

IMDB Movie Reviews Sentiment Statistical analysis

Demeke Kasaw

2023-06-17

Project Overview

I used the IMDb movie review dataset to perform sentiment analysis on movie reviews. Sentiment analysis involves determining the opinion expressed in a piece of text.

#Data Source The dataset is taken from kaggle website as the information provided from fellow.ai.

Load the required libraries

```
library(tm) # topic modeling
library(e1071) # build model
library(gmodels) # cross tabulation
library(dplyr) # data manipulation
library(readr) # read data
library(tidytext)
library(SnowballC)
library(ggplot2)
```

Import the dataset

```
movie_reviews <- read_csv("IMDB Dataset.csv")
```

Explore the dataset

```
glimpse(movie_reviews)
```

```
## Rows: 50,000
## Columns: 2
## $ review    <chr> "One of the other reviewers has mentioned that after watchin...
## $ sentiment <chr> "positive", "positive", "positive", "negative", "positive", ...
```

```
# check the distribution of the sentiment
movie_reviews |>
  count(sentiment)
```

```
## # A tibble: 2 × 2
##   sentiment      n
##   <chr>      <int>
## 1 negative  25000
## 2 positive  25000
```

##The data set has two columns and 50k observations ##Tokenization

```
reviews_df <- movie_reviews |>
  # select(review) |>
  unnest_tokens(word, review)

head(reviews_df)
```

```
## # A tibble: 6 × 2
##   sentiment word
##   <chr>      <chr>
## 1 positive  one
## 2 positive  of
## 3 positive  the
## 4 positive  other
## 5 positive  reviewers
## 6 positive  has
```

```
reviews_df <- reviews_df |>
  mutate(stem = wordStem(word))

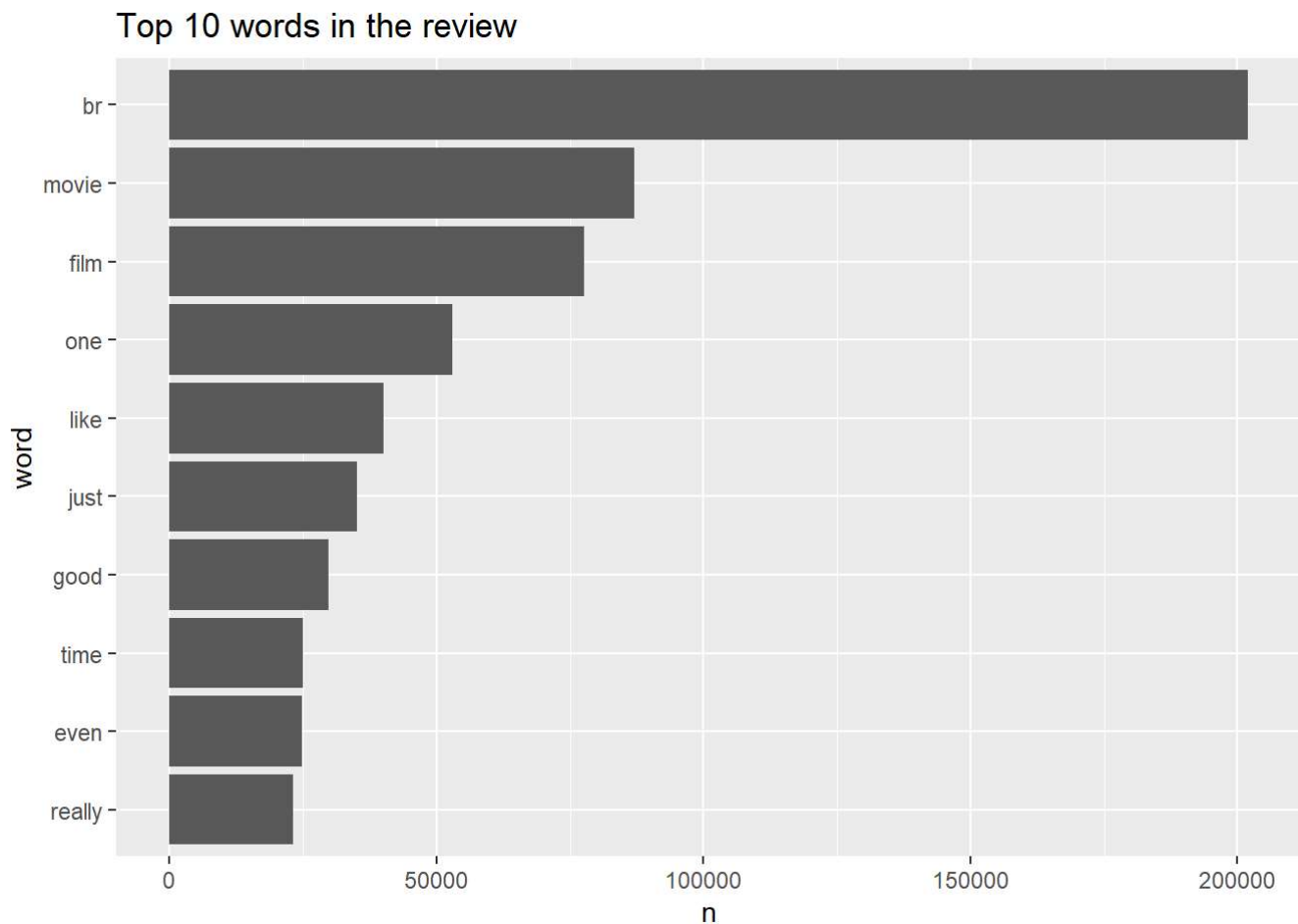
reviews_df <- reviews_df |>
  anti_join(stop_words[stop_words$lexicon == 'snowball',], by = "word")
```

```
reviews_df |>
  group_by(word) |>
  summarise(count = n()) |>
  arrange(desc(count)) |>
  head()
```

```
## # A tibble: 6 × 2
##   word      count
##   <chr>    <int>
## 1 br      201951
## 2 movie   86953
## 3 film    77608
## 4 one     53002
## 5 like    40150
## 6 just    35158
```

```
# visualize
```

```
reviews_df |>  
  count(word, sort = TRUE) |>  
  mutate(word = reorder(word,n)) |>  
  top_n(10) |>  
  ggplot(aes(x = n, y = word))+  
  geom_col()+  
  labs(title = "Top 10 words in the review")
```



