Projektaufgaben Block 3

Carlo Michaelis, 573479; David Hinrichs, 572347; Lukas Ruff, 572521

10 Januar 2017

1 SPAM vs. HAM: Naive Bayes

In dieser Aufgabe beschäftigen wir uns mit dem Spam vs. Ham Klassifizierungsproblem.

1.1 Einlesen der Daten

```
# Read train/test data and dictionary
colFeatures <- c("docID", "wordID", "wordCount")

trainFeatures <- read.table(file = './data/train-features.txt',
                            col.names = colFeatures)
testFeatures <- read.table(file = './data/test-features.txt',
                           col.names = colFeatures)

YTrain <- read.table(file = './data/train-labels.txt', col.names = "Y")
YTest <- read.table(file = './data/test-labels.txt', col.names = "Y")

# Read dictionary
colDict <- c("wordID", "word")
dict <- read.table(file = './data/dictionary.txt', col.names = colDict)
```

1.2 Erzeugen der Featurematrizen

```
# Build (sparse) feature matrices
XTrain <- sparseMatrix(i = trainFeatures$docID, j = trainFeatures$wordID,
                       x = trainFeatures$wordCount)
XTest <- sparseMatrix(i = testFeatures$docID, j = testFeatures$wordID,
                      x = testFeatures$wordCount)
```

1.3 Erzeuge Wahrscheinlichkeiten für den Naive-Bayes-Classifier

```
fnNBTrain <- function(X, Y) {
  # This function generates the probabilities for the Naive-Bayes-Classifier.
  #
  # Args:
  #   X: Matrix of features
  #   Y: Vector of labels
  #
  # Returns:
  #   A list containing the following elements:
  #     $phiSpam:   Probabilities for words occuring in SPAM message
  #     $phiHam:    Probabilities for words occuring in HAM message
```

```
  #     $gammaSpam: The relative frequency of SPAM messages
  #     $gammaHam:  The relative frequency of HAM messages

  nWords <- length(Y)
  indSpam <- as.logical(Y)
  indHam <- !(indSpam)
  NBClass <- list()

  # Generate probabilities
  NBClass$phiSpam <- (colSums(X[indSpam, ]) + 1) / (sum(X[indSpam, ]) + nWords)
  NBClass$phiHam <- (colSums(X[indHam, ]) + 1) / (sum(X[indHam, ]) + nWords)

  # Get relative frequencies
  NBClass$gammaSpam <- sum(indSpam)
  NBClass$gammaHam <- sum(indHam)

  return(NBClass)
}

NBClass1 <- fnNBTrain(XTrain, YTrain[[1]])
```

Zähler und Nenner wurden derart angepasst, dass für den Fall, dass keine Trainingsdaten vorliegen, für die bedingten Verteilungen der Wörter in einem Dokument a priori diskrete Gleichverteilungen mit Wahrscheinlichkeiten $\frac{1}{|V|} = \frac{1}{2500}$ angenommen werden.

1.4 Vorhersage auf den Testdaten

```
fnNBPredict <- function(X, NBClass) {
  # This function predicts one of the two classes (SPAM or HAM) for some test
  # data X using a Naive-Bayes-Classifier trained by fnNBTrain.
  #
  # Args:
  #   X:        Matrix of features
  #   NBClass:  A list returned by fnNBTrain which contains
  #               $phiSpam:   Probabilities for words occuring in SPAM message
  #               $phiHam:    Probabilities for words occuring in HAM message
  #               $gammaSpam: The relative frequency of SPAM messages
  #               $gammaHam:  The relative frequency of HAM messages
  #
  # Returns:
  #   A vector with predictions for each example.

  # Predict labels (using logarithm)
  postSpam <- rowSums(t(t(X) * log(NBClass$phiSpam))) + log(NBClass$gammaSpam)
  postHam <- rowSums(t(t(X) * log(NBClass$phiHam))) + log(NBClass$gammaHam)
  pred <- (postSpam > postHam) * 1

  return(pred)
}

# Predict test labels
predTest1 <- fnNBPredict(XTest, NBClass1)
```

```
# Number of errors
nErr1 <- sum(YTest[[1]] != predTest1)
```

Insgesamt gibt es lediglich 6 falsche Klassifikationen, was einer Fehlerrate von 2.31% entspricht.

Schauen wir uns an, welche Wörter besonders gute Indikatoren für SPAM oder HAM sind. Dazu betrachten wir jeweils die 25 Wörter mit den größten $\Phi_{i|1}$ bzw. $\Phi_{i|0}$:

```
par(mfrow = c(1,2))

# Plot SPAM indicators
wordcloud(dict$word[rank(-NBClass1$phiSpam) <= 50],
          NBClass1$phiSpam[rank(-NBClass1$phiSpam) <= 50],
          scale = c(2.5, 0.2), main = "SPAM indicators")

# Plot HAM indicators
wordcloud(dict$word[rank(-NBClass1$phiHam) <= 50],
          NBClass1$phiHam[rank(-NBClass1$phiHam) <= 50],
          scale = c(2.5, 0.2))
```

