 분석하기

- 단어 빈도 분석이란?

[?] 어떤 단어가 얼마나 쓰였는지 분석함으로써 글쓴이의 의도를 간접적이거나 확인할 수 있음

- dplyr 패키지에 있는 count 함수 사용

[?] tibble 형태의 word_space에 n이라는 빈도수 관련 변수 생성

[?] n변수는 빈도수가 높은 순서대로 내림차순으로 정렬

↳ 제일 위에게 빈도수 높은 것

```
word_space <- word_space %>%  
  count(word, sort = T)
```


단어 빈도 분석하기

- 두 글자 이상으로 된 단어만 남기기

ex) 조사. 말

[?] word_space의 word 변수에서 한 글자 단어는 의미 파악이 어려움

[?] 따라서 두 글자 이상으로 구성된 단어만 남길 필요가 있음

[?] str_count는 글자수를 세는 함수

```
word_space <- word_space %>%  
  filter(str_count(word) > 1)
```

한계: 협동의.협동하여 처럼 같은 뜻 다르게 빈도 파악

↳ word 변수에 저장된 단어

- 이상의 작업을 한 줄로 코딩하기

```
word_space <- word_space %>%  
  count(word, sort = T) %>%  
  filter(str_count(word) > 1)
```

단어 빈도 분석하기

- 빈도수 상위 20위 데이터 프레임 만들기

[?] word_space는 빈도수 내림차순으로 정렬되어 있으므로, head 함수를 이용함

```
top20 <- word_space %>%  
  head(20)
```

- 막대 그래프 그리기

[?] geom_text: 막대 그래프에 빈도수 표시

[?] labs & theme 그래프 제목 및 서식

```
ggplot(top20, aes(reorder(word, 13 n), 13 fill = word)) + geom_bar(stat =  
  "identity") + ↑geom_text(aes(label = n), ↑hjust = -0.3) + labs(title =  
  "문재인 출마 연설문 단어 빈도") + theme(title = element_text(size = 12))  
  ↑  
  ↑내림차순 13축  
  ↑약간 왼쪽에서 표시  
  ↑서식
```

워드 클라우드 만들기

- 워드 클라우드는?

- ☐ 단어 빈도를 구름 모양으로 표현한 그래프

- ☐ 빈도에 따라 글자 크기와 색을 다르게 표현

- ☐ 어떤 단어가 얼마나 많이 사용됐는지 한눈에 파악

- ggwordcloud 패키지에 있는 geom_text_wordcloud 함수 사용

- ☐ geom_text_wordcloud는 난수(random number)를 사용하므로 seed 명령문 필요함

```
ggplot(word_space, aes(label = word, size = n)) +  
  geom_text_wordcloud(seed = 1234) +  
  scale_radius(limits = c(3, NA),      # 최소, 최대 단어 빈도  
               range = c(3, 30))      # 최소, 최대 글자 크기
```

워드 클라우드 만들기

- 워드 클라우드 가다듬기

[?] 참고할 사이트 - <https://lepenec.github.io/ggwordcloud>

```
ggplot(word_space,
      aes(label = word,
           size = n,
           col = n)) +                                # 빈도에 따라 색깔 표현
  geom_text_wordcloud(seed = 1234) +
  scale_radius(limits = c(3, NA),
               range = c(3, 30)) +
  scale_color_gradient(low = "#66aaf2",              # 최소 빈도 색깔
                       high = "#004EA1") +          # 최고 빈도 색깔
  theme_minimal()                                     # 배경 없는 테마 적용
```

워드 클라우드 만들기

- 글자체 바꾸기

[?] showtext 패키지 설치

[?] 관련 사이트(<https://fonts.google.com>)에서 필요한 글자체 확인

```
install.packages("showtext")  
library(showtext)  
  
font_add_google(name = "Nanum Gothic", family = "nanumgothic")  
showtext_auto()
```

[?] Rstudio 실행할 때마다 글자체 설정해 주어야 함

ggplot 그래프 글자체 바꾸기

- Wordcloud 글자체 바꾸기

```
ggplot(word_space, aes(label = word, size = n, col = n)) +  
geom_text_wordcloud(seed = 1234, family = "nanumgothic") + scale_radius  
(limits = c(3, NA), range = c(3, 30)) + scale_color_gradient(low =  
"#66aaf2", high = "#004EA1") + theme_minimal()
```

- ggplot 그래프 글자체 바꾸기

```
ggplot(top20, aes(reorder(word, -n), n, fill = word)) + geom_bar(stat =  
"identity") + geom_text(aes(label = n), hjust = -0.3) + labs(title =  
"문재인 출마 연설문 단어 빈도") + theme(title = element_text(size = 12),  
text = element_text(family = "nanumgothic"))
```

텍스트 마이닝(2)

숙명여자대학교 경영학부 오중산

형태소 분석의 필요성

- 기존 ‘단어’ 중심 토큰화의 문제점

- ❓ 띄어쓰기를 기준으로 하다 보니 의미 없는 단어가 포함됨

- ❓ 예: ‘있습니다’, ‘합니다’ 등...

- 형태소 분석(Morphological Analysis)이란?

- ❓ 형태소(morpheme)는 의미를 지니고 있는 가장 작은 말의 단위로 , 더 나누면 의미가 없어짐

- ❓ 문장에서 형태소를 추출해 명사, 동사, 형용사 등 품사로 분류하는 작업

- ❓ 문장 내용 파악을 위해 ‘명사’가 중요함

형태소 분석 준비

- 형태소 분석을 위한 KoNLP 패키지 설치

[?] 자바와 rJAVA 패키지 설치

[?] `install.packages("multilinguer")`

[?] `library(multilinguer)`

[?] `install_jdk()`

[?] KoNLP 의존성 패키지 설치

[?] `install.packages(c("stringr", "hash", "tau", "Sejong", "RSQLite", "devtools"), type = "binary")`

형태소 분석 준비

- 형태소 분석을 위한 KoNLP 패키지 설치

[?] KoNLP 패키지 설치

[?] `install.packages("remotes")`

예러 [[?] `remotes::install_github("haven-jeon/KoNLP", upgrade = "never", INSTALL_opts=c("--no-multiarch"))`
[?] `library(KoNLP)`

[?] 형태소 사전 설치하기

[?] `useNIADic()`

[?] KoNLP 설치 후에 한 번만 설치하면 됨

형태소 분석 준비

- KoNLP 패키지 설치가 안 될 경우

[?] 방법1

[?] [참고할 사이트 클릭!](#)

[?] 방법2

[?] JAVA 설치(Windows 온라인): <https://www.java.com/ko/download/manual.jsp>

[?] Rtools 4.0 설치: <https://cran.r-project.org/bin/windows/Rtools/>

[?] 방법3

[?] 다음 그림과 같은 메시지가 나오면 cli 패키지를 3.1.0으로 업데이트(설치)

```
** byte-compile and prepare package for lazy loading
loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck = vI[[j]])에서 다
음과 같은 에러가 발생했습니다:
  네임스페이스 'cli' 3.0.1는 이미 로드되었으나 >= 3.1.0가 필요합니다
```

명사를 기준으로 토큰화 실행

- unnest_tokens 함수 예시

```
[?] library(tidytext)
```

```
[?] text <- tibble(value = c("대한민국은  
민주공화국이다.", "대한민국의 주권은  
국민에게 있고, 모든 권력은 국민으로부터  
나온다."))
```

```
[?] text %>% unnest_tokens(input =  
value, output = word, token =  
extractNoun)
```

↳ 명사 기준 토큰화

```
# A tibble: 7 x 1
```

```
word  
<chr>  
1 대한민국  
2 민주공화국  
3 대한민국  
4 주권  
5 국민  
6 권력  
7 국민
```

