Classification K-Nearest-Neighbors

Chris Bentz

16/01/2023

Load Packages

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2"). For the Hrate package this is different, since it comes from github. The devtools library needs to be installed, and then the install_github() function is used.

```
library(ggplot2)
library(dplyr)
library(class)
library(gridExtra)
library(gmodels)
library(caret)
library(ggExtra)
library(ggpubr)
```

Load Data

Load data table with values per text file.

```
# load estimations from stringBase corpus
estimations.df <- read.csv("~/Github/NaLaFi/results/features.csv")
#head(features.csv)
```

Exclude subcorpora (if needed).

```
#selected <- c("random","shuffled")
#estimations.df <- estimations.df[!(estimations.df$subcorpus %in% selected), ]
```

Split into separate files by length of chunks in characters.

```
# choose number of characters
num.char = 100
# subset data frame
estimations.df <- estimations.df[estimations.df$num.char == num.char, ]
```

Select relevant columns of the data frame, i.e. the measures to be included in classification and the ''corpus" or ''subcorpus" column.

```
estimations.subset <- estimations.df[c("corpus", "subcorpus",
                                       "huni.chars",
                                       "hrate.chars",
                                       "ttr.chars",
```

```
                                        "rm.chars"
                                        )]
```

Remove NAs (whole row)

```
estimations.subset <- na.omit(estimations.subset)
```

Center and scale the data

```
estimations.scaled <- cbind(estimations.subset[1:2], scale(estimations.subset[3:ncol(estimations.subset)
```

Create Training and Test Sets

```
# Generating seed
set.seed(1234)
# Randomly generating our training and test samples with a respective ratio of 2/3 and 1/3
datasample <- sample(2, nrow(estimations.scaled), replace = TRUE, prob = c(0.67, 0.33))
# Generate training set
estimations.training <- estimations.scaled[datasample == 1, 3:ncol(estimations.scaled)]
# Generate test set
estimations.test <- estimations.scaled[datasample == 2, 3:ncol(estimations.scaled)]
```

Get training and test labels

```
# Generate training labels
training.labels <- estimations.scaled[datasample == 1, 1]
# Generate test labels
test.labels <- estimations.scaled[datasample == 2, 1]
```

Initialize data frame

```
knn.results <- data.frame(k = numeric(0), precision = numeric(0),
                          recall = numeric(0), f1 = numeric(0))
```

Building knn classifier

```
# choose maximum number of neighbors n
n = 10
# run a loop over different numbers of neighbors up to n
for (k in 1:n){
  # knn estimation of labels
  predictions.knn <- knn(train = as.data.frame(estimations.training),
                         test = as.data.frame(estimations.test),
```

```
                        cl = training.labels, k = k)

  # model evaluation
  # creating a dataframe from known (true) test labels
  test.labels <- data.frame(test.labels)
  # combining predicted and known classes
  class.comparison <- data.frame(predictions.knn, test.labels)
  # giving appropriate column names
  names(class.comparison) <- c("predicted", "observed")
  # inspecting our results table
  head(class.comparison)
  # get confusion matrix
  cm <- confusionMatrix(class.comparison$predicted,
                        reference = class.comparison$observed)
  # print(cm)
  # get precision, recall, and f1 from the output list of confusionMatrix()
  f1 <- cm[["byClass"]]["F1"]
  recall <- cm[["byClass"]]["Recall"]
  precision <- cm[["byClass"]]["Precision"]

  # prepare data frame with results
  local.results <- data.frame(k, precision, recall, f1, row.names = NULL)
  local.results.rounded <- round(local.results, 2)
  # print(local.results.rounded)
  knn.results <- rbind(knn.results, local.results.rounded)
}
print(knn.results)
```

```
##     k precision recall   f1
## 1   1      0.91   0.92 0.91
## 2   2      0.91   0.90 0.90
## 3   3      0.94   0.92 0.93
## 4   4      0.93   0.93 0.93
## 5   5      0.94   0.93 0.93
## 6   6      0.92   0.93 0.93
## 7   7      0.93   0.93 0.93
## 8   8      0.93   0.92 0.93
## 9   9      0.93   0.92 0.93
## 10 10      0.93   0.93 0.93
```

