Assignment 03

Assignment_03

Code By: Xan Lamoreux

Advanced Machine Learning

TASK 00

Processing the labels of the raw IMDB data

```
library(keras)


imdb_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/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)
  }
}
```

Tokenizing the text of the raw IMDB data

```
maxlen <- 150
training_samples <- 100
validation_samples <- 10000
max_words <- 10000

tokenizer <- text_tokenizer(num_words = max_words) %>%
  fit_text_tokenizer(texts)

## Loaded Tensorflow version 2.6.0

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

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]
```

Pre-trained Glove Embedding Model

Parsing the GloVe word-embeddings file

```
glove_dir = "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/"
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))
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.
```

Preparing the GloVe word-embeddings matrix

```
embedding_dim <- 100
embedding_matrix <- array(0, c(max_words, embedding_dim))
for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < max_words) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

Model definition

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, 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_1 (Dense)                     (None, 32)                      480032
## ________________________________________________________________________
_______
## dense (Dense)                       (None, 1)                       33
## ========================================================================
======
## Total params: 1,480,065
## Trainable params: 1,480,065
## Non-trainable params: 0
## ________________________________________________________________________
_______
```

Loading pretrained word embeddings into the embedding layer

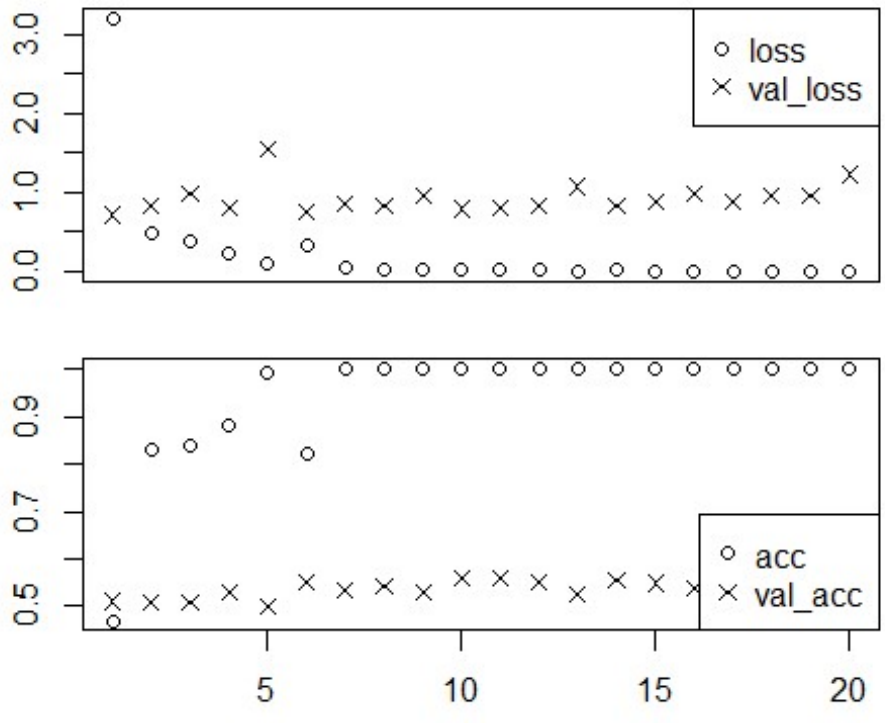
```
get_layer(model, index = 1) %>%
  set_weights(list(embedding_matrix)) %>%
  freeze_weights()
```

