Detecting Vandalism on Wikipedia

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2
```

```
## -- Attaching packages ---------------------------------------------------------------------------------
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.3
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.3
```

```
## -- Conflicts -----------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caTools)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.4.3
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.4.3
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

Wikipedia is a free online encyclopedia that anyone can edit and contribute to. It is available in many languages and is growing all the time. On the English language version of Wikipedia:

- There are currently 4.7 million pages.
- There have been a total over 760 million edits (also called revisions) over its lifetime.
- There are approximately 130,000 edits per day.

One of the consequences of being editable by anyone is that some people vandalize pages. This can take the form of removing content, adding promotional or inappropriate content, or more subtle shifts that change the meaning of the article. With this many articles and edits per day it is difficult for humans to detect all instances of vandalism and revert (undo) them. As a result, Wikipedia uses **bots - computer programs that automatically revert edits that look like vandalism.** In this assignment we will attempt to develop a vandalism detector that uses machine learning to distinguish between a valid edit and vandalism.

The data for this problem is based on the revision history of the page Language. Wikipedia provides a history for each page that consists of the state of the page at each revision. Rather than manually considering each revision, a script was run that checked whether edits stayed or were reverted. If a change was eventually reverted then that revision is marked as vandalism. This may result in some misclassifications, but the script performs well enough for our needs.

As a result of this preprocessing, some common processing tasks have already been done, including lower-casing and punctuation removal. The columns in the dataset are:

Vandal = 1 if this edit was vandalism, 0 if not.

Minor = 1 if the user marked this edit as a "minor edit", 0 if not.

Loggedin = 1 if the user made this edit while using a Wikipedia account, 0 if they did not.

Added = The unique words added.

Removed = The unique words removed.

Notice the repeated use of *unique*.

The data we have available is not the traditional bag of words - rather it is the set of words that were removed or added.

For example, if a word was removed multiple times in a revision it will only appear one time in the "Removed" column.

1.1) Bag of Words

Load the data wiki.csv with the option stringsAsFactors=FALSE, calling the data frame "wiki". Convert the "Vandal" column to a factor using the command

```r
wiki <- read.csv('wiki.csv')
wiki$Vandal = as.factor(wiki$Vandal)
```

How many cases of vandalism were detected in the history of this page?

```r
wiki = read.csv('wiki.csv', stringsAsFactors = FALSE)
```

```r
glimpse(wiki)
```

```
## Observations: 3,876
## Variables: 7
## $ X.1      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ X        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ Vandal   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Minor    <int> 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,...
## $ Loggedin <int> 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,...
## $ Added    <chr> "  represent psycholinguisticspsycholinguistics ortho...
## $ Removed  <chr> " ", " talklanguagetalk", " regarded as technologytec...
```

```r
wiki$Vandal = as.factor(wiki$Vandal)

table(wiki$Vandal)
```

```
##
##    0    1
## 2061 1815
```

Explanation

You can load the data using the command:

wiki = read.csv("wiki.csv", stringsAsFactors=FALSE)

And then convert Vandal to a factor with the command:

wiki$Vandal = as.factor(wiki$Vandal)

You can then use the table command to see how many cases of Vandalism there are:

table(wiki$Vandal)

There are 1815 observations with value 1, which denotes vandalism.

1.2) Bag of Word

We will now use the bag of words approach to build a model. We have two columns of textual data, with different meanings. For example, adding rude words has a different meaning to removing rude words. We'll start like we did in class by building a document term matrix from the Added column. The text already is lowercase and stripped of punctuation. So to pre-process the data, just complete the following four steps:

1) Create the corpus for the Added column, and call it "corpusAdded".

```r
corpus_added = VCorpus(VectorSource(wiki$Added))
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time(), tz = "GMT"): unknown timezone
## 'zone/tz/2018i.1.0/zoneinfo/America/Chicago'
```

```r
corpus_added
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 3876
```

2) Remove the English-language stopwords.

```r
corpus_added = tm_map(corpus_added, removeWords, stopwords('english'))
```

