08

비정형데이터분석

텍스트 데이터의 전처리(2)

통계·데이터과학과 장영재 교수

학습목차

1. 대소문자 변환과 문장부호 삭제
2. 어간추출(stemming)과 원형복원(lemmatization)
3. 불용어(stopwords) 삭제
4. 실제 텍스트 데이터의 전처리

01

대소문자 변환과 문장부호 삭제

1. 대소문자 변환과 문장부호 삭제 - ① 대소문자 변환

- 고유명사의 일부로 사용된 경우 대문자, 보통명사인 경우에는 소문자로 쓰이는 점에 유의할 필요가 있음
  (ex: "Boston University" vs "universities in Boston")
  - 대소문자를 바르게 변환하기 위해서는 많은 경우 고유명사와 일반명사를 잘 구분해야 함
    - → 문장의 첫 글자로 쓰인 대문자들은 소문자로 변환 : tolower( )
    - → 문자열 내의 소문자를 대문자로 변환 : toupper( )

1. 대소문자 변환과 문장부호 삭제 - ① 대소문자 변환

```
> x <- c("Kim was a 12-year-old boy.", "It's 11 O'clock.",
        "Everybody loved Kim.")

> tolower(x)
[1] "kim was a 12-year-old boy." "it's 11 o'clock."
[3] "everybody loved kim."
```

- 문장의 첫 단어나 제목에 고유명사가 쓰이는 경우도 있고, 문장의 일부 단어를 강조하기 위해 일반명사도 대문자로 표현하는 경우가 있음에 유의
  - → "Boston University", "New York" 등과 같이 대문자로 쓰인 많은 고유명사의 경우 n-그램 사전을 잘 작성함으로써 대소문자 변환의 어려움을 해결

1. 대소문자 변환과 문장부호 삭제 - ② 문장부호의 삭제

- 문장부호는 대부분의 경우 특별한 의미를 가지고 있지 않으므로 텍스트 데이터 분석의 효율성을 높이기 위해 전처리 과정에서 삭제
  - 일부 연구 결과에 따르면 문장부호 중 일부는 상당히 중요한 의미를 지니며 문서의 전반적인 구조에 대한 단서를 제공하므로 무조건적인 삭제가 좋지 않다는 주장도 제기(Manning and Schütze, 1999)
    - → "Wash."는 "Wahsington" 주를 나타내는 약자로 마침표를 삭제하고, 대문자를 소문자로 변환하면 "씻다"라는 의미를 가진 "wash"와 구분하기 어려워짐
    - → "'s"의 경우 소유격을 표현하는 경우도 있고 "is"가 축약된 경우도 있어 의미를 처리하기 까다로움

1. 대소문자 변환과 문장부호 삭제 - ② 문장부호의 삭제

- R을 이용하여 문장부호를 삭제하기 위해서는 먼저 정규표현식 (regular expression)을 이해할 필요

문자클래스	의미	문자클래스	의미
[:alpha:]	알파벳	[:digit:]	숫자
[:lower:]	알파벳 중 소문자	[:alnum:]	영문자와 숫자
[:upper:]	알파벳 중 대문자	[:punct:]	문장부호
[:blank:]	공백, 탭 등	[:space:]	공백, 탭, 줄바꾸기 등

1. 대소문자 변환과 문장부호 삭제 - ② 문장부호의 삭제

→ 정규식을 이용하여 문장부호를 제거

```
> x <- c("Kim was a 12-year-old boy!", "It's 11 O'clock.",
        "Everybody loved Kim.")
> gsub(x, pattern = "([[:punct:]])", replacement = "")
[1] "Kim was a 12yearold boy" "Its 11 Oclock"
[3] "Everybody loved Kim"
```

→ 과도하게 삭제된 경우 정규식을 수정해 줄 필요가 있는데, 정규표현식에 쐐기표(caret) "^"를 사용하면 해당 패턴을 삭제 대상에서 제외하라는 의미

```
gsub(x, pattern = "([^[:alnum:][:blank:]'-])", replacement = "") # [ ] 내 문자, 숫자, 어퍼스트로피,
                                                                 # 공백, 줄표는 삭제하지 않음
[1] "Kim was a 12-year-old boy" "It's 11 O'clock"
[3] "Everybody loved Kim"
```

1. 대소문자 변환과 문장부호 삭제 - ② 문장부호의 삭제

- 정규식을 이용하여 첫 글자로 사용된 대문자를 소문자로 변환
  - pattern = ([[:upper:]])를 지정하고 해당 문자를 소문자로 변환

