Group final project - BAN400 fall 2020

Candidate: (. . .)

6/11/2020

Contents

Project suggestion Group 30 **2**

Initial proposal: . . . 2
Method: . . . 2
Analysis: . . . 2
Sources . . . 2
Workflow: . . . 3
1. getting the data to run in R and tidyverse . . . 3
2. Separate the data into training and test data. . . . 4
3 Visualizing the data . . . 5
4. Training the machine learning model . . . 10
5. make a shiny app, which will work as an interface to plug in new data and see if the news stories is true or fake. . . . 11

Project suggestion Group 30

Initial proposal:

"Textual data analysis combined with regression analysis using data on fake/true news"

Method:

- We create regressors based on the words present in the fake and real news. The regressors are based on both the formatting of the article, such as: *Number of words in the body and title of the article; Number of punctuation characters in the body and title; Number of exclamation marks in the body and title; Number of characters in upper case in the body and title.* And on the words present within the article, such as: *Words expressing positive or negative sentiment, words relating to relevant American politicians, words that relate to media itself.*
- Then use a predictive model to see if the news are real or fake based on these factors. We run multiple models and choose the one with the best predictive score against the test dataset.
- Finally, we can create a "shiny ap" which allows up to paste news articles in, the ap will then preprocess the article and give us a score of "fake probability".

Analysis:

- *Tokenizing and prepossessing before doing any textual analysis.* We need to prepossess the data, which mean shaping it in order for the different models to read them. Before that, we store the peculiarities of the text (such as the amount of punctiation characters present, for example).
- *Visualization of the data.* Since we are working with a large corpus (body of text), it is important to employ some techniques in order to better understand what the text we are using is talking about. We use frequency tables, wordclouds and KWIC (Keyword in context) tables to define what words are likely to carry information.
- *Sentiment analysis.* It is used to gather data on how negative /positive the fake news is compared with true news. Maybe there is a specific sentiment present which would add value to our analysis.
- *Regression*, where 1 is fake news and 0 is true news. First we use a linear regression to understand the relationship between our regressors and our prediction. Then we run multiple machine learning algorithms, and we keep whichever performs best.

Sources

For this we will use a data set from Kaggle: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset (Links to an external site.) with data from 2016 to 2017.

All graphs will be visualized using ggplot package.

Sentiment dictionary:

Baccianella S., Esuli, A. and Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. International Conference on Language Resources and Evaluation.

Workflow:

1. getting the data to run in R and tidyverse

1.1 Merging true and fake news datasets, adding a new dummy column if news is true or fake.

1.2 Prepossesing and getting the data ready for modelling.

—- **Extracting counts of the formatting choices** —-

Prepossesing the data means removing most of the formatting. However, we will use some formatting choices as regressors, so before shaping the text we need to extract the information we need.

—- *Prepossessing the data* —-

Now we can preposses the data. In the case of our corpus, we:

- Removed datapoints with no title, and bodies with less than 30 words.
- Removed uppercase letters.
- Removed any character that isn't alphabetical
- Removed any words that were shorter than 2 letters and larger than 21 letters
- Removed stopwords (words that are frequent but carry no meaning).

—- *Tokenizing the data and creating bigrams* —-

Here we will split the data word by word. We will do the analysis with unigrams (which is every word separated individually), and bigrams (which are words separated in twos). The individual words will be used for the sentiment analysis, while bigrams will give us better keywords.

—- *Adding the sentiment score* —-

The code will run through every single word in our text corpus and rate them with a sentiment ranging from -1 (bad) to 1 (good). Then it will return the average score for each article.

NOTE: Adding the sentiment score takes a long time, *over 2 hours with 8 cores*, the sentiment.rds file should have been included with this code, and it contains the data we get after we run the code. If you would prefer not to wait for the code to run, *skip the next chunk.*

—- *Adding more predictors based on our visualization* —-

We added this step after performing the visualization. Here we create a score based on the keyword topics we decided after looking at the data. We decided to create a politician per word score, and a media per word score. We decided that bigrams would carry more meaning in our analysis, and they were picked based on what we saw within the training data and what we believe to be relevant. This does mean that there is a higher potential for overfitting these predictors to the training data.

The bigrams we chose for politicians were:

- *barack_obama, bernie_sanders, donald_trump, hillary_clinton and joe_biden*

The bigrams we chose for media were:

- *twitter_com, pic_twitter, getty_images, fox_news, social_media, fake_news, york_times, washington_post, mainstream_media, via_youtube, via_getty, via_video, facebook_page, youtube_com and abc_news*

2. Separate the data into training and test data.

Here we split our data into training and test. First, we set a seed so the test is replicable. Then the split is made with 75% of our data in the training set, and 25% on the test set. The data is stratified with the Fake dummy variable in mind. It is important we go through this step before visualizing the data. Because we create some variables through visualization, and we don't want to introduce bias into our analysis.