3) Stem the words.

```r
corpus_added = tm_map(corpus_added, stemDocument)
```

```r
strwrap(corpus_added[[1]]$content)
```

```
##  [1] "repres psycholinguisticspsycholinguist orthographyorthographi help"
##  [2] "text action human ethnologu relationship linguist regard write"
##  [3] "languag list xmlspacepreservelanguag metavers formal term philolog"
##  [4] "common includ phonologyphonolog often ten list human affili see"
##  [5] "comput speechpathologyspeech way dialect pleas artifici written"
##  [6] "bodi quit hypothesi found alon refer languag profan studi program"
##  [7] "prioriti rosenfeld technologytechnolog make first among use"
##  [8] "languagephilosophi one sound use area creat phrase mark genet"
##  [9] "basic famili complet sapirwhorfhypothesissapirwhorf"
## [10] "talklanguagetalk popul anim scienc vocal can concept call topic"
## [11] "locat number patholog differ develop 4000 thing idea group complex"
## [12] "anim mathemat fair literatur httpwwwzompistcom philosophi import"
## [13] "meaning historicallinguisticsorphilologyhistor semanticssemant"
## [14] "pattern oral"
```

4) Build the DocumentTermMatrix, and call it dtmAdded.

```r
dtm_added = DocumentTermMatrix(corpus_added)
dtm_added
```

```
## <<DocumentTermMatrix (documents: 3876, terms: 6675)>>
## Non-/sparse entries: 15368/25856932
## Sparsity           : 100%
## Maximal term length: 784
## Weighting          : term frequency (tf)
```

If the code length(stopwords("english")) does not return 174 for you, then please run the line of code in this file, which will store the standard stop words in a variable called sw. When removing stop words, use tm_map(corpusAdded, removeWords, sw) instead of tm_map(corpusAdded, removeWords, stopwords("english")).

How many terms appear in dtmAdded?

Explanation

The following are the commands needed to execute these four steps:

corpusAdded = VCorpus(VectorSource(wiki$Added))

corpusAdded = tm_map(corpusAdded, removeWords, stopwords("english"))

corpusAdded = tm_map(corpusAdded, stemDocument)

dtmAdded = DocumentTermMatrix(corpusAdded)

If you type dtmAdded, you can see that there are 6675 terms.

1.3) Bags of Words

Filter out sparse terms by keeping only terms that appear in 0.3% or more of the revisions, and call the new matrix sparseAdded. How many terms appear in sparseAdded?

```r
sparse_added = removeSparseTerms(dtm_added, 0.997)
sparse_added
```

```
## <<DocumentTermMatrix (documents: 3876, terms: 166)>>
## Non-/sparse entries: 2681/640735
## Sparsity           : 100%
## Maximal term length: 28
## Weighting          : term frequency (tf)
```

Explanation

You can create the sparse matrix with the follow line:

sparseAdded = removeSparseTerms(dtmAdded, 0.997)

If you type sparseAdded, you can see that there are 166 terms.

1.4) Bags of Words

Convert sparseAdded to a data frame called wordsAdded, and then prepend all the words with the letter A, by using the command:

```
typeof(sparse_added)
```

```
## [1] "list"
```

```
words_added = as.data.frame(as.matrix(sparse_added))
```

```
colnames(words_added) = paste("A", colnames(words_added))
```

Now repeat all of the steps we've done so far (create a corpus, remove stop words, stem the document, create a sparse document term matrix, and convert it to a data frame) to create a Removed bag-of-words dataframe, called wordsRemoved, except this time, prepend all of the words with the letter R:

```
corpus_removed = VCorpus(VectorSource(wiki$Removed))

corpus_removed = tm_map(corpus_removed, removeWords, stopwords('english'))

corpus_removed = tm_map(corpus_removed, stemDocument)
```

```
dtm_removed = DocumentTermMatrix(corpus_removed)
dtm_removed
```

```
## <<DocumentTermMatrix (documents: 3876, terms: 5403)>>
## Non-/sparse entries: 13293/20928735
## Sparsity           : 100%
## Maximal term length: 784
## Weighting          : term frequency (tf)
```

```
words_removed = removeSparseTerms(dtm_removed, 0.997)
words_removed
```

```
## <<DocumentTermMatrix (documents: 3876, terms: 162)>>
## Non-/sparse entries: 2552/625360
## Sparsity           : 100%
## Maximal term length: 28
## Weighting          : term frequency (tf)
```

```
words_removed = as.data.frame(as.matrix(words_removed))
```

```
colnames(words_removed) = paste("R", colnames(words_removed))
```

