Assignment 2: Classification for the Detection of Opinion Spam

Anouk van der Lee (6620590), Shu Zhao (6833519), Fleur Petit (5583837)

21, October, 2020

Contents

Dataset for training, validation and test **1**

Dataset split . . . 1

Feature selection . . . 2

Classifiers **3**

Multinomial naive Bayes . . . 3

Regularized logistic regression . . . 5

Classification trees . . . 6

Random forests . . . 8

Hyper-parameters **9**

Questions **9**

Dataset for training, validation and test

Dataset split

```r
# 1 fold = 80 samples
# Fold 1-4 for training = 1:320 from true + 1:320 from false
index.train <- c(c(1:320), 400+c(1:320))

# Training document-term matrix
train.dtm <- DocumentTermMatrix(reviews.all[index.train])
# Test document-term matrix
test.dtm <- DocumentTermMatrix(reviews.all[-index.train],
                               list(dictionary = dimnames(train.dtm)[[2]]))

# Training document-term matrix for bigrams
BigramTokenizer <- function (x) {
  unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
}

train.dtm.bigram <- DocumentTermMatrix(reviews.all[index.train],
                                       control = list(tokenize = BigramTokenizer))
```

```r
# Test document-term matrix for bigrams
test.dtm.bigram <- DocumentTermMatrix(reviews.all[-index.train],
                                      list(dictionary = dimnames(train.dtm.bigram)[[2]]))
```

Feature selection

Frequency

```r
# Remove terms that occur in less than 5% of the documents
# Training document-term matrix
train.dtm.freq <- removeSparseTerms(train.dtm, 0.95)
# Test document-term matrix
test.dtm.freq <- DocumentTermMatrix(reviews.all[-index.train],
                                    list(dictionary = dimnames(train.dtm.freq)[[2]]))

# Training document-term matrix for bigrams
train.dtm.bigram.freq <- removeSparseTerms(train.dtm.bigram, 0.99)
# Test document-term matrix for bigrams
test.dtm.bigram.freq <- DocumentTermMatrix(reviews.all[-index.train],
                                           list(dictionary = dimnames(train.dtm.bigram.freq)[[2]]))

# Combine unigrams and bigrams
# Training document-term matrix for unigrams and bigrams
train.dtm.bigram.freq <- cbind(train.dtm.freq, train.dtm.bigram.freq)
# Test document-term matrix for unigrams and bigrams
test.dtm.bigram.freq <- cbind(test.dtm.freq, test.dtm.bigram.freq)
```

Mutual Information

```r
# Select top-n mutual-information terms from total vocabularies
index.top300 <- calculate.topn(train.dtm, topn = c(1:300))
train.dtm.top300 <- train.dtm[, index.top300[[1]]]
test.dtm.top300 <- test.dtm[, index.top300[[1]]]

# Show part of mutual-info-only top-300 words
index.top300[[2]][1:30]
```

```
##      chicago     location        smell       luxury      decided    recently
##   0.12628024   0.04740995   0.04549257   0.04545347   0.03733789  0.03461556
##      finally   millennium    elevators       seemed        great     cleaned
##   0.03141372   0.03025502   0.02881032   0.02666364   0.02653943  0.02571245
##   experience         open      smelled         rude        relax        star
##   0.02558607   0.02443184   0.02344442   0.02228039   0.02222793  0.02135557
##       turned      arrived    priceline       walked      windows        tiny
##   0.02127947   0.02080563   0.02061632   0.01989954   0.01985828  0.01860112
##         cool     security     elevator   originally construction     counter
##   0.01839637   0.01839637   0.01834595   0.01740433   0.01732006  0.01692180
```

```r
# Select top-n mutual-information terms from frequent terms
index.freq.top200 <- calculate.topn(train.dtm.freq, topn = c(1:200))
train.dtm.freq.top200 <- train.dtm[, index.freq.top200[[1]]]
test.dtm.freq.top200 <- test.dtm[, index.freq.top200[[1]]]

# Show part of frequent top-200 words
index.freq.top200[[2]][1:30]
```

```
##     chicago    location       smell      luxury     decided    recently
##  0.12628024  0.04740995  0.04549257  0.04545347  0.03733789  0.03461556
##     finally  millennium      seemed       great     cleaned  experience
##  0.03141372  0.03025502  0.02666364  0.02653943  0.02571245  0.02558607
##        open     smelled        rude        star     arrived    elevator
##  0.02443184  0.02344442  0.02228039  0.02135557  0.02080563  0.01834595
## comfortable        many       floor       found        make        wait
##  0.01641703  0.01631449  0.01603756  0.01570719  0.01544503  0.01480751
##     website       smoke      suites        like       clerk      sheets
##  0.01463610  0.01427308  0.01379217  0.01325652  0.01313606  0.01254666
```

