CYO Project

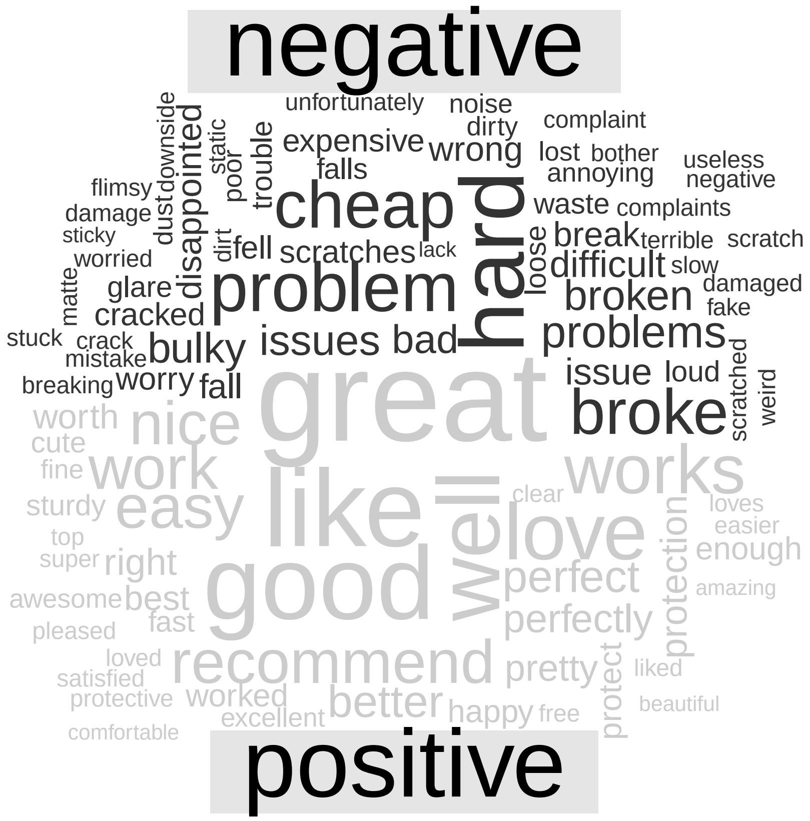
Ahmad Syed Anwar

Jan 06, 2021

#Cell phones and accessories. rating prediction: Introduction For my capstone project, I will create a statistical model for user reviews of mobile phones and accessories. Although the purpose would be to identify by doing sentiment analysis, the review text itself will be the only data that will be used to predict ratings. The initial files can be found on http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Cell_Phones_and_Accessories_5.json.gz and there are millions of records in the json format. Owing to hardware limitations, only a limited subset can be required for this initiative. This will cause the code to run in a timely manner. In order to make it easy to follow through, the R code will be included in the article. Accuracy and F1 are the metrics to be used in testing the models. With no choice for either, F1 is a balanced, weighted average of accuracy and recall. As no result is better or worse than another, F1 is a good metric for this problem.

#Methods and Analysis We will load the required packages and data into R first. The data for our review has a lot of redundant details which would need to be cleaned up. In the dataset, only 1000 documents can be included. The ratings for this classification issue would be classified into two categories. A negative rating would be called anything less than a three star rating and anything else a positive.

Before the data is split, some exploratory analysis is conducted. A wordcloud will reveal frequency and sentiment at the same time. The wordcloud below uses the bing sentiment lexicon which assigns words into positive and negative categories.



This doesn't reveal how balanced the dataset is, however. Using the classification rubric, a two dimensional matrix will be constructed using just the text and the rating sentiment. After, a comparison of the count of each label will reveal the disparity between them. This means it will be a difficult task to increase the F1 statistic.

```
##
## negative positive
##      207      793
```

The bag of words approach will be used. Each word is represented as a feature and each document a vector of features. Word order is disregarded. Prior to building the model, the vectors are converted into a corpus, a large and structured set of texts. The corpus will be cleaned for punctuation, numbers, whitespaces, stop words, and will be converted to lowercase for easier comparison. The corpus is then converted into a Document Term Matrix. Each row (document) is a review's text. The words are laid out in matrix with words and the occurrence of the words in the documents. The data is then split into training and testing sets.

```r
corpus <- VCorpus(VectorSource(text))
corpus.clean <- corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords(kind="en")) %>%
  tm_map(stripWhitespace)

dtm <- DocumentTermMatrix(corpus.clean)

split_val <- floor(0.8 * nrow(reviews))
train_ind <- sample(seq_len(nrow(reviews)), size = split_val)

df.train <- reviews[train_ind, ]
df.test <- reviews[-train_ind, ]

dtm.train <- dtm[train_ind, ]
dtm.test <- dtm[-train_ind, ]

corpus.clean.train <- corpus.clean[train_ind]
corpus.clean.test <- corpus.clean[-train_ind]
```

Three models for comparison are shown below. The first restricts the document term matrix to only terms that appear at least five times. The second uses a normalized term frequency, Tf-Idf, which measures the relative importance of a word to a document. The third uses bigrams and removes sparse bigrams.

```r
fivefreq <- findFreqTerms(dtm.train, 5)

dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = fivefreq))
dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = fivefreq))

tfidf.dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(weighting = weightTfIdf))
tfidf.dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(weighting = weightTfIdf))
```

```
## Warning in weighting(x): empty document(s): 694
```

```r
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
bi.dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control = list(tokenize = BigramTokenizer))
bi.dtm.train.nb <- removeSparseTerms(bi.dtm.train.nb,0.99)
bi.dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control = list(tokenize = BigramTokenizer))
bi.dtm.test.nb <- removeSparseTerms(bi.dtm.test.nb,0.99)
```

