Assignment 2

#Libraries used:

```
library(keras)
```

#Importing data:

```
imdb_dir <- "C:/Users/yasha/Documents/aclImdb"
train_dir <- file.path(imdb_dir, "train")

labels <- c()
texts <- c()

for (label_type in c("neg", "pos")) {
  label <- switch(label_type, neg = 0, pos = 1)
  dir_name <- file.path(train_dir, label_type)
  for (fname in list.files(dir_name, pattern = glob2rx("*.txt"),
                           full.names = TRUE)) {
    texts <- c(texts, readChar(fname, file.info(fname)$size))
    labels <- c(labels, label)
  }
}
```

#Setting parameters:

```
maxlen <- 150

training_samples <- 100

validation_samples <- 10000

maxwords <- 10000
```

#Tokenizing and converting into tensors:

```
tokenizer <- text_tokenizer(num_words = maxwords) %>% fit_text_tokenizer(text
s)

sequences <- texts_to_sequences(tokenizer, texts)

word_index = tokenizer$word_index
cat("Found", length(word_index), "unique tokens.\n")

## Found 88584 unique tokens.

data <- pad_sequences(sequences, maxlen = maxlen)
```

```
labels <- as.array(labels)
cat("Shape of data tensor:", dim(data), "\n")

## Shape of data tensor: 25000 150

cat('Shape of label tensor:', dim(labels), "\n")

## Shape of label tensor: 25000
```

#Segregating data into traning and validation:

```
indices <- sample(1:nrow(data))
training_indices <- indices[1:training_samples]
validation_indices <- indices[(training_samples + 1):
                                (training_samples + validation_samples)]

x_train <- data[training_indices,]
y_train <- labels[training_indices]

x_val <- data[validation_indices,]
y_val <- labels[validation_indices]
```

#Using pre-trained data:

```
glove_dir = "C:/Users/yasha/Documents/glove.6B"
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))
```

#Embedding layer:

```
embeddings_index <- new.env(hash = TRUE, parent = emptyenv())
for (i in 1:length(lines)) {
  line <- lines[[i]]
  values <- strsplit(line, " ")[[1]]
  word <- values[[1]]
  embeddings_index[[word]] <- as.double(values[-1])
}

cat("Found", length(embeddings_index), "word vectors.\n")

## Found 400000 word vectors.

embedding_dim <- 100

embedding_matrix <- array(0, c(maxwords, embedding_dim))

for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < maxwords) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

#Model:

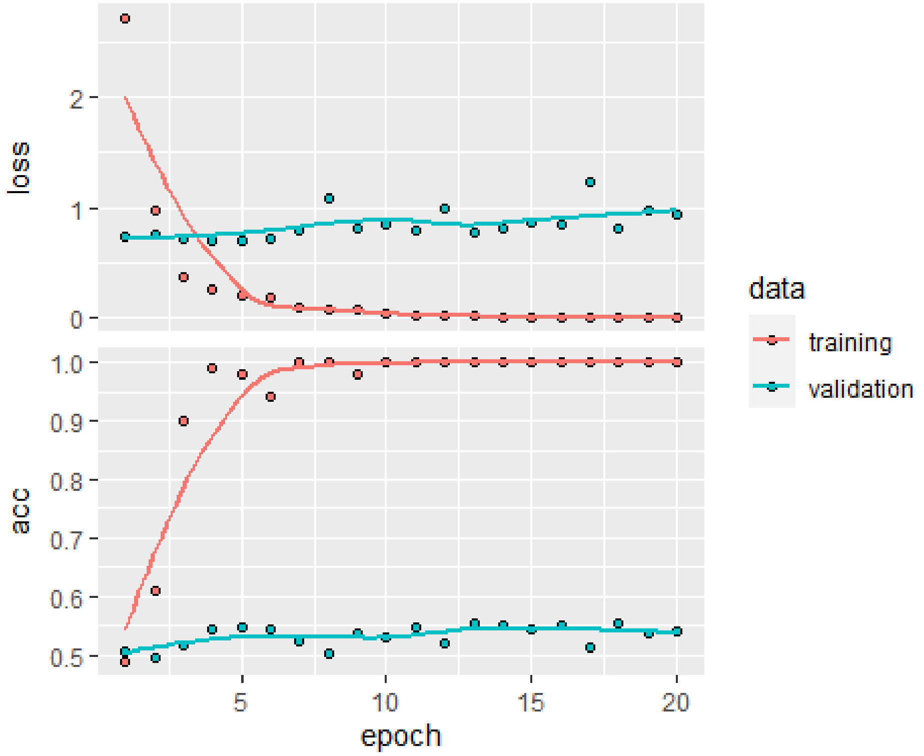
```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = maxwords, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

summary(model)