  → perl을 참(perl = TRUE)으로 지정하고 Perl(프로그래밍 언어) 방식의 정규 표현식을 사용하여 replace = "₩₩L₩₩1"로 지정

  - ₩₩L은 소문자를, ₩₩1은 첫 문자 하나를 의미

1. 대소문자 변환과 문장부호 삭제 - ② 문장부호의 삭제

```
>x <-  gsub(x, pattern = "([[:upper:]])", replacement = “₩₩L₩₩1", perl = T)
> x <- gsub(x, pattern = "([^[:alnum:][:blank:]'-])", replacement = "")
> strsplit(x, " ")
[[1]]
[1] "kim"        "was"        "a"          "12-year-old"
[5] "boy"
[[2]]
[1] "it's"    "11"      "o'clock"
[[3]]
[1] "everybody" "loved"     "kim"
```

02

어간추출(stemming)과 원형복원(lemmatization)

2. 어간추출(stemming)과 원형복원(lemmatization) - ① 어간추출 (stemming)

- 어간추출은 여러 단어의 공통된 문자열을 추출하는 것을 목표로 하지만 단수형과 복수형, 현재형과 과거형, 과거분사 등에서 불규칙 변화
  ex)"produce", "producer", "produced", "producing" → "produc-" "take"와 그 과거형 "took", 과거분사 "taken" → ?

  - 미리 정해둔 사전을 이용하고 부분적으로는 규칙성을 이용하여 어간추출하는 방식이 많이 사용(Weiss et al., 2010)

2. 어간추출(stemming)과 원형복원(lemmatization) - ① 어간추출 (stemming)

- 어간추출을 위해서는 복잡하고 많은 명령문들을 사용하며 정규표현식이 많이 사용됨

- 토큰 길이가 3글자 이하이면 해당 토큰을 그대로 반환
- 숫자로 표현된 토큰은 그대로 반환
- 토큰이 s로 끝나면 s를 삭제한 단어가 사전에 있는지 확인하고 사전에 있는 경우 s를 삭제한 토큰 반환
- 토큰이 es로 끝나면 es를 삭제한 단어가 사전에 있는지 확인하고 사전에 있는 경우 es를 삭제한 토큰 반환
- 토큰이 ed로 끝나면 ed를 삭제한 단어가 사전에 있는지 확인하고 사전에 있는 경우 ed를 삭제한 토큰 반환
- 토큰의 길이가 6 이상이고 ing로 끝나면 ing를 삭제한 토큰 반환
- 토큰의 길이가 5 이하이고 ing로 끝나면 토큰을 그대로 반환

<표> 어간추출 알고리즘

2. 어간추출(stemming)과 원형복원(lemmatization) - ① 어간추출 (stemming)

- 영어문장의 어간추출을 위해 흔히 사용되는 어간추출기(stemmer)는 포터의 어간추출기(Porter stemmer)와 랭커스터 대학 어간추출기 (Lancaster stemmer)

원문장	The	Williams	sisters	are	leaving	this	tennis	centre
포터	the	Williams	sister	are	leav	thi	tennis	centr
랭커스터	the	Williams	sist	ar	leav	thi	ten	cent

<표> 어간추출기의 결과 비교(Eisenstein, 2019)

2. 어간추출(stemming)과 원형복원(lemmatization) - ① 어간추출 (stemming)

→ 포터는 대체로 원 단어를 보존하는 방식으로 어간추출을 하는 반면 랭커스터 방식은 더 많은 문자를 삭제

```
> x <- c("Kim was a 12-year-old boy!", "It's 11 O'clock.",
        "Everybody loved Kim.")
> gsub(x, pattern = "([[:punct:]])", replacement = "")
[1] "Kim was a 12yearold boy" "Its 11 Oclock"
[3] "Everybody loved Kim"
```

→ textstem 패키지의 stem_strings( ) 함수는 snowballC 패키지의 wordStem( ) 함수를 호출하여 사용하고 wordStem( ) 함수는 기본적으로 포터 어간추출기를 활용

2. 어간추출(stemming)과 원형복원(lemmatization)- ② 원형복원 (lemmatization)

- 원형복원이란 변형된 단어들의 공통된 부분만 찾아내어 불완전한 측면이 있는 어간추출과 달리 단어의 변형되기 전 원래 형태, 즉 원형(lemma)으로 단어를 복원하는 방식
  - 불규칙 과거형이나 비교급 단어가 불규칙적으로 변할 경우 적절한 어간추출이 어려움

→ 원형으로 표현함으로써 의미상의 공통성을 표현할 수 있음

ex) "run"과 "ran" → "run" / "good"과 "better"와 "best" → "good"

