# TEXTUAL ANALYSIS WITH R

# 1. Introduction

The objective of this practice is to provide students with a global vision of textual analysis. Specifically, data from the Author Profiling task of the PAN in the CLEF will be used to construct a classifier that discriminates authors by gender and variety of language, according to what they write.

To carry out the practice, R 3.4.0 and the following libraries are required:

- caret: machine learning
- qdap: natural language processing
- tm: text mining
- splitstackshape: data frame manipulation
- e1071: auxiliary statistical functions
- kernlab: support vector machine kernels
- readr: auxiliary functions to read files

In this practice we will build a representation-based bag of words weighted with absolute frequencies, taking the 1,000 most frequent words from the training corpus. We will preprocess the text to pass it to lowercase, eliminate numbers and punctuation marks and eliminate empty words.

# 2. Dataset

This section describes the dataset.

## 2.1. Dataset construction methodology

The following methodology has been followed to collect the data:

**Step 1. Selection of languages and varieties of language**

The following languages and their varieties are selected:

- English: Australia, Canada, Great Britain, Ireland, New Zealand, United States
- Spanish: Argentina, Chile, Colombia, Spain, Mexico, Peru, Venezuela
- Portuguese: Brazil, Portugal
- Arabic: Egypt, Gulf, Levantine, Maghreb

The selection has been made following the state of the art. For the Arabic we also selected the Iraqi, which we had to discard since it did not have enough tweets.

**Step 2. Retrieve tweets by region**

For each variety, select the capitals of the region where the variety is used. Specifically:

| Language | Variety | City |
|---|---|---|
| EN | Australia | Canberra, Sydney |
| | Canada | Toronto, Vancouver |
| | Great Britain | London, Edinburgh, Cardiff |
| | Ireland | Dublin |
| | New Zealand | Wellington |
| | United States | Washington |
| ES | Argentina | Buenos Aires |
| | Chile | Santiago |
| | Colombia | Bogota |
| | Mexico | Mexico |
| | Peru | Lima |
| | Spain | Madrid |
| | Venezuela | Caracas |
| PT | Brazil | Brasilea |
| | Portugal | Lisbon |
| AR | Egypt | Cairo |
| | Gulf | Kuwait, Riyadh, Abu Dhabi, Doha, Manama, Mascate, Sana'a |
| | Levantine | Damascus, Beirut, Amman, Jerusalem |
| | Maghrebi | Rabat, Algiers, Tunis, Tripoli |

Tweets are retrieved within a 10km radius from the center of the previous cities.

**Step 3. Selection of authors**

Unique authors are identified in the previous dataset. For each author, their timeline is retrieved, which provides meta-data such as the following:

- Full name
- Location, as a text description or place names

- Language identified by the author, which may not correspond to the language in which he writes the tweets

**Step 4. Preselection of authors**

For each author we make sure that it contains at least 100 tweets fulfilling the following conditions:

- Tweets are not retweets
- The tweets are written in the corresponding language

**Step 5. Annotation of the language variety**

An author writes down with the corresponding variety if he complies that:

- It has recovered in the corresponding region
- At least 80% of the locations provided as meta-data of their tweets matches one of the toponyms of the corresponding region.

**NOTE:** The assumption is that a person living in a region uses the variety of the said region. This implies two other assumptions:

- We assume that a person lives in a region when their location throughout their timeline reflects that location. The timeline has up to 3,200 tweets per author, which implies in most cases a couple of years, so we consider feasible the assumption.

- We assume that language in social media is dynamic and easily influenced, against the more formal language used for example in the press. This means that it reflects the current and daily use of language, and how this reflects the basic social and personality processes of the author who uses it. In this sense, if there is a high number of immigrants in a region, they can influence the regional use of language, and this can give a valuable clue to detect the possible location of a person

**Step 6. Gender annotation**

The gender annotation is done in two steps:

- Automatically, by using a dictionary of proper names (ambiguous names are discarded)
- Manually, visiting each profile and looking at the photo, description, etc.

**Step 6. Final dataset selection**

The final dataset is balanced in the number of tweets by variety and by gender, as well as by the number of tweets per author.

