Chapter 7 exercises

Anna Wohlmann

2025-01-07

1) See my example in PDF format.
2) In Chapter 6, you were asked to create word embeddings. Improve that code based on the new information on word embeddings you learned in this chapter.

For this task, we need to load the data and preprocessing from Chapter 6:

```r
library(quanteda)
```

```
## Warning: Paket 'quanteda' wurde unter R Version 4.2.3 erstellt

## Package version: 3.3.1
## Unicode version: 13.0
## ICU version: 69.1

## Parallel computing: 12 of 12 threads used.

## See https://quanteda.io for tutorials and examples.
```

```r
datadir <- "./data_exercises"

load(paste0(datadir, "\\supercorpus_unstemmed_V4.RData")) #load where you saved it
library(word2vec)
```

```
## Warning: Paket 'word2vec' wurde unter R Version 4.2.3 erstellt
```

```r
clean_doc <- txt_clean_word2vec(corpus_unstemmed[["documents"]][["texts"]], ascii = TRUE, alpha = TRUE,
```

```r
head(clean_doc, n =2)
```

```
## [1] "today is a little like a change of the guard the outgoing chairman of the european council herma
## [2] "during his speech guy verhofstadt mentioned a letter by bernard foccrouille the director of the
```

Now, we need to make decisions based on the data and our task. Waiting long for our test model to load does not sound fun, so we use the faster "cbow" algorithm. We don't want to overfit and reduce time efforts, but we do have quite a large corpus here; therefore, we initially go with 50 dimensions. The bigger the context window, the more calculations are required. We use a context window of 5 words for cbow, because our text has some complexity. Next, we need to choose the number of iterations. The default of iterations is 5. Tt is probably a good idea to compare 5, 15, and 50 iterations; let us choose 15 initially. Next, we set alpha, which word2vec calls "lr" which we keep at 0.05. We keep negative sampling and the sampling value as is. For the minimum occurrence of words, we use 3 (min_count). We already removed stopwords in the clean_doc, so we do not add a stopword vector here. Of course, we choose the encoding "UTF-8".

```r
chosen_model <- word2vec(
  clean_doc,
  type = "cbow",
  dim = 50,
  window = 5L,
  iter = 15,
  lr = 0.05,
  hs = FALSE,
  negative = 5L,
  sample = 0.001,
  min_count = 3,
  #  stopwords = character(),
  threads = 1L,
  encoding = "UTF-8"
)
```

You can check the tasks from exercise 6 and see if they changed:

```
set.seed(5)
predict(chosen_model, "ideology", type = "nearest", top_n = 6)
```

```
## $ideology
##      term1       term2 similarity rank
## 1 ideology  conspiracy  0.8969203    1
## 2 ideology   worldview  0.8736076    2
## 3 ideology       enemy  0.8730977    3
## 4 ideology  intolerant  0.8608248    4
## 5 ideology      hatred  0.8605962    5
## 6 ideology ideological  0.8588906    6
```

Now we have a first model. In a next step we would evaluate this by looking at models with different hyperparameter values. We want to do this initial check based on our task, data and literature knowledge so we don't run an unnecessary number of models. You will do the evaluation step in exercise 3.

3) In Chapter 6, you were asked to replicate the LDA model based on party manifestos. Use the code from this chapter to tune hyperparameters and evaluate your model.

We start with the model from Chapter 6: You can either just run the Chapter 6 code or save it via:

```
library(topicmodels)
```

```
## Warning: Paket 'topicmodels' wurde unter R Version 4.2.3 erstellt
```

```
#load dataframe:
load("data_exercises/Austriadf.RData")
corpus_manifestos <- corpus(manifestos_df) #turn dataframe into corpus
corpus_sent <- corpus_reshape(corpus_manifestos, to = "sentences")
dfm_sent_aus <-
  quanteda::dfm(corpus_sent %>%
                quanteda::tokens(
                  remove_punct = TRUE,
                  remove_numbers = TRUE,
                  remove_symbols = TRUE,
                  remove_url = TRUE)) %>%
    quanteda::dfm_remove(stopwords("de")) %>%
  quanteda::dfm_wordstem(language = "de")
#dfm to dtm

dtm_aus <- convert(dfm_sent_aus, to = "topicmodels")

dtm_aus
```

```
## <<DocumentTermMatrix (documents: 5742, terms: 13193)>>
## Non-/sparse entries: 60620/75693586
## Sparsity           : 100%
## Maximal term length: 39
## Weighting          : term frequency (tf)
```

