Learning from Big Data: Module 1 - Final Assignment Template

Student name: Alen Gabbassov

9/6/2022

Understanding the audience: Does the movie perform better when the review mentions the movie characters?

Motivation.

Every time people watch a movie, they associate themselves with the characters of these movies rather than the real actor themselves. There are numerous examples of that. As kids, we all wanted to be the main character in our favorite film. Some wanted to be Harry Potter and defeat Lord Voldemort and some people wanted to be Luke Skywalker from Star Wars. But do we really remember the real names of the actors playing these roles? Even when we write reviews, opinions, and blogs we tend to stick to the characters, which are fantasy names, rather than real people who were playing them.

Leonardo DiCaprio is arguable the most famous and influential actor in the Hollywood of the recent century and has recently played in the Netflix movie “Don’t Look up”. On the first weekend, the film was good enough to earn a total of 700.000 US Dollars - 1.5 Mio US Dollars.(varies depending on a different sources,mainly: Wikipedia). Netflix production was generous enough to pay Leonardo 30 Mio. US Dollars for playing the main role, which caused a massive scandal between the following actors/actresses.

On the other side, a well-known cartoon movie probably by all the younger generation “Minions”, which is a sequel of the other famous cartoon movie “Despicable Me”, made approximately 125 Mio. US Dollars for the first weekend. (Source: https://www.boxofficemojo.com/release/rl2271380993/). At least, it was not hard to find someone to voice cast minions since they only produce 10 words for the entire film.

In the assignment, I would like to analyze the importance of mentioning the movie characters either positive or negative in reviews rather than the real actor themselves. If mentioning the movie characters in the reviews section before the release makes the film perform better, then it can be judged, that the scenario-writer (or Movie Studios) should pay as much attention to the easy remembering (or catchy) name of the main character as the plot, production, budgeting or even actors cast itself.

Install Packages

```r
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tm, openNLP, nnet, dplyr, tidyr, ggplot2, reshape2,latex2exp)
```

Load the reviews

```
setwd("~/Desktop/data")
library(readr)
Reviews_Raw <- read.csv("Reviews_tiny.csv")
Reviews_Raw <- Reviews_Raw %>%
  select(movie_name,review_code,    reviewer,   review_date, num_eval,
         prob_sentiment,words_in_lexicon_sentiment_and_review, ratio_helpful,
         raters,
         prob_storyline,    prob_acting,    prob_sound_visual,
         full_text, processed_text,
         release_date,  first_week_box_office,MPAA, studio, num_theaters )
```

Data aggregation and formatting.

Please, see the reference for "mbti.txt" file in Appendix.

1. Loading new training data for charachters and drop unnecessary columns.

For answering our research question leave only the characters and the movie_name column. Then, split the data of the movie characters.

```
mbti <- read.csv("~/Desktop/data/mbti.csv")
mbti_aggregated <- mbti[,-1:-3]
mbti_aggregated <- mbti_aggregated[,-3]
charachters_dictionary <- mbti_aggregated
charachters_dictionary<- charachters_dictionary[1:2000,]
```

Aggregation.

Please, see the reference for "Aggregated Data.xlsx" file in Appendix.

2. Aggregate the review data on the review level.

As for the review data, the reviews that interest us, are only those, that were made before the release date. The reason is that we want eventually predict how the first reviews, or better said, the reviews made before the release of the movie had an influence on the box office. It can be also be simply explained that we can not watch the past.

2.1 First take the data from the Reviews_Raw.csv.

2.2 Filter the reviews on movies.(500 Days of summer / 127 Hours / 2012). Then create the pivot table, where:

2.2 A **Review Data** - date of the review

2.2 B **Number of Reviews on the specific Date** - count of the Reviews on the specific date.

2.2 C **Review_Code for Identifying the review** - we would use it later for filtering the data & computing the regression in the Analysis section. We need to know the Review_id in order to understand which reviews we actually made before the release.