- 500 tweets by gender and variety.
- 100 tweets per author
- The dataset is divided into training / test following the 80/20 ratio

## 2.2. Official Dataset of PAN'17

| ENGLISH | SPANISH | PORTUGUESE | ARABIC |
|---|---|---|---|
| • Australia<br>• Canada<br>• Great Britain<br>• Ireland<br>• New Zealand<br>• United States | • Argentina<br>• Chile<br>• Colombia<br>• Mexico<br>• Peru<br>• Spain<br>• Venezuela | • Brazil<br>• Portugal | • Egypt<br>• Gulf<br>• Levantine<br>• Maghrebi |

- 500 authors per gender and language variety
  - 300 training / 200 tests
- 100 tweets per author

## 2.3. Subset of the course

From the previous dataset, we selected the English training subset:

| ENGLISH |
|---|
| • Australia<br>• Canada<br>• Great Britain<br>• Ireland<br>• New Zealand<br>• United States |

- 300 authors by gender and language variety (from the original training set)
  - 200 training / 100 tests
- 100 tweets per author

The dataset is available at:

https://s3-eu-west-1.amazonaws.com/autoritas.academy/pan-ap17-training.zip

## 2.4. The format of the files

Once the dataset is decompressed, the structure is as follows:

- A couple of folders: training and test
- In each folder:
  - ✓ A truth.txt file with the truth information.
  - ✓ One xml file per author.Truth.txt
  - ✓ Id:::gender:::variety

```
e4c2bef9fcc41f2681fc502d6fba5703:::female:::canada
45e0b78a892812f89a7ef0626bf4d81:::female:::canada
3f951fa8e1340f6ed863ae78298bd1ab:::female:::canada
9ae581dda7fd9c3587b5c146ad9972b1:::female:::canada
5f2a1b80dfc890965384d9847cf46d4c:::female:::canada
c6abda9596747ddc85c8ae3a70d0b9ed:::female:::canada
4517849641711cc18606b7b8a99380e:::female:::canada
ba053834517708a073203aef1756e63e:::female:::canada
da1b54c48d3219b7913d23bce557fb58:::female:::canada
6c8dc5a74ad0b5faba9aab90907fef8e:::female:::canada
efb47ba3a0b0a35fcb24d989c7b1f8b:::female:::canada
f09bdd87bf3002aa3df93cdfa4b7c432:::female:::canada
193d6ccf6ad722c17e781c89c13325b:::female:::canada
a15c52ca5d6d6f8bb21ebf610b590ade:::female:::canada
b6c9d97d1bcba9303ba3c5d3af30100e:::female:::canada
9fba3236a9e2bcaa987322d52209192:::female:::canada
```

- ✓ The name of each xml file corresponds to an id in the file truth.txt
- ✓ Each xml file is an author with a set of 100 tweets with the following structure:

```
<author lang="en">
    <documents>
        <document><![CDATA[Euro banknote counterfeiting remains low in second half of 2016 https://t.co/Cik
        <document><![CDATA[@MeliaHtlResorts getting a bounce from your customer service email, is there anc
        <document><![CDATA[@ColmODonoghue thought D Ryan was badly missed yesterday]]></document>
        <document><![CDATA[@gavinmortimer7 i think there maybe a suprise on the cards....]]></document>
        <document><![CDATA[@IrishRugby no sound on presser on the website...]]></document>
        <document><![CDATA[@MOReganIT @IrishTimes at this stage he objects to everything shoudl be with AA
        <document><![CDATA[@DCCTraffic absolute disgrace that this is happening during Sat daytime! Traffic
        <document><![CDATA[I #ScrumTogether with @DoveMen to win #IREvFRA RBS 6 Nations tickets. Pick your
https://t.co/GGoOUOaOxe]]></document>
```

# 3. Building the model

This section describes how to build the machine learning model through a bag of word based representation.

You can download the code snippets from the following url:

https://drive.google.com/file/d/1BgdBl7slClO3eAXlIiTvThUPhFGguhLj/view?usp=sharing

It is suggested to follow the following steps, although the fragment codes are copied and executed.

### 3.1. Library R

The following libraries must be included (after installation):

```
library(qdap)
library(XML)
library(tm)
library(splitstackshape)
library(caret)
```

### 3.2. Setting global parameters

```
n <- 10      # Number of words in the vocabulary. Usually used 1000 or 10000
k <- 3       # Number of folds in cross-validation. Usually used 10
r <- 1       # Number of repeats in cross-validation. Usually used 3
path_training <- "HERE YOU SHOULD PUT YOUR PATH/training"
path_test <- "HERE YOU SHOULD PUT YOUR PATH/test"
lang <- "en"
```

