Lab 6

Conie O'Malley

2025-02-23

Table of contents


```r
# install packages
packages <- c("word2vec", "text2vec", "magrittr")


for (i in packages) {
```

```r
    if (!requireNamespace(i, quietly = TRUE)) {
        renv::install(i)
    }
    library(i, character.only = TRUE)  # Load the package
}
```

```
Warning: package 'word2vec' was built under R version 4.3.3

Warning: package 'text2vec' was built under R version 4.3.3
```

0.1 Deliverable 1: Get your working directory and paste below

```r
getwd()
```

```
[1] "/Users/coniecakes/Library/CloudStorage/OneDrive-Personal/001. Documents - Main/023. Prog
```

1 Part 1: Building and Using Word Embeddings

1.1 Deliverable 1: Load the data and inspect the first few rows

```r
data("movie_review")
head(movie_review)
```

```
      id sentiment
1 5814_8         1
2 2381_9         1
3 7759_3         0
4 3630_4         0
5 9495_8         1
6 8196_8         1

1
2
3 The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine)
4
5
6
```

1.2 Deliverable 2: Preprocess the data

```r
tokens <- movie_review$review %>%
    tolower() %>%
    text2vec::word_tokenizer()
```

1.3 Deliverable 3: Create a Vocabulary and Term Co-Occurrence Matrix

```r
it <- text2vec::itoken(tokens, progressbar = FALSE)
vocab <- text2vec::create_vocabulary(it)
vectorizer <- text2vec::vocab_vectorizer(vocab)

tcm <- text2vec::create_tcm(it, vectorizer, skip_grams_window = 5L)
```

1.4 Deliverable 4: Fit the GloVe Model to the TCM

```r
glove_model <- text2vec::GlobalVectors$new(rank = 50, x_max = 10)
word_vectors <- glove_model$fit_transform(tcm, n_iter = 20)
```

```
INFO  [13:12:21.791] epoch 1, loss 0.1505
INFO  [13:12:23.040] epoch 2, loss 0.0973
INFO  [13:12:24.242] epoch 3, loss 0.0830
INFO  [13:12:25.408] epoch 4, loss 0.0751
INFO  [13:12:26.603] epoch 5, loss 0.0694
INFO  [13:12:27.794] epoch 6, loss 0.0651
INFO  [13:12:28.985] epoch 7, loss 0.0615
INFO  [13:12:30.159] epoch 8, loss 0.0585
INFO  [13:12:31.339] epoch 9, loss 0.0560
INFO  [13:12:32.527] epoch 10, loss 0.0538
INFO  [13:12:33.700] epoch 11, loss 0.0518
INFO  [13:12:34.873] epoch 12, loss 0.0501
INFO  [13:12:36.042] epoch 13, loss 0.0486
INFO  [13:12:37.220] epoch 14, loss 0.0472
INFO  [13:12:38.389] epoch 15, loss 0.0460
INFO  [13:12:39.609] epoch 16, loss 0.0449
INFO  [13:12:40.786] epoch 17, loss 0.0439
INFO  [13:12:41.984] epoch 18, loss 0.0430
```

```
INFO  [13:12:43.188] epoch 19, loss 0.0422
INFO  [13:12:44.378] epoch 20, loss 0.0414
```

1.5 Deliverable 5: Explore the word embeddings.

```r
king_vector <- word_vectors["king", , drop = FALSE]
print(king_vector)
```

```
          [,1]      [,2]       [,3]      [,4]       [,5]      [,6]      [,7]
king 0.2048331 0.1033692 -0.5105069 0.3859732 -0.1183013 0.1526028 0.1755947
          [,8]     [,9]      [,10]      [,11]      [,12]      [,13]      [,14]
king 0.7517204 0.209195 -0.2875485 -0.3756137 -0.1118976 -0.3348667 -0.4776994
         [,15]     [,16]      [,17]      [,18]       [,19]      [,20]     [,21]
king 0.1896476 0.2930924 0.02056652 0.01906195 -0.08077735 0.05754792 -0.195102
         [,22]      [,23]     [,24]      [,25]     [,26]     [,27]     [,28]
king 0.2521377 -0.3297842 0.1747278 -0.4108376 0.4921499 0.2512966 0.3926649
            [,29]      [,30]      [,31]        [,32]      [,33]      [,34]
king -0.008526148 -0.1595909 0.02619935 -0.007153479 -0.3324712 -0.1743605
          [,35]     [,36]     [,37]     [,38]    [,39]      [,40]      [,41]
king -0.0318132 0.3163234 0.1759345 -0.268418 0.424578 0.03814595 -0.3217611
          [,42]     [,43]      [,44]        [,45]     [,46]      [,47]
king -0.6259316 0.0884071 -0.1251529 -0.009555229 0.2984651 -0.5912164
         [,48]      [,49]     [,50]
king 0.6793077 0.07639147 0.6631689
```

