Assignment 2: Classification for the Detection of Opinion Spam

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

25, October, 2020

Contents

Dataset for training, validation and test

Dataset split

```
# 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]]))
```

Mutual Information

```r
# Select top-n mutual-information terms from total vocabularies
index.top50 <- calculate_topn(train.dtm, topn = 50)
train.dtm.top50 <- train.dtm[, index.top50[[1]]]
test.dtm.top50 <- test.dtm[, index.top50[[1]]]

# Show part of mutual-info-only top-500 words
index.top50[[2]][1:30]

# Select top-n mutual-information bigrams from total bigrams
index.bigram.top50 <- calculate_topn(train.dtm.bigram, topn = 50)
train.dtm.bigram.top50 <- train.dtm.bigram[, index.bigram.top50[[1]]]
test.dtm.bigram.top50 <- test.dtm.bigram[, index.bigram.top50[[1]]]

# Show part of mutual-info-only top-300 words
index.bigram.top50[[2]][1:30]
```

Classifiers

Multinomial naive Bayes

1. Multinomial naive Bayes - Unigram

metric	value
recall	0.9625000
miss-rate	0.0375000
fall-out	0.1093750
selectivity	0.8906250
prevalence	0.5000000
precision	0.8979592
false omission rate	0.0404040
pos likelihood ratio	8.8000000
neg likelihood ratio	0.0421053
accuracy	0.9265625
false discovery rate	0.1020408
neg predictive value	0.9595960
diagnostic odds ratio	209.0000000
F1	0.9291101

```
##            Top-307 mutual info Top-307 frequent
## chicago             0.12628024       0.12628024
## location            0.04740995       0.04740995
## smell               0.04549257       0.04549257
## luxury              0.04545347       0.04545347
## decided             0.03733789       0.03733789
## recently            0.03461556       0.03461556
## finally             0.03141372       0.03141372
## millennium          0.03025502       0.03025502
## elevators           0.02881032               NA
## seemed              0.02666364       0.02666364
## great               0.02653943       0.02653943
## cleaned             0.02571245       0.02571245
## experience          0.02558607       0.02558607
## open                0.02443184       0.02443184
## smelled             0.02344442       0.02344442
## rude                0.02228039       0.02228039
## relax               0.02222793               NA
## star                0.02135557       0.02135557
## turned              0.02127947               NA
## arrived             0.02080563       0.02080563
```

2. Multinomial naive Bayes - Bigram

metric	value
recall	0.9812500
miss-rate	0.0187500
fall-out	0.0406250
selectivity	0.9593750
prevalence	0.5000000
precision	0.9602446
false omission rate	0.0191693
pos likelihood ratio	24.1538462
neg likelihood ratio	0.0195440
accuracy	0.9703125

metric	value
false discovery rate	0.0397554
neg predictive value	0.9808307
diagnostic odds ratio	1235.8717949
F1	0.9706337

```
##                     Top-356 mutual info Top-356 frequent
## cigarette smoke             0.008411447      0.008411447
## hotel first                 0.008411447      0.008411447
## park hotel                  0.008411447      0.008411447
## stay fairmont               0.008411447      0.008411447
## good location               0.007876911      0.007876911
## room looked                 0.007876911      0.007876911
## hyatt regency               0.007864525      0.007864525
## across street               0.007856875               NA
## business center             0.007856875               NA
## coffee table                0.007856875               NA
## construction site           0.007856875               NA
## desk rude                   0.007856875               NA
## general manager             0.007856875               NA
## go room                     0.007856875               NA
## good nights                 0.007856875               NA
## heard good                  0.007856875               NA
## hotel downtown              0.007856875               NA
## hotels ever                 0.007856875               NA
## husband recently            0.007856875               NA
## late checkout               0.007856875               NA
## like nice                   0.007856875               NA
```

