고려대학교
빅데이터 연구회

KU-BIG

Project 최종 발표

- 자유 주제 조 -

조원 : 박정진 이길재 정석원 최지인

자유 주제?

자유 주제 조의 “Identity”?

주제와 알고리즘에서 자유롭기에 하나의 목표를
다양한 수단과 방법으로 탐구해볼 수 있다!

CONTENTS

1. 문제 정의

- TEXT 데이터에 나타나는 Attitude [긍정, 부정 등]

➔ 학습할 수 있을까?

➔ 만약 가능하다면, 다양한 분야에서 다양한 가치 창출 가능 예상

➔ **TEXT 데이터 감정 분석 시도**

2. 데이터 소개

- DATA : Rotten Tomatoes 리뷰
- 목표 : 리뷰 데이터 감정 분석을 통한

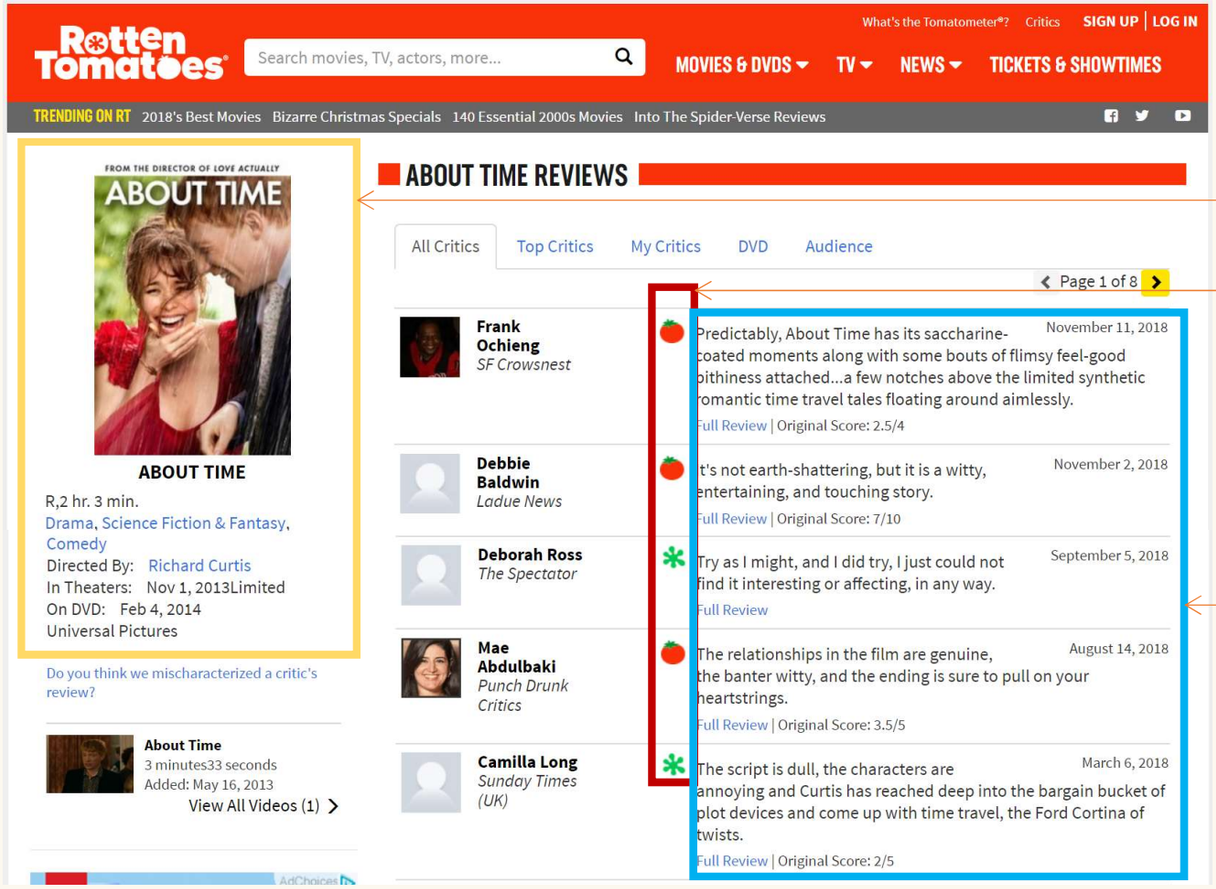
Tomatometer 예측

2. 데이터 소개 – Tomatometer?

