Data Prediction Model and Machine Learning

Online course #12

Text mining & analysis

Preview

■ 텍스트 데이터도 모델링이 가능한가?

- 신동엽 시인과 안도현 시인의 아래 시를 가지고 모델링하면 새로운 시 “껍데기는 가라 사월도 알맹이만 남고 껍데기는 가라 " 가 어느 시인의 것인지 예측할 수 있나?

샘플1: “우리들의 어렸을 적 황토 벗은 고갯마을 할머니 등에 업혀 누님과 난, 곧잘 파랑새 노랠 배웠다.” _신동엽

샘플2: “누가 하늘을 보았다 하는가 누가 구름 한 자락 없이 맑은 하늘을 보았다 하는가” _신동엽

샘플3: “너에게 묻는다 연탄재 함부로 발로 차지 마라 너는 누구에게 한번이라도 뜨거운 사람이었느냐” _안도현

샘플4: “눈 내리는 만경들 건너가네 해진 짚신에 상투 하나 떠가네 가는 길 그리운 이 아무도 없네” _안도현

Preview

■ 텍스트 모델링이 가능하다면 유용한 응용이 많음

- 영화 관람평을 모델링하면 흥행 예측 가능
- 상품에 대한 댓글을 분석하여 마케팅 전략 세움
- 트윗을 분석하여 대선이나 총선 결과 예측
- 주식 관련 댓글을 보고 주가 예측

■ 텍스트는 소통의 원천적 수단

- 스마트폰으로 문자 전송
- 트위터, 페이스북에 텍스트 전송

■ 텍스트는 지금까지 다룬 데이터와 확연히 다른 성질을 가짐

01.1 텍스트 데이터의 성질

■ 텍스트 데이터는 다음과 같은 독특한 성질을 가짐

- 비정형 데이터다. 문서마다 길이가 천차만별이며, 문서에 나타난 단어의 종류도 제각각이다. 문장 중간에 나타나는 숫자와 특수 기호, 외국어의 종류와 위치가 다양하다.
- 잡음이 많은 데이터다. '하다', '그리고', '위해' 등과 같은 불용어가 많으며, 구두점이 자주 나타난다. 그리고 어미가 심하게 변형되어 나타난다. 예를 들어, '뛰다'의 어미는 '뛰니, 뛴, 뜀' 등으로 변형되어 문서에 나타난다.
- 애매성이 많다. 예를 들어, 'time flies like an arrow'를 '시간은 화살처럼 빠르다'로 해석할 수도 있고 '시간 파리는 화살을 좋아한다'로 해석할 수도 있다.
- 텍스트 분석에는 구문론syntax과 의미론semantic이 있다. 문법만 따지는 구문론 수준에서는 위의 영어 문장을 파악하는 데 혼란이 있지만, 의미론에서는 '시간은 화살처럼 빠르다'로 해석할 수 있다. 의미론은 단어의 의미를 파악하여 문서를 분석해야 하므로 훨씬 어렵다.
- 언어가 다양하다. 영어에서는 'I'가 목적어가 되면 'me'이지만, 한국어에서는 '나는'이 '나를'이 되는 것처럼 조사를 이용해 목적어가 된다.

01.1 텍스트 데이터의 성질

■ 위키피디아의 “data science” 문서 예제

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured,[1][2] similar to data mining.

Data science is a “concept to unify statistics, data analysis, machine learning and their related methods” in order to “understand and analyze actual phenomena” with data.[3] It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science.

Turing award winner Jim Gray imagined data science as a “fourth paradigm” of science (empirical, theoretical, computational and now data-driven) and asserted that “everything about science is changing because of the impact of information technology” and the data deluge.[4][5]

In 2012, when Harvard Business Review called it “The Sexiest Job of the 21st Century”,[6] the term “data science” became a buzzword. It is now often used interchangeably with earlier concepts like business analytics,[7] business intelligence, predictive modeling, and statistics. Even the suggestion that data science is sexy was paraphrasing Hans Rosling, featured in a 2011 BBC documentary with the quote, "Statistics is now the sexiest subject around."[8] Nate Silver referred to data science as a sexed up term for statistics.[9] In many cases, earlier approaches and solutions are now simply rebranded as “data science” to be more attractive, which can cause the term to become “dilute[d] beyond usefulness.”[10] While many university programs now offer a data science degree, there exists no consensus on a definition or suitable curriculum contents.[7] To its discredit, however, many data-science and big-data projects fail to deliver useful results, often as a result of poor management and utilization of resources.[11][12][13][14]

텍스트 문서의 예(위키피디아에서 ‘data science’ 검색)

이런 텍스트 데이터를
어떻게 처리할까?

