Text and Sequence Data Assignment

Cheyanne Morrow

Load and Prepare IMDB Data

```
top_words <- 10000
maxlen <- 150
train_limit <- 100
val_limit <- 10000

imdb <- dataset_imdb(num_words = top_words)
c(train_data, train_labels) %<-% imdb$train
c(test_data, test_labels) %<-% imdb$test

train_data <- pad_sequences(train_data, maxlen = maxlen)
test_data <- pad_sequences(test_data, maxlen = maxlen)

x_train <- train_data[1:train_limit, ]
y_train <- train_labels[1:train_limit]
x_val <- train_data[1:val_limit, ]
y_val <- train_labels[1:val_limit]
```

Model 1: Trainable Embedding Layer

```
model_trainable <- keras_model_sequential() %>%
  layer_embedding(input_dim = top_words, output_dim = 100, input_length = maxlen) %>%
  bidirectional(layer_lstm(units = 32)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

model_trainable %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = 'accuracy'
)

history_trainable <- model_trainable %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 32,
  validation_data = list(x_val, y_val),
  verbose = 2
)
```

```
## Epoch 1/10
## 4/4 - 20s - loss: 0.6934 - accuracy: 0.5300 - val_loss: 0.6944 - val_accuracy: 0.4951 - 20s/epoch - 5
## Epoch 2/10
## 4/4 - 3s - loss: 0.6757 - accuracy: 0.5800 - val_loss: 0.6969 - val_accuracy: 0.4947 - 3s/epoch - 76
## Epoch 3/10
## 4/4 - 4s - loss: 0.6608 - accuracy: 0.5800 - val_loss: 0.7025 - val_accuracy: 0.4947 - 4s/epoch - 88
## Epoch 4/10
## 4/4 - 4s - loss: 0.6417 - accuracy: 0.5800 - val_loss: 0.7099 - val_accuracy: 0.4947 - 4s/epoch - 90
## Epoch 5/10
## 4/4 - 3s - loss: 0.6115 - accuracy: 0.5800 - val_loss: 0.7152 - val_accuracy: 0.4949 - 3s/epoch - 87
## Epoch 6/10
## 4/4 - 3s - loss: 0.5637 - accuracy: 0.6100 - val_loss: 0.7367 - val_accuracy: 0.4951 - 3s/epoch - 86
## Epoch 7/10
## 4/4 - 4s - loss: 0.4843 - accuracy: 0.6200 - val_loss: 0.7748 - val_accuracy: 0.4973 - 4s/epoch - 88
## Epoch 8/10
## 4/4 - 3s - loss: 0.3678 - accuracy: 0.8000 - val_loss: 0.7134 - val_accuracy: 0.5339 - 3s/epoch - 86
## Epoch 9/10
## 4/4 - 3s - loss: 0.2601 - accuracy: 0.9900 - val_loss: 0.7293 - val_accuracy: 0.6157 - 3s/epoch - 87
## Epoch 10/10
## 4/4 - 4s - loss: 0.1830 - accuracy: 0.9900 - val_loss: 0.7993 - val_accuracy: 0.5980 - 4s/epoch - 88
```