띄어쓰기 기준 추출

```
text %>%  
  unnest_tokens(input = value,  
                 output = word,  
                 token = "words")
```

```
## # A tibble: 10 x 1
```

```
##   word  
##   <chr>  
## 1 대한민국은  
## 2 민주공화국이다  
## 3 대한민국의  
## 4 주권은  
## 5 국민에게  
## 6 있고  
## 7 모든  
## 8 권력은  
## 9 국민으로부터  
## 10 나온다
```

명사를 기준으로 토큰화 실행

- `unnest_tokens` 실행

[?] 앞서 만들어 놓은 moon 데이터 프레임에서 명사
기준으로 토큰화

```
[?] word_noun <- moon %>%
```

```
  unnest_tokens(input = value, output = word,
```

```
    token =  extractNoun   
           
```

[?] 출력결과를 보면, 아직 KoNLP가 완전하지
않음을 알 수 있음

```
# A tibble: 1,757 x 1  
  word  
  <chr>  
1 "정권교체"  
2 "하겠습니"  
3 "정치"  
4 "교체"  
5 "하겠습니"  
6 "시대"  
7 "교체"  
8 "하겠습니"  
9 ""  
10 "불비불명"  
# ... with 1,747 more rows
```

명사 빈도 분석하기

- word_noun에서의 명사 빈도 분석하기

[?] 빈도가 높을수록 해당 단어가 강조되었음

[?] `word_noun <- word_noun %>%`

`count(word, sort = T) %>%`

↳ 내림차순 정렬

`filter(str_count(word) > 1)`

↳ 문자의 개수

[?] 글자수가 2 이상인 명사에 대해 내림차순으로 빈도수 정리

A tibble: 704 x 2

	word	n
	<chr>	<int>
1	국민	21
2	일자리	21
3	나라	19
4	우리	17
5	경제	15
6	사회	14
7	성장	13
8	대통령	12
9	정치	12
10	하계	12

... with 694 more rows

막대 그래프 만들기

- 상위 20개 단어에 대한 막대 그래프 그리기

[?] 글자체 선정

[?] library(showtext)

[?] font_add_google(name = "Black Han Sans", family = "BHS")

[?] showtext_auto()

↳ 우리가 정한 이름

[?] 막대 그래프 그리기

```
[?] ggplot(top20, aes(reorder(word, -n), n, fill = word)) + geom_bar(stat = "identity") +  
  geom_text(aes(label = n), hjust = -0.3) + labs(title = "문재인 출마 연설문 명사 빈도 ") + theme(title =  
  element_text(size = 12), text = element_text(family = "BHS"))
```

단어 ↗

워드 클라우드 만들기

- word_noun에 대한 워드 클라우드 만들기

```
[?] library(ggwordcloud)
```

```
[?] ggplot(word_noun, aes(label = word, size = n, col = n)) + geom_text_wordcloud(seed =  
1234, family = "BHS") + scale_radius(limits = c(2, NA), range = c(3, 15)) +  
scale_color_gradient(low = "darkgreen", high = "darkred") + theme_minimal()
```

빈도에 따른 크기
배경 없음
최소 크기 3
최대 크기 15

빈도 < 최소 (2 ~ 무한대) < 최대

특정 단어가 사용된 문장 살펴보기

- 문장 기준으로 토큰화

↳ 두 칸 띄백 중 한 칸 제거

↳ tibble 데이터로 만들기

```
[?] sentences_moon <- raw_moon %>% str_squish() %>% as_tibble() %>%  
  unnest_tokens(input = value, output = sentence, token = "sentences")
```

[?] 마침표를 기준으로 문장이 구분되므로, 특수문자 제거하지 않음
↳ . 기준으로

str_replace_all 가 - 될 X

1. 전처리 과정에서 특수문자 제거 X

2. sentences 기준으로 토큰화

sentence
1 정권교체 하겠습니다!
2 정치교체 하겠습니다!
3 시대교체 하겠습니다!
4 '불박불명(不韋不穢)'이라는 고사가 있습니다.
5 남북 연락 나뉘기까지 앉아, 3년 동안 날지도 울지도 않는 새.
6 그러나 그 새는 한번 날면 하늘 끝까지 날고, 한번 울면 천지...
7 그 동안 정치와 거리를 뒀 왔습니다.
8 그러나 암울한 시대가 저를 정치로 불러냈습니다.
9 더 이상 남북 나뉘기까지 아무를 수 없었습니다.
10 이제 저는 국민과 함께 높이 날고 크게 울겠습니다.
11 오늘 저는 제18대 대통령선거 출마를 국민 앞에 엄숙히 선언...
12 '우리나라 대통령이 되겠습니다.
13 존경하는 국민 여러분!
14 저는 대통령이 되겠습니다.
15 우리나라 대통령이 되겠습니다.
16 소수 특권층의 나라가 아니라 보통사람들이 주인인 '우리나...

↳ . 기준으로

특정 단어가 사용된 문장 살펴보기

- 빈도수가 가장 많은 단어가 포함된 문장 확인하기

3.

[?] 빈도수가 가장 많은 단어(‘국민’과 ‘일자리’)를 포함한 문장을 `str_detect` 함수로 찾기

```
[?] sentences_moon %>% filter(str_detect(sentence, "국민"))
```

```
[?] sentences_moon %>% filter(str_detect(sentence, "일자리"))
```

```
[?] 왼쪽 정렬로 모든 내용 출력하려면 %>% print.data.frame(right = F)를 붙임
```

↳ 좌측정렬

텍스트 마이닝(3)

숙명여자대학교 경영학부 오중산

두 텍스트에서 어떤 단어 많이 쓰였냐!

단어 빈도 비교하기

- 비교 분석이란?

- [?] 여러 개의 텍스트를 분석하고 비교하여 공통점과 차이점을 확인

- 비교 분석을 위한 준비

- [?] 두 개 텍스트(speech_moon.txt와 speech_park.txt)를 불러오기

- [?] 텍스트를 **tibble** 형태로 변경하고, 식별을 위한 새로운 변수 (president) 만들기

원래는 문자열 벡터

↳ 이게 문꺼인지, 박꺼인지

- [?] 두 tibble 데이터를 합치기

단어 빈도 비교하기

- 비교 분석을 위한 준비에 필요한 소스코드

```
[?] raw_moon <- readLines("speech_moon.txt", encoding = "UTF-8")
```

```
[?] 문자열moon <- raw_moon %>% as_tibble() %>% mutate(새변수president = "moon")
```

```
[?] raw_park <- readLines("speech_park.txt", encoding = "UTF-8")
```

```
[?] park <- raw_park %>% as_tibble() %>% mutate(president = "park")
```

```
[?] bind_speeches <- bind_rows(moon, park) %>% relocate(president, .before = value)  
통합 tibble data
```

단어 빈도 비교하기

- 데이터 전처리

[?] value 변수에 있는 특수문자 제거 및 연속된 공백에 대한 삭제

[?] bind_speeches는 문자열 벡터가 아니라 tibble

[?] 따라서 mutate 함수 안에 str_replace_all 함수와 str_squish 함수를 사용해야 함

[?] 관련 소스코드

원래는 문자열에 함 → 이번엔 tibble에 적용

[?] library(stringr)

↳ value 변수 업데이트

[?] speeches <- bind_speeches %>% mutate(value = str_replace_all(value, "[^가-힣]", " "),
value = str_squish(value))

president	value	⇒ value에 전처리
moon	—	
moon	—	
:	—	
pork	—	

단어 빈도 비교하기

- ^{KoNLP}명사 기준 토큰화

[?] 관련 소스코드

[?] `library(tidytext) / library(KoNLP)`

[?] `speeches <- speeches %>% unnest_tokens(input = value, output = word, token = extractNoun)` ^{새 변수명}

[?] 데이터 구조의 변화

[?] $213 \times 2(\text{president \& value})$ 구조에서 $2,997 \times 2(\text{president \& word})$ 구조로 변화

↓
문장 단위

↓
명사 단위

단어 빈도 비교하기

- 하위 집단별 단어 빈도 구하기

[?] count 함수를 이용한 하위 집단별 단어 빈도 구하기

case
[?] count(var1, var2): 사례를 var1 측정결과에 따라
소집단으로 구분하고, / 소집단별로 var2 측정결과
빈도를 구하라!