- Fresh / (Fresh + Rotten)

➔ 즉 전체 비평 중 FRESH로 평가한비평의 비율

2. 데이터 소개 – Review Page



Movie Information

Fresh / Rotten Index

Reviews

2. 데이터 소개 – 데이터 선택 이유

- 흥미로운 주제 [영화]
- 적당한 난이도의 TEXT 데이터 [비평가의 영화 비평]

  ➔ 신문기사, 논문 : 매우 정제된 데이터

  ➔ SNS, 댓글 : 매우 복잡한 데이터

2. 데이터 소개 – Crawler [Web Crawling]

```
# Crawler
CrawlerbySuper <- function(movie_name, directory) {
  if (missing(directory)) directory <- getwd()
  url0<-paste("https://www.rottentomatoes.com/m/",movie_name,"/reviews/?page=",sep="")
  htxt0<-read_html(url0)
  pageinfo<-html_node(htxt0,".pageInfo")
  pagenum0<-html_text(pageinfo)
  pages<-word(pagenum0,-1)
  for(page in 1:pages){
    url <- paste("https://www.rottentomatoes.com/m/", movie_name,
                 "/reviews/?page=", page, "&sort=", sep="")
    htxt <- read_html(url)
    icon <- html_nodes(htxt, ".review_icon")
    tomato <- character()
    for (i in 1:length(icon))
      tomato[i] <- xml_attrs(icon[[i]])[["class"]]
    fi <- (tomato == "review_icon icon small fresh")
    ri <- (tomato == "review_icon icon small rotten")
    table <- html_nodes(htxt, ".review_table")
    content <- html_nodes(table, ".the_review")
    review <- html_text(content)
```

2. 데이터 소개 – Crawler [Web Crawling]

```
    fresh <- review[fi]
    for (j in 1:length(fresh)) {
      write.table(paste(fresh[j], "dlrlfwo"),
                  file=paste(directory,"/",movie_name,"_fresh_",page,"_",j,".txt", sep=""),
                  quote = FALSE, row.names = FALSE, col.names = FALSE)
    }
    rotten <- review[ri]
    for (k in 1:length(rotten)) {
      write.table(rotten[k],
                  file=paste(directory,"/",movie_name,"_rotten_",page,"_",k,".txt", sep=""),
                  quote = FALSE, row.names = FALSE, col.names = FALSE)
    }
    if(length(fresh) == 0 & length(rotten) == 0) {break}
    print(page)
  }
}
```

3. Model 1 – 지도학습 감정분석

〈텍스트 데이터 전처리〉

We are students.
The book is interesting.

We be student
book be interest

〈Text 정형화〉

We be student
book be interest

Term / Document	be	book	interest	student	we
Document 1	1	0	0	1	1
Document 2	1	1	1	0	0

