AML Rohit Assignment-4

ROHIT VURADI

2024-04-21

Introduction

This demonstrates how to perform sentiment analysis on the IMDB movie review dataset using Recurrent Neural Networks (RNNs) with the Keras library in R. We'll modify the example from Chapter 6 of the book to align with the specific requirements of our assignment.

RNNs and Transformers are the commonly used models in the field of NLP (Natural Language Processing). They are used for processing sequential data such as text to derive meaning/understanding. ElseGRUs, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are very capable of keeping a memory over time since they preserve the memory by sequencing. And different, to transformation which apply attention self-mechanisms to the model work, there is a capturing of long-range aspects in sequences. RNNs and transformers can be used when it comes to text and data in sequences both for tasks such as sentiment analysis, machine translation, and text generation.

To apply RNNs or Transformers to text data:To apply RNNs or Transformers to text data:

Preprocess the text data: Let's take the text and break it down into words or subwords, convert those tokens into numerical form (such as word embeddings) , and finally ensure the sequences of the sequence length are equal.

Define the model architecture: Construct a neural network model, which is going to have suitable layers for processing sequential data. For RNNs this is where it embeds the words in different ways and then these are concatenated (LSTM, GRU). To the Transformers, forming a complementary relation between the self-attention layers and the feedforward layers would be desirable.

Compile and train the model: Collect the model with the proper loss function and optimizer, then provide the training data for the model successively.

Evaluate the model: Evaluate the model on the validation or test set, which is different from the training set and contains the information about how well the model performs at text data processing.

```r
# load the neccessary library
library(keras)
```

Loading the IMDB Dataset

We'll load the IMDB dataset with the following modifications:

Cutoff reviews after 150 words. Restrict training samples to 100. Validation on 10,000 samples. Consider only the top 10,000 words.

```r
# load the IMDB dataset
imdb <- dataset_imdb(num_words = 10000)
maxlen <- 150
x_train <- pad_sequences(imdb$train$x, maxlen = maxlen)
x_test <- pad_sequences(imdb$test$x, maxlen = maxlen)
sample_size <- 100
```

```r
x_train <- x_train[1:sample_size, ]
y_train <- as.matrix(imdb$train$y)[1:sample_size, ]
x_val <- x_test[1:10000, ]
y_val <- as.matrix(imdb$test$y)[1:10000, ]
```

```r
# check the dimensions of the training and validation sets
dim(x_train)
```

```
## [1] 100 150
```

```r
dim(y_train)
```

```
## NULL
```

```r
dim(x_val)
```

```
## [1] 10000   150
```

```r
dim(y_val)
```

```
## NULL
```