Training and 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)
```



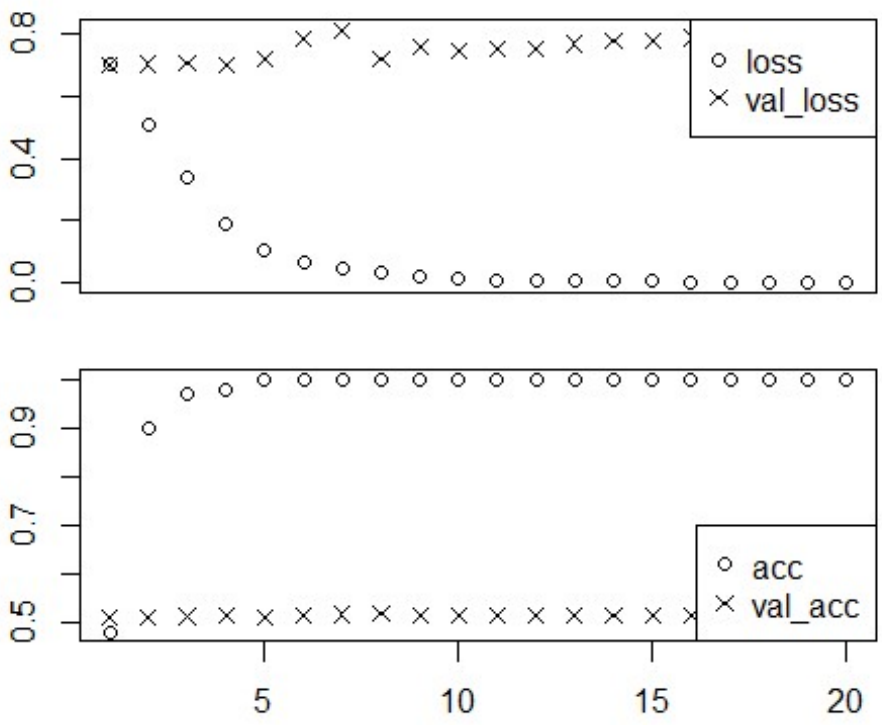
Training the same model without pretrained word embeddings

```
model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history2 <- model2 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
```

```
plot(history2)
```



Tokenizing the data of the test set

```
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)

results2 <- model2 %>% evaluate(x_test, y_test)
results2

##      loss       acc
## 0.7763946 0.5173600
```

Evaluating the model on the test set

```
model %>%
  load_model_weights_hdf5("pre_trained_glove_model.h5") %>%
  evaluate(x_test, y_test)
```

```
##     loss      acc
## 1.208684 0.530280
```

TASK 01 - (Using RNN – training samples=100)

Processing the labels of the raw IMDB data

```
library(keras)


imdb_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/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)
  }
}
```

Tokenizing the text of the raw IMDB data

```
maxlen <- 150
training_samples <- 100
validation_samples <- 10000
max_words <- 10000

tokenizer <- text_tokenizer(num_words = max_words) %>%
  fit_text_tokenizer(texts)

## Loaded Tensorflow version 2.6.0

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

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]
```

Pre-trained Glove Embedding Model

Parsing the GloVe word-embeddings file

```
glove_dir = "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/"
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))
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.
```

Preparing the GloVe word-embeddings matrix

```
embedding_dim <- 100
embedding_matrix <- array(0, c(max_words, embedding_dim))
for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < max_words) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

Model definition

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

```
## Model: "sequential"
## ________________________________________________________________________
_____
## Layer (type)                        Output Shape                    Param
#
## ========================================================================
======
```

```
## embedding (Embedding)                (None, 150, 100)                100000
0
## ________________________________________________________________________
_____
## simple_rnn (SimpleRNN)               (None, 32)                      4256
## ________________________________________________________________________
_____
## dense (Dense)                        (None, 1)                       33
## ========================================================================
======
## Total params: 1,004,289
## Trainable params: 1,004,289
## Non-trainable params: 0
## ________________________________________________________________________
_____
```

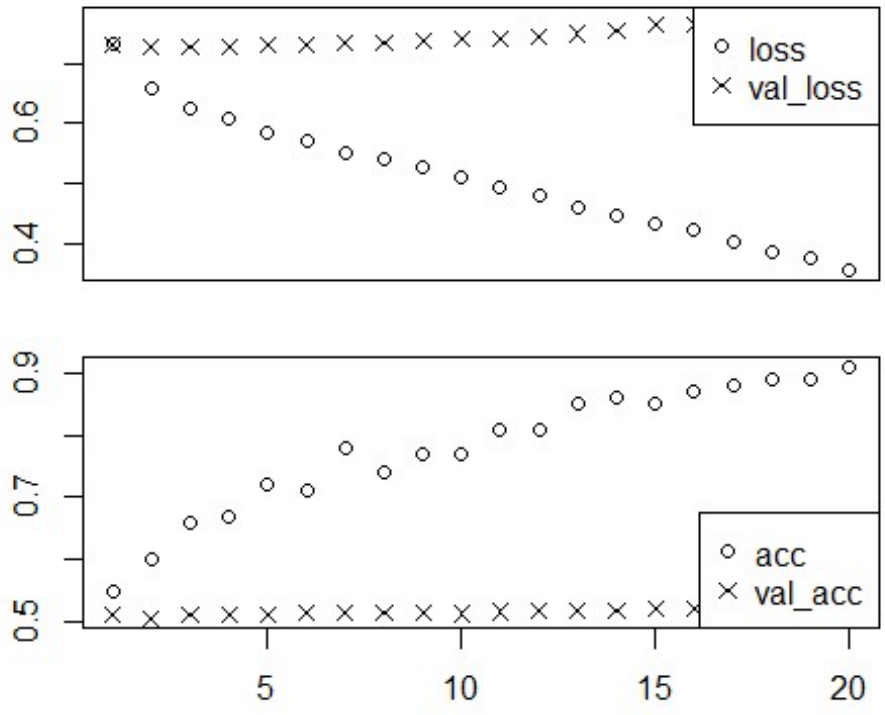
Loading pretrained word embeddings into the embedding layer