A binary conversion function is used to label word frequencies as present or absent and is then applied to the document term matrices. The reasoning behind this is that for sentiment classification word occurrence matters more than frequency.

```r
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
```

```r
trainNB <- apply(dtm.train.nb, 2, convert_count)
testNB <- apply(dtm.test.nb, 2, convert_count)
tfidf.trainNB <- apply(tfidf.dtm.train.nb, 2, convert_count)
tfidf.testNB <- apply(tfidf.dtm.test.nb, 2, convert_count)
bi.trainNB <- apply(bi.dtm.train.nb, 2, convert_count)
bi.testNB <- apply(bi.dtm.test.nb, 2, convert_count)
```

The data is now ready for training and prediction. Naive Bayes evaluates the products of probabilities, which creates problems for the model due to words that do not occur in the sample (0 probability). Therefore, Laplace smoothing is used to assign a small, non-zero probability.

```r
classifier <- naiveBayes(trainNB, as.factor(df.train[,"sc"]), laplace = 1)
tfidf.classifier <- naiveBayes(tfidf.trainNB, as.factor(df.train[,"sc"]), laplace = 1)
bi.classifier <- naiveBayes(bi.trainNB, as.factor(df.train[,"sc"]), laplace = 1)

pred <- predict(classifier, newdata=testNB)
tfidf.pred <- predict(tfidf.classifier, newdata=tfidf.testNB)
bi.pred <- predict(bi.classifier, newdata=bi.testNB)
```

Truth tables will show what was and wasn't correctly classified. Then the confusion matrices will be constructed.

```
## [1] "Five"
```

```
##             Actual
## Predictions negative positive
##    negative       34      121
##    positive        0       45
```

```
## [1] "Tf-Idf"
```

```
##             Actual
## Predictions negative positive
##    negative       34      152
##    positive        0       14
```

```
## [1] "Bigrams"
```

```
##             Actual
## Predictions negative positive
##    negative        0        0
##    positive       34      166
```

The results of the confusion matrix will help determine which model tested better.

```
## [1] "Five"

##          Sensitivity          Specificity       Pos Pred Value
##            1.0000000            0.2710843            0.2193548
##       Neg Pred Value            Precision               Recall
##            1.0000000            0.2193548            1.0000000
##                   F1           Prevalence       Detection Rate
##            0.3597884            0.1700000            0.1700000
## Detection Prevalence    Balanced Accuracy
##            0.7750000            0.6355422

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   3.950000e-01   1.122524e-01   3.267650e-01   4.663964e-01   8.300000e-01
## AccuracyPValue  McnemarPValue
##   1.000000e+00   1.042941e-27

## [1] "Tf-Idf"

##          Sensitivity          Specificity       Pos Pred Value
##           1.00000000           0.08433735           0.18279570
##       Neg Pred Value            Precision               Recall
##           1.00000000           0.18279570           1.00000000
##                   F1           Prevalence       Detection Rate
##           0.30909091           0.17000000           0.17000000
## Detection Prevalence    Balanced Accuracy
##           0.93000000           0.54216867

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   2.400000e-01   3.036489e-02   1.825719e-01   3.053063e-01   8.300000e-01
## AccuracyPValue  McnemarPValue
##   1.000000e+00   1.727912e-34

## [1] "Bigrams"

##          Sensitivity          Specificity       Pos Pred Value
##                 0.00                 1.00                  NaN
##       Neg Pred Value            Precision               Recall
##                 0.83                   NA                 0.00
##                   F1           Prevalence       Detection Rate
##                   NA                 0.17                 0.00
## Detection Prevalence    Balanced Accuracy
##                 0.00                 0.50

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.300000e-01   0.000000e+00   7.706284e-01   8.793044e-01   8.300000e-01
## AccuracyPValue  McnemarPValue
##   5.456387e-01   1.518560e-08
```

The range of the accuracy is not too large, but the F1 score is either or very low. In order to boost this k-fold cross validation will be used. The steps used before will be repeated 10 times in a loop to find the optimal number of folds and the best model.

#Results

```
## Five - # of folds:9

## Tf-Idf - # of folds:9

## Bigrams - # of folds:1

## Five - F1:0.547169811320755

## Tf-Idf - F1:0.504347826086956

## Bigrams - F1:NA

## Five - Accuracy:0.52

## Tf-Idf - Accuracy:0.43

## Bigrams - Accuracy:0.85

## Always guess positive:0.943
```

The frequency model of at least five has advanced quite a bit and now exceeds the Tf-idf model in both F1 and precision. Blindly estimating a positive sentiment (3 or above) would result in a high accuracy, higher than anything but the model of bigrams. This highlights how imbalanced the dataset is and why the F1 score is important. In terms of accuracy and F1 score, the Bi-gram model preformed better and is thus the best model.

#Conclusion Even if the precision was good, the original models that did not use cross validation have no potential to predict true negatives. Both the precision and F1 scores improved with cross validation. The Naive Bayes algorithm performs fairly well, considering the simplicity of the assumptions. Owing to hardware limitations, only a limited subset of the data was included. Future analysis will use more data and better models could be made.