텍스트 마이닝(3)

숙명여자대학교 경영학부 오중산

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
  - 두 tibble 데이터를 합치기

단어 빈도 비교하기

- 비교 분석을 위한 준비에 필요한 소스코드
  - raw_moon <- readLines("speech_moon.txt", encoding = "UTF-8")
  - moon <- raw_moon %>% as_tibble() %>% mutate(president = "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)

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

단어 빈도 비교하기

- 명사 기준 토큰화
  - 관련 소스코드
    - library(tidytext) / library(KoNLP)
    - speeches <- speeches %>% unnest_tokens(input = value, output = word, token = extractNoun)
  - 데이터 구조의 변화
    - 213 × 2(president & value) 구조에서 2,997 × 2(president & word) 구조로 변화

단어 빈도 비교하기

- 하위 집단별 단어 빈도 구하기

◆ count 함수를 이용한 하위 집단별 단어 빈도 구하기

❖ 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 한 글자 이상으로 구성된 단어 빈도를 가나다 순서로 정렬

```
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
```

```
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)
    - ❖ 대통령별로 분류한 후, n을 기준으로 내림차순 정렬한 후, 상위 10개 추출하라는 명령
  - ◆ 빈도수 동점인 경우 제한을 가하려면 with_ties = F 파라미터 지정
    - ❖ 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)
  - ◆ 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”)

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)
    - ❖ 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)