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

Training and 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)
```



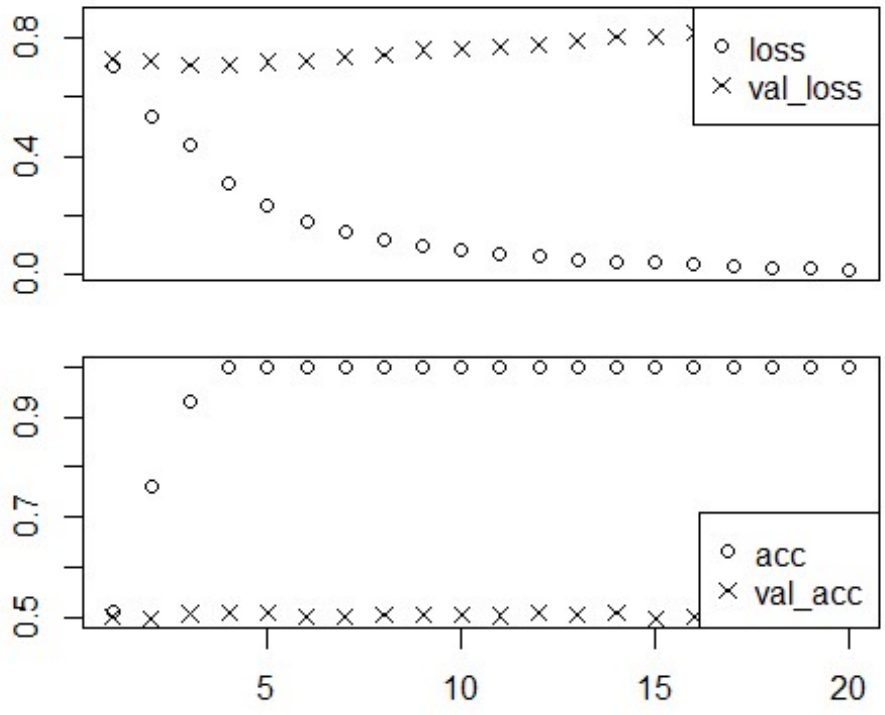
Training the same model without pretrained word embeddings

```
model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")


model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history2 <- model2 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)

plot(history2)
```



Tokenizing the data of the test set

```
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)

results2 <- model2 %>% evaluate(x_test, y_test)
results2

##      loss       acc
## 0.8566806 0.5072000
```

Evaluating the model on the test set

```
model %>%
  load_model_weights_hdf5("pre_trained_glove_model.h5") %>%
  evaluate(x_test, y_test)
```

```
##      loss       acc
## 0.7793868 0.5156000
```

TASK 02 - (Using RNN – Training samples=500)

Processing the labels of the raw IMDB data

```
library(keras)


imdb_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/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)
  }
}
```

Tokenizing the text of the raw IMDB data

```
maxlen <- 500
training_samples <- 500
validation_samples <- 10000
max_words <- 10000

tokenizer <- text_tokenizer(num_words = max_words) %>%
  fit_text_tokenizer(texts)

## Loaded Tensorflow version 2.6.0

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 500

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

## Shape of label tensor: 25000

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]
```

Pre-trained Glove Embedding Model

Parsing the GloVe word-embeddings file

```
glove_dir = "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/"
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))
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.
```

Preparing the GloVe word-embeddings matrix

```
embedding_dim <- 100
embedding_matrix <- array(0, c(max_words, embedding_dim))
for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < max_words) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

Model definition

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
_____
## Layer (type)                        Output Shape                      Param
```