〈Text -> Matrix〉

3. Model 1 – 감정분석 by 지도학습

숫자 표현 제거

⬇

특수 문자 처리

⬇

공란 처리

⬇

대소문자 통일

⬇

불용단어 제거

⬇

어근 동일화

- 텍스트 데이터 전처리에는 명확한 해답이 없기 때문에, 텍스트의 특성을 분석하고 이를 비판적으로 검토하는 과정이 필요

- 다음과 같은 순서로 정형화 과정을 진행

- TM 패키지의 여러 함수들을 이용함

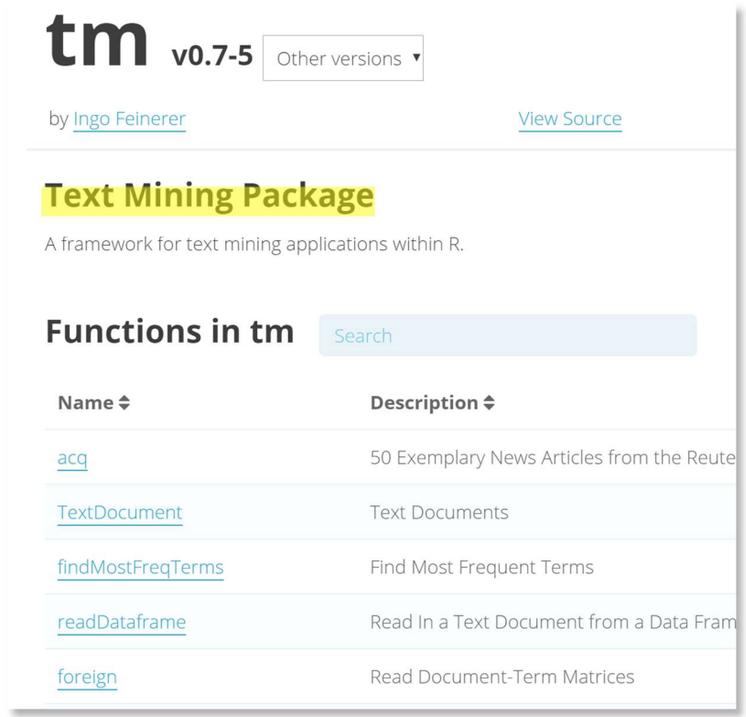
3. Model 1 – 감정분석 by 지도학습

〈정형화 과정〉

```r
library(tm)


TomatoProcessor <- function(review) {
  ppreview <- tm_map(review, removeNumbers)
  #숫자 제거
  ppreview <- tm_map(ppreview, removePunctuation)
  #문자 제거
  ppreview <- tm_map(ppreview, stripWhitespace)
  #공란 처리
  ppreview <- tm_map(ppreview, content_transformer(tolower))
  #대소문자 통일
  ppreview <- tm_map(ppreview, removeWords, words=stopwords("SMART"))
  #불용단어 제거 처리
  ppreview <- tm_map(ppreview, stemDocument, language="en")
  #어근 동일화 처리
  return(ppreview)
}
```

〈Review data에 적용시킨 정형화 과정〉



〈"tm" library in R 〉

3. Model 1 – 감정분석 by 지도학습

```
> inspect(myDTM[[1]][1:5, 25:33])
<<DocumentTermMatrix (documents: 5, terms: 9)>>
Non-/sparse entries: 0/45
Sparsity           : 100%
Maximal term length: 7
Weighting          : term frequency - inverse document frequency (tf-idf)
Sample             :
                                Terms
Docs                             activ actor actress actual acut ada adapt add addit
  bohemian_rhapsody_fresh_1_1.txt      0     0       0      0    0   0     0   0     0
  bohemian_rhapsody_fresh_1_10.txt     0     0       0      0    0   0     0   0     0
  bohemian_rhapsody_fresh_1_11.txt     0     0       0      0    0   0     0   0     0
  bohemian_rhapsody_fresh_1_12.txt     0     0       0      0    0   0     0   0     0
  bohemian_rhapsody_fresh_1_13.txt     0     0       0      0    0   0     0   0     0
```

〈DTM 구조〉

3. Model 1 – 감정분석 by 지도학습

```r
# 전처리
TomatoProcessor <- function(review) {
  ppreview <- tm_map(review, removeNumbers)
  ppreview <- tm_map(ppreview, removePunctuation)
  ppreview <- tm_map(ppreview, stripwhitespace)
  ppreview <- tm_map(ppreview, content_transformer(tolower))
  ppreview <- tm_map(ppreview, removeWords, words=stopwords("SMART"))
  ppreview <- tm_map(ppreview, stemDocument, language="en")
  return(ppreview)
}

# Text -> Matrix
TomatoDTM <- function(directory) {
  if (missing(directory)) directory <- getwd()
  else directory <- paste(getwd(),directory,sep='/')
  label <- numeric()
  review <- VCorpus(DirSource(directory))
  ppreview <- TomatoProcessor(review)
  for (i in 1:length(ppreview))
    if (sum(strsplit(ppreview[[i]]$content, split = " ")[[1]] == "dlrlfwo") != 0)
      label[i] <- 1
  else label[i] <- 0
  ppreview <- tm_map(ppreview, removeWords, words=("dlrlfwo"))
  DTMreview <- DocumentTermMatrix(ppreview, control = list(weighting = function(x) weightTfIdf(x, normalize = "FALSE")))
  return(list(DTMreview, label))
}
```