01.2 텍스트 데이터의 처리 파이프라인

■ 텍스트 마이닝

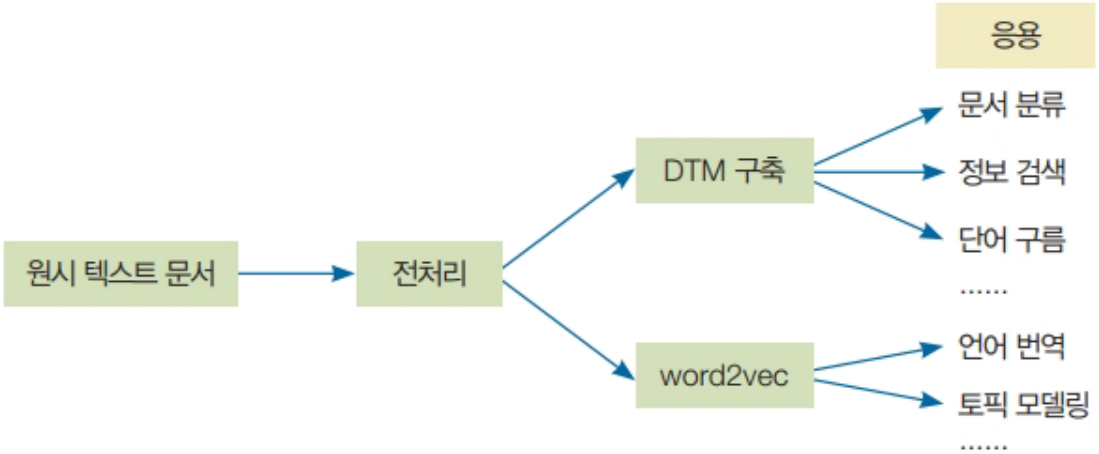
- 텍스트 데이터에서 유용한 정보 또는 지식을 찾아내는 일

■ 용어

- 문서$_{document}$
  - 예) 위키 설명문, 뉴스에서 뉴스 꼭지 하나하나, 트윗에서 트윗 하나, 댓글에서 댓글 하나, 신동엽 시인의 시 하나하나
- 말뭉치$_{corpus}$
  - 특정 분야에서 발생하는 문서의 집합
  - 예) 특정 연도에 치러지는 대선 관련 기사, 사회학자가 모은 한 달간 트윗 문서 전체, 국문학자가 모은 신동엽 시인의 시 전체

01.2 텍스트 데이터의 처리 파이프라인

■ 텍스트 처리 파이프라인

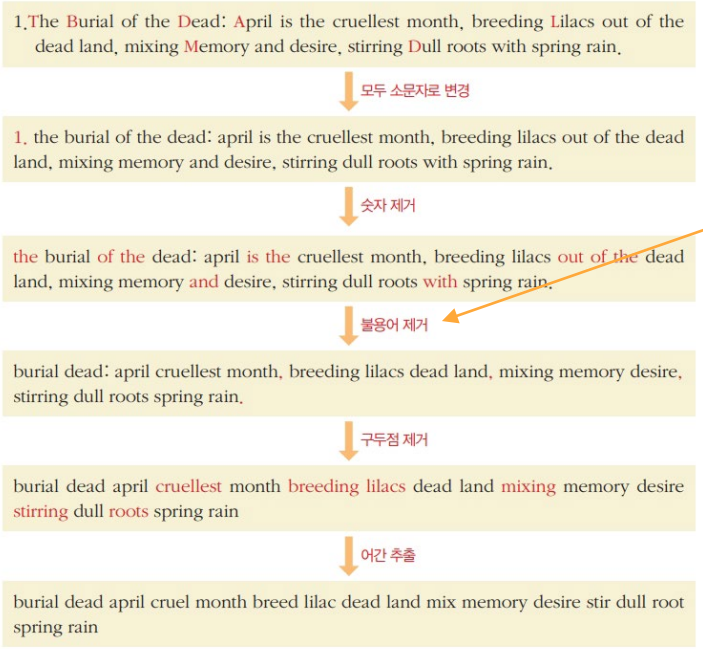


텍스트 처리 파이프라인

01.2 텍스트 데이터의 처리 파이프라인

■ 전처리 과정

- 전처리를 마치면 어느 정도 정보 손실이 있으나 다음 단계 처리에 적합한 형태가 됨



텍스트 데이터의 전처리 예

불용어$_{\text{stop word}}$란 검색 색인 단어로 의미가 없는 단어
예) a, the, and, 그리고, 또는, 및

02 DTM 구축

■ 텍스트 데이터는,

- 비정형 데이터라 그 상태로는 시각화 함수를 적용할 수 없고 모델링할 수도 없음
- 문서를 이들 함수에 적용하려면 일정한 크기의 벡터로 변환해야 함
- DTM은 문서를 벡터로 변환하는 기술

NOTE 문서 단어 행렬, 즉 DWMDocument Word Matrix이라 부르는 대신 DTMDocument Term Matrix이라 부르는 이유는 사전을 만들 때 단어만 대상으로 하지 않고 일반적으로 n-그램을 대상으로 하기 때문이다. n-그램이란 연속으로 나타나는 n개 단어를 말한다. 예를 들어 "Data science is exciting and motivating."의 2-그램은 data-science, science-is, is-exciting, exciting-and, and-motivating이다. n-그램을 사용하면 단어가 나타나는 순서 정보를 어느 정도 보완할 수 있다는 장점이 있다.

02.1 DTM이란?

DTM

- 문서에 나타난 단어의 빈도를 표현하는 행렬
- 예제) 말뭉치에 다음과 같은 세 개의 문서가 있다고 가정

D1 : "Data science is exciting and motivating."
D2 : "I like literature class and science class."
D3 : "What is data science?"