1.6 Deliverable 6: Find Words Similiar to "king"

```r
cos_sim <- text2vec::sim2(x = word_vectors, y = king_vector, method = "cosine", norm = "l2")
head(sort(cos_sim[,1], decreasing = TRUE), 5)
```

```
      king government       paul     hitler       thin
 1.0000000  0.5866684  0.5761881  0.5599212  0.5326028
```

2 Part 2: Building Simple Bi-Gram Language Models

2.1 Deliverable 7: Collect and Prepare Your Data

```
book <- gutenbergr::gutenberg_download(158)
```

```
Determining mirror for Project Gutenberg from https://www.gutenberg.org/robot/harvest
```

```
Using mirror http://aleph.gutenberg.org
```

```
bigrams <- book %>%
    tidytext::unnest_tokens(bigram, text, token = "ngrams", n = 2)
```

2.2 Deliverable 8: Calculate the Frequency of Bigrams

```
bigrams_separated <- bigrams %>%
    tidyr::separate(bigram, c("word1", "word2"), sep = " ")
bigram_counts <- bigrams_separated %>%
    dplyr::count(word1, word2, sort = TRUE)
```

2.3 Deliverable 9: Calculate the Probability of Bigrams

```
word1_counts <- bigrams_separated %>%
    dplyr::count(word1, sort = TRUE) %>%
    dplyr::rename(total = n)

bigram_probabilities <- bigram_counts %>%
    dplyr::left_join(word1_counts, by = "word1") %>%
    dplyr::mutate(probability = n/total)
```

2.4 Deliverable 10: Use the Bigram Model to Predict the Next Word

```r
predict_next_word <- function(current_word) {
    bigram_probabilities %>%
        dplyr::filter(word1 == current_word) %>%
        dplyr::arrange(desc(probability)) %>%
        utils::head(5)
}
```

```r
predict_next_word("mr")
```

```
# A tibble: 5 x 5
  word1 word2         n total probability
  <chr> <chr>     <int> <int>       <dbl>
1 mr    knightley   237  1018      0.233
2 mr    elton       171  1018      0.168
3 mr    weston      125  1018      0.123
4 mr    woodhouse    94  1018      0.0923
5 mr    frank        48  1018      0.0472
```

3 Part 3: Word Embeddings in Python

```python
#import tensorflow as tf
#import torch
#import keras
import nltk
#from transformers import pipeline
from nltk.corpus import movie_reviews
from gensim.models import Word2Vec
nltk.download("movie_reviews")
```

```
True
```

3.1 Deliverable 11: Load and Prepare the Data in Python

```python
documents = [list(movie_reviews.words(fileid)) for fileid in movie_reviews.fileids()]
```

3.2 Deliverable 12: Train a Word2Vec model using gensim

```
model = Word2Vec(sentences=documents, vector_size=50, window=5, min_count=2, workers=4)
```

3.3 Deliverable 13: Explore the word embeddings

```
king_vector = model.wv["king"]
print(king_vector)
```

```
[-0.2856428  -0.17786421  0.2799243   0.15504603  0.04463694  0.43373644
  1.7166221  -0.5387969  -0.05513414 -0.35213506  0.5866303  -1.0010692
  1.0382105  -0.2377805   0.96553653 -0.2529933  -0.06705428  0.00373106
 -1.0715698  -1.4174579   1.0173542   0.24079442  1.856158   -1.100083
  0.79750067 -0.38062474  0.18164986 -0.47823712 -1.079246   -0.16276346
  0.28239533 -0.8364547   0.03266196  1.0683049  -0.16547683  0.3906276
 -0.31353572  1.2597526  -0.9338115  -0.6319033  -0.36353156 -0.02366525
  0.13028109  0.29832596  0.69743013  0.55217856  0.01818331  0.09821928
 -0.09170736  0.847732  ]
```

3.4 Deliverable 14: Find similar words to a "king"

```
similar_words = model.wv.most_similar("king", topn=5)
print(similar_words)
```

```
[('jackson', 0.8497037887573242), ('jane', 0.8455929756164551), ('ryan', 0.8443729877471924)
```

3.5 Deliverable 15: Perform Analogies

```
result = model.wv.most_similar(positive=["woman", "king"], negative=["man"], topn=1)
print(result)
```

```
[('jane', 0.8494290113449097)]
```

Transferred Part 4 to a separate ipynb file