Regularized logistic regression

3. Regularized logistic regression - Unigram

metric	value
recall	0.4625000
miss-rate	0.5375000
fall-out	0.4875000
selectivity	0.5125000
prevalence	0.5000000
precision	0.4868421
false omission rate	0.5119048
pos likelihood ratio	0.9487179
neg likelihood ratio	1.0487805
accuracy	0.4875000
false discovery rate	0.5131579
neg predictive value	0.4880952
diagnostic odds ratio	0.9045915
F1	0.4743590

```
##             term    weight
```

```
## 1        relax 1.7485656
## 2     presence 1.7342649
## 3     decision 1.5013831
## 4       recent 1.2977094
## 5   relatively 1.2866138
## 6      crowded 1.2260760
## 7       arrive 1.1514206
## 8     recently 1.1148370
## 9     bringing 1.1016687
## 10  millennium 1.0271524
## 11      afford 1.0139360
## 12      luxury 0.9990958
## 13        mine 0.9672197
## 14      prices 0.9539196
## 15 cleanliness 0.9358000
## 16       rowdy 0.8803829
## 17      attend 0.8742987
## 18     chicago 0.8618676
## 19      turned 0.8533932
## 20     smelled 0.7950719
```

4. Regularized logistic regression - Bigram

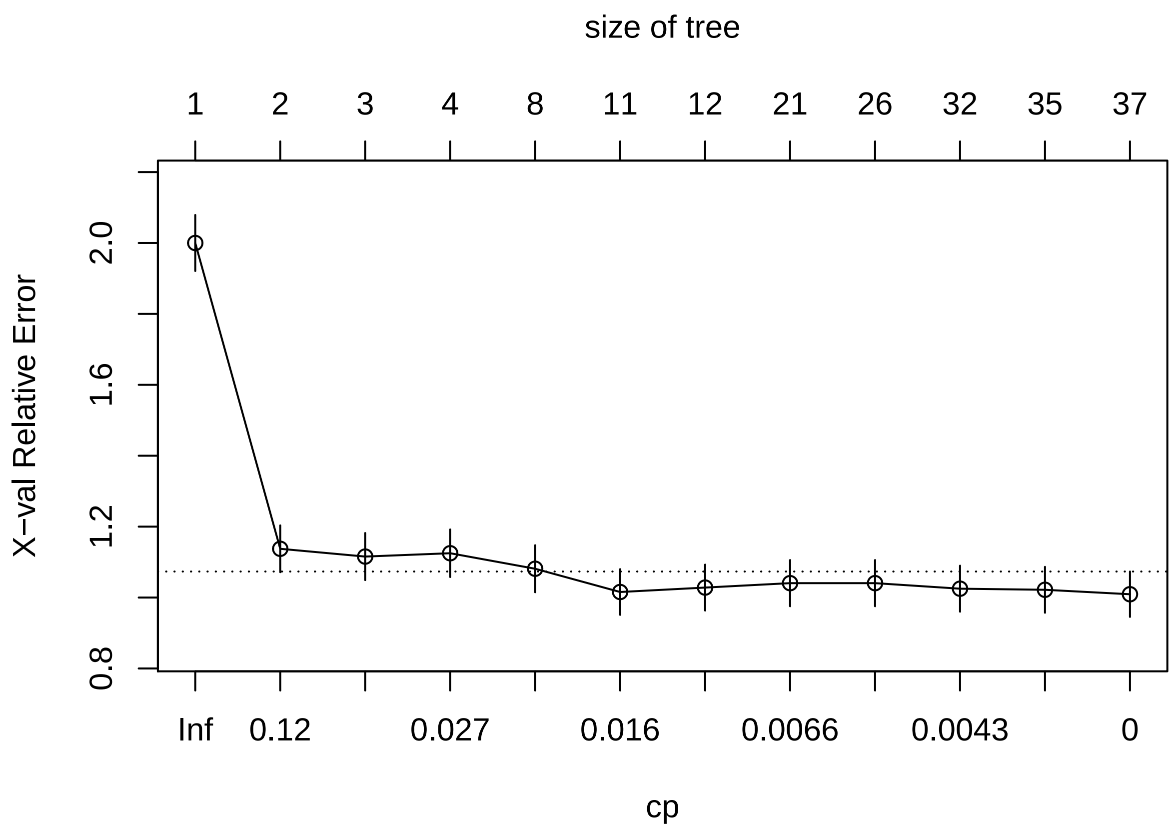
metric	value
recall	0.1000000
miss-rate	0.9000000
fall-out	0.2500000
selectivity	0.7500000
prevalence	0.5000000
precision	0.2857143
false omission rate	0.5454545
pos likelihood ratio	0.4000000
neg likelihood ratio	1.2000000
accuracy	0.4250000
false discovery rate	0.7142857
neg predictive value	0.4545455
diagnostic odds ratio	0.3333333
F1	0.1481481