문서별 단어 발생 빈도

	data	science	exciting	motivating	I	like	literature	class	what
D1	1	1	1	1	0	0	0	0	0
D2	0	1	0	0	1	1	1	2	0
D3	1	1	0	0	0	0	0	0	1

- 사전$_{dictionary}$ 만들기 (문서에 나타난 단어를 모으면 사전이 됨)
- 예제에서는 9개의 단어가 사전을 구성
- 위 표는 사전에 있는 단어별로 발생 빈도를 나타냄

02.1 DTM이란?

DTM 예제

- 3개 문서를 다음과 같이 9차원 벡터로 표현

$$D1=(1, 1, 1, 1, 0, 0, 0, 0, 0)$$
$$D2=(0, 1, 0, 0, 1, 1, 1, 2, 0)$$
$$D3=(1, 1, 0, 0, 0, 0, 0, 0, 1)$$

- DTM 형태로 쓰면,

$$DTM = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

02.1 DTM이란?

DTM 부연 설명

- 사전 구축
  - 실제에서는 사전의 크기는 보통 수만~수십만
  - 말뭉치에서 수집할 수도 있고, 국어사전에 실려있는 단어 집합을 사용할 수도 있음
- 문서가 벡터로 표현되므로 거리 또는 유사성 측정 가능
  - 앞의 예제에서 D1 벡터는 D2 벡터보다 D3 벡터와 가까움
  - 랜덤 포리스트나 SVM 등의 적용이 가능해짐
- 정규화 필요성
  - 문서가 길면 벡터의 길이가 커져서 유사한 문서와 거리가 멀어짐
  - 벡터의 길이를 1로 만드는 정규화를 적용하여 해결 가능

02.1 DTM이란?

DTM 부연 설명

- DTM은 희소 행렬
  - 한번도 발생하지 않아 0인 칸이 아주 많음
- DTM은 단어 사이의 상호작용을 표현 못함
  - “Data science is exciting and motivating”과 “Data is exciting and science is motivating”은 같은 벡터로 변환됨
  - 2-그램이나 3-그램으로 해결 가능하나 열의 개수가 기하급수적으로 커짐

02.2 R을 이용한 전처리와 DTM 구축

■ 위키피디아의 “data science” 문서로 실험

- RCurl 라이브러리로 웹 서버에 접속
- XML 라이브러리로 웹 문서 처리

```
> library(RCurl)
> library(XML)

> t = readLines('https://en.wikipedia.org/wiki/Data_science')
> d = htmlParse(t, asText = TRUE)
> clean_doc = xpathSApply(d,"//p", xmlValue)
```

- readLines 함수는 지정된 URL에서 html 파일을 읽어옴
- htmlParse와 xpathSApply 함수는 웹 문서를 R의 데이터 형으로 변환해줌

02.2 R을 이용한 전처리와 DTM 구축

■ 전처리

- tm 라이브러리는 데이터 마이닝 함수 제공
- SnowballC 라이브러리는 어간을 추출하는 함수 제공
- tm_map 함수는 지정된 매개변수 값에 따라 전처리 수행

```
> library(tm)
> library(SnowballC)

> doc = Corpus(VectorSource(clean_doc))
> inspect(doc)

> doc = tm_map(doc, content_transformer(tolower))
> doc = tm_map(doc, removeNumbers)
> doc = tm_map(doc, removeWords, stopwords('english'))
> doc = tm_map(doc, removePunctuation)
> doc = tm_map(doc, stripWhitespace)
```

02.2 R을 이용한 전처리와 DTM 구축

DocumentTermMatrix 함수로 DTM 구축

```
> dtm = DocumentTermMatrix(doc)
> dim(dtm)
[1]  17 576

> inspect(dtm)
<<DocumentTermMatrix (documents: 17, terms: 576)>>
Non-/sparse entries: 788/9004
Sparsity           : 92%
Maximal term length: 22
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs data field many methods now science scientists statistical statistics term
  10    8     0    0       1   0       5           2           1          0    0
  13   14     0    0       0   0       9           0           4          0    0
  14   11     0    1       0   0       4           3           2          1    3
  15    9     1    1       0   2      10           0           0          0    0
  16   13     3    2       0   1       7           1           0          3    0
  17    8     0    0       0   0       6           0           1          0    0
  5     5     0    3       0   4       5           0           0          3    3
  6     3     0    0       2   0       3           0           0          0    4
  8     7     0    0       0   0       5           1           2          3    1
  9     6     3    0       1   0       3           0           1          2    0
```

dim 함수는 DTM의 행과 열의 개수를 알려줌

inspect 함수는 상세 내용을 요약하여 보여줌

위키에 있는 문장 17개 각각을 문서로 간주함
576개의 단어를 추출하여 사전 구축

17*576=9792개의 칸 중에서
9004개는 0이라는 사실을 알려줌

03 단어 구름word cloud

■ DTM을 구성하는 단어를 가시화

- 단어마다 중요성이 다르고, 단어 사이에 연관관계 정보가 있음. 예) data는 field보다 science와 연관성이 더 높음
- 단어 구름은 이런 정보를 2차원 공간에 표시하는 가시화 기법
- 중요도가 높은 단어는 큰 폰트를 써서 중앙에 배치. 연관성이 높은 단어는 가까이 배치

