# Learning from Big Data: Module 1 - Final Assignment Template

Student name: .....

9/6/2022

## Introduction

This file provides a template for assignment 1 of the Learning from Big Data course. This file was prepared to save you time so that you can focus on the theory and technical parts of the methods seen in class. This was prepared with a specific application in mind: movie reviews. For the supervised learning tasks, we will focus on three topics: acting, storyline and visual/sound effects.

You have by now received the dataset of reviews, the three dictionaries with the training set of words for each topic, a list of stopwords and a validation dataset containting sentences classified by a panel of human judges. This R markdown file has lot of code created to handle things such as loading these data files and the general settings of the environment we use to run the analysis. The supervised learning code in this file was covered in Session #2.

This R markdown file will upload all the above files and make them available for you to use them when solving the NLP problems listed here. The questions you are to complete are marked as "QUESTION". The parts you are expected to add your code are marked as "# YOUR CODE HERE". There, you are expected to write your own code based on the in-class discussions and the decisions you will make as you study the theory, materials, and models.

This tutorial has the following structure:

1. **General Guidelines**
2. **Research Question**
3. **Load libraries**
4. **Load the reviews**
5. **Data aggregation and formatting**
6. **Supervised Learning - The Naive Bayes classifier (NBC)**
7. **Supervised Learning - Inspect the NBC performance**
8. **Unsupervised Learning - Predict Box office using LDA**
9. **Unsupervised Learning - Predict Box office using Word2Vec**
10. **Analysis - answering the research question**
11. **OPTIONAL - run and interpret the VADER lexicon for sentiment**
12. **Appendix**

Before going further: check that you have R, R markdown and tinytex correctly installed. Tutorial 0 provides instructions for doing so.

# 1. General Guidelines

Page length. The template has 8 pages. You are allowed to add 7 to 10 pages, not including the appendix. There is a limit of pages, but you have the possibility of using appendices, which are not limited in number of pages. Use your pages wisely. For example, having a table with 2 rows and 3 columns that uses 50% (or even 25%) of a page is not really wise.

# 2. Research Question

QUESTION I. Present here the main research question you are going to answer with your text analysis. You are free to choose the problem and change it until the last minute before handing in your report. However, your question should not be so simple that it does not require text analysis. For example, if your question can be answered by reading two reviews, you do not need text analysis; all you need is 10 seconds to read two reviews. Your question should not be so difficult that you cannot answer in your report. Your question needs to be answered in these pages.

# 3. Loading libraries

Before starting the problem set, make sure you have all the required libraries properly installed. Simply run this chunk of code below.

```
# Packets required for subsequent analysis. P_load ensures these will be installed and loaded.
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tm, openNLP, nnet, dplyr, tidyr, ggplot2, reshape2,latex2exp)
```

# 4. Load the reviews

We will explore the concepts from this problem set with a dataset of online movie reviews. The reviews were written from 2009 to 2013. The data was collected in 2014. Each observation is a movie review. Each observation in the data includes the textual review, a numerical rating from 1 to 10 (i.e., the number of stars), the movie title, the reviewer and the date the review was written. The observation includes data from the movie being reviewed: the movie release date, the box office in the first week (as that is the strongest predictor of movie success), the studio that produced the movie, the number of teathers that the movie was released and the MPAA rating. The review also includes two pieces of information on the quality of the review itself: the number of readers who found the review useful, and the number of readers who rated the review as useful or not useful. There are reviews that non one rated as useful or not useful. The date in which a review was rated is not available.

The data set contains 19 columns.

- movie name: title of the movie being reviewed
- review code: a unique serial identifier for all reviews in this dataset
- reviewer: the reviewer who wrote the review
- num eval: the number of stars
- review date: the date the review was writte
- prob sentiment: a placeholder variable to store the probability the review is positive. This is to be computed by you.

- words in lexicon sentiment and review: the number of words that are found both in the review and in the sentiment lexicon you will be using
- ratio helpful: number of people that rated the review as useful divided by the total number of people that rated the review
- raters: number of people that rated the review as either useful or not useful
- prob storyline: a placeholder variable to store the probability the review is about the movie storyline.
- prob acting: a placeholder variable to store the probability the review is about acting.
- prob sound visual: a placeholder variable to store the probability the review is about the movie special effects (sound or visual)
- full text: raw review text
- processed text: the cleaned review text, free of punctuation marks.
- release date:day the movie was released.
- first week box office: number of movie teather tickets sold in the first week from movie release. Data from boxofficemojo.com
- MPAA: MPAA rating of the movie (e.g., PG-rated)
- studio: movie studio that produced the movie.
- num theaters: number of movie theaters that this movie was shown on the release date. Data from boxofficemojo.com

```
Reviews_Raw <- read.csv('../data/reviews/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 )
```

# 5. Data aggregation and formatting.

QUESTION II. Decide on how to aggregate or structure your data. The data you received is at the review level (i.e., each row is a review). However, the variables in the data are very rich and allow you to use your creativity when designing your research question. For example, there are timestamps, which allow you to aggregate the data at the daily level or even hourly level. There is information on reviewers, which allow you to inspect patterns of rating by reviewers. There is information on studios, and more. Please explicitly indicate how you structured your dataset, and what is your motivation to do so. Even if you are using the data at the review level, indicate how and why that is needed for your research question.