How many words are in the words_removed data frame?

```
str(words_removed)
```

```
## 'data.frame':    3876 obs. of  162 variables:
```

```
##  $ R 2000000                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R 40000                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R accord                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R actual                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R ago                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R agre                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R analog                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R appar                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R arbitrari                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R believ                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R bigger                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R biolog                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R biologyanalog              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R bodi                       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R call                       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R care                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R clear                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R close                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R combin                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R communic                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R communiti                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R compar                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R complet                    : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R concept                    : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R contact                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R contain                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R correl                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R cours                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R creat                      : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R cromagnon                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R defin                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R deriv                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R develop                    : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R differ                     : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R direct                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R discuss                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R distinct                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R dutch                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R easytolearn                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R estim                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R exampl                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R experiment                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R express                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R famili                     : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R fantasi                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R featur                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R first                      : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ R follow                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R forc                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R forti                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R fuck                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R general                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R geograph                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R german                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R gestur                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R get                           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R given                         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R govern                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R group                         : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R habili                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R homo                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R homolog                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R icon                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R idea                          : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R ideolog                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R imposs                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R includ                        : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R individu                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R intern                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R just                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R know                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R langesnada                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R languageenglish               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R languagespanish               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R learn                         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R least                         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R linguist                      : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R linguisticssent               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R literari                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R logic                         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R mainlinguist                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R mainorigin                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R make                          : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ R manipul                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R map                           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R mean                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R method                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R million                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R must                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R name                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R nation                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R notion                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R object                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R onlin                         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R onto                          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R origin                        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R parallel                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ R part                          : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ R pattern                       : num  0 0 0 1 0 0 0 0 0 0 ...
##   [list output truncated]
```

Explanation

To repeat the steps for the Removed column, use the following commands:

corpusRemoved = VCorpus(VectorSource(wiki$Removed))

corpusRemoved = tm_map(corpusRemoved, removeWords, stopwords(“english”))

corpusRemoved = tm_map(corpusRemoved, stemDocument) dtmRemoved = DocumentTermMatrix(corpusRemoved)

sparseRemoved = removeSparseTerms(dtmRemoved, 0.997)

wordsRemoved = as.data.frame(as.matrix(sparseRemoved))

colnames(wordsRemoved) = paste("R", colnames(wordsRemoved))

To see that there are 162 words in the wordsRemoved data frame, you can type ncol(wordsRemoved) in your R console.

1.5) Bags of Words

Combine the two data frames into a data frame called wikiWords with the following line of code:

```r
wiki_words = cbind(words_added, words_removed)
```

The cbind function combines two sets of variables for the same observations into one data frame. Then add the Vandal column (HINT: remember how we added the dependent variable back into our data frame in the Twitter lecture). Set the random seed to 123 and then split the data set using sample.split from the "caTools" package to put 70% in the training set.

```r
wiki_words$Vandal = wiki$Vandal
```

What is the accuracy on the test set of a baseline method that always predicts "not vandalism" (the most frequent outcome)?

```r
set.seed(123)

spl = sample.split(wiki_words$Vandal, SplitRatio = 0.7)

train = filter(wiki_words, spl == TRUE)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```r
test = filter(wiki_words, spl == FALSE)

table(test$Vandal)
```

```
##
##   0   1
## 618 545
```

```r
618/ (618 + 545)
```

```
## [1] 0.5313844
```

Explanation

You can compute this number using the table command:

table(wikiTest$Vandal)

It outputs that there are 618 observations with value 0, and 545 observations with value 1. The accuracy of the baseline method would be 618/(618+545) = 0.531.

1.6) Bags of Words

Build a CART model to predict Vandal, using all of the other variables as independent variables. Use the training set to build the model and the default parameters (don't set values for minbucket or cp).

What is the accuracy of the model on the test set, using a threshold of 0.5? (Remember that if you add the argument type="class" when making predictions, the output of predict will automatically use a threshold of 0.5.)

```r
wiki_cart = rpart(Vandal ~ ., data = train, method = 'class')
```

```r
pred = predict(wiki_cart, newdata = test, type = 'class')
```

```r
table(test$Vandal, pred)
```

```
##    pred
##       0   1
##   0 618   0
##   1 533  12
```

```r
(618 + 12) / (618 + 12 + 533)
```

```
## [1] 0.5417025
```

Explanation

You can build the CART model with the following command:

wikiCART = rpart(Vandal ~ ., data=wikiTrain, method="class")

And then make predictions on the test set:

testPredictCART = predict(wikiCART, newdata=wikiTest, type="class")

And compute the accuracy by comparing the actual values to the predicted values:

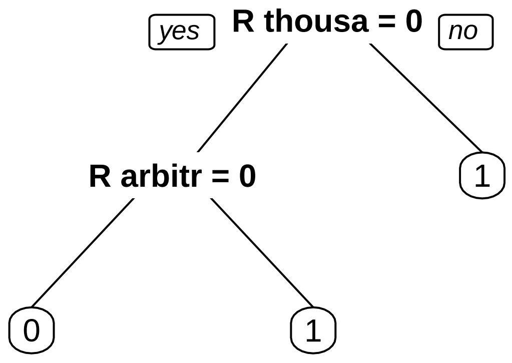
table(wikiTest$Vandal, testPredictCART)

The accuracy is (618+12)/(618+533+12) = 0.5417.