```
#
## ================================================================================
======
## embedding (Embedding)                (None, 500, 100)                  100000
0
## ________________________________________________________________________________
______
## simple_rnn_3 (SimpleRNN)             (None, 500, 32)                   4256
## ________________________________________________________________________________
______
## simple_rnn_2 (SimpleRNN)             (None, 500, 32)                   2080
## ________________________________________________________________________________
______
## simple_rnn_1 (SimpleRNN)             (None, 500, 32)                   2080
## ________________________________________________________________________________
______
## simple_rnn (SimpleRNN)               (None, 32)                        2080
## ________________________________________________________________________________
______
## dense (Dense)                        (None, 1)                         33
## ================================================================================
======
## Total params: 1,010,529
## Trainable params: 1,010,529
## Non-trainable params: 0
## ________________________________________________________________________________
______
```

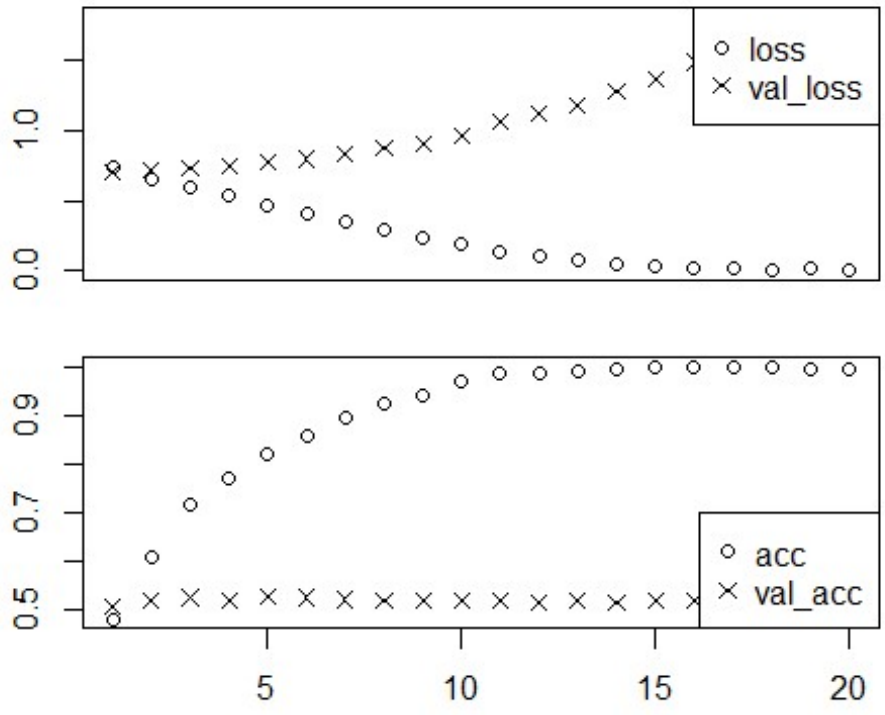
Loading pretrained word embeddings into the embedding layer

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

Training and 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)
```



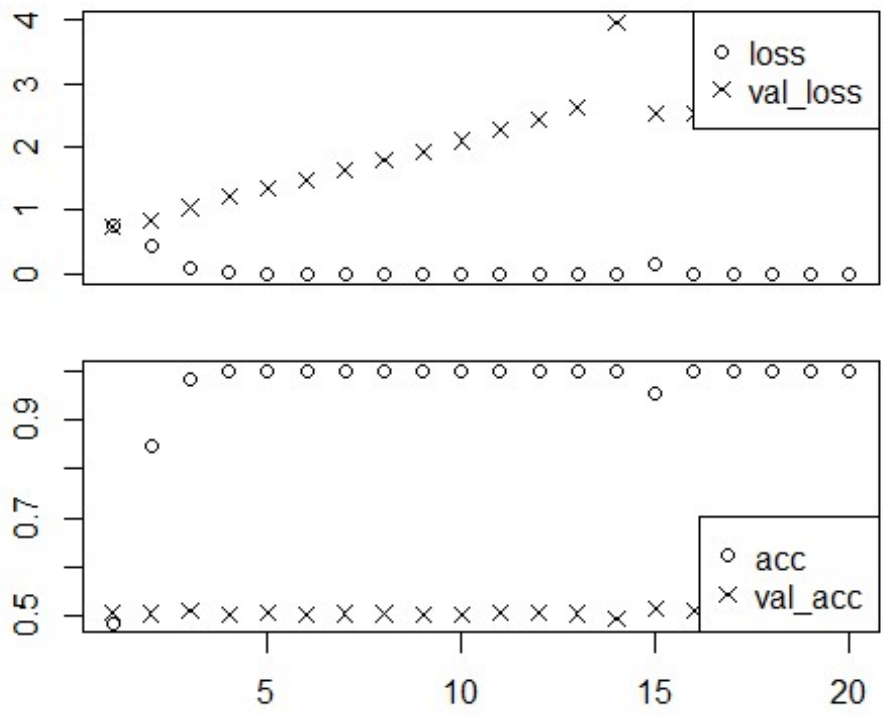
Training the same model without pretrained word embeddings

```
model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")


model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history2 <- model2 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)

plot(history2)
```