2.2 D Afterwards, we use If-function in order to match the reviews that we made before the release of the movie. Please,see the columns (P),(V),(AB) for the reference.

2.2 E **Column Total number of Reviews before the release** represent the sum of the reviews that were matched to the criteria described above (2.2 D).

Eventually, we can see in the Aggregates.xslx file that there are only 32 out of 1000 reviews that were made before the release of movies. Moreover, there was no review made before the release of the movie "127 Hours" in our Review_raw dataset.

Supervised Learning - the Naive Bayes classifier

```
#  FUNCTIONS.
Compute_posterior_sentiment = function(prior, corpus_in, dict_words,
p_w_given_c,TOT_DIMENSIONS){
  output <-  capture.output
  (word_matrix <-inspect(
    DocumentTermMatrix(corpus_in, control=list(stemming=FALSE,
                                                language = "english",

dictionary=as.character(dict_words)))))

  # Check if there are any relevant words in the review.
  #       If there are, treat them. If not, use prior
  if (sum(word_matrix) == 0) {posterior<-prior ; words_ <- c("")} else
  {
    # Positions in word matrix that have words from this review
    word_matrix_indices <- which(word_matrix>0)
    textual_words_vec   <- colnames(word_matrix)[word_matrix_indices]

    # Loop around words found in review
    WR <- length(word_matrix_indices) ;word_matrix_indices_index=1
    for (word_matrix_indices_index in 1: WR)
    {
      word <-
colnames(word_matrix)[word_matrix_indices[word_matrix_indices_index]]
      p_w_given_c_index <- which(as.character(p_w_given_c$words) == word)

      # Loop around occurrences  | word
      occurrences_current_word=1
      for (occurrences_current_word in 1:
           word_matrix[1,word_matrix_indices[word_matrix_indices_index]] )
      {
        # initialize variables
        posterior     <- c(rep(0, TOT_DIMENSIONS))
        vec_likelihood<-
as.numeric(c(p_w_given_c$pos_likelihood[p_w_given_c_index],

p_w_given_c$neg_likelihood[p_w_given_c_index]))
```

```
        # positive - this is the first element in the vector
        numerat          <-  prior[1] *
          as.numeric(p_w_given_c$pos_likelihood[p_w_given_c_index])
        denomin          <-  prior %*% vec_likelihood
        posterior[1]     <- numerat /  denomin

        # negative - this is the second element in the vector
        numerat          <-  prior[2] *
as.numeric(p_w_given_c$neg_likelihood[p_w_given_c_index])
        denomin          <-  prior %*% vec_likelihood
        posterior[2]     <- numerat /  denomin


        if (sum(posterior)>1.01) { ERROR <- TRUE }
        prior <- posterior
      } # close loop around occurrences
    } # close loop around words in this review
    words_ <-  colnames(word_matrix)[word_matrix_indices]
  } # close if review has no sent words

  return(list(posterior_=posterior, words_=words_) )
}

Compute_posterior_content = function(prior,  word_matrix, p_w_given_c,
BIGRAM, TOT_DIMENSIONS){

  # Check if there are any relevant words in the review.
  # If there are, treat them. If not, use prior
  if (sum(word_matrix) == 0) {posterior<-prior  } else
  {
    # Positions in word matrix that have words from this review
    word_matrix_indices <- which(word_matrix>0)
    textual_words_vec   <- colnames(word_matrix)[word_matrix_indices]

    # Loop around words found in review
    WR <- length(word_matrix_indices) ;word_matrix_indices_index=1
    for (word_matrix_indices_index in 1: WR)
    {
      word <-
colnames(word_matrix)[word_matrix_indices[word_matrix_indices_index]]
      p_w_given_c_index <- which(as.character(p_w_given_c$words) == word)

      # Loop around occurrences  | word
      occurrences_current_word=1
      for (occurrences_current_word in
           1:  word_matrix[1,word_matrix_indices[word_matrix_indices_index]])
      {
        # initialize variables
```

```
        posterior     <- c(rep(0, TOT_DIMENSIONS))
        vec_likelihood <-
as.numeric(c(p_w_given_c$storyline[p_w_given_c_index],
                                      p_w_given_c$acting[p_w_given_c_index],
                                      p_w_given_c$visual[p_w_given_c_index],

p_w_given_c$charachters[p_w_given_c_index]) )

        # storyline - this is the first element in the vector
        numerat         <-  prior[1] *
as.numeric(p_w_given_c$storyline[p_w_given_c_index])
        denomin         <-  prior %*% vec_likelihood
        posterior[1]    <-  numerat /  denomin

        # acting - this is the second element in the vector
        numerat         <-  prior[2] *
as.numeric(p_w_given_c$acting[p_w_given_c_index])
        denomin         <-  prior %*% vec_likelihood
        posterior[2]    <-  numerat /  denomin

        # visual - this is the third element in the vector
        numerat         <-  prior[3] *
as.numeric(p_w_given_c$visual[p_w_given_c_index])
        denomin         <-  prior %*% vec_likelihood
        posterior[3]    <-  numerat /  denomin

        # charachter - this is the fourth element in the vector
        numerat         <-  prior[4] *
as.numeric(p_w_given_c$charachters[p_w_given_c_index])
        denomin         <-  prior %*% vec_likelihood
        posterior[4]    <-  numerat /  denomin

        if (sum(posterior)>1.01) { ERROR <- TRUE }
        prior <- posterior
      } # close loop around occurrences
    } # close loop around words in this review

  } # close if review has no sent words

  return (posterior_= posterior  )
}

# GLOBAL PARAMETERS
PRIOR_SENT  = 1/2
PRIOR_TOPIC = 1/4
TOT_REVIEWS = length(Reviews_Raw[,1])
```