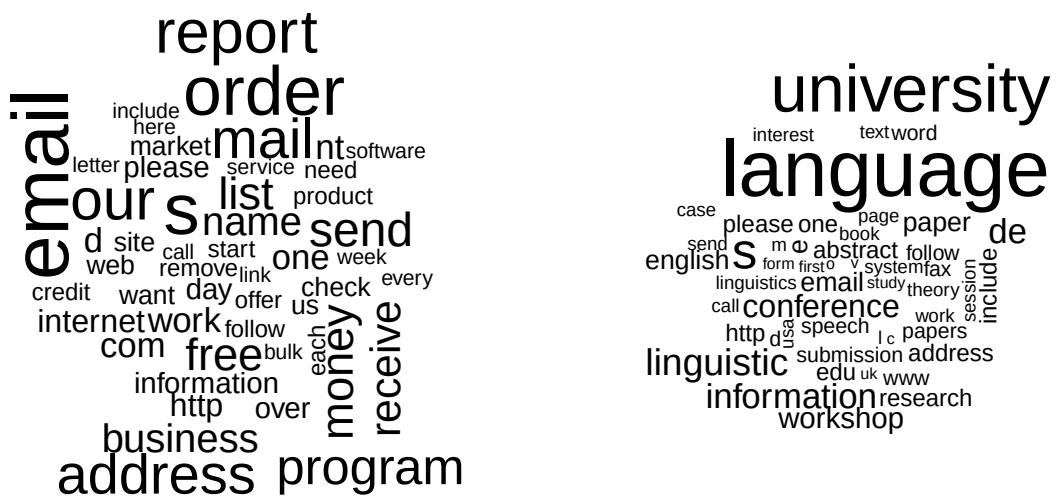
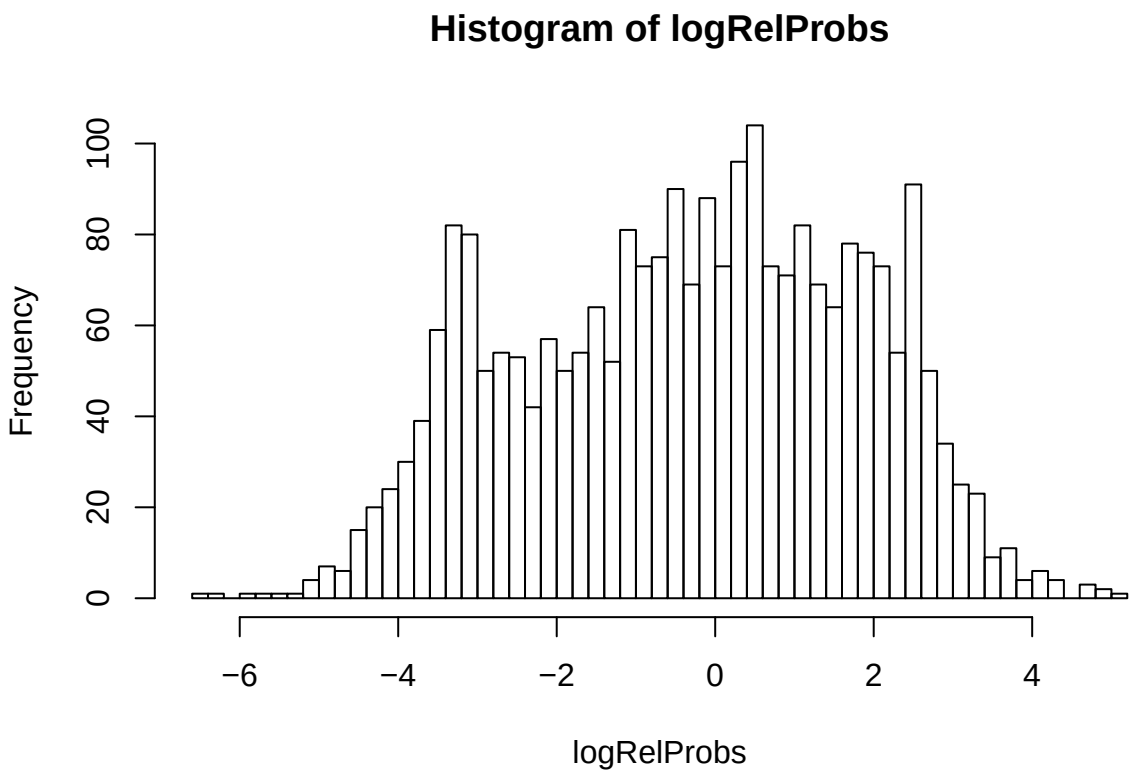


Figure 1: Wordclouds for SPAM (left) and HAM (right) indicators.

1.5 Feature Engineering

Für die Klassifizierung von SPAM und HAM bedeutunglose Wörter sollten in SPAM- und HAM-Nachrichten relativ betrachtet ähnlich häufig vorkommen. Um Ähnlichkeit zu quantifizieren, können wir das Verhältnis von $\Phi_{i|1}$ zu $\Phi_{i|0}$ betrachten. Logarithmieren liefert eine bessere Skalierung.

```
logRelProbs <- log(NBClass1$phiSpam/NBClass1$phiHam)
hist(logRelProbs, breaks = 50)
```




```
# Remove some words
indMeaningful <- !((logRelProbs >= (-1.5)) & (logRelProbs <= 1.5))

# Test classification after removal of meaningless words
NBClass2 <- fnNBTrain(XTrain[, indMeaningful], YTrain[[1]])
predTest2 <- fnNBPredict(XTest[, indMeaningful], NBClass2)

# New classification error
nErr2 <- sum(YTest[[1]] != predTest2)
```

Durch die Entfernung bedeutungsloser Wörtern hat sich die Anzahl der Fehler von 6 auf `nErr2` halbiert.