Detecting Vandalism on Wikipedia

Contents

Introduction

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 lowercasing 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.

Exercices

1. Bags of Words

Problem 1.1 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

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

```
## [1] "C/C/C/C/C/en_CA.UTF-8"
```

```
## 'data.frame':    3876 obs. of  7 variables:
##  $ X.1     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Vandal  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Minor   : int  1 1 0 1 1 0 0 0 1 0 ...
##  $ Loggedin: int  1 1 1 0 1 1 1 1 1 0 ...
##  $ Added   : chr  "  represent psycholinguisticspsycholinguistics orthographyorthography help text al
##  $ Removed : chr  " " " talklanguagetalk" " regarded as technologytechnologies human first" "  repres
```

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

Answer : 1815

Explanation :
You can load the data using the command:

And then convert Vandal to a factor with the command:

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

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

Problem 1.2 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".
2) Remove the English-language stopwords.
3) Stem the words.
4) Build the DocumentTermMatrix, and call it dtmAdded.

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?

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

Answer : 6675

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

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

Problem 1.3 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?

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

Answer : 166

Explanation :
You can create the sparse matrix with the follow line:

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

Problem 1.4 Convert sparseAdded to a data frame called wordsAdded, and then prepend all the words with the letter A, by using the command:

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:

How many words are in the wordsRemoved data frame?

```
## [1] 162
```

Answer : 162

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

To see that there are 162 words in the wordsRemoved data frame, you can type in your R console:

Problem 1.5 Combine the two data frames into a data frame called wikiWords with the following line of code:

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.

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

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

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

Answer : 0.5313844

Explanation :
You can combine the two data frames by using the command:

And then add the Vandal variable by using the command:

To split the data, you can use the following commands:

You can compute this number using the table command:

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.

Problem 1.6 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.)

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

Answer : 0.5417025

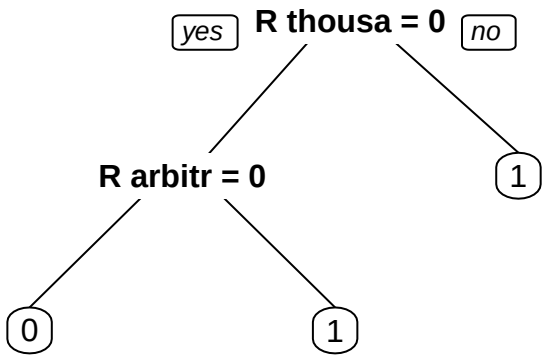
Explanation :
You can build the CART model with the following command:

And then make predictions on the test set:

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

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

Problem 1.7 Plot the CART tree.
How many word stems does the CART model use?



Answer : 2

Explanation :
If you plot the tree with

you can see that the tree uses two words: “R arbitr” and “R thousa”.

Problem 1.8 Given the performance of the CART model relative to the baseline, **what is the best explanation of these results?**

Answer :

1. We have a bad testing/training split.

2. The CART model overfits to the training set.
3. **Although it beats the baseline, bag of words is not very predictive for this problem.**
4. We over-sparsified the document-term matrix.

Explanation :
There is no reason to think there was anything wrong with the split. CART did not overfit, which you can check by computing the accuracy of the model on the training set. Over-sparsification is plausible but unlikely, since we selected a very high sparsity parameter. The only conclusion left is simply that bag of words didn't work very well in this case.

2. Problem specific Knowledge

Problem 2.1 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, and 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.

Create a copy of your dataframe from the previous question:

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

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

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

Answer : 217

Explanation :

You can find this number by typing

and seeing that there are 217 observations with value 1.

Problem 2.2 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:

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

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

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

Answer : 0.5726569

Explanation :
You can compute this by running the following commands:

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

Problem 2.3 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:

What is the average number of words added?

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

Answer : 4.050052

Explanation :
You can get this answer with

Problem 2.4 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?

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

Answer : 0.6552021

Explanation :
To split the data again, use the following commands:

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

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

3. Using Non-Textual Data

Problem 3.1 We have two pieces of "metadata" (data about data) that we haven't yet used. Make a copy of wikiWords2, and call it wikiWords3:

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

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?

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

Answer : 0.7188306

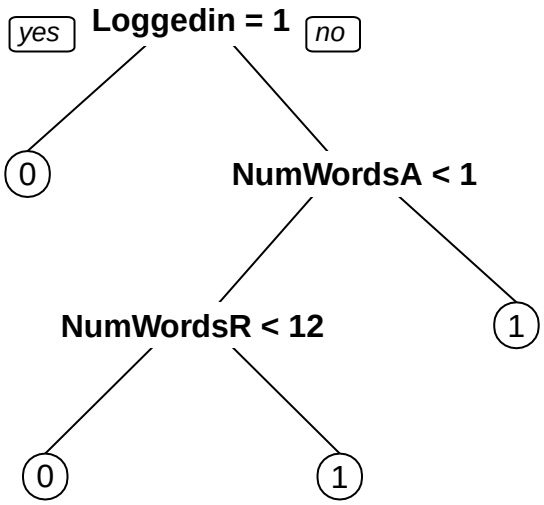
Explanation :
This can be done with the following two commands:

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

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

Problem 3.2 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?



Answer : 3

Explanation :
You can plot the tree with :

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!