Likelihoods

```
# Computing likelihoods

dictionary_storyline<-read.csv2("~/Desktop/data/acting_33k.txt")
dictionary_accting <-read.csv2("~/Desktop/data/storyline_33k.txt")
dictionary_visual <-read.csv2("~/Desktop/data/visual_33k.txt")
stopword_dictionary <-
read.csv2("https://raw.githubusercontent.com/guiliberali/Learning-from-Big-
Data-Module-1/main/data/stopwords/Sentiment_stopwords.csv")

# Standardize dictionaries by keeping all words in lower caps and eliminating
the stop words.
#Lowering the chars of the first word in each dictionary.

dictionary_storyline <- data.frame(tolower(dictionary_storyline$for.))
dictionary_accting <- data.frame(tolower(dictionary_accting$moth))
dictionary_visual <- data.frame(tolower(dictionary_visual$challenge))
charachters_dictionary <- data.frame(tolower(charachters_dictionary$role))


low_dictionary_storyline <- do.call("paste", dictionary_storyline)
low_dictionary_acting <- do.call("paste", dictionary_accting)
low_dictionary_visual <- do.call("paste", dictionary_visual)
low_stopword_dictionary <- do.call("paste", stopword_dictionary)
low_charachters_dictionary <- do.call("paste", charachters_dictionary)

`%notin%` = Negate(`%in%`)

final_dictionary_storyline <-
data.frame(word_col=dictionary_storyline[low_dictionary_storyline%notin%low_s
topword_dictionary, ])
final_dictionary_acting <-
data.frame(word_col=dictionary_accting[low_dictionary_acting %notin%
low_stopword_dictionary, ])
final_dictionary_visual <-
data.frame(word_col=dictionary_visual[low_dictionary_visual %notin%
low_stopword_dictionary, ])
final_dictionary_charachters <-
data.frame(word_col=charachters_dictionary[low_charachters_dictionary %notin%
low_stopword_dictionary, ])

likelihood_dic_story <-
data.frame(prop.table(table(final_dictionary_storyline$word_col))) #frequency
of a given word over the cardinality of the dictionary
likelihood_dic_act <-
data.frame(prop.table(table(final_dictionary_acting$word_col)))
likelihood_dic_vis <-
data.frame(prop.table(table(final_dictionary_visual$word_col)))
likelihood_dic_charachters <-
```

```
data.frame(prop.table(table(final_dictionary_charachters$word_col)))

all_likelihoods <- merge(likelihood_dic_story, likelihood_dic_act,
by.x='Var1', by.y='Var1', all=TRUE)
all_likelihoods <- merge(all_likelihoods, likelihood_dic_vis, by.x='Var1',
by.y='Var1', all=TRUE)
all_likelihoods <- merge(all_likelihoods, likelihood_dic_charachters,
by.x='Var1', by.y='Var1', all=TRUE)
```

```
## Warning in merge.data.frame(all_likelihoods, likelihood_dic_charachters, :
## column names 'Freq.x', 'Freq.y' are duplicated in the result
```

```
colnames(all_likelihoods) <- c("words", "storyline", "acting",
"visual","charachters")

# Smoothing ( Replace the value of words that only appear in 1 or 2 of the
dictionaries with small, non-zero values)
# Instead of smoothing by adding 1 to all tokens in order to get rid of the
"0" values, we keep the count of tokens and consider the likelihood of
picking a token which results in "NA" as 1 over the number of tokens NA in
the given column.

all_likelihoods$storyline[is.na(all_likelihoods$storyline)] <-
1/sum(is.na(all_likelihoods$storyline))
# Dividing 1 by the number of NA's in each column for the likelihood of
picking an NA; replace NA's by probability.
all_likelihoods$acting[is.na(all_likelihoods$acting)] <-
1/sum(is.na(all_likelihoods$acting))
all_likelihoods$visual[is.na(all_likelihoods$visual)] <-
1/sum(is.na(all_likelihoods$visual))
all_likelihoods$charachters [is.na(all_likelihoods$charachters)] <-
1/sum(is.na(all_likelihoods$charachters))

likelihoods <- all_likelihoods
lexicon_content     <- as.character(likelihoods[ ,1] )
```