1.7) Bags of Words

Plot the CART tree. How many word stems does the CART model use?

```r
prp(wiki_cart)
```

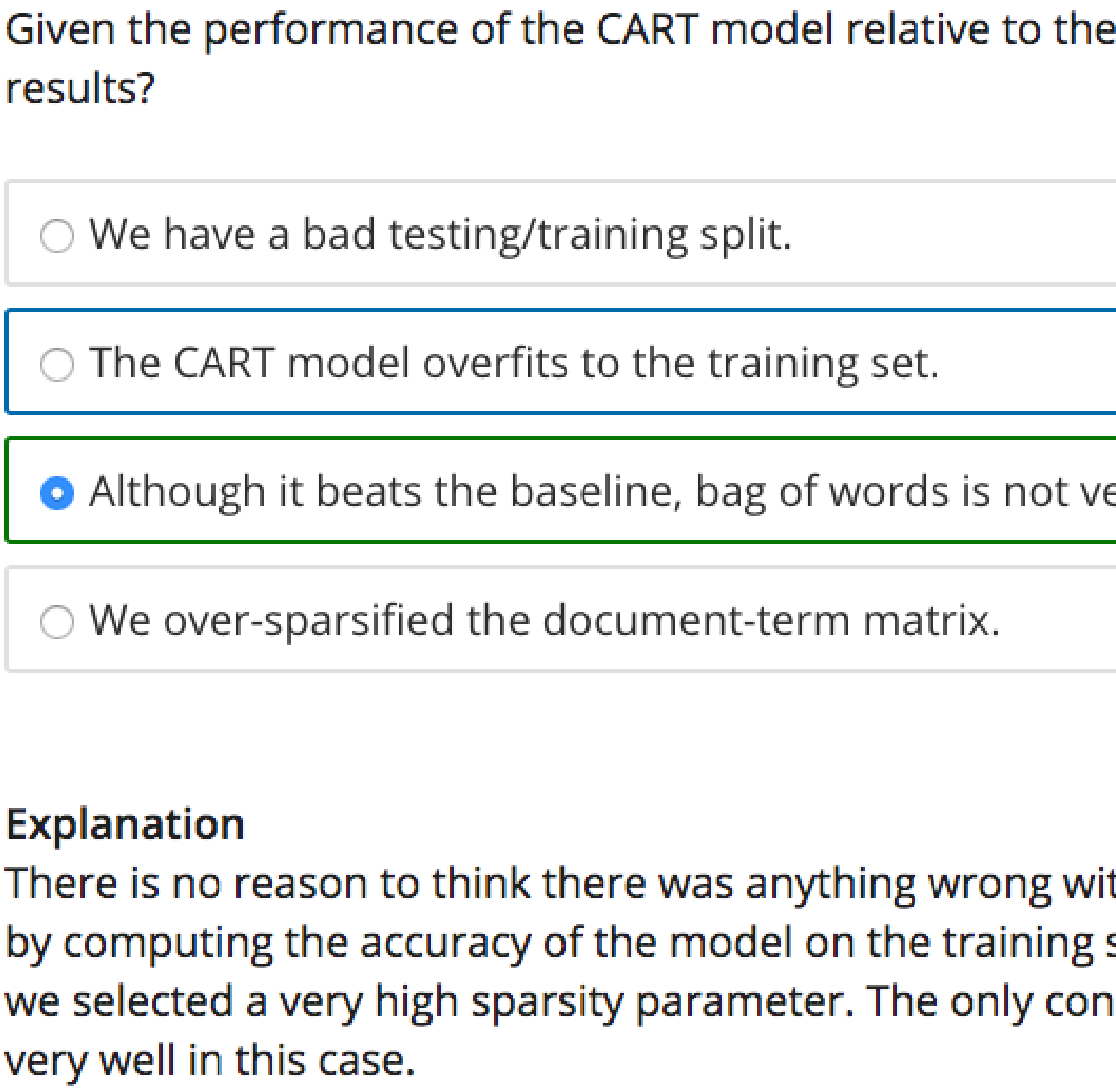


Explanation If you plot the tree with prp(wikiCART), you can see that the tree uses two words: "R arbitr" and "R thousa"

1.8) Bags of Words

Given the performance of the CART model relative to the baseline, what is the best explanation of these results?

```r
knitr::include_graphics('cart.png')
```



Given the performance of the CART model relative to the
results?