[?] frequency <- speeches %>% count(president,
word) %>% filter(str_count(word) > 1)

[?] 대통령별로 두 글자 이상으로 구성된 단어 빈도를
가나다 순서로 정렬

a < $\begin{matrix} F \\ M \end{matrix}$
b < $\begin{matrix} F \\ M \end{matrix}$

a반.b반에 따라 분리



df

```
## # A tibble: 6 x 2
##   class sex
##   <chr> <chr>
## 1 a     female
## 2 a     male
## 3 a     female
## 4 b     male
## 5 b     male
## 6 b     female
```

① moon (국민 0개, 알라리 0, 경제 0)
park

df %>% count(class, sex)

```
## # A tibble: 4 x 3
##   class sex      n
##   <chr> <chr> <int>
## 1 a     female    2
## 2 a     male      1
## 3 b     female    1
## 4 b     male      2
```


단어 빈도 비교하기

- 자주 사용된 단어 추출하기

[?] 하위 집단별 단어 빈도에서 n 변수는 내림차순 정렬이 아님

[?] slice_max 함수를 활용하여 빈도(n) 높은 상위 10개 사례 추출

[?] top10 <- frequency %>% group_by(president) %>% slice_max(n, n = 10) %>% print(n = Inf)

→ 개수

[?] 대통령별로 분류한 후, n을 기준으로 내림차순 정렬한 후, 상위 10개 추출하라는 명령

↳ 변수

= arrange(-n) %>% head(10)

[?] 빈도수 동점인 경우 제한을 가하려면 with_ties = F 파라미터 지정

n=10

m:10

p:12 (9.10.11.12 동률)

```
[?] top10 <- frequency %>% group_by(president) %>% slice_max(n, n = 10, with_ties = F) %>%  
  print(n = Inf)
```

[?] 단점은 가나다/ABC 순서로 앞에 있는 단어에서 잘림

단어 빈도 비교하기

- 빈도수 상위 10개에 대한 막대 그래프 만들기

❓ facet_wrap을 활용한 president 측정값별 막대 그래프 그리기

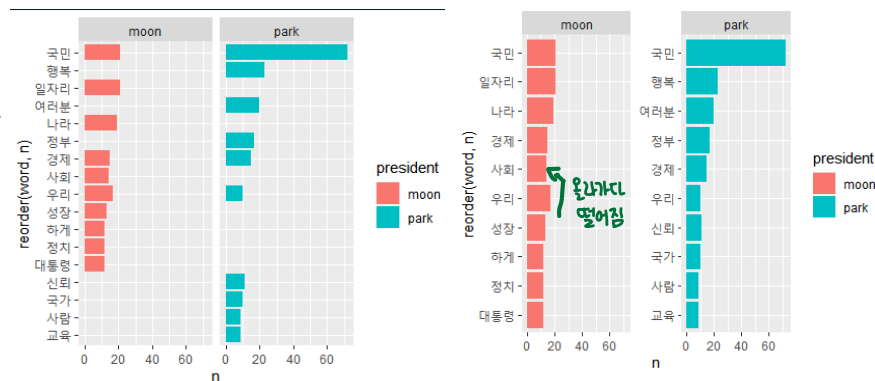
```
❓ ggplot(top10, aes(reorder(word, n), n, fill = president)) + geom_bar(stat = "identity") + coord_flip() +  
  facet_wrap(~ president)
```

전체빈도
↘
문. 박
X-Y 변경
대통령별로
→ 이름차순

❓ X축 설정을 president 측정값에 따라 구분하기

```
❓ ggplot(top10, aes(reorder(word, n), n, fill = president)) + geom_bar(stat = "identity") + coord_flip() +  
  facet_wrap(~ president, scales = "free_y")
```

수직축에 오는 측정 결과값이 상이해도 된다.



단어 빈도 비교하기

- 빈도수 상위 10개에 대한 막대 그래프 만들기

❓ 특정 단어 제외하고 막대 그래프 그리기

❓ park에서 “국민”이 지나치게 많으므로, 이를 제거하여 top10 재구성

```
❓ top10 <- frequency %>% filter(word != "국민") %>% group_by(president) %>% slice_max(n, n = 10,  
with_ties = F) %>% print(n = Inf)
```

```
❓ ggplot(top10, aes(reorder(word, n), n, fill = president)) + geom_bar(stat = "identity") + coord_flip() +  
facet_wrap(~ president, scales = "free_y")
```

단어 빈도 비교하기

- 빈도수 상위 10개에 대한 막대 그래프 만들기

- [?] 빈도수 정렬 구분하여 다시하기

- [?] reorder는 moon/park 구분없이 전체 빈도에 따라 정렬하므로 들쭉날쭉한 모양

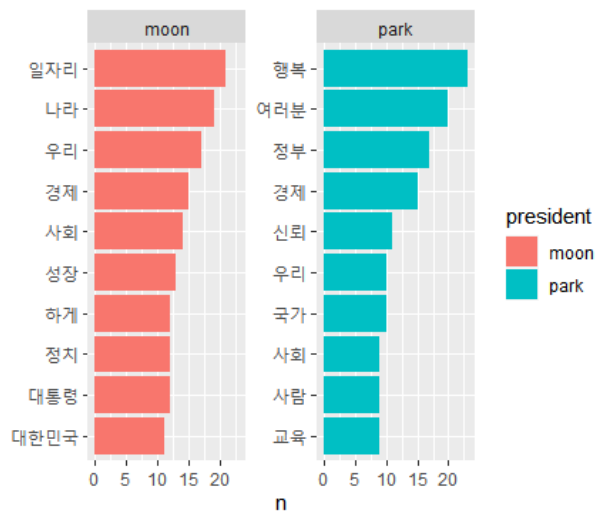
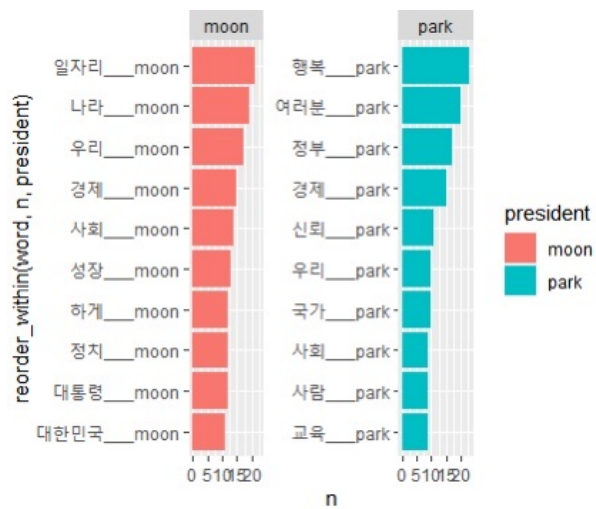
- [?] 이 문제를 해결하기 위해 tidytext 패키지에 있는 `reorder_within` 함수를 사용

- [?] `reorder_within(X축 표시 변수, 정렬 기준 변수, 그래프 구분 기준 변수)`
 - ↳ 박 내에서 내림차순
 - 문 내에서 내림차순

- [?] `ggplot(top10, aes(reorder_within(word, n, president), n, fill = president)) + geom_bar(stat = "identity") + coord_flip() + facet_wrap(~ president, scales = "free_y")`

- [?] X축 항목(word)의 명칭 조정

- [?] `ggplot(top10, aes(reorder_within(word, n, president), n, fill = president)) + geom_bar(stat = "identity") + coord_flip() + facet_wrap(~ president, scales = "free_y") + scale_x_reordered() + labs(x = NULL)`