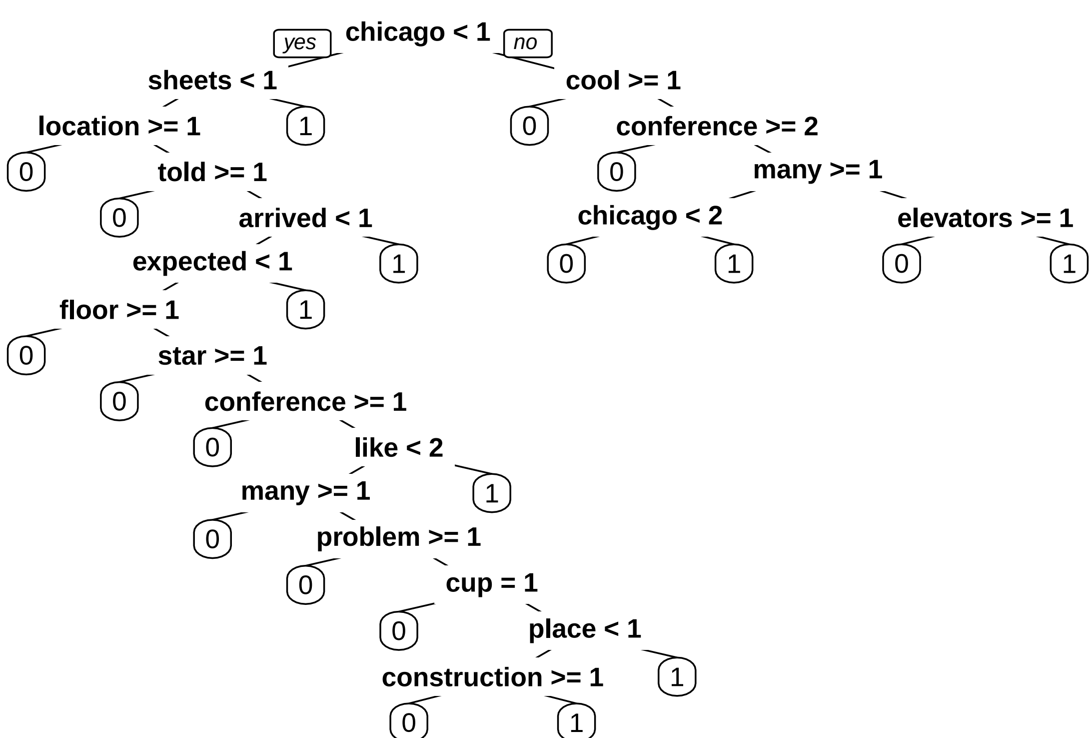
```
##                 term   weight
## 1        hotel thing 2.789793
## 2          hotel bad 2.544291
## 3     really looking 2.507040
## 4          room loud 2.426803
## 5     neighbors room 2.333558
## 6       classy hotel 2.258400
## 7      cleaning crew 2.213976
## 8         hotel full 2.133660
## 9     desperate need 2.016941
## 10      room without 1.958818
## 11             faint 1.932192
```

```
## 12   stayed sofitel 1.926388
## 13            relax 1.818194
## 14   dining options 1.815381
## 15          past pm 1.813917
## 16        bar staff 1.803770
## 17      light didnt 1.795437
## 18             mine 1.729006
## 19     settled room 1.725645
## 20 service horrible 1.699756
```

