Group final project - BAN400 fall 2020

Candidate: (. . .)

6/11/2020

Contents

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. So far we can create a sentiment factor, on both the body and the title of the article, and a keyword per word factor, also both on the title and the body.
- Then use a predictive model to see if the news are real or fake based on these factors. We can run a factor regression or perhaps a machine learning model, like an XGBoost model.
- 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 prepossesing before doing any textual analysis, we need to preposess the data, which mean shaping it in order for the different models to read them.
- Topic modelling: Using the words in the dataset to define a concrete topic in each of the files. This could help us see which words to look out for, perhaps there is a topic which is "Hype up words", that might be more present in the fake news. We can create a "topic per word" score.
- Sentiment analysis to gather data on how negative /positive the fake news is compared with true news. Maybe there is a specific sentiment in the fake news that we can extract, we can add these numbers to our regression, to make a predictive model to see if the news are fake or not.
- Run a regression, were 1 is fake news and 0 is true news. It could be a linear regression, or we could experiment with some machine learning model. 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.

- Use machine learning
- Apply it to live data

https://www.r-bloggers.com/2020/10/sentiment-analysis-in-r-with-custom-lexicon-dictionary-using-tidytext/"

https://rstudio-education.github.io/tidyverse-cookbook/import.html https://www.tidyverse.org/blog/2020/06/recipes-0-1-13/

*check out vip package

Workflow:

1. get the data to run in R and tidyverse

1.0 merge true and fake news datasets, adding a new dummy column if news is true or fake.

1.1 Prepossesing and getting the data ready for modelling. —- Cleaning the data —-

Here we do some simple prepossesing steps, like removing characters and words that don't carry any meaning to our analysis.

—- 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). They may give us slightly different topics, though should be somewhat similar, so its important to check for overfitting if using both in a regression.

—- Adding the sentiment score —-

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 —-

2. Separate the data into training and test data.

3 Visualizing the data

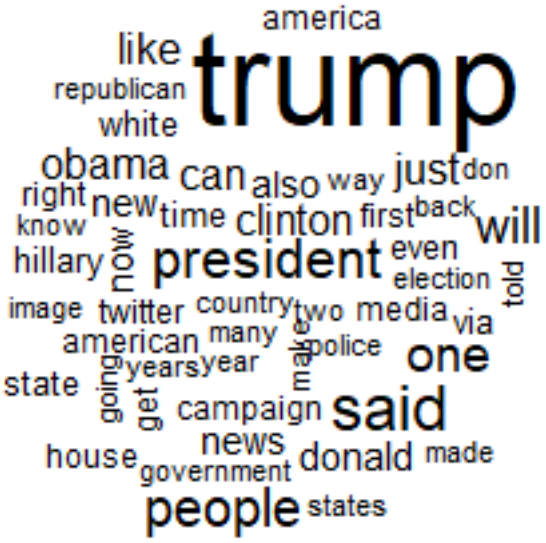
Here we have the steps we took to see how our data actually looked like. Since this data is filled with so much text, we used a Key words in context matrix to see specific words in the context of where they appear. We also can visuallize the frequency of the words in a wordcloud, to better understand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

splitting the clean data back into true and false for the visualization

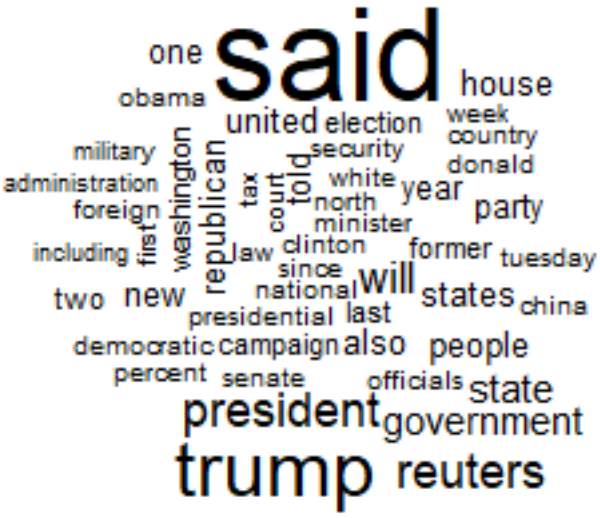
3.1 Making a KWIC (Key words in context) table

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

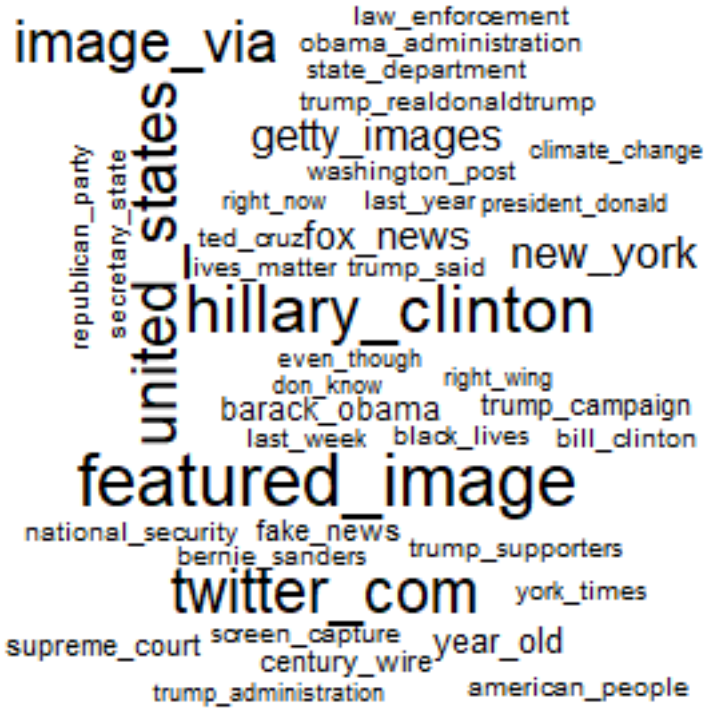
3.2 Visualizing word frequency Fake News word frequency