텍스트 마이닝(4)

숙명여자대학교 경영학부 오중산

단어 빈도 비교 좀 더 정확하게

오즈비: 상대적으로 중요한 단어 비교하기

- 단어 빈도 비교하기의 한계

[?] 어떤 텍스트에서든 많이 사용되는 범용적 단어의 경우 비교의 의미가 없음

어디에나

[?] 예: "우리", "사회", "경제", "일자리" ⇒ 두 현설문 모두 많이 쓰임 ⇒ 범용적 . 특징이 없음

[?] 특정 텍스트에서는 많이 사용되지만, 다른 텍스트에서는 덜 사용되는 ‘상대적 빈도가 높은 단어’가

무엇인지 파악하는 것이 중요함

ex) A B
데이터 많이 적게
 쓰임 쓰임
 ↓
 특색 있는 단어

ex) 지게인해서

A B ⇒ 스쿠버다이빙 (상대적 빈도 높음)

범용적 ⇒ 저는. 봉사. 가족

오즈비: 상대적으로 중요한 단어 비교하기

- Long form 형태 데이터를 wide form으로 바꾸기

세로로 긴

가로로 넓은

[?] Long form 형태 데이터의 한계

↳ moon/park

[?] 같은 단어가 범주별로 다른 행을 구성하여 빈도 비교가 어렵고,

연산하기도 불편함

[?] `df_long <- frequency %>% group_by(president) %>%`

`slice_max(n, n = 10) %>% filter(word %in% c("국민", "우리", "`

`정치", "행복"))` ↳ 값이 큰 것 순서대로 정렬

```
# A tibble: 6 x 3
# Groups:   president [2]
  president word      n
  <chr>      <chr> <int>
1 moon      국민      21
2 moon      우리      17
3 moon      정치      12
4 park      국민      72
5 park      행복      23
6 park      우리      10
```

↳ moon 우리

park 우리 비교 어려움

오즈비: 상대적으로 중요한 단어 비교하기

- Long form 형태 데이터를 wide form으로 바꾸기

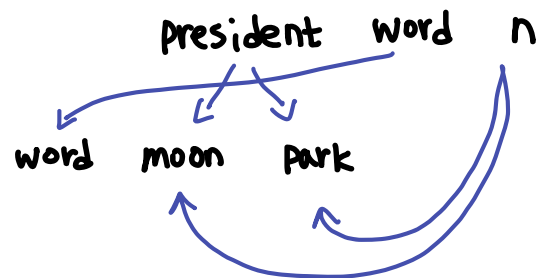
↗ 기준점 놓고 돌린다

[?] tidyR 패키지의 pivot_wider 함수를 이용하기

[?] names_from: 변수명을 가져올 변수

[?] values_from: 변수에 채워 넣을 값이 있는 변수

[?] `df_wide <- df_long %>% pivot_wider(names_from = president, values_from = n, values_fill = list(n = 0))`



moon/park

측정값

```
# A tibble: 4 x 3
  word    moon  park
<chr> <int> <int>
1 국민      21    72
2 우리      17    10
3 정치      12     NA
4 행복      NA    23
```



```
# A tibble: 4 x 3
  word    moon  park
<chr> <int> <int>
1 국민      21    72
2 우리      17    10
3 정치      12     0
4 행복       0    23
```

오즈비: 상대적으로 중요한 단어 비교하기

- 연설문 단어 빈도 데이터 프레임 (frequency)을

wide form으로 바꾸기

[?] 두 대통령의 연설문 단어 빈도를 저장한 frequency
를 wide form으로 변경

~~*~~ [?] `frequency_wide <- frequency %>%`
`pivot_wider(names_from = president,`
`values_from = n, values_fill = list(n = 0))`

```
# A tibble: 955 x 3
  word      moon park
  <chr>    <int> <int>
1 가동          1     0
2 가사          1     0
3 가슴          2     0
4 가족          1     1
5 가족구조      1     0
6 가지          4     0
7 가치          3     1
8 각종          1     0
9 감당          1     0
10 강력          3     0
# ... with 945 more rows
```

오즈비: 상대적으로 중요한 단어 비교하기

- 오즈비(odds ratio) 구하기 ex) $P(\text{event}_A) = 0.6$ $P(\text{event}_B) = 0.2$
3배 (= odds ratio)

[?] 오즈비란?

[?] 어떤 사건의 A조건에서 발생 확률이 B조건에서 발생할 확률에 비해 얼마나 더 큰지 나타낸 값

[?] 단어가 두 텍스트 중 어디 등장할 확률이 높은지, 즉 단어의 상대적인 중요도를 알 수 있음

[?] 연설문별 단어 비중 구하기 ex) 행복 < $\left(\begin{array}{l} \text{moon: 차지하는 비율 } 0.1 \\ \text{park: " } 0.1 \end{array} \right) \frac{0.1}{0.1} = 10\text{배 더 높음}$
(박)

[?] 연설문별로 '각 단어의 빈도'를 '모든 단어 빈도의 합'으로 나눔 박 전체 단어 빈도 = 2000 행복 = 5 $\Rightarrow \frac{5}{2000}$

[?] 단어 빈도가 0이면 오즈비 구할 때 문제가 발생해서 분모 / 분자에 각각 1을 더함

[?] `frequency_wide <- frequency_wide %>% mutate(ratio_moon = ((moon + 1)/(sum(moon + 1))),`
`ratio_park = ((park + 1)/(sum(park + 1))))`
빈도
...

\Rightarrow 연설문 별로 특정 단어가 차지하는 비중

오즈비: 상대적으로 중요한 단어 비교하기

강정애 대비 장원준

- 오즈비(odds ratio) 구하기

장원준: 강정애

[?] 오즈비 변수 추가하기

[?] 한 텍스트의 단어 비중을 다른 텍스트의 단어 비중으로 나눔

→ park 텍스트의 단어 비중

[?] `frequency_wide <- frequency_wide %>% mutate(odds_ratio = ratio_moon/ratio_park)`

↳ moon 텍스트의 단어 비중

(같은 단어에 대해)

A tibble: 955 x 6

	word <chr>	moon <int>	park <int>	ratio_moon <dbl>	ratio_park <dbl>	odds_ratio <dbl>
1	가동	1	0	0.000873	0.000552	1.58
2	가사	1	0	0.000873	0.000552	1.58
3	가슴	2	0	0.00131	0.000552	2.37
4	가족	1	1	0.000873	0.00110	0.791
5	가족구조	1	0	0.000873	0.000552	1.58
6	가지	4	0	0.00218	0.000552	3.96
7	가치	3	1	0.00175	0.00110	1.58
8	각종	1	0	0.000873	0.000552	1.58
9	감당	1	0	0.000873	0.000552	1.58
10	강력	3	0	0.00175	0.000552	3.17

... with 945 more rows

$\frac{1+1}{955개\ 단어(moon+1)}$
빈도합계

오즈비: 상대적으로 중요한 단어 비교하기

- 오즈비 해석 $\frac{\text{ratio}-m}{\text{ratio}-p} > 1$ $\text{ratio}-m > \text{ratio}-p$

$m = \text{강}$

$p = \text{강}$

[?] 오즈비가 1보다 크거나 작을 때의 의미

[?] 1보다 크면, 박 전 대통령 연설문에 비해 문대통령 연설문에서 해당 단어가 더 많이 사용됨

[?] `frequency_wide %>% arrange(-odds_ratio)`

[?] 1보다 작으면, 문대통령 연설문에 비해 박 전 대통령 연설문에서 해당 단어가 더 많이 사용됨

[?] `frequency_wide %>% arrange(odds_ratio)` \Rightarrow 오름차순 \rightarrow 작은 순서 \rightarrow 그 단어가 박에서 더 많이 차지