3 Visualizing the data

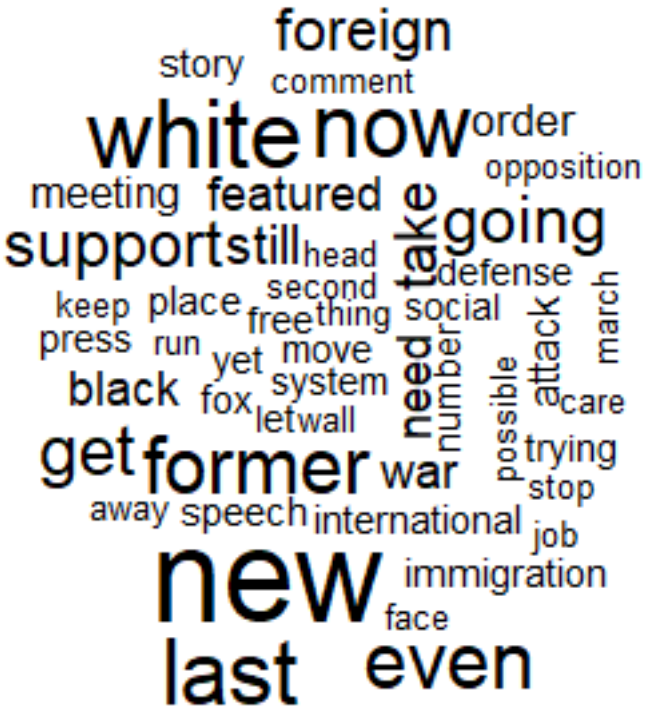
Here we have the steps we took to see how our data actually looked like. Since this data is filled with so much text, we used a Key words in context table to see specific words in the context of where they appear. We also can visuallize the frequency of the words in a wordcloud, to better understand what is being most spoke on in our dataset.

3.1 Visualizing the sentiment analysis

For the sentiment analysis, we decided to go with the sentiment dictionary sentiword. We chose that dictionary after looking at the words present in our data, and checking them in the KWIC matrix to see them in context. We ffound this dictionary in the lexicon package, and its description goes as follows: "*A data.table dataset containing an augmented version of Baccianella, Esuli and Sebastiani's (2010) positive/negative word list as sentiment lookup values.*"

—- *Positive words present in our data* —-

—- *Negative words present in our data* —-



3.2 Making a KWIC (Key words in context) table

this helps us better visualize the text and understand the context of some words

—- *Example of keyword in context: trump on the tokenized data* —-

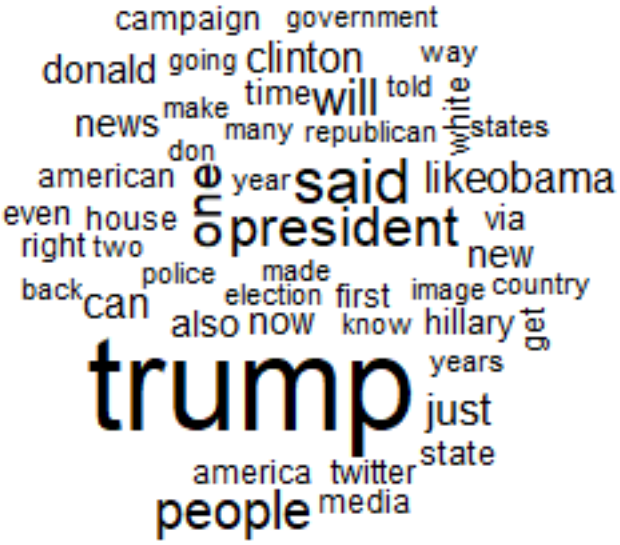
```
## # A tibble: 7 x 3
##   left                             keyword        right
##   <chr>                            <chr>          <chr>
## 1 trump realdonaldtrump december   trump          tweet went welll
## 2 petty infantile gibberish        trump          lack decency won
## 3 pollitt korencarpenter december  trump          new year eve
## 4 know love donald                 trump          realdonaldtrump december nothi~
## 5 love donald trump                realdonaldtru~ december nothing new
## 6 december nothing new             trump          years trump directed
## 7 new trump years                  trump          directed messages enemies
```

