Sentiment analysis

```
library(keras)
```

```
dat<-read.csv("~/Docencia/curs21_22/UB/MESIO/DL/sa.csv",header=F, sep=";")
```

```
texts<-dat[,1]
labels<-dat[,2]
```

```
maxlen <- 25
max_words <- 5000
tokenizer <- text_tokenizer(num_words = max_words) %>%
fit_text_tokenizer(texts)
sequences <- texts_to_sequences(tokenizer, texts)
word_index = tokenizer$word_index # les paraules que tenim en la nostra base de dades
```

```
texts[1:5]
```

```
## [1] "Wow... Loved this place."
## [2] "Crust is not good."
## [3] "Not tasty and the texture was just nasty."
## [4] "Stopped by during the late May bank holiday off Rick Steve recommendation and loved it."
## [5] "The selection on the menu was great and so were the prices."
```

```
#names(word_index)
sequences[1:5]
```

```
## [[1]]
## [1] 507 268   9  26
##
## [[2]]
## [1] 967   7  11  17
##
## [[3]]
## [1]  11 269   2   1 730   8  50 731
##
## [[4]]
##  [1]  732  112  396    1  968  354 1427 1428  136 1429 1430  733    2  268    5
##
## [[5]]
##  [1]   1 291  21   1 195   8  18   2  27  40   1 292
```

```
cat("Found", length(word_index), "unique tokens.\n")
```

```
## Found 3258 unique tokens.
```

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

```
## [1] 2000   25
```

```
data[5,]
```

```
##  [1]   0   0   0   0   0   0   0   0   0   0   0   0   0   1 291  21   1 195   8
## [20]  18   2  27  40   1 292
```

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

```
## Shape of data tensor: 2000 25
```

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

```
## Shape of label tensor: 2000
```

```
training_samples <- 1500 # small subset of examples
validation_samples <- 500
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]
```

```
embedding_dim<-20
model <- keras_model_sequential() %>%
layer_embedding(input_dim = max_words, output_dim = embedding_dim, input_length = maxlen) %>%
layer_flatten() %>%
layer_dense(units = 10, activation = "relu") %>%
layer_dense(units = 1, activation = "sigmoid")
```

```
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## embedding (Embedding)               (None, 25, 20)                  100000
## ________________________________________________________________________________
## flatten (Flatten)                   (None, 500)                     0
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 10)                      5010
## ________________________________________________________________________________
## dense (Dense)                       (None, 1)                       11
## ================================================================================
## Total params: 105,021
## Trainable params: 105,021
## Non-trainable params: 0
## ________________________________________________________________________________
```