[?] 두 연설문에서 비중이 동일하면 오즈비는 1이 됨

[?] `frequency_wide %>% arrange(abs(1 - odds_ratio))`

\hookrightarrow 작은 순서 \hookrightarrow 절대값

0 ~ 높은 거까지

\downarrow

odds ratio 1

오즈비: 상대적으로 중요한 단어 비교하기

- 오즈비가 가장 높은 10개 단어와 가장 낮은 10개

단어 추출하여 top10 만들기

[?] 전자는 문대통령 연설문에서 상대적으로 비중이 더 높음

⇒ 오즈비 가장 높은 10개 단어

[?] 후자는 박 전 대통령 연설문에서 상대적으로 비중이 더

높음 ⇒ 오즈비 가장 낮은 단어

[?] 값이 작은 순서대로 순위를 구하는 rank 함수 이용

1등~10등 ⇒ 박에서 차지 큼

[?] `top10 <- frequency_wide %>% filter(rank(odds_ratio) <= 10)`

`| rank(-odds_ratio) <= 10) %>% arrange(-odds_ratio)`

↳ 큰 순서

⇒ 문에서 차지 큼

ex)

3. 5. 7

1등 2등 3등

A tibble: 20 x 6

	word <chr>	moon <int>	park <int>	ratio_moon <dbl>	ratio_park <dbl>	odds_ratio <dbl>
<문> 1	복지국가	8	0	0.00393	0.000552	7.12
2	세상	6	0	0.00306	0.000552	5.54
3	여성	6	0	0.00306	0.000552	5.54
4	정의	6	0	0.00306	0.000552	5.54
5	강자	5	0	0.00262	0.000552	4.75
6	공평	5	0	0.00262	0.000552	4.75
7	대통령의	5	0	0.00262	0.000552	4.75
8	보통	5	0	0.00262	0.000552	4.75
9	상생	5	0	0.00262	0.000552	4.75
10	지방	5	0	0.00262	0.000552	4.75
<박> 11	과제	0	4	0.000436	0.00276	0.158
12	국정운영	0	4	0.000436	0.00276	0.158
13	시작	0	4	0.000436	0.00276	0.158
14	지식	0	4	0.000436	0.00276	0.158
15	행복	3	23	0.00175	0.0132	0.132
16	실천	0	5	0.000436	0.00331	0.132
17	정보	0	5	0.000436	0.00331	0.132
18	투명	0	5	0.000436	0.00331	0.132
19	여러분	2	20	0.00131	0.0116	0.113
20	박근혜	0	8	0.000436	0.00496	0.0879

오즈비: 상대적으로 중요한 단어 비교하기

- 막대 그래프 그리기

[?] 막대 그래프를 그리기 위한 새로운 변수 (president와 n) 만들기

[?] 해당 단어가 어느 대통령 연설문에서 비롯된 것인지, 그리고 그 빈도가 얼마인지 알기 위해 두 개의 새로운 변수 형성

[?] `top10 <- top10 %>% mutate(president = ifelse(odds_ratio > 1, "moon", "park"), n = ifelse(odds_ratio > 1, moon, park))` (1보다 작은 경우)

↪ 변수명

A tibble: 20 x 8

	word <chr>	moon <int>	park <int>	ratio_moon <dbl>	ratio_park <dbl>	odds_ratio <dbl>	president <chr>	n <int>
1	강자	5	0	0.00262	0.000552	4.75	moon	5
2	공평	5	0	0.00262	0.000552	4.75	moon	5
3	대통령의	5	0	0.00262	0.000552	4.75	moon	5

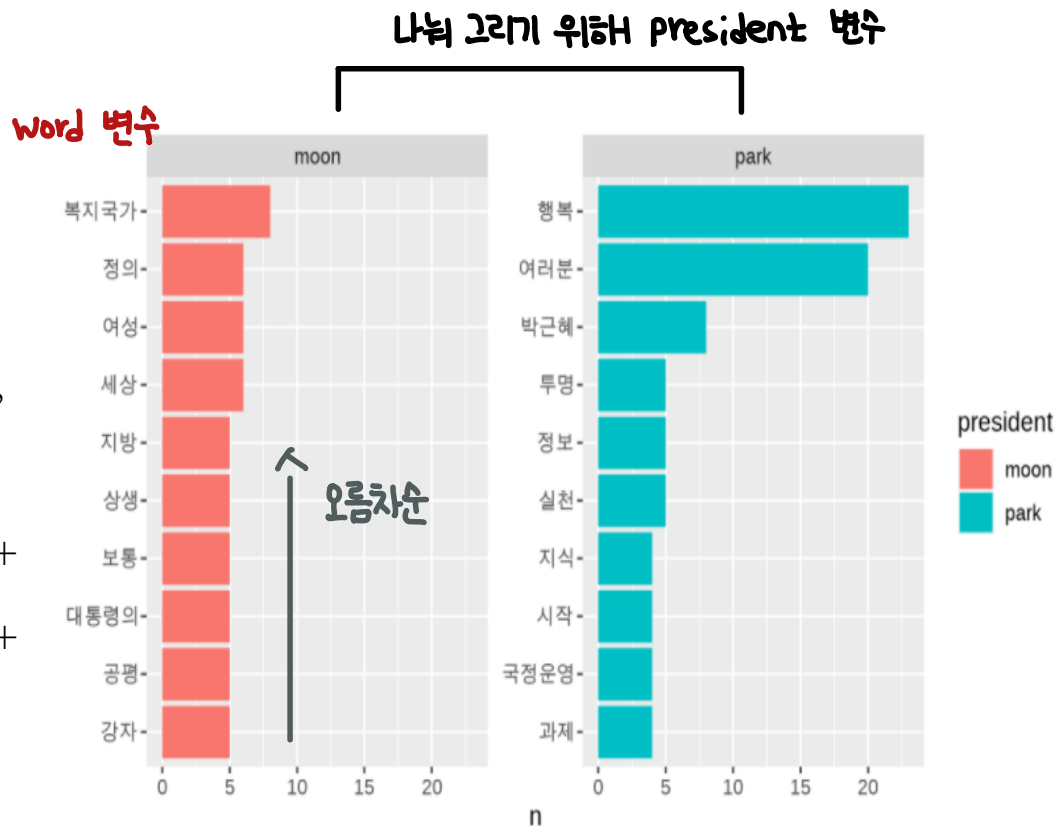
오즈비: 상대적으로 중요한 단어 비교하기

- 막대 그래프 그리기

[?] 두 대통령 연설문에서 상대적으로 많이 사용된 단어 빈도 비교

```
[?] ggplot(top10, aes(x = reorder_within(word, n, president), n, fill = president)) +  
  geom_bar(stat = "identity") + coord_flip() +  
  facet_wrap(~ president, scales = "free_y") +  
  scale_x_reordered() + labs(x = NULL)
```