# 6. Supervised learning - the Naive Bayes classifier

## 6.0 Load support functions and global parameters

These two functions, Compute_posterior_sentiment and Compute_posterior_content, are called one time per review. These functions use the Bayes rule we saw in our session #2 to compute the posterior probabilities

that the review is about each topic (in the 2nd function) and the posterior probability that the sentiment in the review is positive and/or negative (in the 1st function).

```
#  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]) )

        # 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

        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 = 1/3
TOT_REVIEWS = length(Reviews_Raw[,1])
```

## 6.1 Likelihoods

QUESTION III. Create the content likelihoods based on the 3 lists of words below. Be explicit on the decisions you took in the process, and why you made those decisions (e.g., which smoothing approach you used).

```
dictionary_storyline<-read.csv2("../data/lexicons/storyline_33k.txt", header = FALSE)
dictionary_acting <-read.csv2("../data/lexicons/acting_33k.txt", header = FALSE)
dictionary_visual <-read.csv2("../data/lexicons/visual_33k.txt", header = FALSE)

# ADD YOUR CODE HERE, replacing these fake likelihoods
likelihoods <- read.csv2("../data/lexicons/example_100_fake_likelihood_topic.csv", header=TRUE,
                         sep = ",",quote="\"",dec=".",fill=FALSE)
likelihoods <- likelihoods[,1:4]
lexicon_content     <- as.character(likelihoods[ ,1] )
```

QUESTION IV. Locate a list of sentiment words that fits your research question. For example, you may want to look just at positive and negative sentiment (hence two dimensions), or you may want to look at other sentiment dimensions, such as specific emotions (excitement, fear, etc.).
TIP: Google will go a long way finding these, but do check if there is a paper you can cite that uses your list.

```
# ADD YOUR CODE HERE, replacing these fake likelihoods
likelihoods_sentim <-read.csv2("../data/lexicons/example_100 fake_likelihood_sentiment.csv",
                               header=TRUE, sep=",", quote="\"",dec=".",fill=FALSE)
lexicon_sentiment <-  as.character(likelihoods_sentim$words )
```

## 6.2 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 separate
  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 =" ")
}
```

## 6.3 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 = 3
    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]
  }
}
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 )
```

# 7. Supervised Learning - Inspect the NBC performance

## 7.1 Load judges scores

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

## 7.2 Compute confusion matrix, precision and recall

QUESTION V. Compare the performance of your NBC implementation (for content) against the judges ground truth by building the confusion matrix and computing the precision and accuraccy scores. Do not forget to interpret your findings.

```
# YOUR CODE HERE
```

# 8. Unsupervised Learning: Predict box office using LDA

QUESTION VI. Using LDA, predict movie box office. Tip: You can pass a list of reviews to the LDA package, in order to get the posterior probability the reviews are about each topic. If you pass them all in a single document, you wil not get review-specific vectors.

```
# YOUR CODE HERE.You are allowed to use the code from session #3.
```

# 9. Unsupervised Learning: Predict box office using Word embeddings given by Word2Vec

QUESTION VII. Using Word2Vec, predict movie box office. Tip 1. you can reduce the dimensionality of the output of word2vec with PDA/Factor analysis. This will save you computing time. Tip 2. Word2Vec will give you word vectors. You can then compute the average of these word vectors for all words in a review. This will give you vector describing the content of a review, which you can use as your constructed variable(s).

```
# YOUR CODE HERE. You are allowed to use the code from session #3.
```

# 10. Analysis - Use the constructed variables to answer your research question

QUESTION VIII. Now that you have constructed your NLP variables for sentiment and content using supervised and unsupervised methods, use them to answer your original research question.

```
# YOUR CODE HERE
```

# OPTIONAL: run and interpret sentiment with the supervised learning VADER lexicon

QUESTION IX (optional). Using the VADER code you received in the lecture, compute the sentiment using the VADER package/lexicon. Compare the performance of your NBC implementation (for sentiment) assuming that the VADER classification were the ground truth and then build the confusion matrix and computing precision and recall. Note that we are now interested in understanding how much the two classifications differ and how, but we are not implying that VADER is error-free, far from it. We are interested in uncovering sources of systematic differences that can be attributed to the algorithms or lexicons. Do interpret your findings.

```
 # YOUR CODE HERE
```

# APPENDIX