Election Violence Newspaper Article Selection Process Pilot: Evaluate Keyword Selection Method

August 10, 2018

1 Summary

After the code to setup and download data (Sections 2-**??**) the document below includes: First, an implementation of code to select of articles for coding, where initial evaluation of articles is based on keywords from the boolean searches which return those articles (Sections **??**-**??**). This classification is done using actual searches conducted (effectively 4 keywords) with a Naive Bayes classifier.

Second, the document then provides an evaluation of the keyword selection step based on user classified articles from the platform with Naive Bayes and other classifiers and with 4 and 10 keyword sets (Sections 4-**??**).

Finally, to provide an alternative way of evaluating the process, the document includes 10 randomly selected cases which are rejected for coding in the implementation code (Section **??**).

2 Packages used

```
library(reportRx) # note using this just for sanitizing LaTex output, not included in durhamevp package
library(tidyverse)
library(quanteda)
library(durhamevp)
library(RTextTools) # not sure if this is included as part of the durhamevp package
```

3 Cases Used

We assess on the classified case set (items coded on the platform by users, plus items coded by Gary as non-election, and election non-violent).

4 Assess Naive Bayes selection performance on 4 keywords (binary dfm)

Here there are four keywords (election, riot, disturbance, incident) which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
cd<-classdocs
cd<-cd[sample(nrow(cd)),] # random order cases
class_corpus<-quanteda::corpus(cd, text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance")
class_dfm_4<-quanteda::dfm(class_corpus, select=keywords)
#class_dfm<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20)
```

```
the_sets_4<-split_dfm(class_dfm_4, n_train = 1000)
classifier<-quanteda::textmodel_nb(the_sets_4$training_set, y=quanteda::docvars(the_sets_4$training_set
quanteda::docvars(the_sets_4$testing_set, "predicted")<-predict(classifier, newdata = the_sets_4$testing
caret::confusionMatrix(data=quanteda::docvars(the_sets_4$testing_set, "predicted"), reference=factor(qua
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 546 161
##          1 122 477
##
##                Accuracy : 0.7833
##                  95% CI : (0.76, 0.8054)
##     No Information Rate : 0.5115
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5658
##  Mcnemar's Test P-Value : 0.02389
##
##               Precision : 0.7963
##                  Recall : 0.7476
##                      F1 : 0.7712
##              Prevalence : 0.4885
##          Detection Rate : 0.3652
##    Detection Prevalence : 0.4587
##       Balanced Accuracy : 0.7825
##
##        'Positive' Class : 1
##
```

5 Assess Naive Bayes selection performance on 10 keywords (binary dfm)

Here there are ten keywords ((election, riot, disturbance, incident, mob, stone, window, candidate, party, hustings, magistrate) which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
#cd<-classdocs[classdocs$election_article==1,]
cd<-classdocs
cd<-cd[sample(nrow(cd)),]
class_corpus<-quanteda::corpus(cd, text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance", "mob", "stone", "window", "candidate", "party
class_dfm_10<-quanteda::dfm(class_corpus, select=keywords)
#class_dfm<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20)

the_sets_10<-split_dfm(class_dfm_10, n_train = 1000)
classifier_10<-quanteda::textmodel_nb(the_sets_10$training_set, y=quanteda::docvars(the_sets_10$training
quanteda::docvars(the_sets_10$testing_set, "predicted")<-predict(classifier_10, newdata = the_sets_10$te
caret::confusionMatrix(data=quanteda::docvars(the_sets_10$testing_set, "predicted"), reference=factor(qu

## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 530 158
##          1 137 481
##
##                Accuracy : 0.7741
##                  95% CI : (0.7505, 0.7965)
##     No Information Rate : 0.5107
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5477
##  Mcnemar's Test P-Value : 0.2442
##
##               Precision : 0.7783
##                  Recall : 0.7527
##                      F1 : 0.7653
##              Prevalence : 0.4893
##          Detection Rate : 0.3683
##    Detection Prevalence : 0.4732
##       Balanced Accuracy : 0.7737
##
##        'Positive' Class : 1
##
```

6 Assess Naive Bayes selection performance on full set of keywords (binary dfm)

Here we take all the keywords in the ocr as a theoretical maximum which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
cd<-classdocs[classdocs$election_article==1,]
cd<-classdocs
cd<-cd[sample(nrow(cd)),]
class_corpus<-quanteda::corpus(cd, text_field="ocr")
class_dfm_full<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20, min_docfreq = 20)

the_sets_full<-split_dfm(class_dfm_full, n_train = 1000)
classifier_full<-quanteda::textmodel_nb(the_sets_full$training_set, y=quanteda::docvars(the_sets_full$t
quanteda::docvars(the_sets_full$testing_set, "predicted")<-predict(classifier_full, newdata = the_sets_
caret::confusionMatrix(data=quanteda::docvars(the_sets_full$testing_set, "predicted"), reference=factor

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 325 146
##          1 341 494
##
##                Accuracy : 0.6271
##                  95% CI : (0.6002, 0.6534)
##     No Information Rate : 0.51
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2583
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.5916
##                  Recall : 0.7719
##                      F1 : 0.6698
##              Prevalence : 0.4900
##          Detection Rate : 0.3783
##    Detection Prevalence : 0.6394
##       Balanced Accuracy : 0.6299
##
##        'Positive' Class : 1
##
```