[?] 주의! scale_x_reordered 함수를 사용하려면 tidytext 패키지를 불러와야 함



오즈비: 상대적으로 중요한 단어 비교하기


within \Rightarrow moon / park 내에서

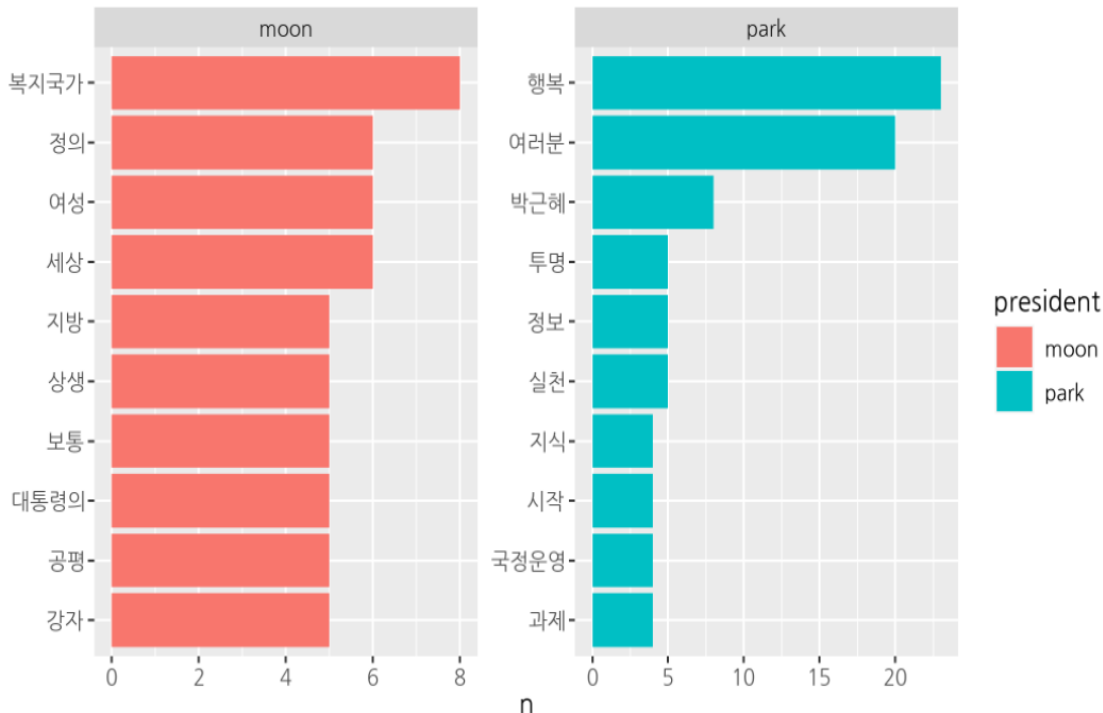
between \Rightarrow "행복" moon 보다 park에 많이 쓰임

- 막대 그래프 그리기

[?] 그래프별로 축 설정하기

```
ggplot(top10, aes(x =  
  reorder_within(word, n, president), n,  
  fill = president)) + geom_bar(stat =  
  "identity") + coord_flip() +  
  facet_wrap(~ president, scales = "free")  
  + scale_x_reordered() + labs(x =  
  NULL)
```

[?]  막대 길이가 같아도 빈도가 다름에 주의해야 함
비슷해도



오즈비: 상대적으로 중요한 단어 비교하기

- 주요 단어가 사용된 문장 살펴보기

[?] 문장 단위로 토큰화하기

```
[?] speeches_sentence <- bind_speeches %>% as_tibble() %>% unnest_tokens(input = value, output =  
sentence, token = "sentences")
```

[?] 주요 단어가 사용된 문장 추출하기

`print.data.frame(right = F)`

```
[?] speeches_sentence %>% filter(president == "moon" & str_detect(sentence, "복지국가"))
```

```
[?] speeches_sentence %>% filter(president == "park" & str_detect(sentence, "행복"))
```

오즈비: 상대적으로 중요한 단어 비교하기

- 중요도가 비슷한 단어 살펴보기

[?] 오즈비가 1에 가까운 보편적인 단어

→ $\frac{\text{빈도수}}{\text{오즈비}}$ \Rightarrow 1과 가까운 수

[?] `frequency_wide %>% arrange(abs(1 - odds_ratio)) %>%`

`head(10)`

$1 - 1 = 0$

$1 - 1.03 = 0.03$

- 중요도가 비슷하고 빈도가 높은 단어 살펴보기

[?] 각 연설문에서 빈도수가 5회 이상이면서 중요도가 비슷한 단어

추출

[?] 두 연설문에서 모두 강조한 단어 확인

[?] `frequency_wide %>% filter(moon >= 5 & park >= 5) %>%`

`arrange(abs(1 - odds_ratio)) %>% head(10)`

A tibble: 10 x 6

word	moon	park	ratio_moon	ratio_park	odds_ratio
<chr>	<int>	<int>	<dbl>	<dbl>	<dbl>
1 사회	14	9	0.00655	0.00552	1.19
2 사람	9	9	0.00436	0.00552	0.791
3 경제	15	15	0.00698	0.00883	0.791
4 지원	5	5	0.00262	0.00331	0.791
5 우리	17	10	0.00786	0.00607	1.29
6 불안	7	8	0.00349	0.00496	0.703
7 산업	9	5	0.00436	0.00331	1.32
8 대한민국	11	6	0.00524	0.00386	1.36
9 국가	7	10	0.00349	0.00607	0.576
10 교육	6	9	0.00306	0.00552	0.554

텍스트 마이닝(5)

숙명여자대학교 경영학부 오중산

로그오즈비(log odds ratio) 로 단어 비교하기

- 로그오즈비의 정의와 이점

❓ 로그오즈비는 오즈비에 자연로그를 취한 값

❓ 상용로그는 밑이 10이지만, 자연로그는 밑이 무리수인 $e(=2.71828182\dots)$

❓ 오즈비와 로그오즈비의 관계

❓ 오즈비 $= 1 \rightarrow$ 로그오즈비 $= 0$: 두 텍스트에서 차지하는 비율 똑같음

❓ 오즈비 $> 1 \rightarrow$ 로그오즈비 > 0

❓ 오즈비 $< 1 \rightarrow$ 로그오즈비 < 0

❓ 로그오즈비 막대그래프는 텍스트에서의 단어 비교하기를 더욱 명확하게 함

로그오즈비(log odds ratio) 로 단어 비교하기

- 로그오즈비 구하기

[?] frequency_wide에서 로그오즈비 구하기

$\log()$ 함수 \Rightarrow 자연로그

[?] frequency_wide <- frequency_wide %>% mutate(log_odds_ratio = $\log(\text{odds_ratio})$)

[?] 로그오즈비값과 텍스트에서의 단어의 비중

[?] 로그오즈비가 클수록 moon에서 비중이 큰 단어이고, 작을수록 park에서 비중이 큰 단어

[?] 로그오즈비가 0이면 양쪽에서 비중이 비슷한 단어

로그오즈비(log odds ratio) 로 단어 비교하기

- 로그오즈비를 이용해서 중요한 단어 비교하기

[?] 두 연설문(moon과 park)에서 로그오즈비 상위 10개 단어 추출하기

↳ 기존에 없던 새로운 변수

```
[?] top10 <- frequency_wide %>% group_by(president = ifelse(log_odds_ratio > 0, "moon", "park"))  
%>% slice_max(abs(log_odds_ratio), n = 10, with_ties = F)
```

↳ 값이 큰 것 순서대로 정렬

✖ mutate가 아닌 group_by를 쓴 것에 주의해야 함! ✖

[?] president라는 새로운 변수를 만들고,

[?] president에 따라 집단을 두 개 (moon / park)로 구분한 후,

[?] 각 집단별로 로그오즈비 절대값이 큰 상위 10개 사례 추출

[?] rank 함수를 이용해서 오즈비 기준 상·하위 10개 사례 추출 결과와 동일함

moon

!

10개

park

!