Creating Sentiment list

Please, look for "sentiment.xlsx" in Appendix for the reference.

```
# 3. In the "sentiment.xlsx" file we first clearn the data with the Trim and
LOWER function in excel.
#3.1 We then use ABS function in order to make the values absolute.
#3.2 We then Formula for Normalizing values and making them from [0,1]. X =
(x-x(min))/(x(max)-x(min)).

install.packages("readxl", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##
/var/folders/x2/qpdlp4yx3gs61bxl763phq8w0000gn/T//Rtmpw4TVVx/downloaded_packa
ges
```

```
library(readxl)
likelihoods_sentim <- read_excel("~/Desktop/data/Sentiment_NOT_fake.xlsx")
lexicon_sentiment <-  as.character(likelihoods_sentim$words )
```

Run NBC for sentiment

```
for (review_index in 1:TOT_REVIEWS) {
  prior_sent     <- c(PRIOR_SENT,1-PRIOR_SENT)   # Reset the prior as each
review is looked at separately
  text_review    <- as.character(Reviews_Raw$processed_text[review_index])

  # 2.2.A Pre-process the review to removepunctuation marks and numbers.
  #       Note that we are not removing stopwords here (nor elsewhere - a
point for improvement)
  corpus_review  <- tm_map(tm_map(VCorpus(VectorSource(text_review)),
removePunctuation), removeNumbers)

  # 2.2.B Compute posterior probability the review is positive
  TOT_DIMENSIONS = 2
  output <-capture.output(sent.results <- Compute_posterior_sentiment(prior =
prior_sent,
                                              corpus_in  = corpus_review,
                                              dict_words = lexicon_sentiment,
                                              p_w_given_c=likelihoods_sentim,
                                              TOT_DIMENSIONS) )
  words_sent  <- sent.results$words_
  posterior_sent <- sent.results$posterior_
  Reviews_Raw$prob_sentiment[review_index] <- posterior_sent[1]
  Reviews_Raw$words_in_lexicon_sentiment_and_review[review_index] <-
paste(words_sent,collapse =" ")
}
```