〈전처리 통합 함수〉

1. 정형화
- “tm” library 사용
- 방금 전 살펴본 논리 적용
- TomatoProcessor 함수로 적용.

2. DTM 구성
- TomatoDTM 함수로 적용
- TomatoProcessor 함수를 내장하였음.

3. Model 1 – 감정분석 by 지도학습

```
myDTM <- TomatoDTM(directory = "action") # DTM 구성

TTdivide <- TrainTestSize(myDTM, c(8,2)) # Training Set, Test Set 분류

mycontainer <- create_container(myDTM[[1]], myDTM[[2]],
                                trainSize = TTdivide[[1]], TTdivide[[2]],
                                virgin = FALSE)
                               #container 생성과정.


mytestlabel <- myDTM[[2]][TTdivide[[2]]] #Test data의 label 따로 지정

myclassifier <- train_models(mycontainer, algorithms = c("SVM")) #모델 트레이닝

myresult <- classify_models(mycontainer, myclassifier) #Train Model을 Test Data에 적용
```

〈RtextTools 이용한 지도 학습 모델 구축〉

모델 학습

- “RTextTools” library 사용
- 간편하게 9가지 알고리즘을 이용 가능(SVM, SLDA, BOOSTING, BAGGING, RF, GLMNET, TREE, NNET, MAXENT)
- 학습은 앞에서 이야기 한 대로 장르를 나누어서 진행함

3. Model 1 – 감정분석 by 지도학습

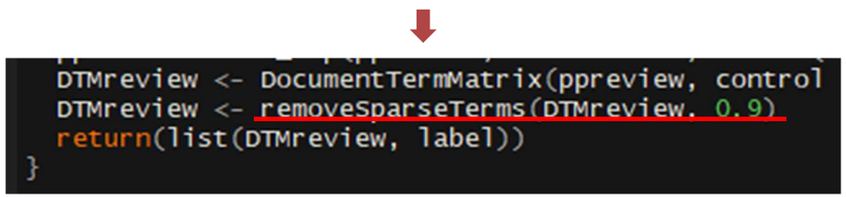
```
> table(myresult$SVM_LABEL, mytestlabel)
   mytestlabel
     0  1
  0 52 65
  1 21 31
> table(myresult$SLDA_LABEL, mytestlabel)
   mytestlabel
     0  1
  0 50 65
  1 23 31
> table(myresult$BAGGING_LABEL, mytestlabel)
   mytestlabel
     0  1
  0 71 95
  1  2  1
> table(myresult$FORESTS_LABEL, mytestlabel)
   mytestlabel
     0  1
  0 65 87
  1  8  9
```

〈다양한 알고리즘 적용한 결과〉

- 생각보다 좋지 않은 모델링 결과
- 전처리를 나름 논리적으로 진행 하였으나 예측력이 Random Guess에 비하여 좋다고 보기 힘듦

3. Model 1 – 감정분석 by 지도학습

```
> myDTM
[[1]]
<<DocumentTermMatrix (documents: 324, terms: 1501)>>
Non-/sparse entries: 3735/482589
Sparsity           : 99%
```



```
  DTMreview <- DocumentTermMatrix(ppreview, control
  DTMreview <- removeSparseTerms(DTMreview, 0.9)
  return(list(DTMreview, label))
}
```

```
> myDTM
[[1]]
<<DocumentTermMatrix (documents: 324, terms: 12)>>
Non-/sparse entries: 826/3062
Sparsity           : 79%
```