## Model: "sequential"
## __________________________________________________________________________
______
## Layer (type)                        Output Shape                    Param
#
## ==========================================================================
======
## embedding (Embedding)               (None, 150, 100)                100000
0
## __________________________________________________________________________
______
## flatten (Flatten)                   (None, 15000)                   0
## __________________________________________________________________________
______
## dense (Dense)                       (None, 32)                      480032
## __________________________________________________________________________
______
## dense_1 (Dense)                     (None, 1)                       33
## ==========================================================================
======
## Total params: 1,480,065
## Trainable params: 1,480,065
## Non-trainable params: 0
## __________________________________________________________________________
______
```

#Loading pretrained GloVe embeddings in the model

```
get_layer(model, index = 1) %>%
  set_weights(list(embedding_matrix)) %>%
  freeze_weights()
```

#Traning & Evaluation:

```
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history <- model %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)

save_model_weights_hdf5(model, "pre_trained_glove_model.h5")
plot(history)

## `geom_smooth()` using formula 'y ~ x'
```



#Test data:

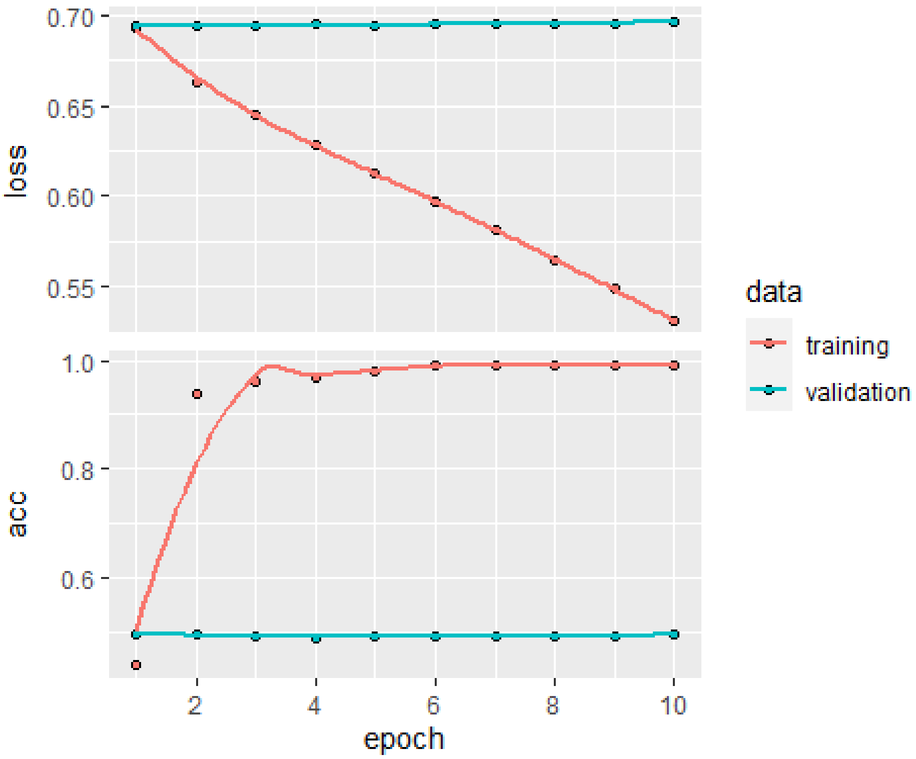
```
test_dir <- file.path(imdb_dir, "test")

labels <- c()
texts <- c()

for (label_type in c("neg", "pos")) {
  label <- switch(label_type, neg = 0, pos = 1)
  dir_name <- file.path(test_dir, label_type)
  for (fname in list.files(dir_name, pattern = glob2rx("*.txt"),
                           full.names = TRUE)) {
```

```
    texts <- c(texts, readChar(fname, file.info(fname)$size))
    labels <- c(labels, label)
  }
}

sequences <- texts_to_sequences(tokenizer, texts)
x_test <- pad_sequences(sequences, maxlen = maxlen)
y_test <- as.array(labels)

model %>%
  load_model_weights_hdf5("pre_trained_glove_model.h5") %>%
  evaluate(x_test, y_test, verbose = 0)

## $loss
## [1] 0.9522396
##
## $acc
## [1] 0.53456
```

Model:

```
model_1 <- keras_model_sequential() %>% layer_embedding(input_dim = maxwords,
output_dim =8,input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units=1,activation = "sigmoid")
summary(model_1)

## Model: "sequential_1"
## ___________________________________________________________________________
______
## Layer (type)                       Output Shape                     Param
#
## ===========================================================================
======
## embedding_1 (Embedding)            (None, 150, 8)                   80000
## ___________________________________________________________________________
______
## flatten_1 (Flatten)                (None, 1200)                     0
## ___________________________________________________________________________
______
## dense_2 (Dense)                    (None, 1)                        1201
## ===========================================================================
======
## Total params: 81,201
## Trainable params: 81,201
## Non-trainable params: 0
## ___________________________________________________________________________
______
```