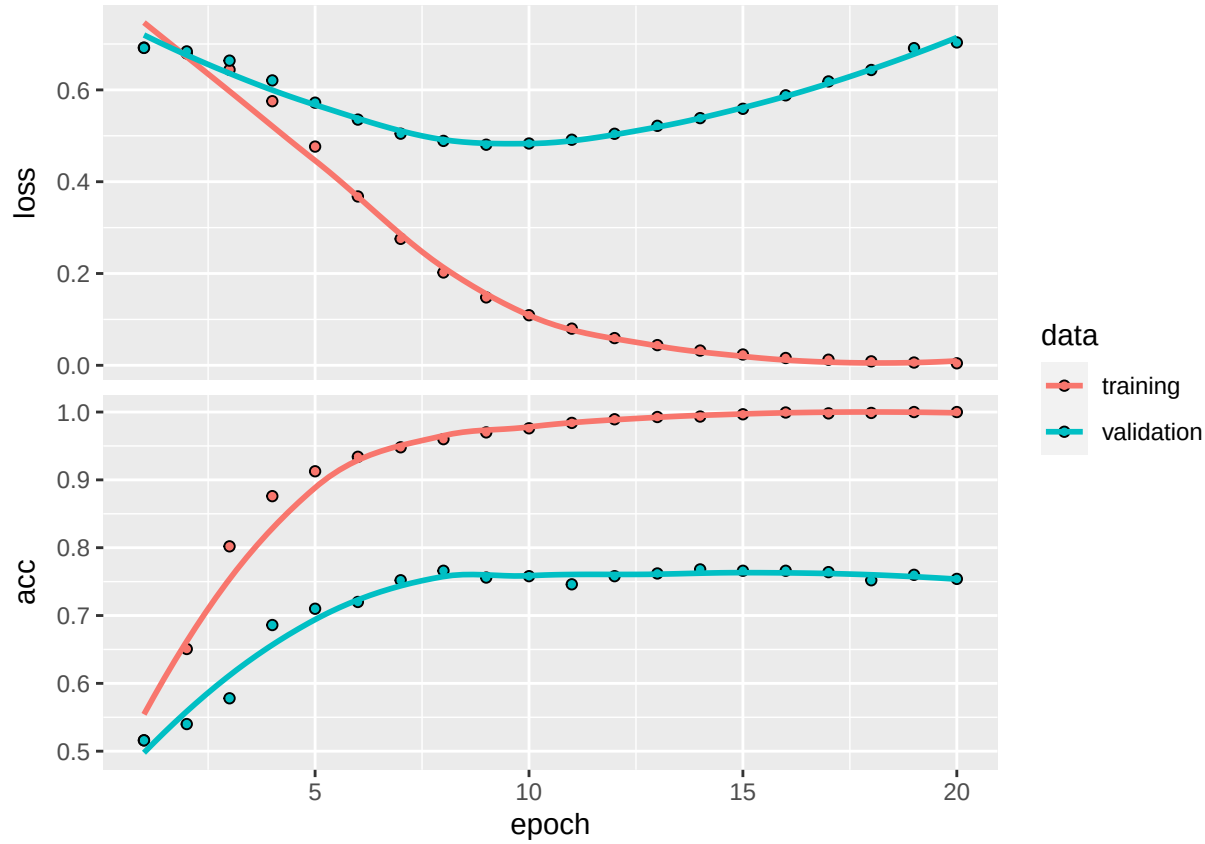
```
model %>% compile(
optimizer = "rmsprop",
loss = "binary_crossentropy",
metrics = c("acc")
)
```

```
history <- model %>% fit(
x_train, y_train,
```

```
epochs = 20,
batch_size = 32,
validation_data = list(x_val, y_val)
)
```

```
plot(history)
```

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




```
model2 <- keras_model_sequential() %>%
layer_embedding(input_dim = max_words, output_dim = embedding_dim) %>%
#layer_gru(units = 28) %>%
layer_simple_rnn(units = 28) %>%
layer_dense(units = 1, activation = "sigmoid")
summary(model2)
```

```
## Model: "sequential_1"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## embedding_1 (Embedding)             (None, None, 20)                100000
## ________________________________________________________________________________
## simple_rnn (SimpleRNN)              (None, 28)                      1372
## ________________________________________________________________________________
## dense_2 (Dense)                     (None, 1)                       29
## ================================================================================
## Total params: 101,401
```