03.1 wordcloud 라이브러리

■ 단어 구름 예제

- 앞 절에서 구한 DTM을 활용

```
> library(wordcloud)

> m = as.matrix(dtm)
> v = sort(colSums(m), decreasing = TRUE)
> d = data.frame(word = names(v), freq = v)
> wordcloud(words = d$word, freq = d$freq, min.freq = 1, max.words = 100, random.
order = FALSE, rot.per = 0.35)
```

as.matrix 함수는 DTM을 행렬 표현으로 변환

sort 함수는 빈도(중요도)가 높은 순서로 단어를 정렬

중요도가 높은 상위 100개만 그리라는 옵션

세로로 배치할 단어의 비율을 35%로 하라는 옵션

03.1 wordcloud 라이브러리

■ 단어 구름의 여러 가지 옵션

- RColorBrewer 라이브러리 (6.3.2절 참조)를 이용해 색상 입히기

```
> library(RColorBrewer)
> pal = brewer.pal(11,"Spectral")
> wordcloud(words = d$word, freq = d$freq, min.freq = 1, max.words = 50, random.
order = FALSE, rot.per = 0.50, colors = pal)
> wordcloud(words = d$word, freq = d$freq, min.freq = 1, max.words = 50, random.
order = FALSE, rot.per = 0.50, colors = pal, family = "mono", font = 2)
```

가로 세로 비율

색상

단어 개수

폰트

(a) 단어 개수 조절

(b) 폰트 조절

03.2 wordcloud2 라이브러리

■ 보다 뛰어난 wordcloud2 라이브러리

- wordcloud2는 wordcloud 이후에 나온 새로운 버전

```
> library(wordcloud2)
> wordcloud2(d)
```

- 자동으로 색상 입혀주고 단어를 다양한 방향으로 배치해 줌

wordcloud2로 만든 단어 구름

03.2 wordcloud2 라이브러리

■ wordcloud2의 여러 가지 옵션들

wordcloud2에는 단어 개수 지정하는 옵션이 없어 미리 추출

배경 모양

```
> d1 = d[1:200, ]          # 200개 단어만 표시
> wordcloud2(d1, shape = 'star')
> wordcloud2(d1, minRotation = pi/4, maxRotation = pi/4, rotateRatio = 1.0)
```

단어 방향 범위 지정

wordcloud2의 변형

03.3 빈도 표시하기

- findFreqTerms와 findAssocs 함수

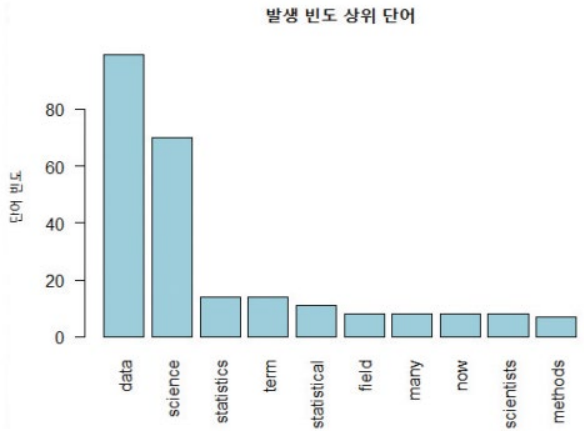
```
> findFreqTerms(dtm, lowfreq = 12)
[1] "data"       "science"    "statistics" "term"

> findAssocs(dtm, terms = 'harvard', corlimit = 0.7)
$harvard
sexiest     job  review
   0.94    0.79    0.79

> barplot(d[1:10, ]$freq, las = 2, names.arg = d[1:10,]$word, col = 'lightblue',
main = '발생 빈도 상위 단어', ylab = '단어 빈도')
```

상위 12개 단어만 표시하라는 옵션

'harvard'와 상관관계가 0.7이상인 단어를 표시하라는 옵션



중요도에 따라 단어를 나열한 막대그래프

03.4 텍스트 이외의 응용: gapminder 예제

- wordcloud는 데이터 프레임을 사용
  - 첫번째 열은 단어, 두번째 열은 해당 단어의 빈도수인 데이터 프레임
  - 텍스트 이외에도 이런 형식을 갖추면 단어 구름 가능함
  - 예) gapminder에서 첫번째 열은 대륙, 두번째 열은 해당 대륙의 인구가 되도록 데이터 추출

```
> library(gapminder)
> library(dplyr)

> pop_siz = gapminder%>%filter(year==2007)%>%group_by(continent)%>%summarize(sum
(as.numeric(pop)))
> d = data.frame(word = pop_siz[, 1], freq = pop_siz[, 2])
> wordcloud(words = d[, 1], freq = d[, 2], min.freq = 1, max.words = 100, random.
order = FALSE, rot.per = 0.35)
> wordcloud2(d)
```

(a) wordcloud 사용 (b) wordcloud2 사용

gapminder를 단어 구름으로 그린 모양

04 문서 분류

■ 지도 학습supervised learning에 속하는 분류 문제

- 예) Preview에서 제기한 시 분류 문제
- 다양한 응용
  - 영화 관람평을 모델링하면 흥행 예측 가능
  - 상품에 대한 댓글을 분석하여 마케팅 전략 세움
  - 트윗을 분석하여 대선이나 총선 결과 예측
  - 주식 관련 댓글을 보고 주가 예측
  - …..