- DTM의 Sparsity가 너무 높음
- Sparsity가 너무 큰 것은 차원의 저주, Overfitting의 문제점을 야기할 가능성이 있음
- removeSparseTerms 함수를 사용하여 Sparsity를 줄이고 난 후 다시 모델링을 해보기로 함

3. Model 1 – 감정분석 by 지도학습

〈다양한 문제해결 시도에도..〉

```
> table(myresult$SVM_LABEL, mytestlabel)
   mytestlabel
      0   1
  0 379 386
  1  87  84
```

```
> table(myresult$FORESTS_LABEL, mytestlabel)
   mytestlabel
      0   1
  0 409 393
  1  57  77
```

3. Model 2 – 감정분석 by 감정사전

〈감정사전을 이용한 분석의 장점〉

- 감정사전은 텍스트에 있는 단어와 사전에 있는 단어를 비교
- 지도학습처럼 DTM을 구축할 필요가 없다

→ 내장함수를 이용하여 stemming만 해줌

3. Model 2 – 감정분석 by 감정사전

〈분류기 - Tidytext를 이용한 토큰화〉

```
> mytxt[1:10]
 [1] " Anchored by powerful, honest performances by Danielsen and Gravli, who play two people at opposite ends of a rifl
e, 22 July is a powerful reminder that democracies are fragile and must be protected with scrutiny and resolve, not bull
ets. dlrlfwo"
 [2] " Violent but affecting docudrama about Norway massacre.  dlrlfwo"

 [3] " There's nothing else in 22 July that you can't find in a good book or newspaper article about Hanssen and his fel
low survivors. dlrlfwo"

 [4] " With 22 July, [director Paul Greengrass] rattles the viewer with a poignant story about the killing of 77 people
in Norway, on that day in 2011. [Full review in Spanish] dlrlfwo"

 [5] " 22 July is a bleak film, but one that focuses on the people involved, and their journey after the atrocity of rig
ht wing extremism, rather than exploiting the violence of that day. dlrlfwo"

 [6] " It's done as thoroughly and as sensitively as you would imagine - yet I couldn't tell whether this was either too
 soon, or gave too much credence to the manifesto and questionings of a far-right lunatic. dlrlfwo"

 [7] " Greengrass manages to avoid turning the film into an utterly dire, hopeless watch. It's devastating, to be sure,
but not defeatist. dlrlfwo"

 [8] " The film is elevated by an impressive Norwegian cast performing in English. [Full review in Spanish] dlrlfwo"

 [9] " A hard-hitting docudrama about the July 22, 2011 mass shooting in Norway. dlrlfwo"

[10] " Leaves you a little disconcerted by its almost feel-good message that democracy can defeat fascism if only the yo
uth of the world make their voices heard. dlrlfwo"
```

```
> my.df.text.word[1:20,]
# A tibble: 20 x 2
   paper.id word
      <int> <chr>
 1        1 anchored
 2        1 by
 3        1 powerful
 4        1 honest
 5        1 performances
 6        1 by
 7        1 danielsen
 8        1 and
 9        1 gravli
10        1 who
11        1 play
12        1 two
13        1 people
14        1 at
15        1 opposite
16        1 ends
17        1 of
18        1 a
19        1 rifle
20        1 22
>
```

3. Model 2 – 감정분석 by 감정사전

〈감정 점수 (Sentiment score) 부여과정〉

```
> emotion[1:20,]
# A tibble: 20 x 4
   review.id fresh.sum rotten.sum fresh.sent
       <int>     <dbl>      <dbl>      <dbl>
 1         1         3          1          2
 2         2         0          2         -2
 3         3         1          0          1
 4         4         1          2         -1
 5         5         1          3         -2
 6         6         2          1          1
 7         7         0          4         -4
 8         8         1          0          1
 9         9         0          1         -1
10        10         2          2          0
11        13         3          1          2
12        14         0          2         -2
13        15         2          4         -2
14        16         1          0          1
15        17         1          0          1
16        18         0          2         -2
17        19         1          0          1
18        20         2          1          1
19        21         0          2         -2
20        22         1          2         -1
> |
```