—- *Example of keyword in context: donald_trump on the bigram data* —-

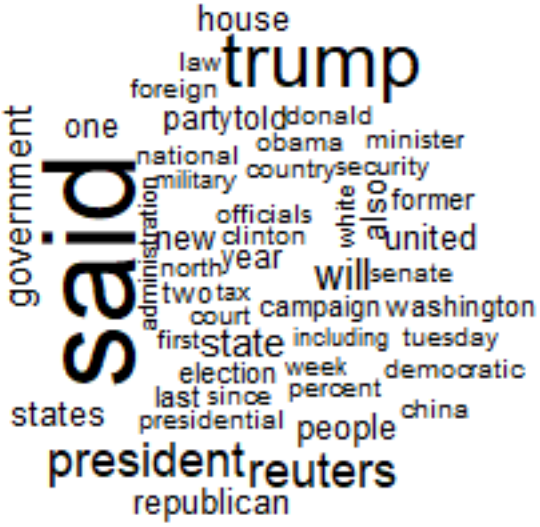
```
## # A tibble: 7 x 3
##   left                            keyword      right
##   <chr>                           <chr>        <chr>
## 1 security_secretary secretary_~  donald_trump trump_administration administrati~
## 2 message_rebuke rebuke_donald    donald_trump trump_without without_even
## 3 decision_derailed derailed_do~  donald_trump trump_plan plan_bar
## 4 getty_images centerpiece_dona~  donald_trump trump_campaign campaign_now
## 5 break_wednesday wednesday_don~  donald_trump trump_looked looked_cameras
## 6 mcconnell_knows knows_donald    donald_trump trump_destroying destroying_gop
## 7 inequality_words words_donald   donald_trump trump_paul paul_ryan
```