04.1 영화평 분류: movie_review 데이터 모델링

movie_review 데이터

- text2vec 라이브러리가 제공
- 5000개의 샘플, 3개의 변수(id, sentiment, review)
  - id는 일련번호이므로 무시, sentiment는 반응 변수에 해당(1은 긍정 평가, 0은 부정 평가)
  - Review 변수는 영화를 평가하는 텍스트

```
> library(text2vec)
> library(caret)

> str(movie_review)
'data.frame':          5000 obs. of  3 variables:
 $ id       : chr  "5814_8" "2381_9" "7759_3" "3630_4" ...
 $ sentiment: int  1 1 0 0 1 1 0 0 0 1 ...
 $ review   : chr  "With all this stuff going down at the moment with MJ i've
 started listening to his music, watching the odd docu"¦ __truncated__ "\\\"The
... 생략 ...
```

```
> head(movie_review)
      id sentiment
1 5814_8         1
2 2381_9         1
3 7759_3         0
4 3630_4         0
5 9495_8         1
6 8196_8         1
... 생략 ...
```

04.1 영화평 분류: movie_review 데이터 모델링

■ 첫 번째 샘플의 review 변수 내용

With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again. Maybe i just want to get a certain insight into this guy who i thought was really cool in the eighties just to maybe make up my mind whether he is guilty or innocent. Moonwalker is part biography, part feature film which i remember going to see at the cinema when it was originally released. Some of it has subtle messages about MJ's feeling towards the press and also the obvious message of drugs are bad m'kay.

Visually impressive but of course this is all about Michael Jackson so unless you remotely like MJ in anyway then you are going to hate this and find it boring. Some may call MJ an egotist for consenting to the making of this movie BUT MJ and most of his fans would say that he made it for the fans which if true is really nice of him.

 … (생략)

movie_review 데이터의 첫 번째 샘플

이 샘플의 sentiment 변수 값은 1(긍정평가)

04.1 영화평 분류: movie_review 데이터 모델링

■ 6:4 비율로 훈련 집합과 테스트 집합으로 분할

```
> # 데이터를 훈련 집합(mtrain)과 테스트 집합(mtest)으로 나눔.
> train_list=createDataPartition(y=movie_review$sentiment, p=0.6, list= FALSE)
> mtrain=movie_review[train_list, ]
> mtest=movie_review[-train_list, ]
```

■ 훈련 집합에 대해 DTM 구축

```
> # 훈련 집합으로 DTM 구축
> doc=Corpus(VectorSource(mtrain$review))
> doc=tm_map(doc, content_transformer(tolower))
> doc=tm_map(doc, removeNumbers)
> doc=tm_map(doc, removeWords, stopwords('english'))
> doc=tm_map(doc, removePunctuation)
> doc=tm_map(doc, stripWhitespace)

> dtm=DocumentTermMatrix(doc)
> dim(dtm)
[1]  3000 36871
```

사전의 크기는 36871 (3000개 문서에서 36871개의 단어가 추출됨)

04.1 영화평 분류: movie_review 데이터 모델링

- inspect 함수로 DTM의 내용을 살펴보면,

```
> inspect(dtm)
<<DocumentTermMatrix (documents: 3000, terms: 36871)>>
Non-/sparse entries: 295077/110317923
Sparsity           : 100%
Maximal term length: 45
Weighting          : term frequency (tf)
Sample             :
      Terms
Docs   even film good just like movie one really story time
  127     3    7   11    2    3     5   3      1     1    2
  1330    3    6    5    6    9    14   4      1     0    1
  1722    2   14    1    5    2     1   5      3     0    5
  1818    2    0    2    0    2     0   6      0     0    2
  1920    2   11    0    2    4     8   4      2     4    0
  2167    2    9    1    1    2     6   7      1     0    2
  2233    1    9    4    1    1     0   6      0     2    1
  2633    0    3    1    1    0     3   2      0     0    0
  2930    1    0    2    0    0     0  10      1     0    2
  40      0    0    0    3    2     2   1      2     0    0
```

3000*36871=110613000개의 칸 중에
295077개만 0이 아니고 나머지 110317923개는 0

04.1 영화평 분류: movie_review 데이터 모델링

DTM을 모델링 가능한 형태로 변환

- 사전이 아주 커서 그대로 적용하면 메모리 오류 발생 → removeSparseTerms 함수로 줄임

- cbind 함수로 반응 변수 sentiment를 덧붙임

```
> dtm_small = removeSparseTerms(dtm, 0.90)
> X = as.matrix(dtm_small)
> dataTrain = as.data.frame(cbind(mtrain$sentiment, X))
> dataTrain$V1 = as.factor(dataTrain$V1)
> colnames(dataTrain)[1] = 'y'
```