"produced", "producing" → "produce"

"productive", "productivity" → "product"

2. 어간추출(stemming)과 원형복원(lemmatization)- ② 원형복원 (lemmatization)

→ 워드넷(WordNet)의 원형복원기(lemmatizer)를 사용하는 경우 앞서 살펴본 포터와 랭커스트 어간추출기에 비해 원래의 형태가 더 잘 남아있음

원문장	The	Williams	sisters	are	leaving	this	tennis	centre
포터	the	Williams	sister	are	leav	thi	tennis	centr
랭커스터	the	Williams	sist	ar	leav	thi	ten	cent
워드넷	The	Williams	sister	are	leaving	this	tennis	centre

<표> 어간추출기와 원형복원 결과 비교(Eisenstein, 2019)

2. 어간추출(stemming)과 원형복원(lemmatization)- ② 원형복원 (lemmatization)

- R의 textstem 패키지에 탑재된 lemmatize_strings( ) 함수를 실행하여 원형복원

→ "are"이 원형 "be"로, "leaving"이 원형 "leave"로 나타난다는 점이 <표>와 차이점

```
> lemmatize_strings(txts)
[1] "The Williams sister be leave this tennis centre."
```

03

불용어(stopwords) 삭제

3. 불용어(stopwords) 삭제 - ① 기능어 (function words)

- 문장에서 관사(the)와 전치사(in, of, for), 접속사(and) 등이 자주 등장하는데, 이처럼 문법적인 기능과 의미를 나타내는 단어들을 기능어(function word)라 함
  - 동사, 명사, 형용사, 부사 등 주된 의미를 표현하는 단어를 내용어 (Content word)라 함

3. 불용어(stopwords) 삭제 - ②지프의 법칙 (Zipf's law)

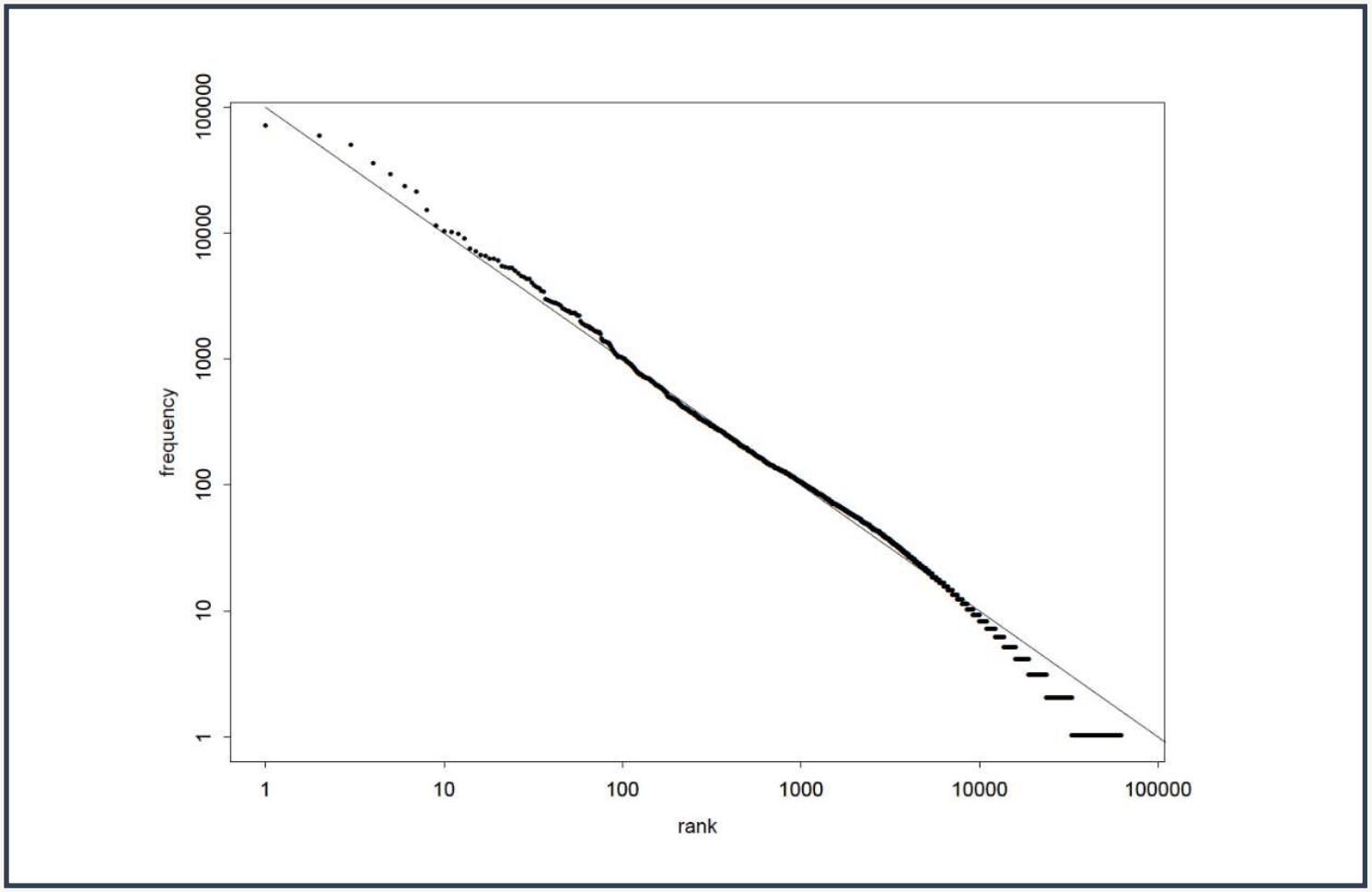
- 지프의 법칙은 언어학자 지프에 의해 발견된 경험적 법칙으로 텍스트 데이터 내의 단어의 출현빈도를 설명하는 데 유용하게 사용