We have already evaluated k, so we will keep 15 topics here. We now want to choose further hyperparameters. Let us start with one initial model again. Besides our input data and k, we need to choose the sampling algorithm we test both here. For alpha, we keep the recommended 0.5, and we look at different burnin and iter values for Gibbs.

Here are models with different hyperparameters we can test:

```
?topicmodels::LDA
```

```
## starte den http Server für die Hilfe fertig
```

```
set.seed(50)
initial_model <- topicmodels::LDA(dtm_aus, k = 15, method = "Gibbs", control = list(alpha = 0.5, burnin

vem_model <-topicmodels::LDA(dtm_aus, k= 15, method = "VEM", control = list(alpha = 0.5))

ch6_model <- topicmodels::LDA(dtm_aus, k = 15, method = "Gibbs", control = list(alpha = 0.5, burnin = 1
```

We go deeper than in exercise 2 and evaluate some hyperparameter options: Because logLik exists in multiple packages, we need to specify that the topicmodels version is used.

```
set.seed(50)
topicmodels::logLik(initial_model)
```

```
## 'log Lik.' -466061.7 (df=197895)
```

```
topicmodels::logLik(vem_model)
```

```
## 'log Lik.' -480597.6 (df=197896)
```

```
topicmodels::logLik(ch6_model)
```

```
## 'log Lik.' -465946.8 (df=197895)
```

The model from Chapter 6 has the best fit, followed by the initial model.

For perplexity, we first need to split the data:

```
set.seed(34)
train_index <- sample(seq_len(ndoc(dfm_sent_aus)), size = 0.8 * ndoc(dfm_sent_aus), replace = FALSE)
dfm_train <- dfm_sent_aus[train_index, ]
dfm_test <- dfm_sent_aus[-train_index, ]
dtm_train <- convert(dfm_sent_aus, to = "topicmodels")
dtm_test <- convert(dfm_sent_aus, to = "topicmodels")
```

Now run the model on 80% of the data:

```
set.seed(50)
initial_model_train <- topicmodels::LDA(dtm_train, k = 15, method = "Gibbs", control = list(alpha = 0.5
set.seed(50)
vem_model_train <- topicmodels::LDA(dtm_train,  k= 15, method = "VEM", control = list(alpha = 0.5))
set.seed(50)
ch6_model_train <- topicmodels::LDA(dtm_train, k = 15, method = "Gibbs", control = list(alpha = 0.5, bu
set.seed(50)
```

And finally, calculate perplexity:

```r
topicmodels::perplexity(initial_model_train, dtm_test)
```

```
## [1] 2181.266
```

```r
topicmodels::perplexity(vem_model_train, dtm_test)
```

```
## [1] 2064.071
```

```r
topicmodels::perplexity(ch6_model_train, dtm_test)
```

```
## [1] 2241.415
```

The lowest number here is the VEM model, followed by the initial model.

Additionally, we can look at the resulting categories as we did in Chapter 6 to see if the results are logical.

Evaluation with the oolong package:

```r
library(oolong)
```

```
## Warning: Paket 'oolong' wurde unter R Version 4.2.3 erstellt
```

```r
oolong_test <- create_oolong(input_model = initial_model, input_corpus = corpus_sent$text)
```

```r
library("shiny")
oolong_test$do_word_intrusion_test()
```

```
##
## Listening on http://127.0.0.1:6726
```

```r
oolong_test$lock()
```

```r
oolong_test
```

```
##

## -- oolong (topic model) --------------------------------------------------------

## v WI x TI x WSI

## i WI: k = 15, 15 coded.

##

## -- Results: --

##

## i 40%  precision
```