Learning from Big Data: Tutorial 1.2

Tayyip Altan

September 2025

Introduction

In this tutorial, we explore unsupervised learning techniques used in NLP, which do not require labeled data. Specifically, we will cover Latent Dirichlet Allocation (LDA), a topic modeling method used to uncover abstract topics within text, and Word2Vec (W2V), which is a technique for embedding words into vectors. These methods will be applied to extract insights from review data. We will also take initial steps toward using Word2Vec embeddings in a predictive model to forecast movie box office performance.

1. Loading libraries

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

```r
# Required packages. 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, topicmodels,word2vec,tokenizers)
```

2. Load the reviews

```r
# Load the review data. Note that we are now using the fileEncoding parameter when
#      calling read.csv() - this helps reading the review text correctly for further
#      processing (by correctly interpreting the non-ASCII symbols).
#      The ISO-8859-1 encoding is used to represent the first 256 uncicode characters
reviews_raw <- read.csv('Reviews_tiny.csv', fileEncoding="ISO-8859-1")

# Selecting only the relevant columns from the entire dataset
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 )

# Determining the total number of reviews in our dataset
```

```
total_reviews <- nrow(reviews_raw)

# Loading fake likelihoods data
likelihoods <- read.csv("example_100_fake_likelihood_topic.csv")
```

Inspect list of words to be passed to LDA:

```
# set out lexicon equal to the first column of the likelihoods data and inspecting its structure
lexicon_content  <- as.character(likelihoods[ ,1] )
str(lexicon_content)
```

3. Unsupervised Learning: Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is an unsupervised learning technique used for discovering topics within a collection of documents or texts. We apply this topic modeling technique in this context to get an understanding of the main topics people are talking about in their reviews.

```
 # Put processed reviews in corpus format
corpus_review <- VCorpus(VectorSource(reviews_raw$processed_text))

# Creates the document term matrix that will be passed to the LDA function
dtm <- DocumentTermMatrix(corpus_review,
                       control = list(stemming = FALSE,  # Which is also the default
                                      language = "english",
                                      removePunctuation = TRUE,
                                      removeNumbers = TRUE,
                                      dictionary = as.character(lexicon_content)) )

# Inpsecting the structure of the DocumentTermMatrix
str(dtm)
```

```
## List of 6
##  $ i       : int [1:15936] 1 1 1 1 1 1 1 1 1 1 ...
##  $ j       : int [1:15936] 12 14 22 23 32 36 39 40 43 48 ...
##  $ v       : num [1:15936] 1 1 1 1 1 2 1 1 4 1 ...
##  $ nrow    : int 1000
##  $ ncol    : int 100
##  $ dimnames:List of 2
##   ..$ Docs : chr [1:1000] "1" "2" "3" "4" ...
##   ..$ Terms: chr [1:100] "action" "also" "anatomy" "argument" ...
##  - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
##  - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

```
# Useful info on the DocumentTermMatrix and its parameters
?DocumentTermMatrix()
```

```
## starting httpd help server ... done
```

```
?termFreq()   # more info on the control settings used within the dtm
```

Next, we will set the LDA parameters. k is the number of topics we ask LDA to estimate. In supervised learning, we set that to 3. In this example, we arbitrarily set k at 10 to obtain 10 topics. Seed is for replicability (i.e., obtain the same random number every time the code is run). Burn-in and number of iterations are for the convergence of the Markov chains in the Gibbs sampler (MCMC-based inference is outside of the scope of this course and not required for the assignment - just use the default values below.) In the unlikely case you have warnings re: not convergence, you can increase ITER to 2k or 4k.

```
# LDA parameters
seed <- 2
burnin <-  2000
iter <- 1000
k <- 10
```

Next, we will run the LDA and save the model. The model produced by LDA() is an object of class LDA (page 11 of https://cran.r-project.org/web/packages/topicmodels/topicmodels.pdf). This class includes the topics, the log-likelihood, and a lot more. To extract this information, it is needed to use functions listed in page 2 of the said pdf, as shown in the code below.

```
#Create an LDA model using GIBBS sampling
model_lda <- LDA(dtm, k, method = "Gibbs", control = list(seed = seed, burnin = burnin, iter = iter) , 

save(model_lda , file = paste("LDA_model_" ,k,".RData" ,sep=""))
```

Inspect posteriors.