- 각 review에서 토큰화된 단어들에 감정사전에 해당하는 점수 부여
- Fresh.sum(긍정점수)-rotten.sum(부정점수) = fresh.sent(해당 리뷰의 감정점수)

3. Model 2 – 감정분석 by 감정사전

<분류기의 구축>

```
# Classifier by Lexicon
LexiClassifier <- function(movie_name, lexicon, adjust = FALSE) {
  directory <- paste(getwd(),'/', movie_name, sep= '')

  review <- VCorpus(DirSource(directory), readerControl =list(language="en"))

  n <- length(review)
  mytxt <- rep(NA,n)

  for (i in 1:n) mytxt[i] <- as.character(review[[i]][1])

  my.df.text <- data_frame(review.id=1:n, doc=mytxt)

  my.df.text.word <- my.df.text %>%
    unnest_tokens(word,doc)

  if (lexicon == "bing" | lexicon == "nrc") {
    myresult.sa <- my.df.text.word %>%
    inner_join(get_sentiments(lexicon)) %>%
    count(word,review.id,sentiment) %>%
    spread(sentiment,n,fill=0)

    if (lexicon == "bing") {
      emotion <- summarise(group_by(myresult.sa, review.id),
                           fresh.sum = sum(positive),
                           rotten.sum = sum(negative),
                           fresh.sent = fresh.sum - rotten.sum)
    }
    else
      emotion <- summarise(group_by(myresult.sa, review.id),
                           fresh.sum = sum(positive) + sum(joy),
                           rotten.sum = sum(negative) + sum(anger) + sum(disgust),
                           fresh.sent = fresh.sum - rotten.sum)
  }

  else if (lexicon == "afinn") {
    myresult.sa <- my.df.text.word %>%
      inner_join(get_sentiments(lexicon))

    emotion <- summarise(group_by(myresult.sa, review.id),
                         fresh.sent=sum(score))
    emotion <- cbind(emotion[,1],numeric(nrow(emotion)),numeric(nrow(emotion)),emotion[,2])
    colnames(emotion) <- c("review.id","d1","d2","fresh.sent")
    }

  myfilenames <- list.files(path=directory,
                            pattern=NULL,all.files = TRUE)
  review.id <- 1:n
  review.name <- myfilenames[3:(n+2)]
  review.inf.df <- data.frame(review.id, review.name)
  emotion <- emotion %>% full_join(review.inf.df)
  if (lexicon == "afinn") {
    sent <- c(emotion[,4])
    split <- strsplit(as.character(emotion[,5]), "_")
    label <- unlist(strsplit(as.character(emotion[,5]), "_"))[(1:n) *  length(split[[1]]) - 2]
  }
LexiClassifier(movie_name, lexicon, adjust)
```

- 변수로는 영화제목, 사용할 감정 사전, 예측치 보정여부
- 지난 시간에 소개한 ‘Bing’, ‘Lexicon’, ‘Afinn’ 세 종류의 사전 사용가능
- "accuracy", "Tomatometer_actual", "Tomatometer_estimator“ 3가지 결과를 도출해줌