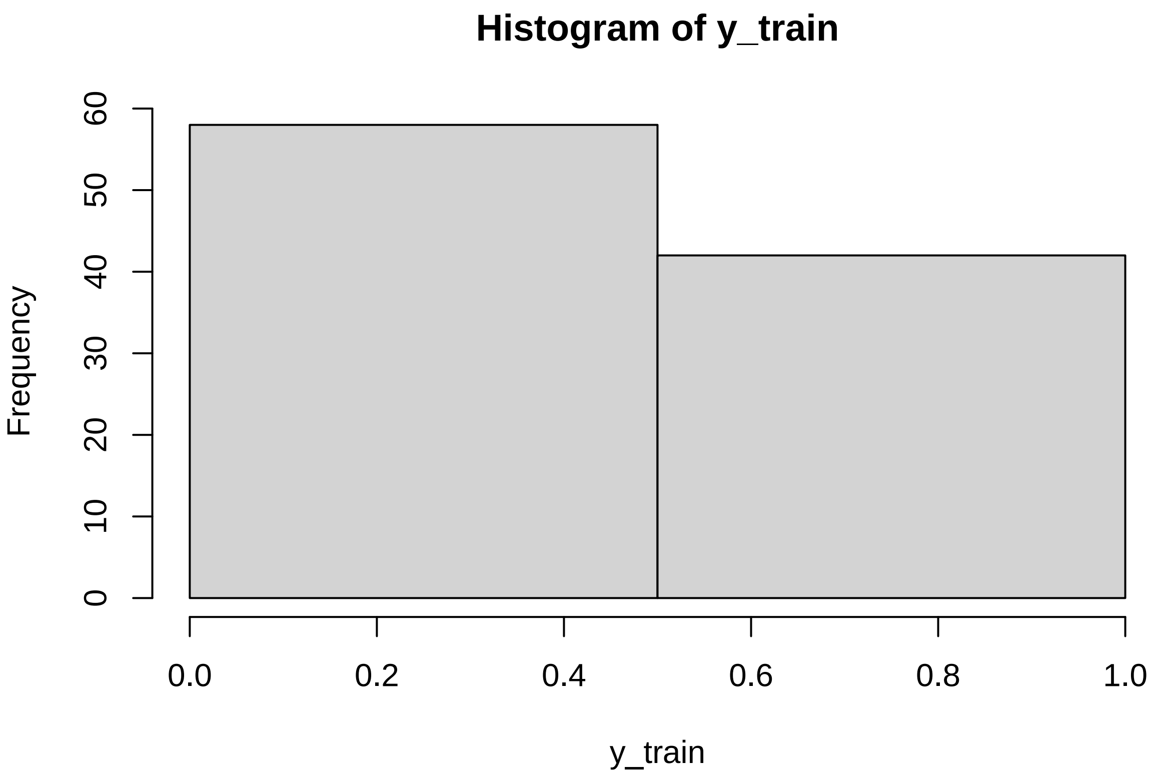
```r
# check the first 5 elements of the training set
head(x_train)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]   12   16   43  530   38   76   15   13 1247     4    22    17   515    17
## [2,]    4  249  126   93    4  114    9 2300 1523     5   647     4   116     9
## [3,]    0    0    0    0    0    0    0    0    0     1    14    47     8    30
## [4,]    4    2    7  154    5    4  518   53    2     2     7  3211   882    11
## [5,]    0    0    0    1  249 1323    7   61  113    10    10    13  1637    14
## [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]    12    16   626    18     2     5    62   386    12     8   316     8
## [2,]    35  8163     4   229     9   340  1322     4   118     9     4   130
## [3,]    31     7     4   249   108     7     4  5974    54    61   369    13
## [4,]   399    38    75   257  3807    19     2    17    29   456     4    65
## [5,]    20    56    33  2401    18   457    88    13  2626  1400    45  3171
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]   106     5     4  2223  5244    16   480    66  3785    33     4   130
## [2,]  4901    19     4  1002     5    89    29   952    46    37     4   455
## [3,]    71   149    14    22   112     4  2401   311    12    16  3711    33
## [4,]     7    27   205   113    10    10     2     4     2     2     9   242
## [5,]    13    70    79    49   706   919    13    16   355   340   355  1696
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]    12    16    38   619     5    25   124    51    36   135    48    25
## [2,]     9    45    43    38  1543  1905   398     4  1649    26  6853     5
## [3,]    75    43  1829   296     4    86   320    35   534    19   263  4821
## [4,]     4    91  1202     2     5  2070   307    22     7  5168   126    93
## [5,]    96   143     4    22    32   289     7    61   369    71  2359     5
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]  1415    33     6    22    12   215    28    77    52     5    14   407
## [2,]   163    11  3215     2     4  1153     9   194   775     7  8255     2
## [3,]  1301     4  1873    33    89    78    12    66    16     4   360     7
```

```
## [4,]    40     2    13   188  1076  3222    19     4     2     7  2348   537
## [5,]    13    16   131  2073   249   114   249   229   249    20    13    28
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]    16    82     2     8     4   107   117  5952    15   256     4     2
## [2,]   349  2637   148   605     2  8003    15   123   125    68     2  6853
## [3,]     4    58   316   334    11     4  1716    43   645   662     8   257
## [4,]    23    53   537    21    82    40     2    13     2    14   280    13
## [5,]   126   110    13   473     8   569    61   419    56   429     6  1513
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
## [1,]     7  3766     5   723    36    71    43   530   476    26   400   317
## [2,]    15   349   165  4362    98     5     4   228     9    43     2  1157
## [3,]    85  1200    42  1228  2578    83    68  3912    15    36   165  1539
## [4,]   219     4     2   431   758   859     4   953  1052     2     7  5991
## [5,]    18    35   534    95   474   570     5    25   124   138    88    12
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