Prepare for model building


The data has 50k observation, which relatively large data set to train the model on my personal laptop. So I decided to take random sample of 20k.

```
# convert the review texts into corpus  
set.seed(1234)  
movie_reviews <- sample_n(movie_reviews, size = 20000)  
reviews_corpus <- VCorpus(  
  VectorSource(movie_reviews$review))  
print(reviews_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:   documents: 20000
```

```
# check actual corpus in the text
as.character(reviews_corpus[[5]])
```

```
## [1] "Watching this movie made me think constantly; why are they making such a problem out of
some broken brakes? There are a million options to slow down the car! In the movie Speed the wri
ters a least thought of a good reason why the car wasn't able to stop...<br /><br />There aren't
many good things to say about this film; all the usual narrative cliché's make their appearance,
the actors are very bad, the story is as leak as a sieve etc. That makes this movie a waste of t
ime and money.<br /><br />"
```



data preparation for the model

```
# clean the text
movie_reviews_clean <- tm_map(reviews_corpus, content_transformer(tolower))
movie_reviews_clean <- tm_map(movie_reviews_clean, removeNumbers)
movie_reviews_clean <- tm_map(movie_reviews_clean, removeWords, stopwords())
movie_reviews_clean <- tm_map(movie_reviews_clean, removePunctuation)
movie_reviews_clean <- tm_map(movie_reviews_clean, stripWhitespace)
```

```
as.character(movie_reviews_clean[[5]])
```

```
## [1] "watching movie made think constantly making problem broken brakes million options slow c
ar movie speed writers least thought good reason car able stopbr br many good things say film us
ual narrative cliches make appearance actors bad story leak sieve etc makes movie waste time mon
eybr br "
```

```
# Tokenize as document term matrix
movie_reviews_dtm <- DocumentTermMatrix(movie_reviews_clean)
```

Split the data set

```
# split the data set to train the model
movie_reviews_train <- movie_reviews_dtm[1:15000,] # 75%
movie_reviews_test <- movie_reviews_dtm[15001:20000,] # 25%

movie_reviews_train_labels <- movie_reviews[1:15000,]$sentiment # 75%
movie_reviews_test_labels <- movie_reviews[15001:20000,]$sentiment# 25%

# check the distribution of train and test data

prop.table(table(movie_reviews_train_labels))
```

```
## movie_reviews_train_labels
## negative positive
##    0.5056    0.4944
```

```
prop.table(table(movie_reviews_test_labels))
```

```
## movie_reviews_test_labels
## negative positive
##    0.4984    0.5016
```

```
# find terms that appear 5 times or more in the review
review_freq_words_tr <- findFreqTerms(movie_reviews_train, 5)
review_freq_words_ts <- findFreqTerms(movie_reviews_test, 5)

# include only the frequent terms in the training and test data
movie_review_freq_train <- movie_reviews_train[, review_freq_words_tr]
movie_review_freq_test <- movie_reviews_test[, review_freq_words_ts]

# function that will check the presence of the each term
check_term <- function(x){
  x <- ifelse(x> 0, "Yes", "No")
}

review_train <- apply(movie_review_freq_train, MARGIN = 2, check_term)
review_test <- apply(movie_review_freq_test, MARGIN = 2, check_term)
```

Build Model

I used the Naive Bayes algorithm for this task. Naive Bayes classifiers have several advantages for text classification tasks.

Simplicity and Speed: Naive Bayes is a simple and fast algorithm it is computationally efficient. It performs well with large data sets and high-dimensional feature spaces.

Ease of Implementation: it is easy to understand and interpret. it has a relatively low complexity compared to more complex machine learning algorithms.

Good Performance with Small Training Sets: it performs well even with small training data sets.

```

review_model <- naiveBayes(review_train, movie_reviews_train_labels)
review_test_pred <- predict(review_model, review_test)

# create confusion matrix to see the accuracy of the model
CrossTable(review_test_pred,movie_reviews_test_labels,
            prop.chisq = FALSE,
            prop.t = FALSE, dnn = c("pred", "observed"))

```

```

##
##
##      Cell Contents
## |-----|
## |                N |
## |      N / Row Total |
## |      N / Col Total |
## |-----|
##
##
## Total Observations in Table:  5000
##
##
##      | observed
##      | negative | positive | Row Total |
## -----|-----|-----|-----|
## negative |      2142 |      424 |      2566 |
##          |      0.835 |      0.165 |      0.513 |
##          |      0.860 |      0.169 |           |
## -----|-----|-----|-----|
## positive |       350 |     2084 |      2434 |
##          |      0.144 |      0.856 |      0.487 |
##          |      0.140 |      0.831 |           |
## -----|-----|-----|-----|
## Column Total |      2492 |      2508 |      5000 |
##          |      0.498 |      0.502 |           |
## -----|-----|-----|-----|
##
##

```

Our model is very good with better accuracy of 85% with test dataset.