### 3.3. Getting the vocabulary

The following code is encapsulated within the GenerateVocabulary function:

```
# GenerateVocabulary: Given a corpus (training set), obtains the n most frequent words

GenerateVocabulary <- function(path, n = 1000, lowcase = TRUE, punctuations = TRUE,
numbers = TRUE, whitespaces = TRUE, swlang = "", swlist = "", verbose = TRUE) {
  setwd(path)

  # Reading corpus list of files
  files = list.files(pattern="*.xml")

  # Reading files contents and concatenating into the corpus.raw variable
  corpus.raw <- NULL
  i <- 0
  for (file in files) {
    xmlfile <- xmlTreeParse(file, useInternalNodes = TRUE)
    corpus.raw <- c(corpus.raw, xpathApply(xmlfile, "//document", function(x) xmlValue(x)))
    i <- i + 1
    if (verbose) print(paste(i, " ", file))
  }

  # Preprocessing the corpus
  corpus.preprocessed <- corpus.raw

  if (lowcase) {
    if (verbose) print("Tolower...")
    corpus.preprocessed <- tolower(corpus.preprocessed)
  }

  if (punctuations) {
    if (verbose) print("Removing punctuations...")
    corpus.preprocessed <- removePunctuation(corpus.preprocessed)
  }
```

```
  if (numbers) {
    if (verbose) print("Removing numbers...")
    corpus.preprocessed <- removeNumbers(corpus.preprocessed)
  }

  if (whitespaces) {
    if (verbose) print("Stripping whitestpaces...")
    corpus.preprocessed <- stripWhitespace(corpus.preprocessed)
  }

  if (swlang!="")           {
    if (verbose) print(paste("Removing stopwords for language ", swlang , "..."))
    corpus.preprocessed <- removeWords(corpus.preprocessed, stopwords(swlang))
  }

  if (swlist!="") {
    if (verbose) print("Removing provided stopwords...")
    corpus.preprocessed <- removeWords(corpus.preprocessed, swlist)
  }

  # Generating the vocabulary as the n most frequent terms
  if (verbose) print("Generating frequency terms")
  corpus.frequentterms <- freq_terms(corpus.preprocessed, n)
  if (verbose) plot(corpus.frequentterms)

  return (corpus.frequentterms)
}
```

Once the function has been defined, it should be called as follows:

***vocabulary <- GenerateVocabulary(path_training, n, swlang=lang)***

### 3.4. Generating the Bag of Word for the variety task in the training subset

The following code is encapsulated in the GenerateBoW function.

```
GenerateBoW <- function(path, vocabulary, n = 100000, lowcase = TRUE, punctuations =
TRUE, numbers = TRUE, whitespaces = TRUE, swlang = "", swlist = "", class="variety",
verbose = TRUE) {
  setwd(path)

  # Reading the truth file
  truth <- read.csv("truth.txt", sep=":", header=FALSE)
  truth <- truth[,c(1,4,7)]
  colnames(truth) <- c("author", "gender", "variety")

  i <- 0
  bow <- NULL
```

```
  # Reading the list of files in the corpus
  files = list.files(pattern="*.xml")
  for (file in files) {
    # Obtaining truth information for the current author
    author <- gsub(".xml", "", file)
    variety <- truth[truth$author==author,"variety"]
    gender <- truth[truth$author==author,"gender"]

    # Reading contents for the current author
    xmlfile <- xmlTreeParse(file, useInternalNodes = TRUE)
    txtdata <- xpathApply(xmlfile, "//document", function(x) xmlValue(x))

    # Preprocessing the text
    if (lowcase) {
      txtdata <- tolower(txtdata)
    }

    if (punctuations) {
      txtdata <- removePunctuation(txtdata)
    }

    if (numbers) {
      txtdata <- removeNumbers(txtdata)
    }

    if (whitespaces) {
      txtdata <- stripWhitespace(txtdata)
    }
```

**# Building the vector space model. For each word in the vocabulary, it obtains the frequency of occurrence in the current author.**

```
    line <- author
    freq <- freq_terms(txtdata, n)
    for (word in vocabulary$WORD) {
      thefreq <- 0
      if (length(freq[freq$WORD==word,"FREQ"])>0) {
        thefreq <- freq[freq$WORD==word,"FREQ"]
      }
      line <- paste(line, ",", thefreq, sep="")
    }

    # Concatenating the corresponding class: variety or gender
    if (class=="variety") {
      line <- paste(variety, ",", line, sep="")
    } else {
      line <- paste(gender, ",", line, sep="")
    }
```

```
    # New row in the vector space model matrix
    bow <- rbind(bow, line)
    i <- i + 1

    if (verbose) {
      if (class=="variety") {
        print(paste(i, author, variety))
      } else {
        print(paste(i, author, gender))
      }
    }
  }

  return (bow)
}
```

Once the function is defined, it should be called as follows:

```
# GENERATING THE BOW FOR THE GENDER SUBTASK FOR THE TRAINING SET
bow_training_gender <- GenerateBoW(path_training, vocabulary, class="gender")

# GENERATING THE BOW FOR THE GENDER SUBTASK FOR THE TEST SET
bow_test_gender <- GenerateBoW(path_test, vocabulary, class="gender")

# GENERATING THE BOW FOR THE VARIETY SUBTASK FOR THE TRAINING SET
bow_training_variety <- GenerateBoW(path_training, vocabulary, class="variety")

# GENERATING THE BOW FOR THE VARIETY SUBTASK FOR THE TEST SET
bow_test_variety <- GenerateBoW(path_test, vocabulary, class="variety")
```