Tokenizing the data of the test set

```
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)

results2 <- model2 %>% evaluate(x_test, y_test)
results2

##     loss      acc
## 2.596861 0.506880
```

Evaluating the model on the test set

```
model %>%
  load_model_weights_hdf5("pre_trained_glove_model.h5") %>%
  evaluate(x_test, y_test)
```

```
##     loss      acc
## 1.767162 0.526520
```

TASK 03 - (Using RNN & Increase training samples=10000)

Processing the labels of the raw IMDB data

```
library(keras)


imdb_dir <- "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/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)
  }
}
```

Tokenizing the text of the raw IMDB data

```
maxlen <- 500
training_samples <- 10000
validation_samples <- 10000
max_words <- 10000

tokenizer <- text_tokenizer(num_words = max_words) %>%
  fit_text_tokenizer(texts)

## Loaded Tensorflow version 2.6.0

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 500

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

## Shape of label tensor: 25000

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]
```

Pre-trained Glove Embedding Model

Parsing the GloVe word-embeddings file

```
glove_dir = "E:/Assignments and Tasks/Xan - Advance Machine Learning/Module 0
5/Data/"
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))
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.
```

Preparing the GloVe word-embeddings matrix

```
embedding_dim <- 100
embedding_matrix <- array(0, c(max_words, embedding_dim))
for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < max_words) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

Model definition

```
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
_____
## Layer (type)                        Output Shape                      Param
```

```
#
## ================================================================================
======
## embedding (Embedding)                (None, 500, 100)                  100000
0
## ________________________________________________________________________________
______
## simple_rnn_3 (SimpleRNN)             (None, 500, 32)                   4256
## ________________________________________________________________________________
______
## simple_rnn_2 (SimpleRNN)             (None, 500, 32)                   2080
## ________________________________________________________________________________
______
## simple_rnn_1 (SimpleRNN)             (None, 500, 32)                   2080
## ________________________________________________________________________________
______
## simple_rnn (SimpleRNN)               (None, 32)                        2080
## ________________________________________________________________________________
______
## dense (Dense)                        (None, 1)                         33
## ================================================================================
======
## Total params: 1,010,529
## Trainable params: 1,010,529
## Non-trainable params: 0
## ________________________________________________________________________________
______
```

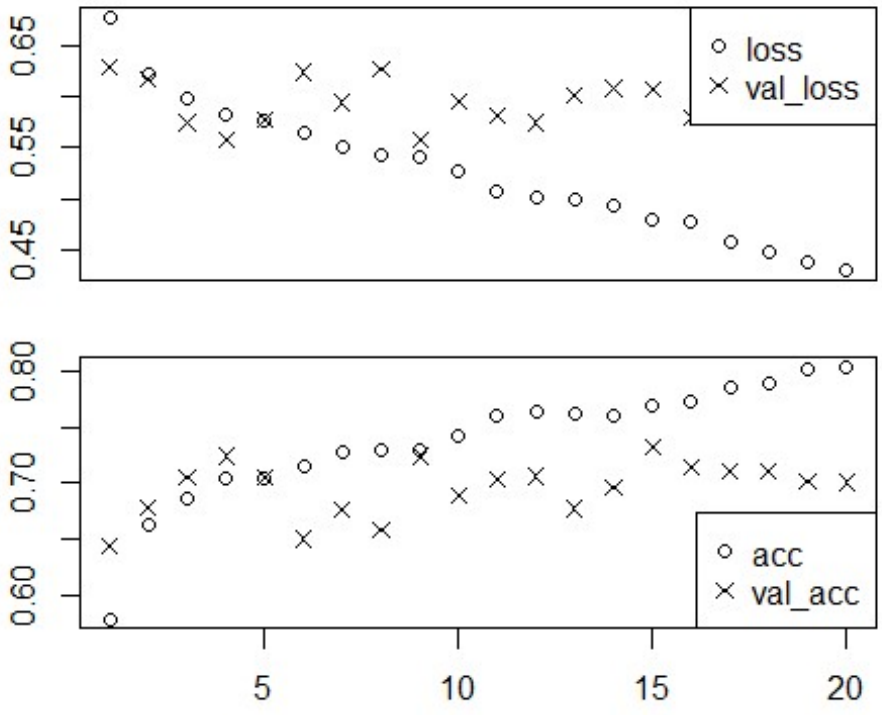
Loading pretrained word embeddings into the embedding layer

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

Training and 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)
```