○ We have a bad testing/training split.

○ The CART model overfits to the training set.

[✓ Although it beats the baseline, bag of words is not ve]

○ We over-sparsified the document-term matrix.

Explanation

There is no reason to think there was anything wrong wit
by computing the accuracy of the model on the training s
we selected a very high sparsity parameter. The only con
very well in this case.

2.1) Problem-specific Knowledge

We weren't able to improve on the baseline using the raw textual information. More specifically, the words themselves were not useful. There are other options though, and in this section we will try two techniques

- identifying a key class of words
- counting words.

The key class of words we will use are website addresses. "Website addresses" (also known as URLs - Uniform Resource Locators) are comprised of two main parts. An example would be "http://www.google.com".

The first part is the protocol, which is usually "http" (HyperText Transfer Protocol).

The second part is the address of the site, e.g. "www.google.com". We have stripped all punctuation so links to websites appear in the data as one word, e.g. "httpwwwgooglecom".

We hypothesize that given that a lot of vandalism seems to be adding links to promotional or irrelevant websites, the presence of a web address is a sign of vandalism.

We can search for the presence of a web address in the words added by searching for "http" in the Added column. The grepl function returns TRUE if a string is found in another string, e.g.

```r
grepl("cat","dogs and cats",fixed=TRUE) # TRUE
```

```
## [1] TRUE
```

```r
grepl("cat","dogs and rats",fixed=TRUE) # FALSE
```

```
## [1] FALSE
```

Create a copy of your dataframe from the previous question:

```r
wiki_words_2 = wiki_words
```

Make a new column in wikiWords2 that is 1 if "http" was in Added:

```r
wiki_words_2$HTTP = ifelse(grepl("http",wiki$Added,fixed=TRUE), 1, 0)
```

Based on this new column, how many revisions added a link?

```r
table(wiki_words_2$HTTP)
```

```
##
##    0    1
## 3659  217
```

Explanation

You can find this number by typing table(wikiWords2$HTTP), and seeing that there are 217 observations with value 1.

2.2) Problem-specific Knowledge

In problem 1.5, you computed a vector called "spl" that identified the observations to put in the training and testing sets. Use that variable (do not recompute it with sample.split) to make new training and testing sets:

```r
wiki_train_2 = subset(wiki_words_2, spl==TRUE)

wiki_test_2 = subset(wiki_words_2, spl==FALSE)
```

Then create a new CART model using this new variable as one of the independent variables.

```r
wiki_cart_2 = rpart(Vandal ~ ., data = wiki_train_2, method = 'class')
```

What is the new accuracy of the CART model on the test set, using a threshold of 0.5?

```r
pred_2 = predict(wiki_cart_2, newdata = wiki_test_2, type = 'class')
```

```r
table(wiki_test_2$Vandal, pred_2)
```

```
##    pred_2
##       0   1
##    0 609   9
##    1 488  57
```

```
(609 + 57) / (609 + 57 + 488 + 9)
```

```
## [1] 0.5726569
```

Explanation

You can compute this by running the following commands:

wikiCART2 = rpart(Vandal ~ ., data=wikiTrain2, method="class")

testPredictCART2 = predict(wikiCART2, newdata=wikiTest2, type="class")

table(wikiTest2$Vandal, testPredictCART2)

Then the accuracy is (609+57)/(609+9+488+57) = 0.5726569.

2.3) Problem-Specific Knowledge

Another possibility is that the number of words added and removed is predictive, perhaps more so than the actual words themselves. We already have a word count available in the form of the document-term matrices (DTMs).

Sum the rows of dtmAdded and dtmRemoved and add them as new variables in your data frame wikiWords2 (called NumWordsAdded and NumWordsRemoved) by using the following commands:

```
wiki_words_2$NumWordsAdded = rowSums(as.matrix(dtm_added))

wiki_words_2$NumWordsRemoved = rowSums(as.matrix(dtm_removed))
```

What is the average number of words added?

```
mean(wiki_words_2$NumWordsAdded)
```

```
## [1] 4.050052
```

Explanation You can get this answer with mean(wikiWords2$NumWordsAdded).