Classifiers

Multinomial naive Bayes

1. Multinomial naive Bayes - Unigram

```r
# 4-fold cross validation
reviews.mnb.pred <- c()
reviews.mnb.actual <- c()

# hyper-parameter: feature selection
# the best option: only mutual information to select top-300 terms out of 6900 total terms
train.dtm.mnb <- train.dtm.top300

for(i in 1:4) {
  # Validation fold
  val.start <- (i - 1) * 80 + 1
  val.end <- val.start + 80 - 1
  val.range <- c(c(val.start:val.end), 320+c(val.start:val.end))

  # Training fold
  train.range <- c(c(1:320)[-val.range], 320+c(1:320)[-val.range])

  # Train the model with priors and conditional probabilities
  reviews.mnb <- train.mnb(as.matrix(train.dtm.mnb)[train.range,], labels[index.train][train.range])
  # Make predictions
  reviews.mnb.pred <- c(reviews.mnb.pred, predict.mnb(reviews.mnb, as.matrix(train.dtm.mnb)[val.range,]
  reviews.mnb.actual <- c(reviews.mnb.actual, labels[index.train][val.range])
}

# Confusion matrix
```

```
conf.mat.mnb <- table(reviews.mnb.actual, reviews.mnb.pred, dnn = c("actual", "predicted"))
perf.mnb <-performance(conf.mat.mnb)
```

```
perf.mnb %>% kable()
```

metric	value
recall	0.9531250
miss-rate	0.0468750
fall-out	0.1218750
selectivity	0.8781250
prevalence	0.5000000
precision	0.8866279
false omission rate	0.0506757
pos likelihood ratio	7.8205128
neg likelihood ratio	0.0533808
accuracy	0.9156250
false discovery rate	0.1133721
neg predictive value	0.9493243
diagnostic odds ratio	146.5042735
F1	0.9186747

2. Multinomial naive Bayes - Bigram

```
perf.mnb.bigram %>% kable()
```

metric	value
recall	0.8000000
miss-rate	0.2000000
fall-out	0.1468750
selectivity	0.8531250
prevalence	0.5000000
precision	0.8448845
false omission rate	0.1899110
pos likelihood ratio	5.4468085
neg likelihood ratio	0.2344322
accuracy	0.8265625
false discovery rate	0.1551155
neg predictive value	0.8100890
diagnostic odds ratio	23.2340426
F1	0.8218299

Regularized logistic regression

3. Regularized logistic regression - Unigram

```r
perf.logreg %>% kable()
```

metric	value
recall	0.7375000
miss-rate	0.2625000
fall-out	0.1000000
selectivity	0.9000000
prevalence	0.5000000
precision	0.8805970
false omission rate	0.2258065
pos likelihood ratio	7.3750000
neg likelihood ratio	0.2916667
accuracy	0.8187500
false discovery rate	0.1194030
neg predictive value	0.7741935
diagnostic odds ratio	25.2857143
F1	0.8027211