```
#posterior probabilities per document by topic
posteriors_lda <- posterior(model_lda)$topics
str(posteriors_lda)
```

```
##  num [1:1000, 1:10] 0.1139 0.1158 0.0909 0.1134 0.1061 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1000] "1" "2" "3" "4" ...
##   ..$ : chr [1:10] "1" "2" "3" "4" ...
```

```
posteriors_lda[999,]
```

```
##          1          2          3          4          5          6          7 
## 0.09230769 0.09230769 0.12307692 0.10769231 0.07692308 0.09230769 0.07692308 
##          8          9         10 
## 0.09230769 0.16923077 0.07692308
```

Additional challenge: choosing which k to use in LDA is a model selection problem. Typically, the best approach is to compute the LDA model like we have done above for each level of k, save the model log-likelihood (produced by applying the function logLik() to the model, stored in model_lda) and choosing the k that produced the highest log-likelihood.

4. Unsupervised Learning: word embeddings

Word embeddings are an unsupervised learning technique that convert words into numerical vectors, enabling machine learning models to process texts. In this vector space, words with similar meanings (e.g., happy, joyful) are located close to each other, capturing semantic relationships between them.

Our word embedding example has three steps. First, run word2vec to train a model using the training data split. Second, it uses the trained model to analyze the prediction data split. Third, it uses the constructed variables to forecast box office.

Step 1 - Training Step

```r
# Obtain the column with the reviews and convert it to lower case
x <-  reviews_raw$full_text
x <- tolower(x)

# TODO: use a split of the data here (say 50%) instead of the entire dataset

# number of topics in Word2Vec
total_topics_word2vec <- 10

# Train the w2v model
model <- word2vec(x = x, type = "cbow", dim = total_topics_word2vec, iter = 20)
embedding <- as.matrix(model)
```

Step 2 - Construct variables from word embeddings

Similar to tutorial 1.1, we loop over all reviews. For a refresher on for loops in R you can visit: https://www.dataquest.io/blog/for-loop-in-r/. If you'd like more practice with for loops, try exercises 1, 3, 6, 9, 14, and 18 from https://www.w3resource.com/r-programming-exercises/control-structure/index.php.

```r
# TODO: Use the other split of the data here (say 50%) instead of the entire dataset

# Create an empty matrix to store the posteriors
posteriors_w2v <- matrix(0, nc = total_topics_word2vec, nr = total_reviews)

# Loop over all reviews
for (k in 1:total_reviews)
{
   # 2.1 get a review and tokenize it - identify the words, separately
   tokenized_review <- unlist(strsplit(reviews_raw$full_text[[k]],"[^a-zA-Z0-9]+"))

   # 2.2 get the word vectors per review using predict()
   embedding_review <- predict(model, tokenized_review, type = "embedding")