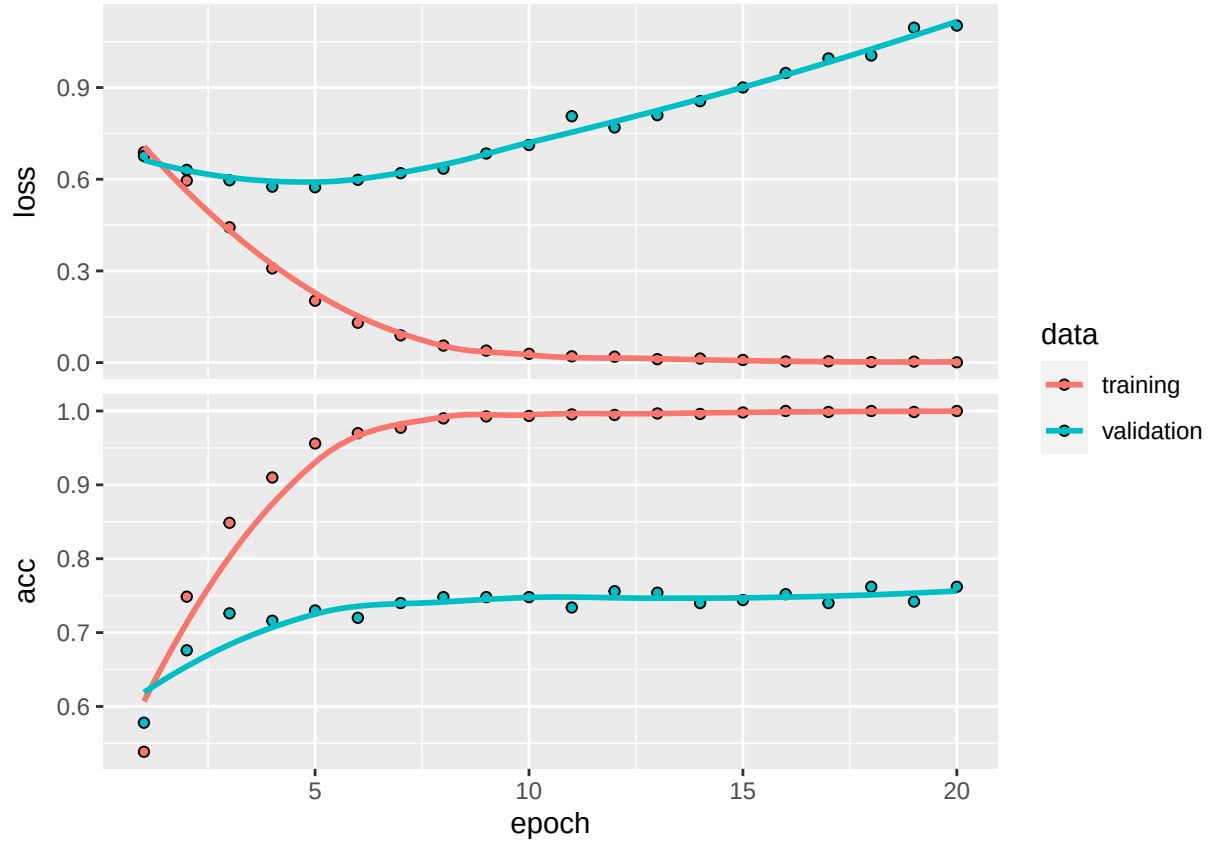
```
## Trainable params: 101,401
## Non-trainable params: 0
## _____________________________________________________________________________
```

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

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

```
plot(history)
```

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




```
glove_dir<-"~/Docencia/curs21_22/UB/MESIO/DL"
lines <- readLines(file.path(glove_dir,"glove.6B.50d.txt"))
```

```
embeddings_index <- new.env(hash = TRUE, parent = emptyenv())
# processat per obtenir un format llista del word embedding
for (i in 1:length(lines)) {
line <- lines[[i]]
```

```
values <- strsplit(line, " ")[[1]]
word <- values[[1]] # la paraula
embeddings_index[[word]] <- as.double(values[-1]) # les coordenades de la paraula
}
cat("Found", length(embeddings_index), "word vectors.\n")
```

```
## Found 400000 word vectors.
```

```
embeddings_index[["after"]]
```

```
##  [1]  0.3831500 -0.3561000 -0.1283000 -0.1952700  0.0476290  0.2146800
##  [7] -0.9876500  0.8296200 -0.4278200 -0.2287900  0.1071200 -0.3087000
## [13] -1.2069000 -0.1771300  0.8884100  0.0056658 -0.7730500 -0.6691300
## [19] -1.3384000  0.3467600  0.5044000  0.5125000  0.2682600 -0.6531300
## [25] -0.0815160 -2.1658000  0.5797400  0.0363450  0.0090949  0.2577200
## [31]  3.4402000  0.2073200 -0.5202800  0.0264530  0.1789500 -0.0178020
## [37]  0.3660500  0.3453900  0.4135700 -0.2497000 -0.4922700  0.1774500
## [43] -0.4376400 -0.3484000 -0.0570610 -0.0395780 -0.1351700 -0.4258000
## [49]  0.1368100 -0.7773100
```

```
embedding_dim <- 50 # dimension compatible with pretrained embedding
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
}
}
dim(embedding_matrix)
```

```
## [1] 5000   50
```

```
model3 <- keras_model_sequential() %>%
layer_embedding(input_dim = max_words, output_dim = embedding_dim) %>%
#layer_gru(units = 28) %>%
layer_simple_rnn(units = 28) %>%
layer_dense(units = 1, activation = "sigmoid")
summary(model2)
```

```
## Model: "sequential_1"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## embedding_1 (Embedding)             (None, None, 20)                100000
## ________________________________________________________________________________
## simple_rnn (SimpleRNN)              (None, 28)                      1372
## ________________________________________________________________________________
## dense_2 (Dense)                     (None, 1)                       29
## ================================================================================
## Total params: 101,401
## Trainable params: 101,401
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
get_layer(model3, index = 1) %>%
set_weights(list(embedding_matrix)) %>%
freeze_weights()
summary(model3)
```