3. Model 2 – 감정분석 by 감정사전

〈분류기의 구축〉

```
myfilenames <- list.files(path=getwd(),
                          pattern=NULL,all.files = TRUE)
review.id <- 1:n
review.name <- myfilenames[3:(n+2)]
review.inf.df <- data.frame(review.id, review.name)
emotion <- emotion %>% full_join(review.inf.df)
if (lexicon == "afinn") {
  sent <- c(emotion[,4])
  split <- strsplit(as.character(emotion[,5]), "_")
  label <- unlist(strsplit(as.character(emotion[,5]), "_"))[(1:n) *  length(split[[1]]) - 2]
}
sent <- c(emotion[,4])[[1]]
split <- strsplit(as.character(c(emotion[,5])[[1]]), "_")
label <- unlist(strsplit(as.character(c(emotion[,5])[[1]]), "_"))[(1:n) *  length(split[[1]]) - 2]
else {
  sent <- c(emotion[,4])[[1]]
  split <- strsplit(as.character(c(emotion[,5])[[1]]), "_")
  label <- unlist(strsplit(as.character(c(emotion[,5])[[1]]), "_"))[(1:n) *  length(split[[1]]) - 2]
}
labelall <- label
k <- min(which(is.na(sent))) - 1
sent <- sent[1:k]; label <- label[1:k];
label <- label[sent != 0]; sent <- sent[sent != 0]

fcorrect <- sum(label[sent > 0] == "fresh")
rcorrect <- sum(label[sent < 0] == "rotten")

accuracy <- (fcorrect + rcorrect) / length(sent)
faccuracy <- fcorrect / sum(label == "fresh")
raccuracy <- rcorrect / sum(label == "rotten")

actualmeter <- sum(labelall == "fresh")/ length(labelall)
estimeter <- sum(sent > 0) / length(sent)
output <- data.frame(accuracy, actualmeter, estimeter)
output
```

- 감정점수를 바탕으로 review 감정의 긍정, 부정을 분류하는 과정
- Fresh.sent > 0 → 긍정적인 review
  Fresh.sent < 0 → 부정적인 review
- Review의 실제 감정과 분류기 통해 예측한 감정을 비교하여 accuracy 계산

3. Model 2 – 감정분석 by 감정사전

<예시 – 영화 “블랙 팬서” 리뷰 분석>

```
> CrawlerforLexi("black_panther_2018")
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
[1] 13
[1] 14
[1] 15
[1] 16
[1] 17
[1] 18
[1] 19
[1] 20
[1] 21
[1] 22
Warning message:
In dir.create(movie_name) : 'black_panther_2018' already exists
> LexiClassifier("black_panther_2018","nrc")
Joining, by = "word"
Joining, by = "review.id"
   accuracy Tomatometer_actual Tomatometer_estimator
1 0.7410468          0.9132321              0.738292
>
```

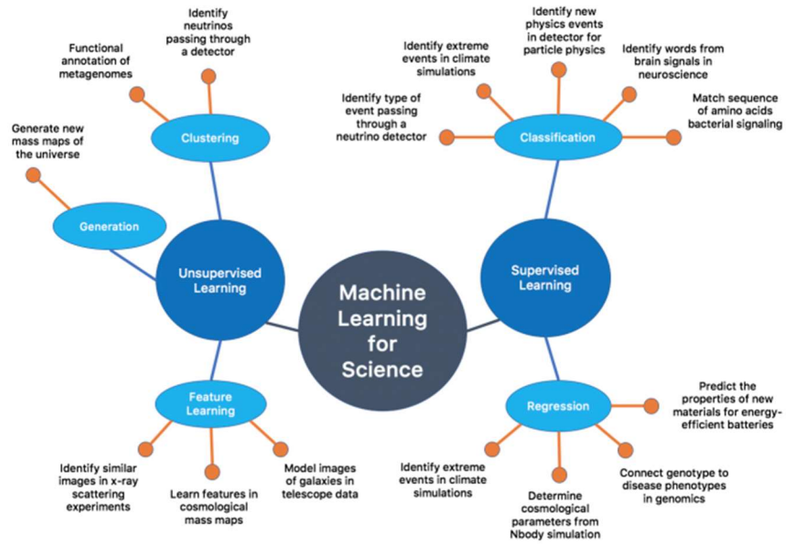
- 앞에서 구축한 Crawler와 Classifier를 이용하여 영화 “블랙 팬서"의 review에 대해 감정분석을 진행해 보았다.

- 약 74%의 accuracy 보여주어, 감정사전을 이용한 감정분석이 훨씬 뛰어난 성능을 보임을 알 수 있었다.

5. 결론

〈한 학기 동안 느낀 점 – 1. 한계〉

- 프로그램의 한계 (R)



- 지식의 한계

5. 결론

〈한 학기 동안 느낀 점 – 2. 의의〉

- 흥미로운 주제

- 다양한 라이브러리 경험

“한 학기동안 수고하셨습니다.”