Run NBC for content

```
for (review_index in 1: TOT_REVIEWS) {
  if ( Reviews_Raw$full_text[review_index]!=""){
    text_review   <- as.character(Reviews_Raw$processed_text[review_index])

 # 3.3.A Pre-process the review to remove numbers and punctuation marks.
    #       Note that we are not removing stopwords here (nor elsewhere - a
point for improvement)
    corpus_review <- VCorpus(VectorSource(text_review))  # put in corpus
format
```

```
    output <-capture.output(content_word_matrix <-
                inspect(DocumentTermMatrix(corpus_review,
                                           control = list(stemming=FALSE,
                                                          language =
"english",

removePunctuation=TRUE,
                                                          removeNumbers=TRUE,

dictionary=as.character(lexicon_content)))))

    # 3.3.B  Compute posterior probability the review is about each topic
    TOT_DIMENSIONS = 4
    posterior <- Compute_posterior_content(prior=matrix(PRIOR_TOPIC,
ncol=TOT_DIMENSIONS),
                                           content_word_matrix,
                                           p_w_given_c=likelihoods,
                                           TOT_DIMENSIONS)
    Reviews_Raw$prob_storyline[review_index]    <- posterior[1]
    Reviews_Raw$prob_acting[review_index]       <- posterior[2]
    Reviews_Raw$prob_sound_visual[review_index] <- posterior[3]
    Reviews_Raw$prob_charachters[review_index]  <- posterior[4]
  }
}
Processed_reviews <- Reviews_Raw

# Saves the updated file, now including the sentiment and content/topic
posteriors.
write.csv(Reviews_Raw,file="output_Reviews_posteriors.csv" , row.names =
FALSE )

# Saves the updated file as excel for building the Confusion Matrix.
library("writexl")
write_xlsx(Reviews_Raw,"~/Desktop/data\\output_ReviewsRaw2.xlsx")
```

Supervised Learning - Inspect the NBC performance

Load judges scores

```
ground_truth_judges <-read.csv("~/Desktop/data/judges.csv")
```

Compute confusion matrix, precision and recall

Please, see the Appendix for the Confusion Matrix.xlsx file.

From the results, we can clearly see the total accuracy of 81%. However, the total recall, which is the number of samples predicted correctly to the positives class out of all samples that actually belong to the positives is only 22%. The model predicts at best the positive samples for “acting” cluster with 68% recall. The reason for that might be the higher predictability for the “acting” cluster by NBC (about 400/1000 reviews), following the “storyline” (about 300/1000 reviews), and “visionary_sound” (about 300/1000 reviews).

The model works worst for predicting the ” visionary_sound” cluster with only 26% precision and 24% recall. It’s also logical since a lot of people writing their reviews would pay most of their attention evaluating the the skills of the actors (“acting” part) or the plot (“storyline” part) as for the sound of the movie or the quality of the picture.

The solution for improving the overall recall and precision might be increasing the number of training examples, helping the machine learning algorithm to a build more generalizable model.

Confusion Matrix

		Predicted		
	n = 1000	storyline	acting	visionary
Actual	storyline	151	182	116
	acting	60	221	42
	visionary	68	105	55

Summary

	storyline	acting	visionary
TP	151	221	55
TN	423	390	614
FP	128	287	158
FN	298	102	173

Precision	54%	44%	26%
Recall	34%	68%	24%
Accuracy	57,40%	61,10%	66,90%

Total Values

Total TP	445
Total FP	573
Total FN	573
Total TN	4517

Total Precision	44%
Total Recall	22%
Total Accuracy	81%

Analysis

Please, see the Appendix for the Regression.xlsx file For answering our original research question we would measure the “performance” as the logarithm of a number of the box office for the first weekend. It will be also our dependent value (=y). As for the independent values (=x1,x2,x3…), we would take 3 models with different controls. The first one would be only the “acting”, “storyline”, and “visionary” controls. The second one would be the same, but with movie characters as an additional dimension. The third would be just the number of theaters where the film was shown.