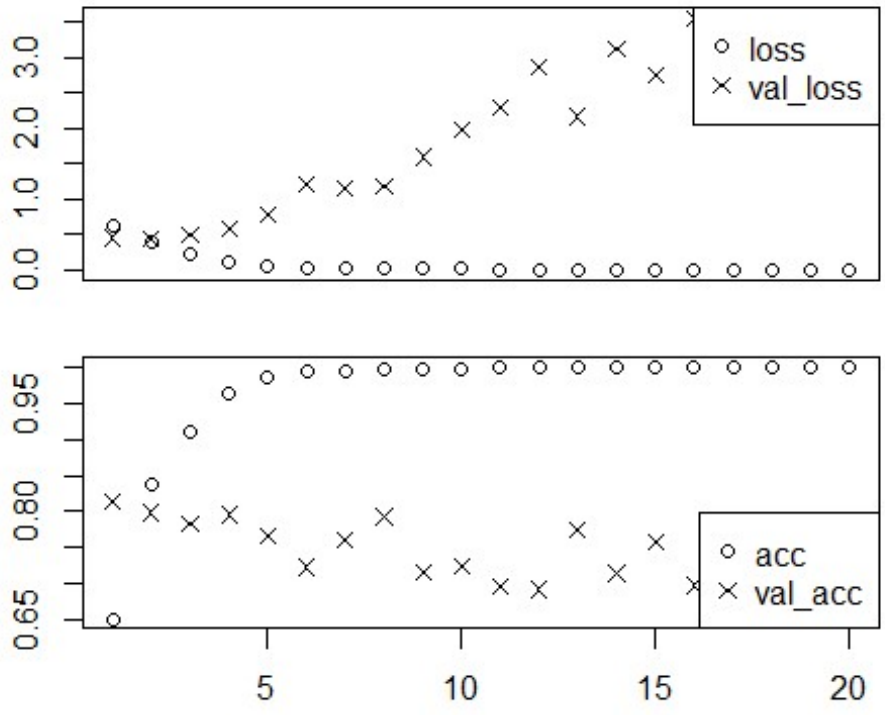
Training the same model without pretrained word embeddings

```
model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32, return_sequences = TRUE) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")


model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history2 <- model2 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)

plot(history2)
```



Tokenizing the data of the test set

```
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)

results2 <- model2 %>% evaluate(x_test, y_test)
results2

##     loss      acc
## 3.317801 0.751160
```

Evaluating the model on the test set

```
model %>%
  load_model_weights_hdf5("pre_trained_glove_model.h5") %>%
  evaluate(x_test, y_test)
```

```
##      loss       acc
## 0.6346291 0.7012800
```

Summary:

	Embedding Layer		Pre-trained Glove	
	Loss	**Accuracy**	**Loss**	**Accuracy**
Task 0 Training samples=**100 Without RNN**	0.7763946	0.5173600	1.208684	0.530280
Task 1 Training samples=**100 With RNN**	0.8566806	0.5072000	0.7793868	0.5156000
Task 2 Training samples=**500 With RNN**	2.596861	0.506880	1.767162	0.526520
Task 3 Training samples=**10000 With RNN**	3.317801	0.751160	0.6346291	0.7012800

The summary table shows that whenever using a small dataset the model did not pass the accuracy above 53% but as soon as the training sample size increase to 10000 then accuracy will reach 75%. When using the embedding layer with the RNN model the model gives an accuracy of 75.12% while loss is high which is 3.32. However, when using the pre-trained glove model, the model accuracy is 70.13% and loss is less which is 0.63 compared to the embedding layer.

The model's results show that it will work better with large training samples. In different training samples pre-trained model outclasses the embedding layer model, which is also expected for the large training samples, but the result shows that the embedding layer has higher accuracy which is 75% approximately and the pre-trained Glove model has an accuracy of 70% approximately.