  - 경험적으로 각 단어의 출현빈도와 출현빈도순위 사이에 반비례 관계가 성립한다는 법칙

    $freq \times rank = k$ ($freq$=출현빈도, $rank$:출현빈도 순위, k:상수)

  → 브라운 대학에서 일반적인 문서들을 모아 만든 말뭉치(corpus)인 브라운 코퍼스(Brown corpus)에 포함된 토큰들의 순위와 출현빈도를 살펴보면 지프의 법칙이 성립 (k=100,000)

3. 불용어(stopwords) 삭제 - ②지프의 법칙 (Zipf's law)



3. 불용어(stopwords) 삭제 - ②지프의 법칙 (Zipf's law)

- 단어주머니 가설에서는 단어의 출현빈도가 중요한 변수이므로 텍스트 데이터의 의미 파악에 도움이 되지 않으면서 출현빈도가 높은 단어들은 제거해주는 것이 필요함
  - 출현빈도는 높지만 텍스트 데이터 분석에 도움이 되지 않는 단어들을 불용어(stopwords)라 하며 이를 삭제하는 과정이 불용어 삭제

3. 불용어(stopwords) 삭제 - ③ R을 이용한 불용어 삭제

- R의 stopwords 패키지에는 자주 등장하는 불용어들이 저장되어 있음
  - 불용어 목록에는 대명사, 관사, 조동사 등의 기능어들이 다수 포함되어 있음

→ 차집합을 구하는 setdiff( ) 함수로서 불용어 제거에 활용됨

3. 불용어(stopwords) 삭제 - ③ R을 이용한 불용어 삭제

```
> install.packages("stopwords")
> library(stopwords)
> txt <- c("He decided to quit his job.", "I do not deny it.", "Can you hear me?")
> txt <- tolower(txt)
> txt <- gsub(txt, pattern = "([^[:alnum:][:blank:]'-])", replacement = "")
> txt <- strsplit(txt, " ")
> lapply(txt, setdiff, y=stopwords())
[[1]]
[1] "decided" "quit"    "job"
[[2]]
[1] "deny"
[[3]]
[1] "can"  "hear"
```