4. Regularized logistic regression - Bigram

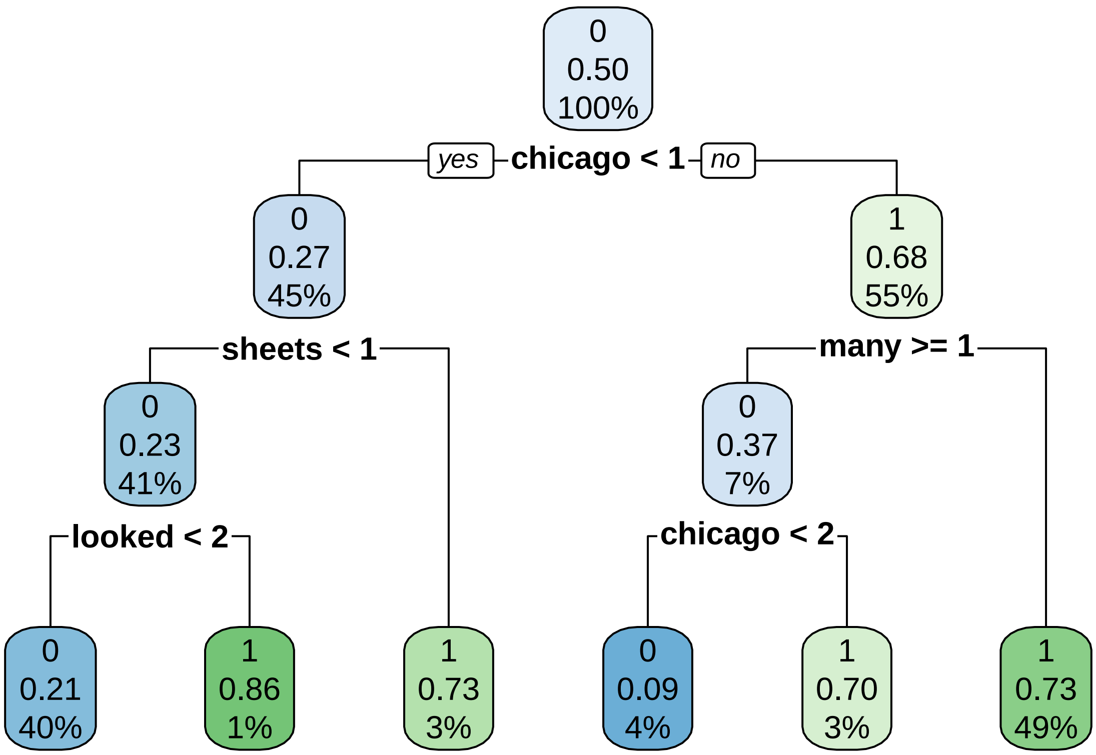
```r
perf.logreg.bigram %>% kable()
```

metric	value
recall	0.6375000
miss-rate	0.3625000
fall-out	0.1000000
selectivity	0.9000000
prevalence	0.5000000
precision	0.8644068
false omission rate	0.2871287
pos likelihood ratio	6.3750000
neg likelihood ratio	0.4027778
accuracy	0.7687500
false discovery rate	0.1355932
neg predictive value	0.7128713
diagnostic odds ratio	15.8275862
F1	0.7338129

