Learning from Big Data: Module 1 - Session 3

9/9/2022

Introduction

This file illustrates LDA and word2vec using the review data. Updated on 12 September at 10am.

1. Loading libraries

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

```
# 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

```
# 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)
Reviews_Raw <- read.csv('../../data/reviews/Reviews_tiny.csv', fileEncoding="ISO-8859-1")
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 )
TOT_REVIEWS = length(Reviews_Raw[,1])


# TO DO - replace here these fake likelihoods with your own content likelihood file
likelihoods <- read.csv2("../../data/lexicons/example_100_fake_likelihood_topic.csv",
                         header=TRUE, sep = ",",quote="\"",dec=".",fill=FALSE)[,1:4]
```

Inspect list of words to be passed to LDA:

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

```
##  chr [1:100] "story" "hero" "world" "character" "moral" "audience" ...
```

3. Unsupervised Learning: Latent Dirichlet Allocation (LDA)

```
 # Put 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,
                                      language = "english",
                                      removePunctuation=TRUE,
                                      removeNumbers=TRUE,
                                      dictionary=as.character(lexicon_content)) )


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"
```

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. 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 convergence of the Markov chains in the Gibbs sampler (ask me for details if you want them - 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=20080809
BURNIN = 1000
ITER = 1000
k =10
```

Tip: choosing which k to use in LDA is a model selection problem. Typically, the best approach is to compute a model 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.

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 these infos 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) , m
save(model_lda , file = paste("../../output/LDA_model_" ,k,".RData" ,sep=""))
```

Inspect posteriors.

```r
#posterior probabilities per document by topic
posteriors_lda=posterior(model_lda)$topics
str(posteriors_lda)
```

```
##  num [1:1000, 1:10] 0.1392 0.1158 0.0545 0.0515 0.0909 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1000] "1" "2" "3" "4" ...
##   ..$ : chr [1:10] "1" "2" "3" "4" ...
```

```r
posteriors_lda[review=999,]
```

```
##          1          2          3          4          5          6          7
## 0.07692308 0.10769231 0.09230769 0.10769231 0.15384615 0.07692308 0.10769231
##          8          9         10
## 0.10769231 0.07692308 0.09230769
```

Tip: for the data splits, if you can, mind the time. Best to train on a split that temporarily precedes the prediction split but sometimes that is not viable. Good to be aware anyhow.

4. Unsupervised Learning: word embeddings

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
x <-  Reviews_Raw$full_text
x <- tolower(x)

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

# number of topics in Word2Vec
TOT_TOPICS_WORD2VEC <- 10

# Train
model<- word2vec(x = x, type = "cbow", dim = TOT_TOPICS_WORD2VEC, iter = 20)
embedding <- as.matrix(model)
```

Step 2 - Construct variables from word embeddings

```r
# TO DO: Use the other split of the data here  (say 50%) instead of the entire dataset
all_embeddings = matrix(0, nc=TOT_TOPICS_WORD2VEC, nr=TOT_REVIEWS)
for (k in 1:TOT_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 in the review
   all_embeddings[k,] = apply(embedding_review, 2, mean, na.rm=TRUE)
}
```

Inspect embeddings

```
# word embeddings per document by topic (these are not probabilities)
str(all_embeddings)
```

```
##  num [1:1000, 1:10] -0.0927 -0.2348 -0.1865 -0.299 -0.1573 ...
```

```
all_embeddings[review=999,]
```

```
##  [1] -0.10804513 -0.28280228  0.01978119 -0.18138249 -0.25284248  0.07896421
##  [7]  0.23168891  0.08062798 -0.45942000 -0.10963170
```