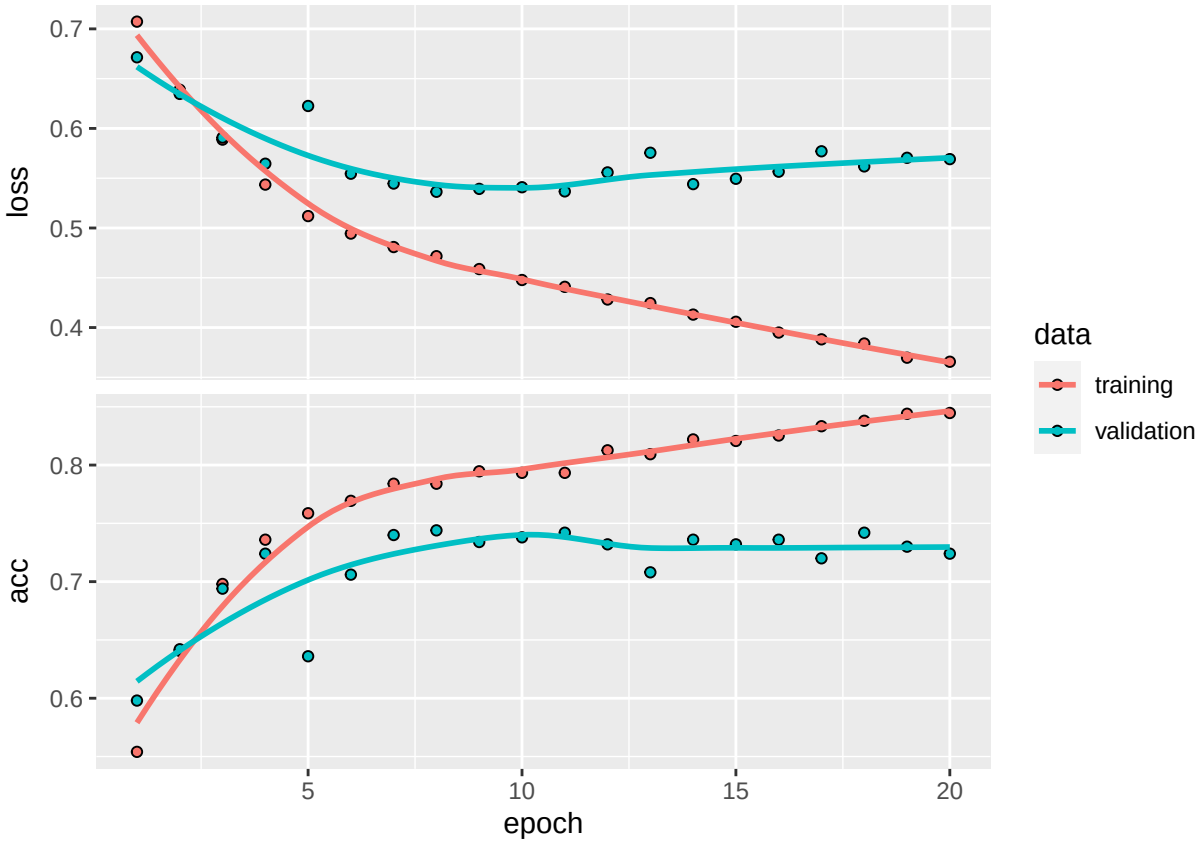
```
## Model: "sequential_2"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## embedding_2 (Embedding)      (None, None, 50)          250000
## _________________________________________________________________________
## simple_rnn_1 (SimpleRNN)     (None, 28)                2212
## _________________________________________________________________________
## dense_3 (Dense)              (None, 1)                 29
## =========================================================================
## Total params: 252,241
## Trainable params: 2,241
## Non-trainable params: 250,000
## _________________________________________________________________________
```

```
model3 %>% compile(
optimizer = "rmsprop",
loss = "binary_crossentropy",
metrics = c("acc")
)
```

```
history <- model3 %>% fit(
x_train, y_train,
epochs = 20,
batch_size = 32,
validation_data = list(x_val, y_val)
)
```

```
plot(history)
```

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


loss
0.7
0.6
0.5
0.4
acc
0.8
0.7
0.6
5
10
15
20
epoch
data
training
validation