### 3.5. Preparing the vector space model for the training and test subsets

We must do this twice, depending on the task (variety / gender):

```
# PREPARING THE VECTOR SPACE MODEL FOR THE TRAINING SET
training_gender <- concat.split(bow_training_gender, "V1", ",")
training_gender <- cbind(training_gender[,2], training_gender[,4:ncol(training_gender)])
names(training_gender)[1] <- "theclass"

# Preparing the vector space model and truth for the test set
test_gender <- concat.split(bow_test_gender, "V1", ",")
truth_gender <- unlist(test_gender[,2])
test_gender <- test_gender[,4:ncol(test_gender)]

# PREPARING THE VECTOR SPACE MODEL FOR THE TRAINING SET
training_variety <- concat.split(bow_training_variety, "V1", ",")
training_variety <- cbind(training_variety[,2], training_variety[,4:ncol(training_variety)])
```

```
names(training_variety)[1] <- "theclass"

# Preparing the vector space model and truth for the test set
test_variety <- concat.split(bow_test_variety, "V1", ",")
truth_variety <- unlist(test_variety[,2])
test_variety <- test_variety[,4:ncol(test_variety)]
```

### 3.6. Learning an SVM model

With the following lines you learn an SVM model and it is evaluated with cross-validation in k-folds, and we can print the results of the evaluation:

```
# Learning a SVM and evaluating it with k-fold cross-validation
train_control <- trainControl( method="repeatedcv", number = k , repeats = r)
model_SVM_gender <- train( theclass~., data= training_gender, trControl = train_control,
method = "svmLinear")
print(model_SVM_gender)

# Learning a SVM and evaluating it with 10-fold cross-validation
train_control <- trainControl( method="repeatedcv", number = k , repeats = r)
model_SVM_variety <- train( theclass~., data= training_variety, trControl = train_control,
method = "svmLinear")
print(model_SVM_variety)
```

And with the following lines, the SV model is learned with all the training without cross evaluation (faster):

```
# Learning a SVM with the whole training set and without evaluating it
train_control <- trainControl(method="none")
model_SVM_gender <- train( theclass~., data= training_gender, trControl = train_control,
method = "svmLinear")

# Learning a SVM with the whole training set and without evaluating it
train_control <- trainControl(method="none")
model_SVM_variety <- train( theclass~., data= training_variety, trControl = train_control,
method = "svmLinear")
```

### 3.7. Applying the model to predict the test

```
pred_SVM_gender <- predict(model_SVM_gender, test_gender)
pred_SVM_variety <- predict(model_SVM_variety, test_variety)
```

### 3.8. Evaluating predictions

```
confusionMatrix(pred_SVM_gender, truth_gender)
confusionMatrix(pred_SVM_variety, truth_variety)
```

### 3.9. Joint evaluation: when it is correct in both gender and variety

```
joint <- data.frame(pred_SVM_gender, truth_gender, pred_SVM_variety, truth_variety)
joint <- cbind(joint, ifelse(joint[,1]==joint[,2],1,0), ifelse(joint[,3]==joint[,4],1,0))
joint <- cbind(joint, joint[,5]*joint[,6])
colnames(joint) <- c("pgender", "tgender", "pvariety", "tvariety", "gender", "variety", "joint")

accgender <- sum(joint$gender)/nrow(joint)
accvariety <- sum(joint$variety)/nrow(joint)
accjoint <- sum(joint$joint)/nrow(joint)

print(paste(accgender, accvariety, accjoint, time.taken))
```

*** time.taken** is calculated as the difference in times from the beginning and end of the execution of the script.

# 4. Result

The following table shows the results for both subtasks in cross validation and evaluating with the test subset:

| APPROACH | GENDER | VARIETY | JOINT |
|---|---|---|---|
| Best PAN'17 | 0.8233 | 0.8988 | 0.7429 |
| N = 10 | 0.5875 | 0.2608 | 0.1442 |
| N = 50 | 0.6850 | 0.3167 | 0.2142 |
| N = 100 | 0.7375 | 0.3383 | 0.2525 |
| N = 500 | 0.7358 | 0.5717 | 0.4175 |
| N = 1,000 | 0.6983 | 0.6167 | 0.4325 |
| N = 5,000 | 0.7550 | 0.7275 | 0.5517 |
| N = 10,000 | Impossible with RStudio | | |

# 5. Conclusions

We have applied the concepts of learning in text analysis with a practical example. This example is part of the task of Author Profiling at PAN. The results show that the bag of words allows to obtain good precision with a simple approximation, although more elaborated representations are necessary to be competitive with the state of the art.

Paolo Rosso, PhD.
prosso@dsic.upv.es

Francisco Rangel Pardo, PhD.
francisco.rangel@autoritas.es

Edwin Puertas, PhD Student
Edwin.puertas@javeriana.edu.co