Now, we can use our aggregated review data from the “Aggregate.xlsx.” file mentioned earlier. For the regression, we would only use the reviews that are relevant to us. For instance, the reviews that were made before the release of the movie since those reviews that were made after the release have no value for answering the research question.

From the regression table below, we can clearly see the results for our first model. The adjusted R squared is 36% which indicates that acting, story, and visionary are not the perfect fit data to predicting the first week's box office. Although the significance is very low (between 0,0003 and 0,0017), the coefficient of "visionary _sound" as our dependent variable is zero.

Coming closer to our research questions we can see the coefficient of zero from the character's data, which means no relationship between the box office and the characters of the movie. It can be explained with several arguments. Firstly, is that we had only a very small sample of only 32 observations of the reviews made before the release which could lead to such results. Secondly, for the most part, we would not expect the names of the characters to have an influence on the first-week box office. If the main character - Frodo, from the "Lord of Rings", would have been called Bill - It can be viewed skeptical that it would have any influence on the first week box office. The same logic is to apply to all other films.

On the other hand, the third model - *number of theaters* should have shown the significant correlation with the box office for the first weekend. The more theaters present the film, the higher the box office. It also depends on the influence of the movie studio showing the film, since all theaters are dependent on Hollywood movie studios (such as Fox or Sony…).

Although the values in the table are probably not informative to say anything about the numbers of theaters since the P-value is zero(what can't be true), the box office for the first weekend increases in 0,0012 times depending on the number of theaters.

To sum up, there are numerous arguments as the low number of observations as well as more insightful controls that would lead us to the higher correlation with the box office. However, it can not be claimed that mentioning the characters in the reviews, before the release of the movie, would have no influence on the box office at all. With more sufficient data, time, and research the opposite argument can be possibly proven.

Regression coeffecients for predicting the performance of the mentioned charachters in the review before the movie release.

Model	Coefficients	R^2	(adjusted) R^2	p - Values	Sig.	*95 % Confidency Interval for B*
1. Probabilities _ act/story/vis.	17,81973308	0,3640	0,2882		0,0003	
acting	-2,241058368			0,51276719		
story	0,634570701			0,00804517		
visionary	0,00			#NUM!		
2. Probabilities _ act/story/vis /char.	17,81973308	0,3730	0,2736		0,0017	
acting	-2,241058368			0,00875426		
story	0,634570701			0,51703042		
visionary				#NUM!		
charachters				#NUM!		
3. Theaters.	14,10039094	1	1	0	0	
number of theathers	0,00120988			0		

a. Dependent Variable : Box office for the first week
b. Number of Observations = 32

APPENDIX

1. Training data for movie charachters: https://www.kaggle.com/datasets/subinium/movie-character-mbti-dataset?resource=download

2. Aggregation part Excel file: https://github.com/Shmalen/Big-Data-Files-for-Ass.1

3. Sentiment dictionary: Stanford SocialSent. File name “2000.tsv”. https://www.tutorialexample.com/a-full-list-of-sentiment-lexicons-for-sentiment-analysis-to-download-machine-learning-tutorial/

4. Sentiment Excel file: https://github.com/Shmalen/Big-Data-Files-for-Ass.1

5. Confusion Matrix & Regression Excel file: https://github.com/Shmalen/Big-Data-Files-for-Ass.1

6. Regression Excel file: https://github.com/Shmalen/Big-Data-Files-for-Ass.1

7. I didn’t have time on adjusting the Regression Table.png file (making it bigger).