04.1 영화평 분류: movie_review 데이터 모델링

■ 결정 트리와 랜덤 포리스트를 학습

```
> library(rpart)
> r = rpart(y~., data = dataTrain)
> printcp(r)
Classification tree:
rpart(formula = y ~ ., data = dataTrain)

Variables actually used in tree construction:
[1] bad     best    even    great   life    nothing

Root node error: 1497/3000 = 0.499

n= 3000

        CP nsplit rel error  xerror     xstd
1 0.233801      0   1.00000 1.03607 0.018283
2 0.032398      1   0.76620 0.76620 0.017780
3 0.021710      3   0.70140 0.70875 0.017493
4 0.012692      5   0.65798 0.69272 0.017401
5 0.010000      6   0.64529 0.66333 0.017217
```

04.1 영화평 분류: movie_review 데이터 모델링

■ 결정 트리와 랜덤 포리스트를 학습

```
> par(mfrow = c(1, 1), xpd = NA)
> plot(r)
> text(r, use.n = TRUE)

> library(randomForest)
> f = randomForest(y~., data = dataTrain)
```

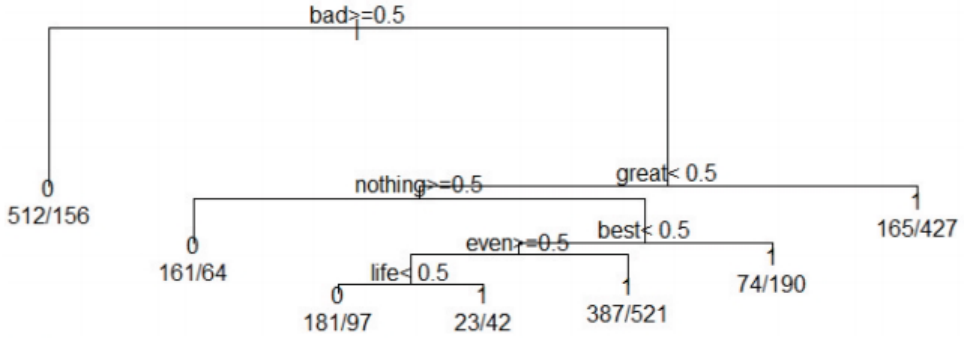


그림 11-11 movie_review 데이터로 학습한 결정 트리

04.2 예측과 성능 평가

■ 테스트 목적으로 남겨둔 mtest로 성능을 평가

- 먼저 mtest에 전처리 적용 (학습 과정과 동일한 과정을 적용해야 함)
- 훈련 집합이 사용한 사전을 테스트 과정에도 그대로 사용해야 함

```
> # 테스트 집합으로 DTM 구축
> docTest=Corpus(VectorSource(mtest$review))
> docTest=tm_map(docTest, content_transformer(tolower))
> docTest=tm_map(docTest, removeNumbers)
> docTest=tm_map(docTest, removeWords, stopwords('english'))
> docTest=tm_map(docTest, removePunctuation)
> docTest=tm_map(docTest, stripWhitespace)

> dtmTest=DocumentTermMatrix(docTest, control=list(dictionary=dtm_
small$dimnames$Terms))
```

control 옵션으로 학습 과정이 사용한 사전을 여기서도 사용함

```
> dim(dtmTest)
[1] 3000  111
> str(dtmTest)
List of 6
 $ i       : int [1:60524] 1 1 1 1 1 1 1 1 1 1 ...
...생략...
> inspect(dtmTest)
<<DocumentTermMatrix (documents: 3000, terms: 111)>>
Non-/sparse entries: 60524/272476
...생략...
```

04.2 예측과 성능 평가

■ 결정 트리와 랜덤 포리스트로 예측 수행

```
> X=as.matrix(dtmTest)
> dataTest=as.data.frame(cbind(mtest$sentiment, X))
> dataTest$V1=as.factor(dataTest$V1)
> colnames(dataTest)[1]='y'
> pr=predict(r, newdata=dataTest, type='class')
> table(pr, dataTest$y)
pr    0   1
  0 570 601
  1 895 934
> pf=predict(f, newdata=dataTest)
> table(pf, dataTest$y)
pf    0   1
  0 738 765
  1 727 770
```

결정 트리 r

랜덤 포리스트 f

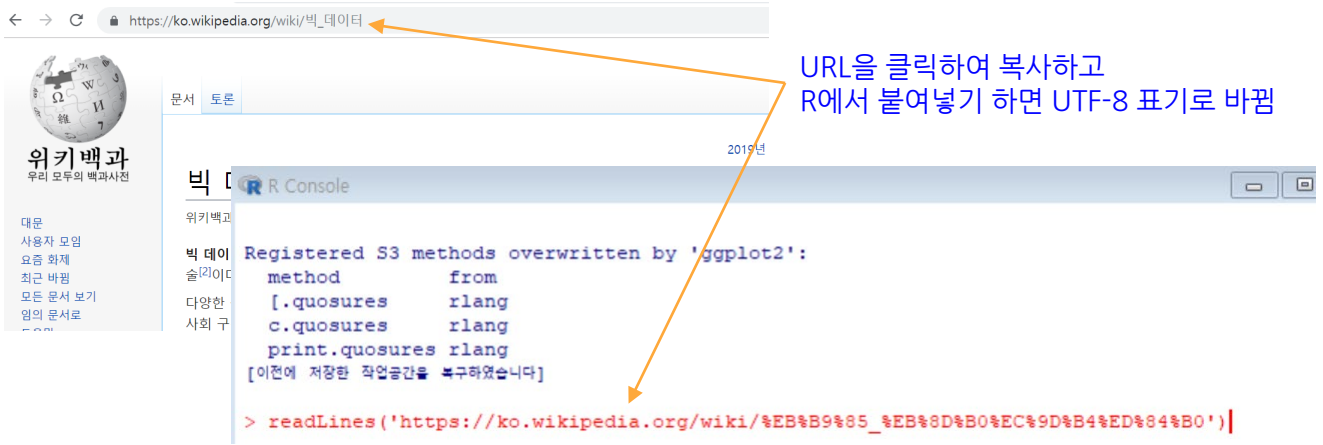
- 결정 트리는 (570+934)/2000=75.2%, 랜덤 포리스트는 (738+770)/2000=75.4%의 정확률