## [1,]    46     7     4     2  1029    13   104    88     4   381    15   297
## [2,]    15   299   120     5   120   174    11   220   175   136    50     9
## [3,]   278    36    69     2   780     8   106    14  6905  1338    18     6
## [4,]     5    94    40    25   238    60     2     4     2   804     2     7
## [5,]   421  1543    52   725  6397    61   419    11    13  1571    15  1543
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## [1,]    98     32   2071     56     26    141      6    194   7486     18
## [2,]  4373    228   8255      5      2    656    245   2350      5      4
## [3,]    22     12    215     28    610     40      6     87    326     23
## [4,]     4   9941    132      8     67      6     22     15      9    283
## [5,]    20     11      4      2      5    296     12   3524      5     15
## [6,]     0      0      0      0      0      0      0      0      0      1
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## [1,]      4    226     22     21    134    476     26    480      5    144
## [2,]   9837    131    152    491     18      2     32   7464   1212     14
## [3,]   2300     21     23     22     12    272     40     57     31     11
## [4,]      8   5168     14     31      9    242    955     48     25    279
## [5,]    421    128     74    233    334    207    126    224     12    562
## [6,]    778    128     74     12    630    163     15      4   1766   7982
##      [,119] [,120] [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128]
## [1,]     30   5535     18     51     36     28    224     92     25    104
## [2,]      9      6    371     78     22    625     64   1382      9      8
## [3,]      4     22     47      6   2307     51      9    170     23    595
## [4,]      2     23     12   1685    195     25    238     60    796      2
## [5,]    298   2167   1272      7   2601      5    516    988     43      8
## [6,]   1051      2     32     85    156     45     40    148    139    121
##      [,129] [,130] [,131] [,132] [,133] [,134] [,135] [,136] [,137] [,138]
## [1,]      4    226     65     16     38   1334     88     12     16    283
## [2,]    168    145     23      4   1690     15     16      4   1355      5
## [3,]    116    595   1352     13    191     79    638     89      2     14
## [4,]      4    671      7   2804      5      4    559    154    888      7
## [5,]     79    120     15    595     13    784     25   3171     18    165
## [6,]    664    665     10     10   1361    173      4    749      2     16
##      [,139] [,140] [,141] [,142] [,143] [,144] [,145] [,146] [,147] [,148]
## [1,]      5     16   4472    113    103     32     15     16   5345     19
```

```
## [2,]     28      6     52    154    462     33     89     78    285     16
## [3,]      9      8    106    607    624     35    534      6    227      7
## [4,]    726     50     26     49   7008     15    566     30    579     21
## [5,]    170    143     19     14      5   7224      6    226    251      7
## [6,]   3804      8      4    226     65     12     43    127     24      2
##       [,149] [,150]
## [1,]    178     32
## [2,]    145     95
## [3,]    129    113
## [4,]     64   2574
## [5,]     61    113
## [6,]     10     10
```

```r
# plot the distribution
hist(y_train, breaks = 2)
```



