

# STAT S4240 002, Homework 3

Brian Weinstein (bmw2148)

July 30, 2015

## Problem 1: Naive Bayes Text Classification: Data Preparation

See hw03\_q1.R for code.

(a) Text pre-processing

```
# load functions from hw03.R
source("hw03.R")

# preprocess text
preprocess.directory("datasets/FederalistPapers/fp_hamilton_train")
preprocess.directory("datasets/FederalistPapers/fp_hamilton_test")
preprocess.directory("datasets/FederalistPapers/fp_madison_train")
preprocess.directory("datasets/FederalistPapers/fp_madison_test")
```

(b) Loading the cleaned data

```
hamilton.train <- read.directory("datasets/FederalistPapers/fp_hamilton_train_clean")
hamilton.test <- read.directory("datasets/FederalistPapers/fp_hamilton_test_clean")
madison.train <- read.directory("datasets/FederalistPapers/fp_madison_train_clean")
madison.test <- read.directory("datasets/FederalistPapers/fp_madison_test_clean")
```

(c) Create a dictionary from all of the documents

```
dictionary <- make.sorted.dictionary.df(c(hamilton.train, hamilton.test,
                                         madison.train, madison.test))
```

(d) Creating document-term-matrices for each of the datasets

```
dtm.hamilton.train <- make.document.term.matrix(infiles=hamilton.train,
                                                dictionary=dictionary)
dtm.hamilton.test <- make.document.term.matrix(infiles=hamilton.test,
                                                dictionary=dictionary)
dtm.madison.train <- make.document.term.matrix(infiles=madison.train,
                                                dictionary=dictionary)
dtm.madison.test <- make.document.term.matrix(infiles=madison.test,
                                                dictionary=dictionary)
```

(e) Compute the log probabilities for the dictionary in each of the document datasets

```
mu=1/nrow(dictionary)

logp.hamilton.train <- make.log.pvec(dtm.hamilton.train, mu)
logp.hamilton.test <- make.log.pvec(dtm.hamilton.test, mu)
logp.madison.train <- make.log.pvec(dtm.madison.train, mu)
logp.madison.test <- make.log.pvec(dtm.madison.test, mu)
```

**Problem 2: Naive Bayes Function**

We first estimate the log priors based on the log of the proportion of training documents attributed to each author.

$$p(\text{author} = \text{author}) = \log \left( \frac{\# \text{ of training documents attributed to } \text{author}}{\text{total } \# \text{ of training documents}} \right)$$

Then, using (1) the log probabilities for the dictionary in a Hamilton-authored document and (2) the log probabilities for the dictionary in a Madison-authored document (as computed in **Problem 1**), we can input a new document-term-matrix and classify each document as belonging to one of the authors.

```
naive.bayes <- function(logp.hamilton.train, logp.madison.train,
                        log.prior.hamilton, log.prior.madison, dtm.test){
  # Performs naive bayes classification
  # Inputs: logp.hamilton.train : vector of log probabilities of words
  #         occurring in the hamilton training data
  #         logp.madison.train  : vector of log probabilities of words
  #         occurring in the madison training data
  #         log.prior.hamilton  : the log prior of hamilton documents
  #         log.prior.madison   : the log prior of madison documents
  #         dtm.test            : a document-term-matrix to classify
  # Output: Classification labels for each document in dtm.test

  # calculate the log posterior probabilities
  log.post.hamilton <- log.prior.hamilton + (dtm.test %*% logp.hamilton.train)
  log.post.madison  <- log.prior.madison + (dtm.test %*% logp.madison.train)

  # compare the log posterior probabilities and assign to the author
  # with highest probability
  prediction <- data.frame(logPostHam=log.post.hamilton,
                           logPostMad=log.post.madison)
  prediction$pred <- (log.post.hamilton >= log.post.madison)
  prediction$pred <- gsub(TRUE, "Hamilton", prediction$pred)
  prediction$pred <- gsub(FALSE, "Madison", prediction$pred)

  # return a vector of the predictions
  return(prediction$pred)
}
```

**Problem 3: question 3**

Using the confusionMatrix function from the caret library:

- **Accuracy:** 63% accurate (% of the test papers that are classified correctly)
- **True Positive Rate:** 100% (Hamilton classified as Hamilton divided by the total amount of testing Hamilton papers)

- **True Negative Rate:** 9% (Madison classified as Madison divided by the total amount of testing Madison papers)
- **False Positive Rate:** 91% (Madison classified as Hamilton divided by the total amount of testing Madison)
- **False Negative Rate:** 0% (Hamilton classified as Madison divided by the total amount of testing Hamilton)

```
> confusionMatrix(data=predictions$pred,
+                 reference=predictions$trueValue,
+                 dnn=c("Prediction", "True Value"),
+                 positive="Hamilton")
```

Confusion Matrix and Statistics

	True Value	
Prediction	Hamilton	Madison
Hamilton	16	10
Madison	0	1

Accuracy : 0.6296  
 95% CI : (0.4237, 0.806)  
 No Information Rate : 0.5926  
 P-Value [Acc > NIR] : 0.427258

Kappa : 0.106  
 McNemar's Test P-Value : 0.004427

Sensitivity : 1.00000  
 Specificity : 0.09091  
 Pos Pred Value : 0.61538  
 Neg Pred Value : 1.00000  
 Prevalence : 0.59259  
 Detection Rate : 0.59259  
 Detection Prevalence : 0.96296  
 Balanced Accuracy : 0.54545

'Positive' Class : Hamilton