Classification trees

5. Classification trees - Unigram

```
# Plot the tree
rpart.plot(reviews.rpart.pruned)
```

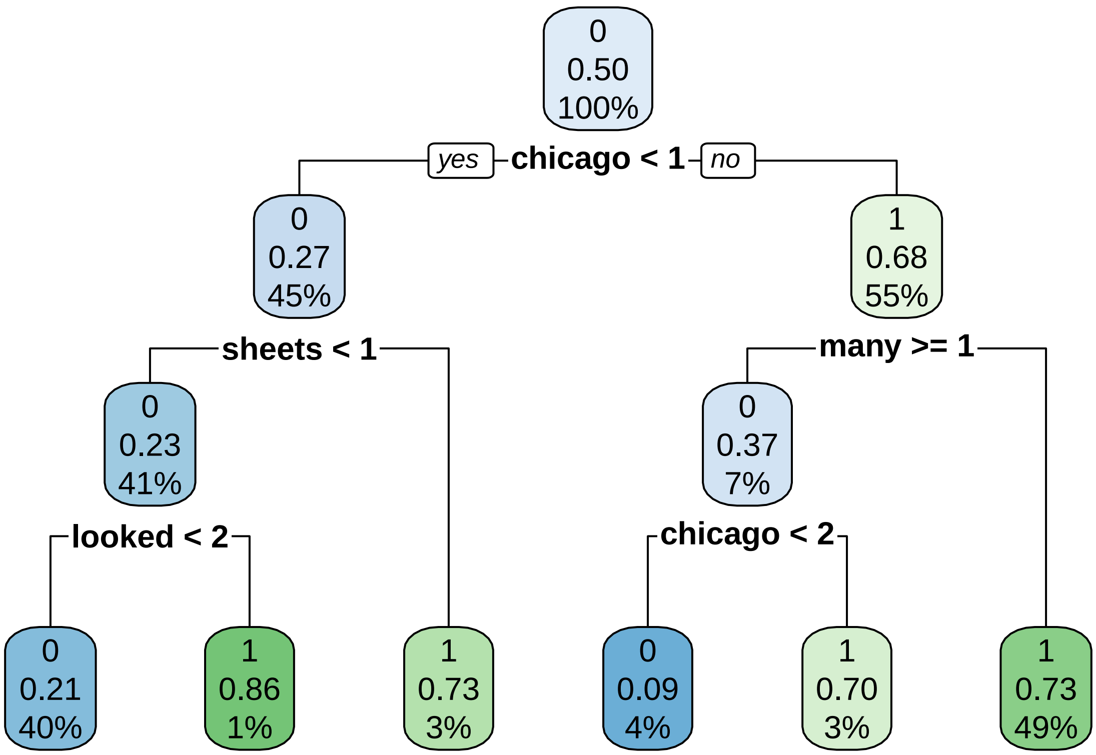



```
perf.rpart %>% kable()
```

metric	value
recall	0.6000000
miss-rate	0.4000000
fall-out	0.2875000
selectivity	0.7125000
prevalence	0.5000000
precision	0.6760563
false omission rate	0.3595506
pos likelihood ratio	2.0869565
neg likelihood ratio	0.5614035
accuracy	0.6562500
false discovery rate	0.3239437
neg predictive value	0.6404494
diagnostic odds ratio	3.7173913
F1	0.6357616

6. Classification trees - Bigram

```r
# Plot the tree
rpart.plot(reviews.rpart.bigram.pruned)
```




```r
perf.rpart.bigram %>% kable()
```

metric	value
recall	0.6000000
miss-rate	0.4000000
fall-out	0.2875000
selectivity	0.7125000
prevalence	0.5000000
precision	0.6760563
false omission rate	0.3595506
pos likelihood ratio	2.0869565
neg likelihood ratio	0.5614035
accuracy	0.6562500
false discovery rate	0.3239437
neg predictive value	0.6404494
diagnostic odds ratio	3.7173913
F1	0.6357616

Random forests

7. Random forests - Unigram

```
perf.rf %>% kable()
```

metric	value
recall	0.7750000
miss-rate	0.2250000
fall-out	0.2000000
selectivity	0.8000000
prevalence	0.5000000
precision	0.7948718
false omission rate	0.2195122
pos likelihood ratio	3.8750000
neg likelihood ratio	0.2812500
accuracy	0.7875000
false discovery rate	0.2051282
neg predictive value	0.7804878
diagnostic odds ratio	13.7777778
F1	0.7848101

8. Random forests - Bigram

```
perf.rf.bigram %>% kable()
```

metric	value
recall	0.6875000
miss-rate	0.3125000
fall-out	0.0875000
selectivity	0.9125000
prevalence	0.5000000
precision	0.8870968
false omission rate	0.2551020
pos likelihood ratio	7.8571429
neg likelihood ratio	0.3424658
accuracy	0.8000000
false discovery rate	0.1129032
neg predictive value	0.7448980
diagnostic odds ratio	22.9428571
F1	0.7746479

Hyper-parameters

Questions

1. For a single classification tree, the impurity reduction is not equal to mutual information?
2. the words found in feature selection (e.g. frequqency or mutual information) = the words found in classification tree by impurity reduction = the words found in logistic regression by coef(s="lambda.1se") (e.g. as.matrix(coef(reviews.glmnet, s="lambda.1se"))[,1]['chicago'])?
3. sum(as.matrix(coef(reviews.glmnet, s="lambda.1se")) != 0) is 52, so top50 is better?