2.4) Problem-Specific Knowledge

In problem 1.5, you computed a vector called "spl" that identified the observations to put in the training and testing sets. Use that variable (do not recompute it with sample.split) to make new training and testing sets with wikiWords2. Create the CART model again (using the training set and the default parameters).

What is the new accuracy of the CART model on the test set?

```
wiki_train_3 = subset(wiki_words_2, spl == TRUE)
wiki_test_3 = subset(wiki_words_2, spl == FALSE)

wiki_cart_3 = rpart(Vandal ~., data = wiki_train_3, method = 'class')

pred_3 = predict(wiki_cart_3, newdata = wiki_test_3, type = 'class')

table(wiki_test_3$Vandal, pred_3)
```

```
##    pred_3
##       0   1
##   0 514 104
##   1 297 248
```

```
(514 + 248) / (514 + 248 + 297 + 104)
```

```
## [1] 0.6552021
```

Explanation

To split the data again, use the following commands:

wikiTrain3 = subset(wikiWords2, spl==TRUE)

wikiTest3 = subset(wikiWords2, spl==FALSE)

You can compute the accuracy of the new CART model with the following commands:

wikiCART3 = rpart(Vandal ~ ., data=wikiTrain3, method="class")

testPredictCART3 = predict(wikiCART3, newdata=wikiTest3, type="class")

table(wikiTest3$Vandal, testPredictCART3)

The accuracy is (514+248)/(514+104+297+248) = 0.6552021.

3.1) Using Non-Textual Data

We have two pieces of "metadata" (data about data) that we haven't yet used. Make a copy of wikiWords2, and call it wikiWords3:

```
wiki_words_3 = wiki_words_2
```

Then add the two original variables Minor and Loggedin to this new data frame:

```
wiki_words_3$Minor = wiki$Minor

wiki_words_3$Loggedin = wiki$Loggedin
```

In problem 1.5, you computed a vector called "spl" that identified the observations to put in the training and testing sets. Use that variable (do not recompute it with sample.split) to make new training and testing sets with wikiWords3.

Build a CART model using all the training data. What is the accuracy of the model on the test set?

```
wiki_train_4 = subset(wiki_words_3, spl == TRUE)

wiki_test_4 = subset(wiki_words_3, spl == FALSE)

wiki_cart_4 = rpart(Vandal ~., data = wiki_train_4, method = 'class')

pred_4 = predict(wiki_cart_4, newdata = wiki_test_4, type = 'class')

table(wiki_test_4$Vandal, pred_4)
```

```
##    pred_4
##       0   1
##   0 595  23
##   1 304 241
```

```
(595 + 241) / (595 + 241 + 23 + 304)
```

```
## [1] 0.7188306
```

Explanation

This model can be built and evaluated using the following commands:

wikiCART4 = rpart(Vandal ~ ., data=wikiTrain4, method="class")

predictTestCART4 = predict(wikiCART4, newdata=wikiTest4, type="class")

table(wikiTest4$Vandal, predictTestCART4)

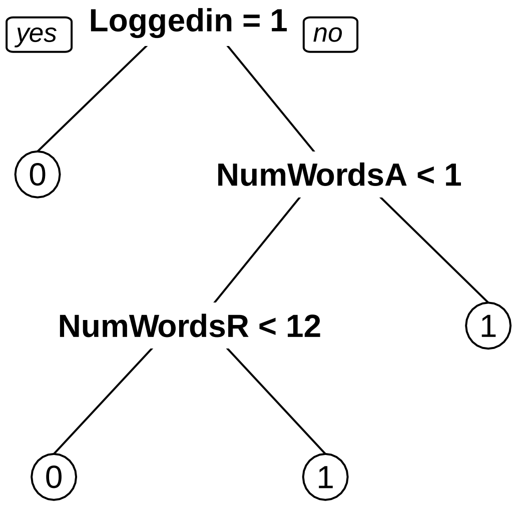
The accuracy of the model is (595+241)/(595+23+304+241) = 0.7188306.

3.2) Using Non-Textual Data

There is a substantial difference in the accuracy of the model using the meta data. Is this because we made a more complicated model?

Plot the CART tree. How many splits are there in the tree?

```r
prp(wiki_cart_4)
```



Explanation

You can plot the tree with prp(wikiCART4). The first split is on the variable "Loggedin", the second split is on the number of words added, and the third split is on the number of words removed.

By adding new independent variables, we were able to significantly improve our accuracy without making the model more complicated!