Learning from Big Data: Module 1 - Natural Language Processing

Student name: . . . .

7/18/2022

Introduction

This tutorial includes the Natural Language Processing problems to be solved in the course Learning from Big Data. This tutorial 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 are now receiving 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.

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 parts you are expected to solve are marked with a "TO DO". 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. **Load the reviews**
2. **Supervised learning - here you will implement the Nayve Bayes classifier (NBC) for sentiment and content**
3. **Supervised learning - inspect the performance of your NBC implementation**
4. **Supervised learning - run and interpret the VADER lexicon for sentiment**
5. **Supervised learning - compare NBC and VADER**
6. **Unsupervised learning - run and interpret LDA to assess the content being discussed in the reviews**
7. **Analysis - use the constructed variables to answer your research question**

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

Loading libraries

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

```r
# 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)
```

1. Load the reviews

We will explore the concepts from this problem set with a dataset of online movie reviews. This dataset contains 10 thousand 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

```r
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    )

# GLOBAL PARAMETERS
PRIOR_SENT  = 1/2
PRIOR_TOPIC = 1/3
TOT_REVIEWS = length(Reviews_Raw[,1])
```

2. Supervised learning classifier - implement the Naive Bayes classifier

2.0 Load support functions and global parameters

```
# YOUR FUNCTIONS
Compute_posterior_sentiment = function(prior, review_corpus, likelihoods,TOT_DIMENSIONS )
{
  # TO DO: write your own function that takes in a prior, a review, and the likelihoods and
  #        compute the posterior probability of the review being positive. Do not forget to clearly
  #        explain each step taken and motivating the decisions you took in the process.



  return(list(posterior_=posterior_sentiment ) )
}

Compute_posterior_content = function(prior,  review_corpus, likelihoods,  TOT_DIMENSIONS)
{
  # TO DO: write your own function that takes in a prior, a review, and the likelihoods and
  #        compute the posterior probability of the review discussing storyline, acting, and visual
  #        effects.Do not forget to clearly explain each step taken and motivating the decisions
  #        you took in the process.


  return (posterior_= posterior_content  )
}
```

2.1 Creating lexicons

```
# TO DO:  create the lexicons based on the lists of words below. Do not forget to clearly explain each
#         step taken and motivating the decisions you took in the process.
dictionary_storyline<-read.csv2("../data/lexicons/storyline_33k.txt")
dictionary_accting <-read.csv2("../data/lexicons/acting_33k.txt")
dictionary_visual <-read.csv2("../data/lexicons/visual_33k.txt")
```

2.2 Run NBC for sentiment

```
# TO DO: write your code to compute the sentiment of the reviews in the dataset.
#   Tip: loop over the reviews and, for each review, call your function Compute_posterior_sentiment()
#        that returns the posterior probability of a review being positive. Do not forget to interpret
#        your findings per the report structure given in class
```

2.3 Run NBC for content

```
# TO DO: write your code to compute the probability of each review discussing each of the three content
#        dimensions (acting, storyline and visual effects).
#   Tip: loop over the reviews and, for each review, call your function Compute_posterior_content()
#        that returns the posterior probability of a review being about one of the dimensions (i.e., a
#        topic. Do not forget to interpret your findings per the report structure given in class


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

3. Supervised Learning - inspect the performance of your (content) naive Bayes classifier

3.1 Load judges scores

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

3.2 Compute confusion matrix, precision and recall

```
# TO DO:  Compare the performance of your NBC implementation (for content) against the
#         judges ground truth  by building the confusion matrix and computing the precison
#         and accuraccy scores.
#         Do not forget to interpret your findings.

# SOLUTION
   # GRETA - can you please write this up, so we can use as a comparison when grading?
```

The above steps were based on manually performing several computational steps in our Naive Bayes classifier. This was important so that you understand what is happening 'under the hood' of classification models. Next, we will use more automated procedures for classification. More specifically, we will first perform a sentiment analysis using the package VADER and look at how to assess the quality of the classification by VADER and by the NBC, given our training data.

4. Supervised Learning - run and interpret sentiment with the VADER lexicon

```
# TO DO: Compute the sentiment using the VADER package/lexicon. In your implementation,
#        do not forget to  indicate, motivate, and explain your decisions regarding
#        parameters used.  Do not forget to interpret the results and possible
#        reasons for differences.

# SOLUTION (confidential - not to be posted, shared nor given to students):

   # GRETA: Can you please code this using the package VADER (e.g.,
   #        https://cran.r-project.org/web/packages/vader/index.html) and the reviews loaded above?
   #        it is just looping over the reviews and for each review call the VADER package and
   #        storing the classification for each review
```

5 Supervised Learning - compare VADER and NBC

```
# TO DO:  Compare the performance of your NBC implementation (for sentiment) assuming that the VADER
#         classification were the ground truth and then building the confusion matrix and computing
#         precision and recall. Note that we are now interested in understanding how much the two
#         classifications (we are not implying that VADER is error-free, far from it). We are intereste
#         in uncovering sources of systmatic differences that can be attributed to the algorithms or
#         lexicons. Do interpret your findings.

# SOLUTION
   # GRETA - can you please write this up, so we can use as a comparison when grading?
```

6. The Unsupervised Learning: run and interpret LDA

```
# TO DO:  Using the LDA package of your preference, identify the topics these reviews are discussing
#         without having labeled the topics upfront i.e., in an unsupervised way.

# SOLUTION:

   # GRETA: Can you please run the reviews through the LDA function in the topicmodels package/library
   #        use the output as a baseline (not ground truth, though)  that will be used to grade student
   #        You can see two-page example of using this package in the MSc Thesis by Donna (two-page app
   #        in Github. There are several packages out there but I figured this documentation would make
   #        easier for you in case you never ran a LDA model
```

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

```
# TO DO: now that you have constructed your NLP variables for sentiment and content, use them to answer
#        original research question per the report template. TIP: Often, it requires reshapoing the dat
#         by changing the observations level (e.g., from 1-observation-is-one-review to 1-observation-is
#         or others). Note also that you camn benefit from having complete timestamps, review rating and
#         characteristics, including box office. These give you lots of food for tought to come up with
#         interesting research questions. You are free and encouradged to search and use extra informati
#         augment your dataset, e.g., by adding the movie genre or competing blockbusters that were rele
#         the same day as these movies.

# SOLUTION (confidential - not to be posted, shared nor given to students):
```