10개

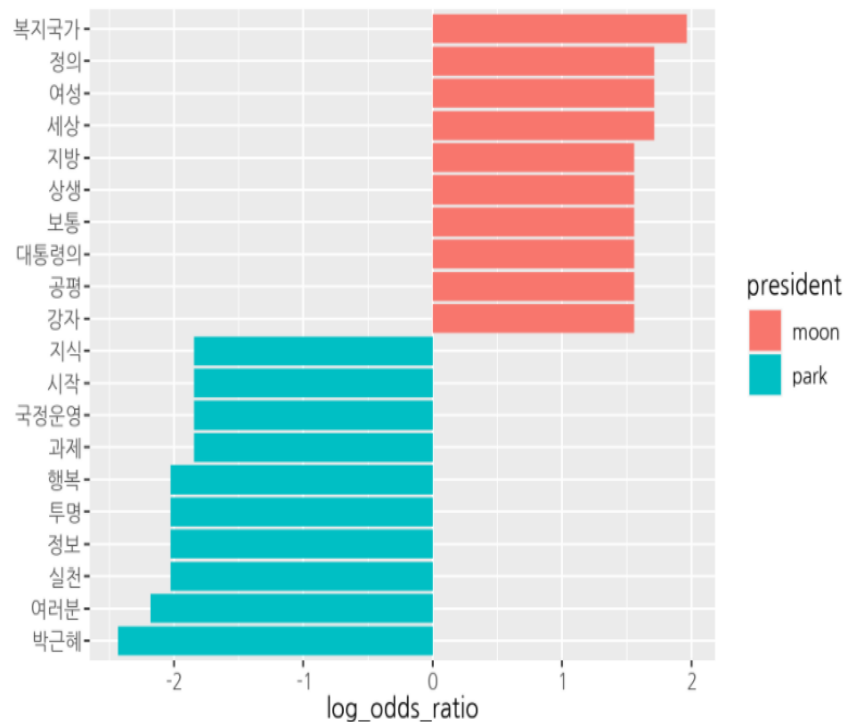
로그오즈비(log odds ratio) 로 단어 비교하기

- 막대 그래프 만들기

[?] 두 연설문에 대해 하나의 그래프로 만들

[?] `ggplot(top10, aes(reorder(word, log_odds_ratio),
log_odds_ratio, fill = president)) + geom_bar(stat =
"identity") + coord_flip() + labs(x = NULL)`

[?] `reorder_within` 함수나 `facet_wrap` 함수 필요 없음!



TF-IDF: 여러 텍스트의 단어 비교하기

- 오즈비나 로그오즈비의 한계

[?] 서로 다른 두 텍스트에서 단어의 상대적 비중 차이를 알 수 있지만 , 세 개 이상 텍스트를 대상으로 비교할 수 없음

- TF-IDF

ex)취미 중 패러글라이딩. 동굴탐험

[?] 중요한 단어는 1)흔하지 않으면서 2)특정 텍스트에서 많이 사용되는 특징이 있음

[?] TF-IDF는 여러 텍스트를 비교하여 특정 텍스트에서 어떤 단어가 자주 사용되었는지 알려주는 지표

[?] 텍스트별로 단어별 TF-IDF 값 계산

	1	2	3
A			
B			
C			

TF-IDF: 여러 텍스트의 단어 비교하기

- TF-IDF

[?] TF(^{단어}term ^{빈도}frequency)란?

[?] 특정 단어가 특정 텍스트에서 사용된 회수 (단어 빈도)

[?] DF(document frequency)란?

[?] 특정 단어가 사용된 텍스트 수(문서 빈도)

[?] IDF(Inverse document frequency)는 DF의 역수에 문서 개수(N)을 곱한 후 자연로그를 취한값

[?] $IDF = \log(N / DF) \frac{N}{DF}$

[?] IDF가 크다는 것은 특정 단어가 특정 텍스트 (들)에서만 사용된다는 것을 의미하며, IDF가 작을 경우 여러 텍스트 (들)에서 흔하게 사용된다는 것을 의미

TF-IDF: 여러 텍스트의 단어 비교하기

ex) 마라톤 10 1 2 1 1

⇒ 5개 문서에 다 쓰여서 IDF 0

$$\frac{5}{5} = 1 \log \Rightarrow 0$$

• TF-IDF

TF-IDF = TF × IDF = $TF \times \log(N / DF)$

흔하지 않으면서 특정 텍스트에서만 많이 나오는 단어일수록 TF-IDF값이 높음

↳ 배우진 않음

TF가 커도 IDF가 0이면 TF-IDF도 0이 되는 문제는 weighted log odds가 대안

ex)

TF				IDF			TF-IDF			
단어	자기소개서 A	자기소개서 B	자기소개서 C	단어	DF	IDF	단어	자기소개서 A	자기소개서 B	자기소개서 C
저는	15	10	10	저는 $N=3$	3	$\log \frac{3}{3} = 0$	저는	$15 \times \log \frac{3}{3} = 0$	$10 \times \log \frac{3}{3} = 0$	$10 \times \log \frac{3}{3} = 0$
스카이다이빙	3	0	0	스카이다이빙	1	$\log \frac{3}{1} = 1.1$	스카이다이빙	$3 \times \log \frac{3}{1} = 3.3$	$0 \times \log \frac{3}{1} = 0$	$0 \times \log \frac{3}{1} = 0$
자기주도적	3	5	3	자기주도적	3	$\log \frac{3}{3} = 0$	자기주도적	$3 \times \log \frac{3}{3} = 0$	$5 \times \log \frac{3}{3} = 0$	$3 \times \log \frac{3}{3} = 0$
데이터	0	5	1	데이터	2	$\log \frac{3}{2} = 0.4$	데이터	$0 \times \log \frac{3}{2} = 0$	$5 \times \log \frac{3}{2} = 2$	$1 \times \log \frac{3}{2} = 0.4$
배낭여행	2	3	5	배낭여행	3	$\log \frac{3}{3} = 0$	배낭여행	$2 \times \log \frac{3}{3} = 0$	$3 \times \log \frac{3}{3} = 0$	$5 \times \log \frac{3}{3} = 0$

딱히 두드러진 단어는
없는데 그나마 데이터

TF-IDF: 여러 텍스트의 단어 비교하기

- TF-IDF 구하기

- ❓ 필요한 텍스트 데이터

- ❓ 역대 대통령(4명)의 출마 선언문 파일 (speeches_presidents.csv)

- ❓ readr 패키지에 있는 read_csv 함수 이용

- ❓ csv 파일을 불러와서 tibble로 바꾸어 주는 동시에, 데이터 불러오는 속도도 빠름

- ❓ `raw_speeches <- read_csv("speeches_presidents.csv")`

- ❓ 전처리, 토큰화(명사 기준), 단어 빈도 구하기

- ❓ `speeches <- raw_speeches %>% mutate(value = str_replace_all(value, "[^가-힣]", " "), value = str_squish(value))`

- ❓ `speeches <- speeches %>% unnest_tokens(input = value, output = word, token = extractNoun)`

- ❓ `frequency_four <- speeches %>% count(president, word) %>% filter(str_count(word) > 1)`

↳ 명사 단위로 토큰화한 변수

TF-IDF: 여러 텍스트의 단어 비교하기

- TF-IDF 구하기

[?] 연설문별로 TF, IDF, and TF-IDF 계산하기

[?] tidytext 패키지에 있는 `bind_tf_idf(term = , document = , n =)` 함수 사용

[?] term은 TF-IDF 대상 단어 변수, document는 텍스트(연설문) 구분 변수, n은 단어 빈도 변수

[?] `frequency_four <- frequency_four %>% bind_tf_idf(term = word, document = president, n = n)`
`%>% arrange(-tf_idf)` ↳ 토큰화된 명사 들어가있음

내림차순 (높은 순서부터) ✖ ✖

[?] 주의!: TF값은 특정 텍스트에서 특정 단어의 비중 (특정 텍스트에서 특정 단어 빈도수 / 특정 텍스트의 전체 단어 빈도수)

[?] 대통령 연설문별로 TF-IDF가 높은 단어 확인 가능

[?] `frequency_four %>% filter(president == "문재인 / 박근혜 / 이명박 / 노무현")`

TF-IDF: 여러 텍스트의 단어 비교하기

- TF-IDF 막대 그래프 그리기

[?] 대통령별로 TF-IDF가 높은 10개 단어 추출하기

```
[?] top10 <- frequency_four %>% group_by(president) %>% slice_max(tf_idf, n = 10, with_ties = F)
```

[?] 그래프 순서 정하기

```
[?] top10$president <- factor(top10$president, levels = c("문재인", "박근혜", "이명박", "노무현"))
```

[?] 막대 그래프 그리기

```
[?] ggplot(top10, aes(reorder_within(word, tf_idf, president), tf_idf, fill = president)) + geom_bar(stat =  
  "identity") + coord_flip() + facet_wrap(~ president, scales = "free") + scale_x_reordered() + labs(x = NULL)
```

텍스트 마이닝(6)

숙명여자대학교 경영학부 오중산

감정 사전 활용하기

- 감정분석(sentiment analysis)이란?