Write to file.

```
# write.csv(knn.results, file = paste("~/Github/NaLaFi/results/knn/knn_results_baselineTTR",
#                                     paste(num.char, ".csv", sep =""),
#                                     sep =""), row.names = F)
```

Visualization

Create dataframe with test data and predictions.

```
plot.df <- cbind(estimations.test, test.labels, predictions.knn)
colnames(plot.df) <- c("huni.chars", "hrate.chars", "ttr.chars",
                       "rm.chars", "observed", "predicted")
```

Unigram entropy vs. TTR for characters

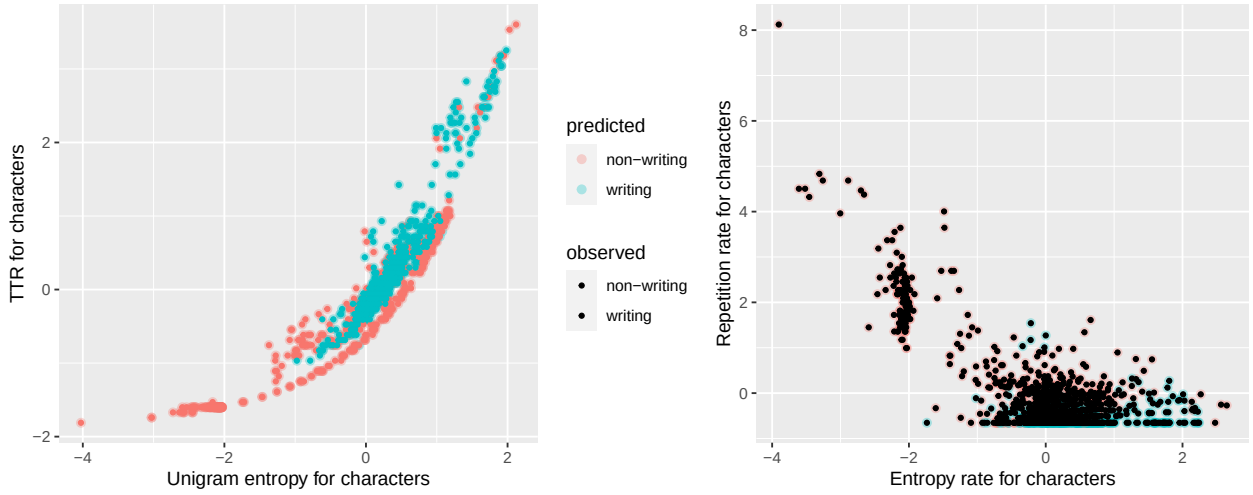
```
huni.hrate.plot <- ggplot(plot.df,
                                aes(x = huni.chars, y = ttr.chars)) +
  geom_point(aes(colour = predicted), size  = 2, alpha = 0.3) +
  geom_point(aes(colour = observed), size  = 1) +
  labs(x = "Unigram entropy for characters", y = "TTR for characters") +
  theme(legend.position = "none")
#huni.hrate.plot
```

Entropy rate vs. repetition rate for characters

```
ttr.rm.plot <- ggplot(plot.df,
                    aes(x = hrate.chars, y = rm.chars)) +
  geom_point(aes(colour = predicted), size  = 2, alpha = 0.3) +
  geom_point(aes(fill = observed), size  = 1) +
  theme(legend.position = "left") +
  labs(x = "Entropy rate for characters", y = "Repetition rate for characters")
#ttr.rm.plot
```

Combined Plots

```
plots.combined <- ggarrange(huni.hrate.plot, ttr.rm.plot,
                    ncol = 2, nrow = 1, widths = c(1, 1.3))
plots.combined
```



Safe complete figure to file

```
ggsave("~/Github/NaLaFi/figures/plots_knn_combined.pdf", plots.combined, width = 10,
       height = 4, dpi = 300, scale = 1, device = cairo_pdf)
```