3.3 Visualizing word frequency

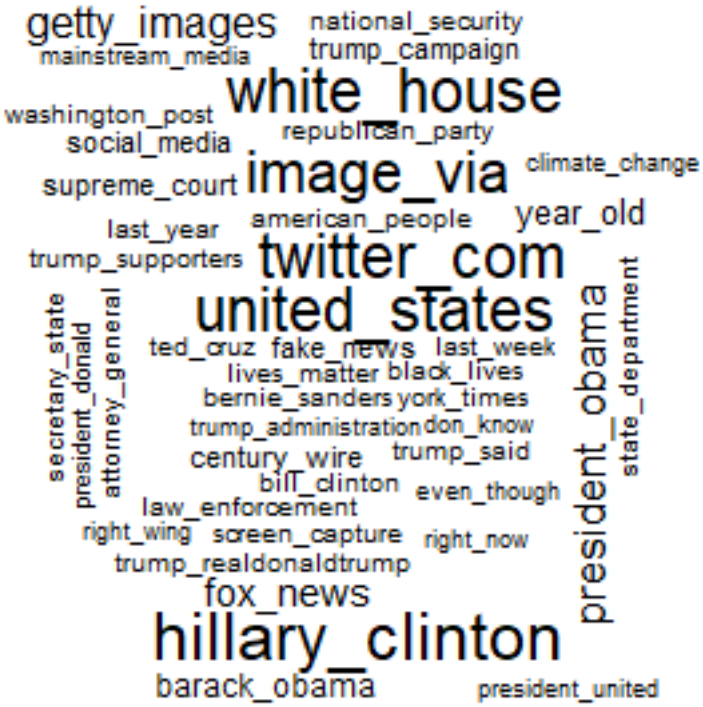
—- *Fake News word frequency* —-



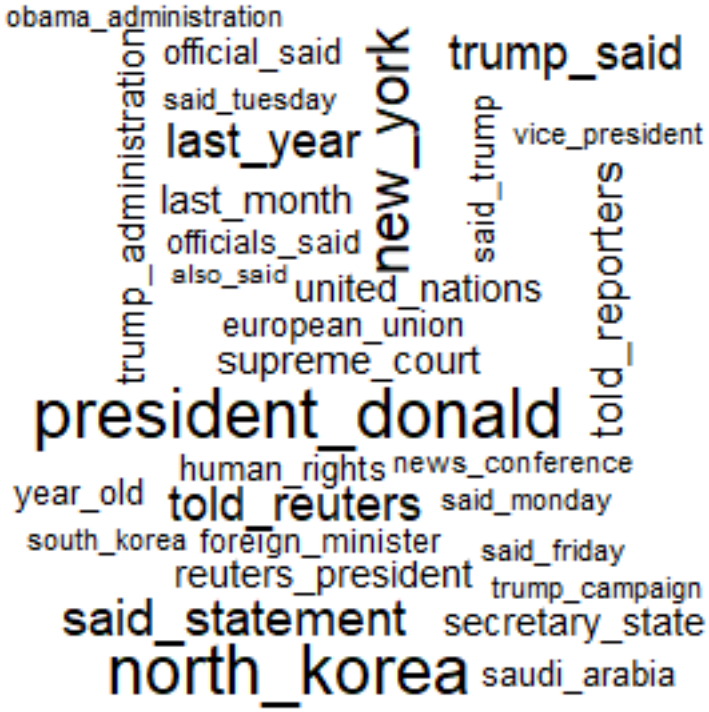
—- *True News word frequency* —-



—- *Fake News bigram frequency* —-

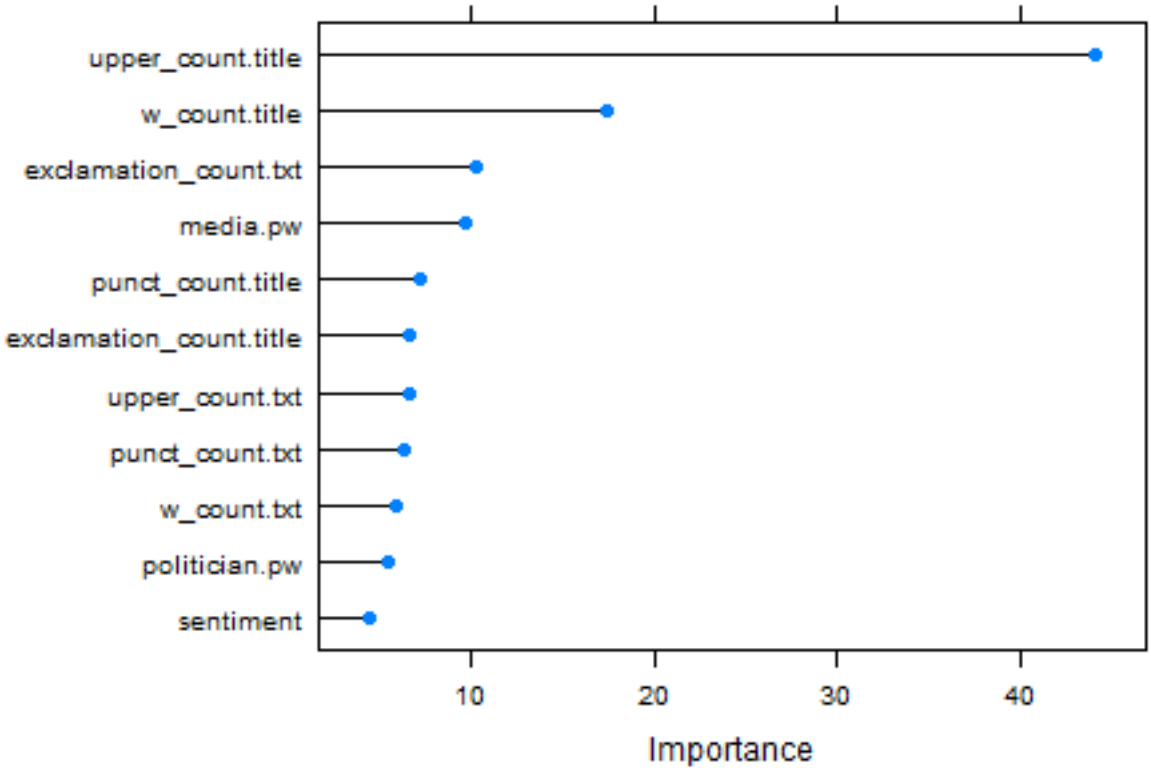


—- *True News bigram frequency* —-



4. Training the machine learning model

—- *Predictor importance plot* —-



Here we can see how well our predictors performed in the linear regression. It gives us a better understanding of how the data is shaped and what was most important on deciding if the news were fake or not. It seems like the formating of the data gives off most information. While the sentiment analysis, despite being statistically relevant, does not give much information.

—- *Testing the trained regression models* —-

Methods	Accuracy	Kappa	Sensitivity	Specificity
glm	0.987171204430088	0.974338777951869	0.988585329341317	0.985794937169914
gbm	0.994185509921551	0.988368267099711	0.992140718562874	0.996175560007285
knn	0.973696354407014	0.947393169918864	0.980538922155689	0.967036969586596
xgbTree	0.990032302722658	0.98005993640473	0.988023952095808	0.991986887634311

Here we see that the GBM machine learning algorithm outperformed the other predictions. It also has a fairly high accuracy, albeit it is important to remember that this accuracy is very dependand on the fact that our test and trainig data comes from the same source.

5. make a shiny app, which will work as an interface to plug in new data and see if the news stories is true or fake.