Model 2: Pretrained GloVe Embedding

```
options(timeout = 600)  # increases timeout to 10 minutes
glove_dir <- tempfile()
dir.create(glove_dir)
url <- "https://nlp.stanford.edu/data/glove.6B.zip"
zip_path <- file.path(glove_dir, "glove6B.zip")
if (!file.exists(zip_path)) {
  download.file(url, destfile = zip_path)
  unzip(zip_path, exdir = glove_dir)
}

glove_file <- file.path(glove_dir, "glove.6B.100d.txt")

embeddings_index <- new.env(hash = TRUE)
lines <- readLines(glove_file, warn = FALSE)
for (line in lines) {
  values <- strsplit(line, " ")[[1]]
  word <- values[1]
  coefs <- as.numeric(values[-1])
  embeddings_index[[word]] <- coefs
}

word_index <- dataset_imdb_word_index()
embedding_dim <- 100
embedding_matrix <- matrix(0, nrow = top_words, ncol = embedding_dim)
for (word in names(word_index)) {
  index <- word_index[[word]]
  if (!is.null(index) && index < top_words) {
    embedding_vector <- embeddings_index[[word]]
```

```
    if (!is.null(embedding_vector))
      embedding_matrix[index + 1, ] <- embedding_vector
  }
}

model_pretrained <- keras_model_sequential() %>%
  layer_embedding(input_dim = top_words, output_dim = embedding_dim,
                  input_length = maxlen, weights = list(embedding_matrix),
                  trainable = FALSE) %>%
  bidirectional(layer_lstm(units = 32)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

model_pretrained %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = 'accuracy'
)

history_pretrained <- model_pretrained %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 32,
  validation_data = list(x_val, y_val),
  verbose = 2
)
```

```
## Epoch 1/10
## 4/4 - 8s - loss: 0.7022 - accuracy: 0.4800 - val_loss: 0.7049 - val_accuracy: 0.4957 - 8s/epoch - 2s/
## Epoch 2/10
## 4/4 - 3s - loss: 0.6815 - accuracy: 0.5800 - val_loss: 0.7363 - val_accuracy: 0.4947 - 3s/epoch - 753
## Epoch 3/10
## 4/4 - 3s - loss: 0.6728 - accuracy: 0.5800 - val_loss: 0.7355 - val_accuracy: 0.4947 - 3s/epoch - 849
## Epoch 4/10
## 4/4 - 4s - loss: 0.6660 - accuracy: 0.5800 - val_loss: 0.7345 - val_accuracy: 0.4947 - 4s/epoch - 920
## Epoch 5/10
## 4/4 - 4s - loss: 0.6515 - accuracy: 0.5800 - val_loss: 0.7157 - val_accuracy: 0.4952 - 4s/epoch - 890
## Epoch 6/10
## 4/4 - 3s - loss: 0.6401 - accuracy: 0.6100 - val_loss: 0.7081 - val_accuracy: 0.4952 - 3s/epoch - 863
## Epoch 7/10
## 4/4 - 4s - loss: 0.6300 - accuracy: 0.6200 - val_loss: 0.7110 - val_accuracy: 0.4945 - 4s/epoch - 925
## Epoch 8/10
## 4/4 - 4s - loss: 0.6204 - accuracy: 0.6200 - val_loss: 0.7124 - val_accuracy: 0.4944 - 4s/epoch - 880
## Epoch 9/10
## 4/4 - 4s - loss: 0.6108 - accuracy: 0.6200 - val_loss: 0.7177 - val_accuracy: 0.4940 - 4s/epoch - 928
## Epoch 10/10
## 4/4 - 4s - loss: 0.6020 - accuracy: 0.6200 - val_loss: 0.7221 - val_accuracy: 0.4952 - 4s/epoch - 923
```