05 영어 텍스트 마이닝을 이용한 한국어 처리

■ 예제) 위키의 "빅 데이터" 설명 문서

```
> library(tm)
> library(XML)
> library(wordcloud2)
> library(SnowballC)
> library(RCurl)
> t = readLines('https://ko.wikipedia.org/wiki/%EB%B9%85_%EB%8D%B0%EC%9D%B4%ED%84%B0')
> d = htmlParse(t, asText = TRUE)
> clean_doc = xpathSApply(d, "//p", xmlValue)
```

'빅 데이터'의 UTF-8 표기



URL을 클릭하여 복사하고
R에서 붙여넣기 하면 UTF-8 표기로 바뀜

05 영어 텍스트 마이닝을 이용한 한국어 처리

■ 전처리 수행

```
> doc = Corpus(VectorSource(clean_doc))
> inspect(doc)
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 39

 [1] 빅 데이터(영어: big data)란 기존 데이터베이스 관리도구의 능력을 넘어서는 대량(수십 테라바
이트)의 정형 또는 심지어 데이터베이스 형태가 아닌 비정형의 데이터 집합조차 포함한[1] 데이터로부터
가치를 추출하고 결과를 분석하는 기술[2]이다.\n
...생략...
> doc = tm_map(doc, content_transformer(tolower))
> doc = tm_map(doc, removeNumbers)
> doc = tm_map(doc, removePunctuation)
> doc = tm_map(doc, stripWhitespace)
```

05 영어 텍스트 마이닝을 이용한 한국어 처리

■ DTM을 구축

```
> dtm = DocumentTermMatrix(doc)
> dim(dtm)
[1]   39 1443

> inspect(dtm)
<<DocumentTermMatrix (documents: 39, terms: 1443)>>
Non-/sparse entries: 1920/54357
Sparsity           : 97%
Maximal term length: 24
Weighting          : term frequency (tf)
Sample             :
    Terms
 Docs 년 데이터 데이터를 데이터의 등 분석 빅 빅데이터 수 있다
   11   0      3       2       1  0    2   1        0   0     1
   15   1      0       0       0  2    2   0        0   1     0
   16   4      1       1       0  2    0   2        0   0     0
   18   2      0       2       3  1    1   1        0   1     1
   22   0      0       3       0  2    1   0        1   1     1
   24   0      3       2       0  1    0   4        0   0     3
   26   0      0       1       0  1    1   0        1   1     1
   39   2      1       2       0  1    0   1        0   1     1
    7   0      4       4       2  0    0   0        1   4     4
    8   2      2       0       3  0    0   1        0   0     0
```

39개 문장 각각을 문서로 간주하여 39개 문서 추출
이들 문서에서 1443개의 단어를 추출하여 사전 구축

'등', '있다 ' 등을 단어로 추출하였고,
'데이터', '데이터를', '데이터의'를 다른 단어로 추출하는 한계

05 영어 텍스트 마이닝을 이용한 한국어 처리

■ 데이터 프레임으로 변환하고 단어 구름을 그리면,

```
> m = as.matrix(dtm)
> v = sort(colSums(m), decreasing = TRUE)
> d = data.frame(word = names(v), freq = v)
> d1 = d[1:500, ]                    # 500개 단어만 표시
> wordcloud2(d1)
```

‘등’, ‘및‘, ‘있다 ' 등이 중요 자리 차지
‘데이터’, ‘데이터를‘, ‘데이터의’가 다른 단어로 간주되어 중요 자리 차지
← 영어 텍스트 마이닝을 한글에 적용한 탓

위키피디아의 ‘빅 데이터’ 문서에서 추출한 단어 구름

06 KoNLP를 이용한 한국어 텍스트 마이닝

- KoNLP는 한국어를 전용으로 처리하는 텍스트 마이닝 라이브러리

  - SystemDic, SejongDic, NIADic이라는 세 종류의 사전을 지원함

```
> library(KoNLP)
> useSystemDic()
Backup was just finished!

283949 words dictionary was built.

> useSejongDic()
Backup was just finished!

370957 words dictionary was built.

> useNIADic()
Backup was just finished!

983012 words dictionary was built.
```