  # 2.3 compute mean across all words for each column in the review
  #     By setting na.rm = TRUE, we ignore the missing values
   posteriors_w2v[k,] <- apply(embedding_review, 2, mean, na.rm=TRUE)
}
```

Tip: for the above data splits, mind the time. Best to train in a split that temporarily precedes the prediction split

Step 3 - Use the constructed variables to forecast

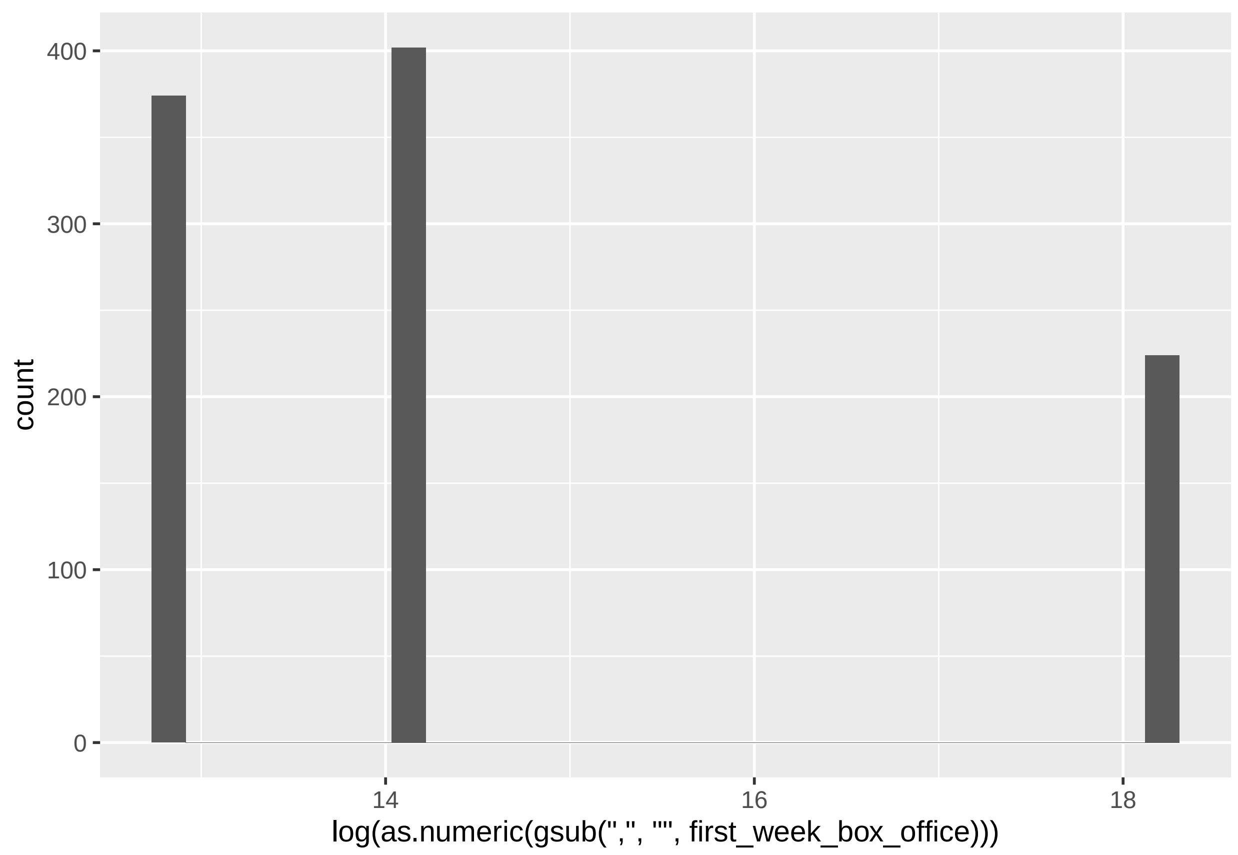
```r
# prepare the constructed variables for analysis
log_BO   <- log(as.numeric(gsub(",", "",reviews_raw$first_week_box_office)))
data_reg <- cbind(c(log_BO), posteriors_w2v)
colnames(data_reg) <- c("LogBoxOffice", paste("w2v_", as.character(1:total_topics_word2vec), sep=""))

# forecast
w2v_BO_lm<- lm(LogBoxOffice ~ posteriors_w2v,  data=as.data.frame(data_reg))
summary(w2v_BO_lm)
```

```
##
## Call:
## lm(formula = LogBoxOffice ~ posteriors_w2v, data = as.data.frame(data_reg))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4270 -0.9475 -0.1007  0.8966  4.1113
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         5.7622     1.3282   4.339 1.58e-05 ***
## posteriors_w2v1     4.3238     0.6929   6.240 6.47e-10 ***
## posteriors_w2v2    -7.0284     0.7293  -9.637  < 2e-16 ***
## posteriors_w2v3    -7.2344     0.9927  -7.287 6.46e-13 ***
## posteriors_w2v4    -8.9536     1.1802  -7.586 7.59e-14 ***
## posteriors_w2v5    -2.8960     1.1141  -2.600  0.00947 **
## posteriors_w2v6    -6.3096     1.0488  -6.016 2.51e-09 ***
## posteriors_w2v7     1.0883     0.9116   1.194  0.23283
## posteriors_w2v8     5.6213     0.4098  13.716  < 2e-16 ***
## posteriors_w2v9    -5.0190     0.5829  -8.610  < 2e-16 ***
## posteriors_w2v10   -2.6435     0.6327  -4.178 3.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.378 on 989 degrees of freedom
## Multiple R-squared:  0.5517, Adjusted R-squared:  0.5472
## F-statistic: 121.7 on 10 and 989 DF,  p-value: < 2.2e-16
```

```r
ggplot(data = reviews_raw, aes(x = log(as.numeric(gsub(",", "",first_week_box_office))))) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
# write.csv(data_reg, "data_reg.csv")
```