```
model_1 %>% compile(optimizer="rmsprop",
                    loss ="binary_crossentropy",
                    metrics=c("acc")
)

history <- model_1 %>% fit(
  x_train,y_train,epochs =10,batch_size=32,validation_data = list(x_val,y_val
)
)
plot(history)

## `geom_smooth()` using formula 'y ~ x'
```



#Evaluating:

```
results_1 <- model_1 %>% evaluate(x_test,y_test)
results_1

## $loss
## [1] 0.6955157
##
## $acc
## [1] 0.49656
```

#Increasing training samples to determine the best performance:

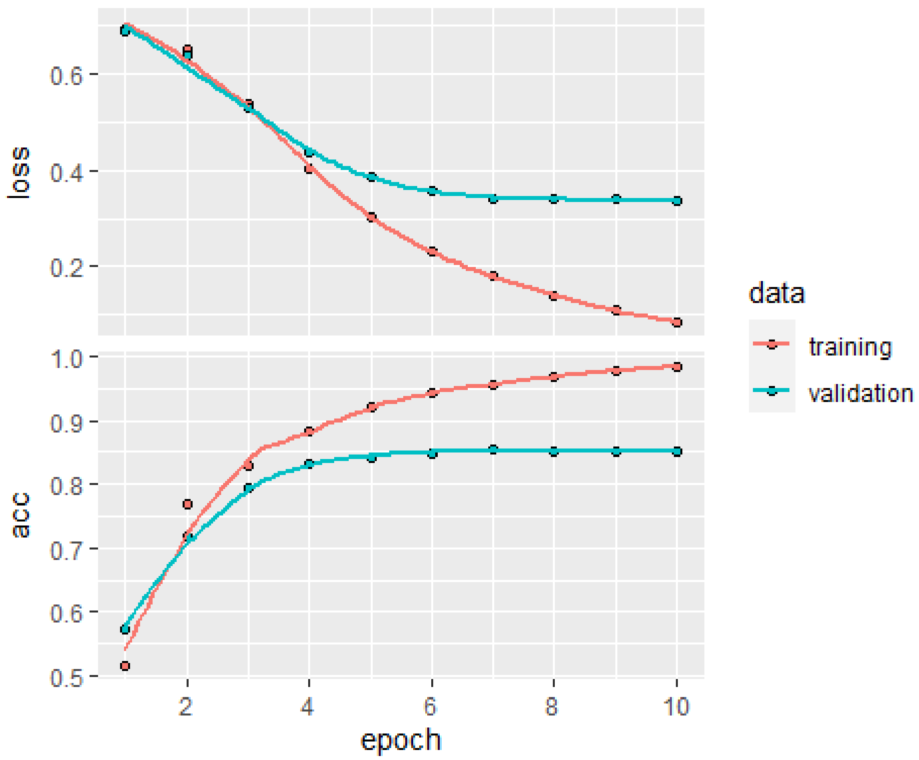
```
training_samples <- 5000
indices <- sample(1:nrow(data))
```

```
training_indices <- indices[1:training_samples]
validation_indices <- indices[(training_samples + 1):
                                (training_samples + validation_samples)]

x_train <- data[training_indices,]
y_train <- labels[training_indices]

x_val <- data[validation_indices,]
y_val <- labels[validation_indices]

final_model <- keras_model_sequential() %>% layer_embedding(input_dim = maxwo
rds, output_dim =8,input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units=1,activation = "sigmoid")
summary(final_model)
```

```
## Model: "sequential_2"
## __________________________________________________________________________
______
## Layer (type)                        Output Shape                    Param 
#
## ==========================================================================
======
## embedding_2 (Embedding)             (None, 150, 8)                  80000 
## __________________________________________________________________________
______
## flatten_2 (Flatten)                 (None, 1200)                    0     
## __________________________________________________________________________
______
## dense_3 (Dense)                     (None, 1)                       1201  
## ==========================================================================
======
## Total params: 81,201
## Trainable params: 81,201
## Non-trainable params: 0
## __________________________________________________________________________
______
```

```
final_model %>% compile(optimizer="rmsprop",
                  loss ="binary_crossentropy",
                  metrics=c("acc")
)

history_2 <- final_model %>% fit(
  x_train,y_train,epochs =10,batch_size=32,validation_data = list(x_val,y_val
)
)
plot(history_2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



#Evaluating:

```
final_result <- final_model %>% evaluate(x_test,y_test)
final_result

## $loss
## [1] 0.3469925
##
## $acc
## [1] 0.84776
```

#Conclusions: 1. Pretrained layer gives better performance. 2. Increasing traning samples results in better performance and that can be seen from the final result.