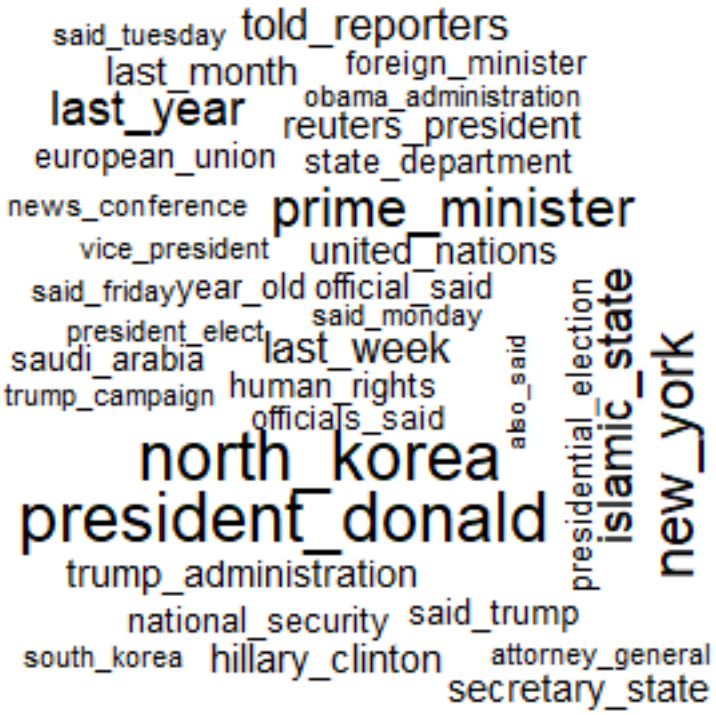
True News word frequency



Fake News bigram frequency



True News bigram frequency



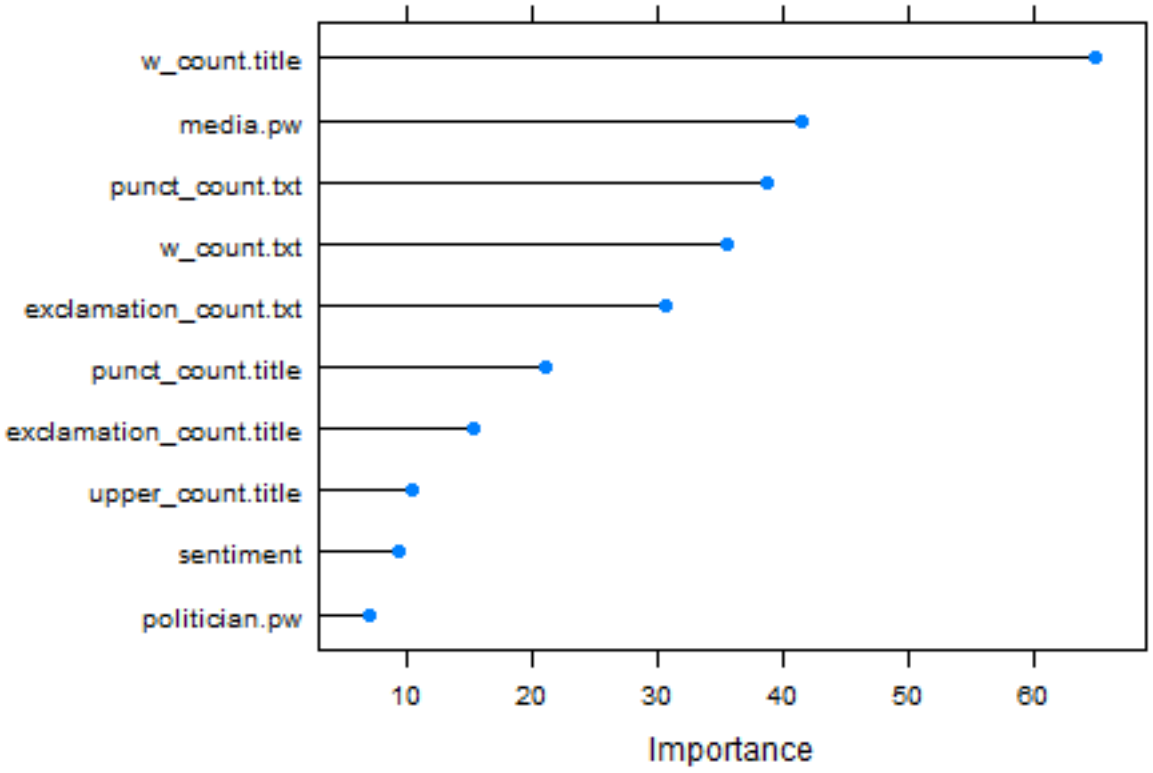
4. Training the machine learning model

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```




```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

Methods	Accuracy	Kappa	Sensitivity	Specificity
glm	0.890447623442547	0.781024391455665	0.917851796407186	0.863777089783282
gbm	0.928934010152284	0.857878163975887	0.9375	0.920597341103624
knn	0.811905860636825	0.623878127256043	0.822791916167665	0.801311236568931
xgbTree	0.908721735117674	0.817530978167484	0.932260479041916	0.885813148788927

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.