- ❓ 텍스트에 어떤 감정이 담겨 있는지 분석하는 방법

- ❓ 긍정적인 감정인가, 아니면 부정적인 감정인가?

- 감정 사전이란?

- ❓ 감정 단어와 감정의 강도를 표현한 숫자로 구성된 사전

- ❓ 사전을 이용해서 문장의 단어에 감정 점수를 부여한 후에 합산

감정 사전 활용하기

- KNU 한국어 감성 사전

[?] 국립 군산대학교 소프트웨어융합공학과에서 개발한 감정 사전

[?] `dic <- read_csv("knu_sentiment_lexicon.csv")`

[?] word: 감정 단어

[?] 단어(한 단어로 구성), 복합어(두 개 이상 단어로 구성), 이모티콘

[?] 총 14,854개 단어로 구성됨

[?] polarity: 감정의 강도

[?] +2~-2의 5개 정수로 구성되며, 긍정단어는 +로 부정단어는 -로 중성단어는 0으로 표시

감정 사전 활용하기

- KNU 한국어 감성 사전 살펴보기

[?] 이모티콘 살펴보기

[?] `library(stringr) / dic %>% filter(!str_detect(word, "[가-힣]")) %>% arrange(word)`

[?] 14,854개 단어 분류

[?] `dic %>% mutate(sentiment = ifelse(polarity >= 1, "pos", ifelse(polarity <= -1, "neg", "neu"))) %>% count(sentiment)`

```
# A tibble: 3 x 2
  sentiment      n
  <chr>      <int>
1 neg       9829
2 neu        154
3 pos       4871
```

감정 사전 활용하기

- 문장의 감정 점수 구하기

[?] STEP1: 단어 기준으로 토큰화하기

[?] `df <- tibble(sentence = c("디자인 예쁘고 마감도 좋아서 만족스럽다.", "디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다."))`

[?] `library(tidytext) / df <- df %>% unnest_tokens(input = sentence, output = word, token = "words", drop = F)`

[?] `drop = F`는 원문(sentence 변수)을 제거하지 않는다는 의미

```
# A tibble: 12 x 2
  sentence                                word
  <chr>                                <chr>
1 디자인 예쁘고 마감도 좋아서 만족스럽다. 디자인
2 디자인 예쁘고 마감도 좋아서 만족스럽다. 예쁘고
3 디자인 예쁘고 마감도 좋아서 만족스럽다. 마감도
4 디자인 예쁘고 마감도 좋아서 만족스럽다. 좋아서
5 디자인 예쁘고 마감도 좋아서 만족스럽다. 만족스럽다
6 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 디자인은
7 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 괜찮다
8 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 그런데
9 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 마감이
10 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 나쁘고
11 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 가격도
12 디자인은 괜찮다. 그런데 마감이 나쁘고 가격도 비싸다. 비싸다
```

감정 사전 활용하기

- 문장의 감정 점수 구하기

[?] STEP2: 단어에 감정 점수 부여하기

[?] word 기준으로 left_join을 이용하여 감정 사전 결합

[?] 만약단어가 감정 사전에 없으면 NA가 되는데, 이때는 0으로 변경

[?] `df <- left_join(df, dic, by = "word") %>% mutate(polarity = ifelse(is.na(polarity), 0, polarity))`

[?] STEP3: 문장별로 감정 점수 합산하기

[?] 합산 점수가 양수면, 문장에서 긍정적인 단어가 상대적으로 많이 사용되었음을 의미함

[?] `score_df <- df %>% group_by(sentence) %>% summarise(score = sum(polarity))`

댓글 감정 분석하기

- 기본적인 전처리

- [?] 기사 댓글 소개

- [?] “news_comment_parasite.csv”에는 영화 기생충의 아카데미상 수상 관련 댓글을 담고 있음

- [?] raw_news_comment <- read_csv("news_comment_parasite.csv")

- [?] 고유 번호 변수(id) 만들기와 html 특수 문자 제거하기

- [?] install.packages("textclean") / library(textclean)

- [?] 웹에서 만든 텍스트의 html 특수문자 제거를 위한 replace_html 함수는 textclean 패키지에 있음

- [?] news_comment <- raw_news_comment %>% mutate(id = row_number(), reply = str_squish(replace_html(reply)))

댓글 감정 분석하기

- 단어 기준 토큰화 및 감정 점수 부여하기

[?] 토큰화

```
[?] word_comment <- news_comment %>% unnest_tokens(input = reply, output = word, token =  
  "words", drop = F)
```

[?] 감정 점수 부여하기

```
[?] word_comment <- left_join(word_comment, dic, by = "word") %>% mutate(polarity =  
  ifelse(is.na(polarity), 0, polarity))
```

댓글 감정 분석하기

- 감정 분류 및 단어 빈도별 막대 그래프 그리기

[?] 감정 분류

```
[?] word_comment <- word_comment %>% mutate(sentiment = ifelse(polarity == 2, "pos",  
  ifelse(polarity == -2, "neg", "neu")))
```

[?] 문장에 대해 감정 분류 기준값을 ± 2 로 강화함

[?] 빈도수 상위 10개 단어 데이터 프레임 만들기

```
[?] top10_sentiment <- word_comment %>% filter(sentiment != "neu") %>% count(sentiment, word)  
%>% group_by(sentiment) %>% slice_max(n, n = 10)
```

[?] count 함수에 의해 n(빈도수) 변수를 만들고, 감정에 따른 문장 유형에 따라 상위 10개 단어 추출

[?] 빈도수 동률인 단어를 모두 포함

댓글 감정 분석하기

- 감정 분류 및 단어 빈도별 막대 그래프 그리기

❓ 빈도수 상위 10개 단어에 대한 막대 그래프 그리기

```
ggplot(top10_sentiment, aes(reorder_within(word, n, sentiment), n, fill = sentiment)) +  
  geom_bar(stat = "identity") + coord_flip() + facet_wrap(~sentiment, scales = "free") +  
  scale_x_reordered() + labs(x = NULL) + geom_text(aes(label = n))
```

❓ geom_text 함수를 통해 그래프에 빈도수를 수치로 표기

댓글 감정 분석하기

- 댓글별 감정 점수 구하고 내용 살펴보기

[?] 댓글별 감정 점수 구하기

```
[?] score_comment <- word_comment %>% group_by(id, reply) %>% summarise(score =  
  sum(polarity)) %>% ungroup()
```

[?] word_comment에서 댓글(reply)이 동일하더라도 id가 다르면 서로 다른 댓글로 취급하기 위해 id와 reply로 집단을 구분함

[?] ungroup 함수를 이용하여 그룹을 해제함

[?] 감정 점수 상/하위 10개 댓글 확인하기

```
[?] score_comment %>% arrange(-score) %>% head(10)
```

```
[?] score_comment %>% arrange(score) %>% head(10)
```

댓글 감정 분석하기

- 댓글별 감정 점수 구하고 내용 살펴보기

[?] 감정 점수별 빈도 구하기

```
[?] score_comment %>% count(score)
```

[?] 감정 점수에 따른 감정 분류 및 유형별 빈도수 확인하기

```
[?] score_comment <- score_comment %>% mutate(sentiment = ifelse(score >= 1, "pos", ifelse(score  
  <= -1, "neg", "neu")))
```

```
[?] score_comment %>% count(sentiment)
```