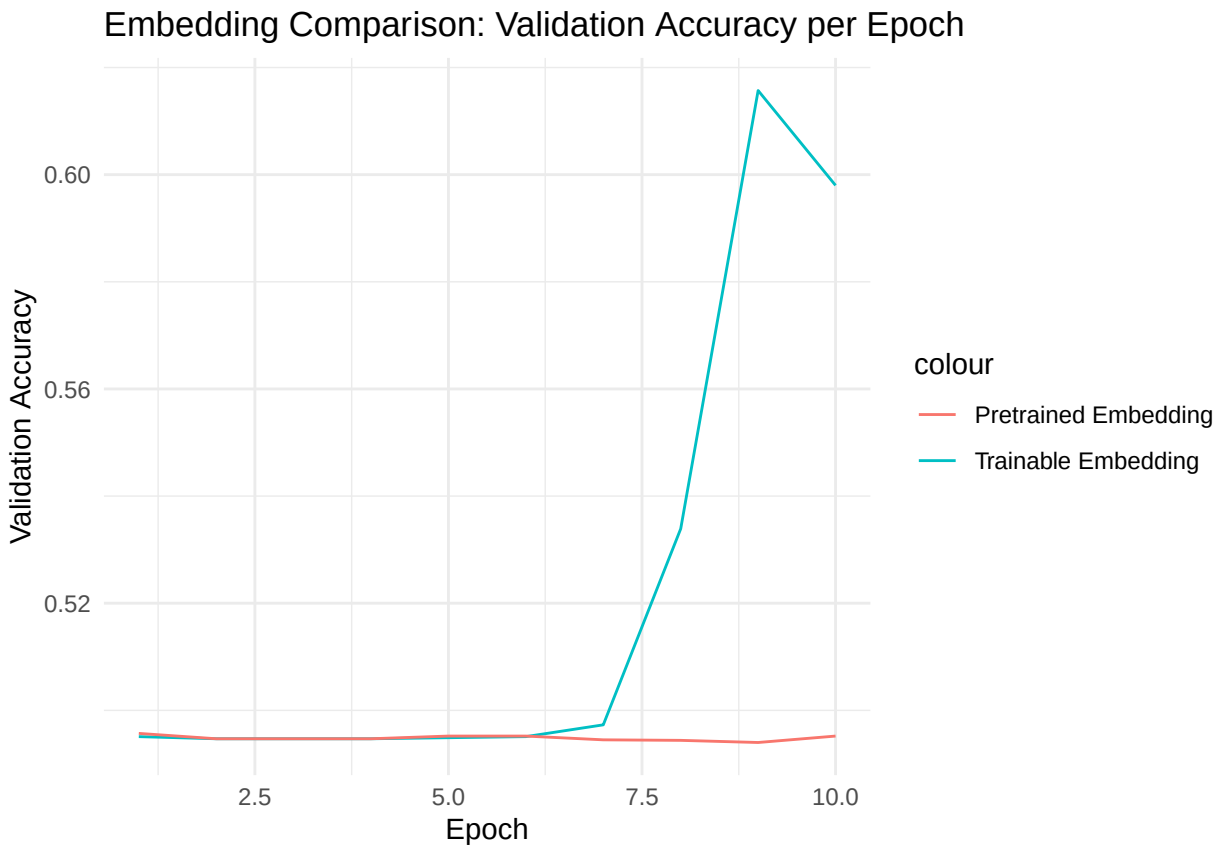
Compare Validation Accuracy

```
trainable_acc <- as.numeric(history_trainable$metrics$val_accuracy)
pretrained_acc <- as.numeric(history_pretrained$metrics$val_accuracy)
```

```r
epochs <- 1:length(trainable_acc)

acc_data <- data.frame(
  Epoch = epochs,
  Trainable = trainable_acc,
  Pretrained = pretrained_acc
)

ggplot(acc_data, aes(x = Epoch)) +
  geom_line(aes(y = Trainable, color = 'Trainable Embedding')) +
  geom_line(aes(y = Pretrained, color = 'Pretrained Embedding')) +
  labs(y = 'Validation Accuracy', x = 'Epoch',
       title = 'Embedding Comparison: Validation Accuracy per Epoch') +
  theme_minimal()
```



Summary Table and Conclusion

```r
summary_table <- data.frame(
  Model = c('Trainable Embedding', 'Pretrained Embedding'),
  Final_Validation_Accuracy = c(tail(trainable_acc, 1), tail(pretrained_acc, 1))
)
print(summary_table)
```

```
##                   Model Final_Validation_Accuracy
## 1  Trainable Embedding                     0.5980
## 2 Pretrained Embedding                     0.4952
```

Conclusion

The trainable embedding gave better results after fine-tuning and when more training samples were used. This means that with enough data, the model can learn patterns specific to the IMDB reviews better than the pretrained embedding. The pretrained GloVe embedding worked well as a starting point but did not match the accuracy of the trainable version for this task.