3. 불용어(stopwords) 삭제 - ③ R을 이용한 불용어 삭제

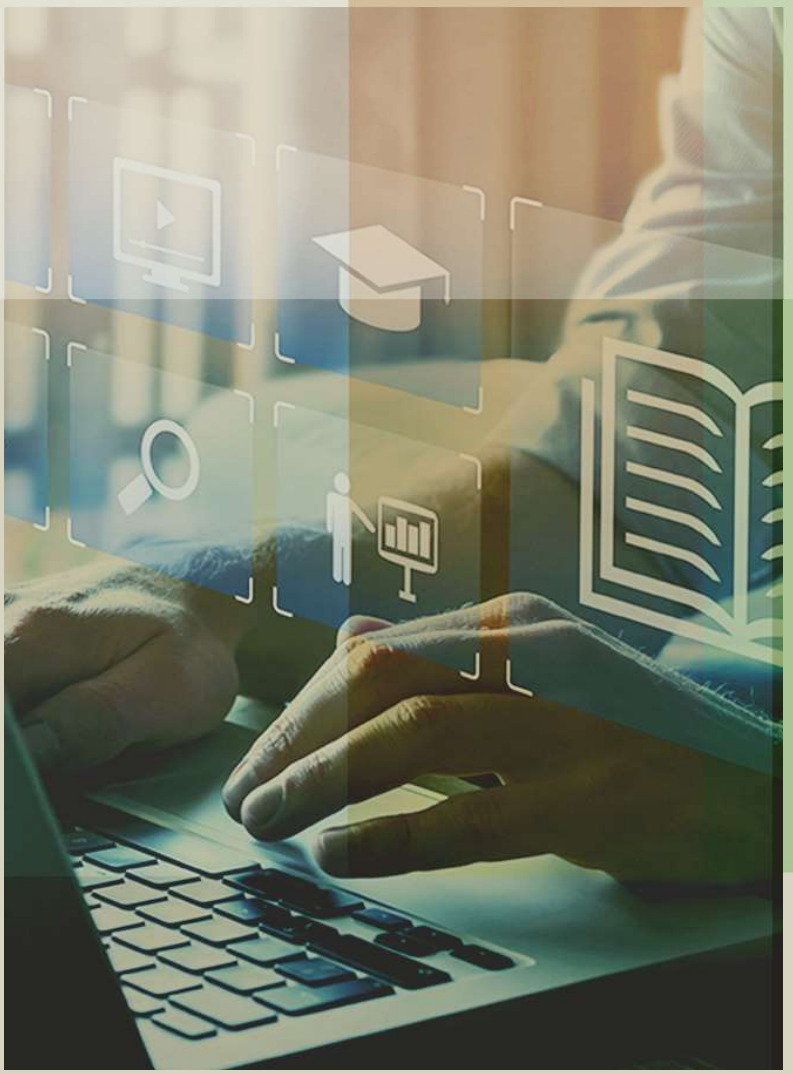
- R의 stopwords 패키지에는 자주 등장하는 불용어들이 저장되어 있음
  - 불용어 목록에 일부 불완전해 보이는 부분도 있음
  - → "can"은 불용어 목록에 포함되어 있지 않아 출력되며 두 번째 문장에서 부정어 "not"이 제거
  - → 불용어 목록을 일부 수정하여 사용할 수 있음

3. 불용어(stopwords) 삭제 - ③ R을 이용한 불용어 삭제

```
> newstop <- c(stopwords(), "can") # 제거목록에 "can"추가
> newstop <- setdiff(newstop, c("no", "not"))
            # 새로운 제거목록 newstop에서 "no", "not" 제외
> lapply(txt, setdiff, y=newstop)
[[1]]
[1] "decided" "quit"    "job"
[[2]]
[1] "not"  "deny"        # not 제거되지 않음
[[3]]
[1] "hear"              # can 제거됨
```

04

실제 텍스트 데이터의 전처리



4. 실제 텍스트 데이터의 전처리

- 구텐베르크 프로젝트(Project Gutenberg)는 지적재산권이 적용되지 않는 다양한 문학 작품들을 전자문서로 만들어 홈페이지 (http://www.gutenberg.org)를 통해 제공
  - 「로빈슨 크루소」를 전처리하여 각 단어들의 도수분포표를 만들기

4. 실제 텍스트 데이터의 전처리

```
> RC <- scan("http://www.gutenberg.org/files/521/521-0.txt", what = "character",  encoding = "UTF-8", sep = "₩n")
> RC_Chpt <- grep(RC, pattern="CHAPTER") #각 장 시작 위치
> RC_End <- grep(RC, pattern="END OF THE PROJECT GUTENBERG EBOOK")-1  # 본문이 끝나는 위치 정보 추출
> RC_body <- RC[(RC_Chpt[1]):RC_End]
> RC_all <- paste(RC_body, collapse = " ")# 각 행들 연결
> RC_all <- gsub(RC_all, pattern = "'s", replacement = "")
> RC_all <- gsub(RC_all, pattern = "([^[:alnum:][:blank:]'-])", replacement = "")
#부호제거
> RC_all <- tolower(RC_all) # 소문자로 변환
>RC_all <- unlist(strsplit(RC_all, " ")) #리스트→벡터
> library(textstem)
> RC_all <- lemmatize_strings(RC_all) # 원형복원
> RC_all_table <- sort(table(RC_all), decreasing = T)
> RC_all_proptable <- sort(prop.table(table(RC_all)), decreasing=T)  # 도수분포표 및 상대도수분포표
```

실습하기

다음시간안내

09

텍스트 데이터에 대한 탐색적 자료분석