Model Definition and Compilation

We define an RNN model with an embedding layer and LSTM layer:

```r
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = 10000, output_dim = 128) %>%
  layer_lstm(units = 64) %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  loss = "binary_crossentropy",
  optimizer = "adam",
  metrics = c("accuracy")
)
```

Model Training

We train the model for 20 epochs with a batch size of 100 samples.

```r
model %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 100,
  validation_data = list(x_val, y_val)
)
```

```
## Epoch 1/20
## 1/1 - 6s - loss: 0.6931 - accuracy: 0.5100 - val_loss: 0.6929 - val_accuracy: 0.5149 - 6s/epoch - 6s
## Epoch 2/20
## 1/1 - 4s - loss: 0.6883 - accuracy: 0.6500 - val_loss: 0.6927 - val_accuracy: 0.5117 - 4s/epoch - 4s
## Epoch 3/20
## 1/1 - 4s - loss: 0.6831 - accuracy: 0.6800 - val_loss: 0.6926 - val_accuracy: 0.5062 - 4s/epoch - 4s
## Epoch 4/20
## 1/1 - 3s - loss: 0.6774 - accuracy: 0.6200 - val_loss: 0.6926 - val_accuracy: 0.5045 - 3s/epoch - 3s
## Epoch 5/20
## 1/1 - 4s - loss: 0.6708 - accuracy: 0.6000 - val_loss: 0.6927 - val_accuracy: 0.5036 - 4s/epoch - 4s
## Epoch 6/20
## 1/1 - 4s - loss: 0.6631 - accuracy: 0.6000 - val_loss: 0.6929 - val_accuracy: 0.5029 - 4s/epoch - 4s
## Epoch 7/20
## 1/1 - 4s - loss: 0.6539 - accuracy: 0.6000 - val_loss: 0.6933 - val_accuracy: 0.5028 - 4s/epoch - 4s
## Epoch 8/20
## 1/1 - 4s - loss: 0.6430 - accuracy: 0.6000 - val_loss: 0.6942 - val_accuracy: 0.5028 - 4s/epoch - 4s
## Epoch 9/20
## 1/1 - 3s - loss: 0.6300 - accuracy: 0.6000 - val_loss: 0.6960 - val_accuracy: 0.5028 - 3s/epoch - 3s
## Epoch 10/20
## 1/1 - 3s - loss: 0.6145 - accuracy: 0.6000 - val_loss: 0.6993 - val_accuracy: 0.5028 - 3s/epoch - 3s
## Epoch 11/20
## 1/1 - 3s - loss: 0.5961 - accuracy: 0.5900 - val_loss: 0.7058 - val_accuracy: 0.5028 - 3s/epoch - 3s
## Epoch 12/20
## 1/1 - 3s - loss: 0.5744 - accuracy: 0.5800 - val_loss: 0.7187 - val_accuracy: 0.5027 - 3s/epoch - 3s
## Epoch 13/20
## 1/1 - 3s - loss: 0.5499 - accuracy: 0.5800 - val_loss: 0.7463 - val_accuracy: 0.5027 - 3s/epoch - 3s
## Epoch 14/20
## 1/1 - 4s - loss: 0.5259 - accuracy: 0.5800 - val_loss: 0.7744 - val_accuracy: 0.5028 - 4s/epoch - 4s
## Epoch 15/20
## 1/1 - 3s - loss: 0.5014 - accuracy: 0.5800 - val_loss: 0.7609 - val_accuracy: 0.5028 - 3s/epoch - 3s
## Epoch 16/20
## 1/1 - 3s - loss: 0.4619 - accuracy: 0.6700 - val_loss: 0.7383 - val_accuracy: 0.5042 - 3s/epoch - 3s
## Epoch 17/20
## 1/1 - 4s - loss: 0.4229 - accuracy: 0.8100 - val_loss: 0.7254 - val_accuracy: 0.5085 - 4s/epoch - 4s
## Epoch 18/20
## 1/1 - 4s - loss: 0.3845 - accuracy: 0.8700 - val_loss: 0.7234 - val_accuracy: 0.5146 - 4s/epoch - 4s
## Epoch 19/20
## 1/1 - 3s - loss: 0.3387 - accuracy: 0.9200 - val_loss: 0.7433 - val_accuracy: 0.5220 - 3s/epoch - 3s
## Epoch 20/20
## 1/1 - 3s - loss: 0.2828 - accuracy: 0.9300 - val_loss: 0.9024 - val_accuracy: 0.5270 - 3s/epoch - 3s
```

Model Evaluation

We evaluate the model on the test set:

```r
loss_and_metrics <- model %>% evaluate(x_test, imdb$test$y, batch_size = 32)
```

```
## 782/782 - 9s - loss: 0.9021 - accuracy: 0.5250 - 9s/epoch - 12ms/step
```

```r
cat("Test Loss:", loss_and_metrics[[1]], "\n")
```

```
## Test Loss: 0.9020851
```

```r
cat("Test Accuracy:", loss_and_metrics[[2]], "\n")
```

```
## Test Accuracy: 0.525
```

Conclusion

We have successfully trained and evaluated an RNN model for sentiment analysis on the IMDB dataset with the specified modifications. Further experiments can be conducted to explore the impact of using pre-trained word embeddings and varying the number of training samples on model performance.