사전 크기

06.1 KoNLP를 이용한 형태소 분석

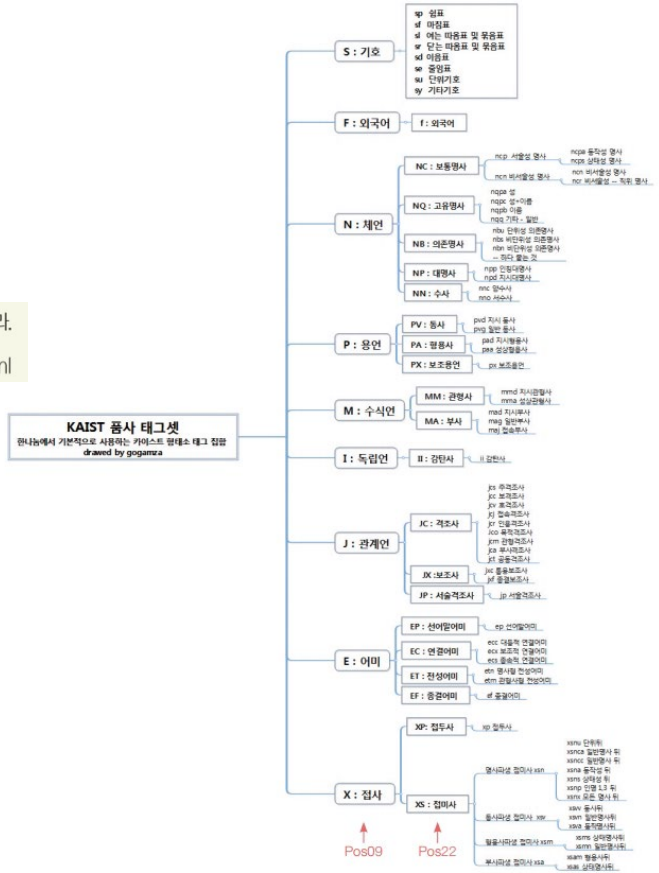
■ 한국어 품사 태그셋

- Pos09는9 종류로 품사 구분

- Pos22는 22 종류로 품사 구분

NOTE KoNLP와 카이스트 품사 태그셋에 관한 자세한 내용은 다음 문서를 참조하라.

https://cran.r-project.org/web/packages/KoNLP/vignettes/KoNLP-API.html



카이스트 품사 태그셋

06.1 KoNLP를 이용한 형태소 분석

■ 한글 형태소 분석 예제

- 형태소$_{morpheme}$란 언어를 구성하는 가장 작은 문법 요소

- 형태소 분석이란 문장을 형태소 단위로 분할하는 작업

```
> useSejongDic()
> s='너에게 묻는다 연탄재 함부로 발로 차지 마라 너는 누구에게 한번이라도 뜨거운 사람이었느냐'
> extractNoun(s)
[1] "너"        "연탄재"    '발'     "차"      "너"     "누구"      "한"       "번"
[9] "사람이었"    "냐"

> simplePoss22(s)
$'너에게'
[1] "너/NP+에게/JC"

$묻는다
[1] "묻/PV+는다/EF"

$연탄재
[1] "연탄재/NC"

$함부로
[1] "함부로/MA"

$발로
[1] "발/NC+로/JC"

$차지
[1] "차/NC+이/JP+지/EC"

$마라
[1] "마르/PV+아/EC"

$너는
[1] "너/NC+는/JX"

$누구에게
[1] "누구/NP+에게/JC"

$한번이라도
[1] "한/NN+번/NB+이라도/JX"

$뜨거운
[1] "뜨겁/PA+은/ET"

$사람이었느
[1] "사람이었느/NC"

$"냐/NC"
[1] "냐/NC"
```

extractNoun 함수는 명사를 추출

simplePos22 함수는 Pos22 단계까지 형태소 분석을 수행

'너에게'를 '너'라는 대명사(NP)와 '에게'라는 격조사(JC)로 분해

06.2 KoNLP를 단어 구름 그리기

■ 위키 “빅 데이터” 예제 (11.5절 참조)

```
> t=readLines('https://ko.wikipedia.org/wiki/%EB%B9%85_%EB%8D%B0%EC%9D%B4%ED%84%B0')
> d = htmlParse(t, asText=TRUE)
> clean_doc = xpathSApply(d, "//p", xmlValue)
```

```
> useSejongDic()

> nouns = extracNoun(clean_doc)
> mnous = unlist(nouns)
> mnous_freq = table(mnous)
> v = sort(mnous_freq, decreasing = TRUE)

> wordcloud2(v)              # 모든 단어 표시 : [그림 11-14(a)]
> v1 = v[1:100]              # 상위 100개 단어만 골라 표시 : [그림 11-14(b)]
> wordcloud2(v1)
```

06.2 KoNLP를 단어 구름 그리기

■ 단어 구름

- 영어 텍스트 마이닝을 활용한 [그림 11-12]에 비해 큰 향상
- 하지만 여전히 한계 (KoNLP의 한계)
  - '빅데이터'와 '빅데이터를'이 함께 나타나는 현상
  -

(a) 모든 단어 표시 (b) 상위 100개 단어만 골라 표시

위키피디아의 '빅 데이터' 문서를 KoNLP로 처리하여 추출한 단어 구름