Experiment

Load data

```
dataset_hs <- read.csv(file="dataset_dummy_classes5.csv", header=TRUE, sep=",")
```

Data conversion and division in train and test set

```
#library(kerasR)

test_train_preparation <- function(){
  # convert to binary
  dataset_hs$Hate_speech <- as.integer(dataset_hs$Hate.speech)
  dataset_hs$Sexism <- as.integer(dataset_hs$Sexism)
  dataset_hs$Racism <- as.integer(dataset_hs$Racism)

  # Split the data into a training set and a validation set
  # But assure shuffled data
  training_samples <- as.integer(0.7*nrow(dataset_hs))      # We will be training on 200 samples # divi
  validation_samples <- nrow(dataset_hs) - training_samples   # We will be validating on 10000 samples

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

  # convert to binary
  labels_hate_speech <- as.integer(dataset_hs$Hate_speech)
  labels_sexism <- as.integer(dataset_hs$Sexism)
  labels_racism <- as.integer(dataset_hs$Racism)
  labels_hate_speech <- as.array(labels_hate_speech)
  labels_sexism <- as.array(labels_sexism)
  labels_racism <- as.array(labels_racism)

  x_train <<- dataset_hs[training_indices,]
  y_train_hate_speech <<- labels_hate_speech[training_indices]
  y_train_sexism <<- labels_sexism[training_indices]
  y_train_racism <<- labels_racism[training_indices]

  x_val <<- dataset_hs[validation_indices,]
  y_val_hate_speech <<- labels_hate_speech[validation_indices]
  y_val_sexism <<- labels_sexism[validation_indices]
  y_val_racism <<- labels_racism[validation_indices]

}

test_train_preparation()
remove(dataset_hs)
```

Training data pre-processing

```
# Remove RT and links.

data_pre_processing <- function(dataset){
  # Get rid of URLs
  dataset$text <- gsub(" ?(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", "", dataset$text)
  # Take out retweet header, there is only one
  dataset$text <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", dataset$text)
  return(dataset)
}

x_train <- data_pre_processing(x_train)
```

word embedding

```
# from https://github.com/kylehamilton/deep-learning-with-r-notebooks/blob/master/notebooks/6.1-using-w

#reticulate::use_python("/Users/paulafortuna/anaconda3/envs/r-tensorflow/bin/python")

maxlen <- 100                 # We will cut reviews after 100 words
max_words <- 10000            # We will only consider the top 10,000 words in the dataset

library(keras)
tokenization <- function(maxlen, max_words, dataset, file_sufix){

  tokenizer <- keras::text_tokenizer(num_words = max_words) %>%
    keras::fit_text_tokenizer(dataset$text)
  keras::save_text_tokenizer(tokenizer, paste0("tokenizers/tokenizer", file_sufix))

  sequences <- keras::texts_to_sequences(tokenizer, dataset$text)

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

  data <- keras::pad_sequences(sequences, maxlen = maxlen)
  return(data)
}

x_train <- tokenization(maxlen, max_words, x_train, "_pt")
```

```
## Found 11018 unique tokens.
```

Recurrent Neural Networks (RNN) including a Long-Short Term Memory Unit (LSTM)

```
lstm_model <- function(x_train, y_train, file_prefix){

  # extracted from https://cran.r-project.org/web/packages/kerasR/vignettes/introduction.html
  X_train <- keras::pad_sequences(x_train, maxlen = 100)
  Y_train <- y_train
```

```
  library(kerasR)

  # including a Long-Short Term Memory Unit (LSTM)
  mod <- Sequential()

  mod$add(Embedding(10000, 32, input_length = 100, input_shape = c(100)))
  mod$add(Dropout(0.25))

  mod$add(LSTM(32))

  mod$add(Dense(256))
  mod$add(Dropout(0.25))
  mod$add(Activation('relu'))

  mod$add(Dense(1))
  mod$add(Activation('sigmoid'))

  keras_compile(mod,  loss = 'binary_crossentropy', optimizer = RMSprop(lr = 0.00025))
  keras_fit(mod, X_train, Y_train, batch_size = 32, epochs = 10, verbose = 1,
            validation_split = 0.1)

  filename <- paste0("models/", file_prefix)
  filename <- paste0(filename, ".h5")
  keras_save(mod, path = filename)
}

lstm_model(x_train, y_train_hate_speech, "hate_pt")
```

```
## successfully loaded keras
```

```
##
## Attaching package: 'kerasR'
```

```
## The following objects are masked from 'package:keras':
##
##     normalize, pad_sequences, text_to_word_sequence,
##     to_categorical
```

```
lstm_model(x_train, y_train_sexism, "sexism_pt")
lstm_model(x_train, y_train_racism, "racism_pt")
```

```
test_model <- function(model_name, x_test, tokenizer_name){
  reticulate::use_python("/Users/paulafortuna/anaconda3/envs/r-tensorflow/bin/python")

  #read tokenizer
  tokenizer <- keras::load_text_tokenizer(tokenizer_name)

  #apply tokenizer to the validation data
  sequences <- keras::texts_to_sequences(tokenizer, x_val$text)
  x_test <- keras::pad_sequences(sequences, maxlen = 100)

  # read model based on hate_type and language
  mod <- keras_load(path = model_name)

  # apply model to the validation data
```

```
  Y_test_hat <- kerasR::keras_predict(mod, x_test)

  return(Y_test_hat)

}

Y_test_hate <- test_model("models/hate_pt.h5", x_val, "tokenizers/tokenizer_pt")
Y_test_sexism <- test_model("models/sexism_pt.h5", x_val, "tokenizers/tokenizer_pt")
Y_test_racism <- test_model("models/racism_pt.h5", x_val, "tokenizers/tokenizer_pt")
```

Confusion matrices

```
cm_hate <- caret::confusionMatrix(as.factor(round(Y_test_hate)), as.factor(y_val_hate_speech), positive=
cm_sexism <- caret::confusionMatrix(as.factor(round(Y_test_sexism)), as.factor(y_val_sexism), positive=
cm_racism <- caret::confusionMatrix(as.factor(round(Y_test_racism)), as.factor(y_val_racism), positive=
```

```
## Warning in confusionMatrix.default(as.factor(round(Y_test_racism)),
## as.factor(y_val_racism), : Levels are not in the same order for reference
## and data. Refactoring data to match.
```

Performance metrics

```
metrics_calculation <- function(cm){
  TP <- cm$table[2,2]
  TN <- cm$table[1,1]
  FP <- cm$table[2,1]
  FN <- cm$table[1,2]

  precision <- TP / (TP + FP)
  recall <- TP / (TP + FN)
  F1 <- 2*(precision*recall)/(precision + recall)

  print(precision)
  print(recall)
  print(F1)
}

metrics_calculation(cm_hate)
```

```
## [1] 0.71875
## [1] 0.2222222
## [1] 0.3394834
```

```
metrics_calculation(cm_sexism)
```

```
## [1] 0.7282609
## [1] 0.1666667
## [1] 0.2712551
```

```
metrics_calculation(cm_racism)
```

```
## [1] NaN
## [1] 0
```

```
## [1] NaN
```

}