Classification trees

5. Classification trees - Unigram

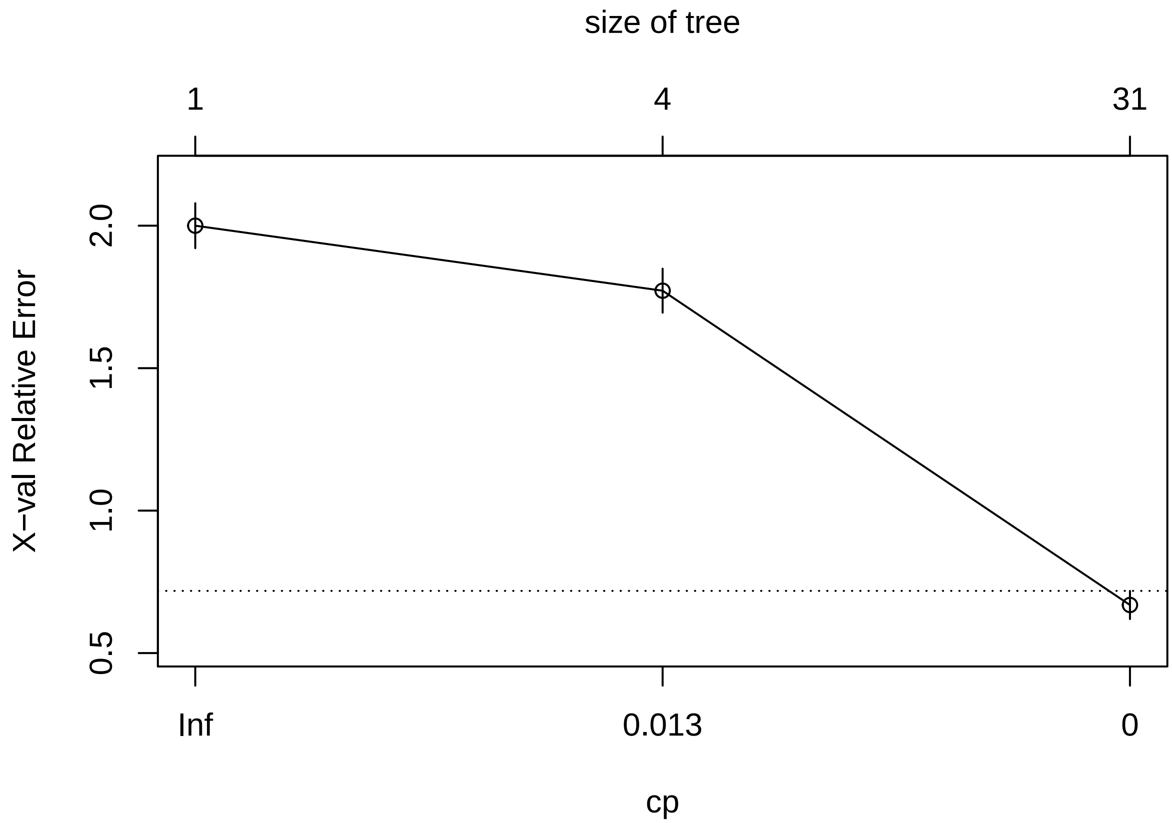


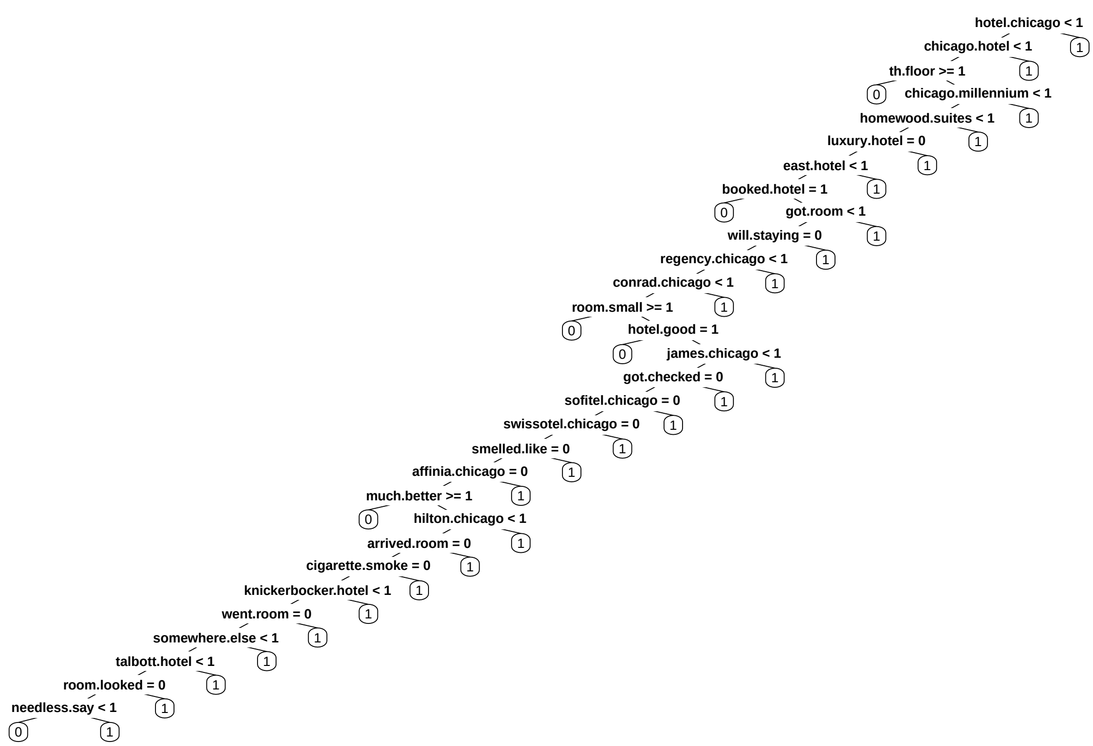


metric	value
recall	0.9562500
miss-rate	0.0437500
fall-out	0.3656250
selectivity	0.6343750
prevalence	0.5000000
precision	0.7234043
false omission rate	0.0645161
pos likelihood ratio	2.6153846
neg likelihood ratio	0.0689655
accuracy	0.7953125
false discovery rate	0.2765957
neg predictive value	0.9354839
diagnostic odds ratio	37.9230769
F1	0.8236878

```
##    chicago     sheets   location       cool       many conference       told
##  52.432155   9.829104   9.667462   8.726672   8.414385   8.177520   7.897861
##      floor    arrived  elevators   expected      hotel       star  attending
##   5.842619   5.093745   4.794941   4.462391   3.797753   2.844444   2.775940
##       like       room    problem    looking        cup       temp
##   2.542112   2.329282   2.218279   2.200153   2.082292   1.979204
```

6. Classification trees - Bigram





metric	value
recall	0.8156250
miss-rate	0.1843750
fall-out	0.1406250
selectivity	0.8593750
prevalence	0.5000000
precision	0.8529412
false omission rate	0.1766467
pos likelihood ratio	5.8000000
neg likelihood ratio	0.2145455
accuracy	0.8375000
false discovery rate	0.1470588
neg predictive value	0.8233533
diagnostic odds ratio	27.0338983
F1	0.8338658

```
##       chicago.hotel       hotel.chicago            th.floor  chicago.millennium
##           11.319163           10.774921           10.474832            6.851645
##      homewood.suites        luxury.hotel     millennium.park          east.hotel
##            6.603086            6.202419            6.045569            5.826870
##     fairmont.chicago      conrad.chicago            got.room        booked.hotel
##            5.642531            5.512321            5.071100            4.972993
##      regency.chicago        will.staying          room.small          hotel.good
##            4.954902            4.929470            3.674140            3.479299
##    swissotel.chicago         got.checked knickerbocker.hotel     sofitel.chicago
##            3.374830            3.320061            3.279437            3.250780
```

Random forests

7. Random forests - Unigram

```
##
## Call:
##  randomForest(formula = as.factor(label) ~ ., data = data.frame(as.matrix(train.dtm.rf),      label =
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 12
##
##         OOB estimate of  error rate: 13.44%
## Confusion matrix:
##     0   1 class.error
## 0 270  50     0.15625
## 1  36 284     0.11250
```

metric	value
recall	0.8875000
miss-rate	0.1125000
fall-out	0.1562500
selectivity	0.8437500
prevalence	0.5000000
precision	0.8502994

metric	value
false omission rate	0.1176471
pos likelihood ratio	5.6800000
neg likelihood ratio	0.1333333
accuracy	0.8656250
false discovery rate	0.1497006
neg predictive value	0.8823529
diagnostic odds ratio	42.6000000
F1	0.8685015

8. Random forests - Bigram

```
##
## Call:
##  randomForest(formula = as.factor(label) ~ ., data = data.frame(as.matrix(train.dtm.rf.bigram),
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 15.78%
## Confusion matrix:
##     0   1 class.error
## 0 310  10    0.031250
## 1  91 229    0.284375
```

metric	value
recall	0.7156250
miss-rate	0.2843750
fall-out	0.0312500
selectivity	0.9687500
prevalence	0.5000000
precision	0.9581590
false omission rate	0.2269327
pos likelihood ratio	22.9000000
neg likelihood ratio	0.2935484
accuracy	0.8421875
false discovery rate	0.0418410
neg predictive value	0.7730673
diagnostic odds ratio	78.0109890
F1	0.8193202

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?