Fake News Detection

Rohan Bansal

Sai Prahladh

18752 Project

Problem Statement and Data Collection

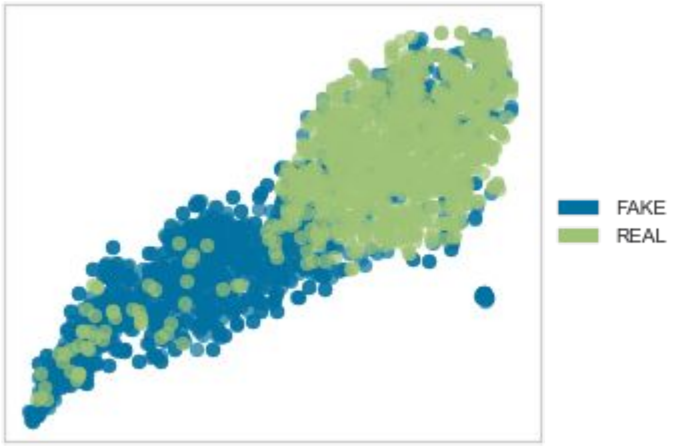
- Problem:
  - The problem we were working on is the classification of an article as fake news or not.
  - Our project looks at various methods which are common in NLP and our aim is to see which one can perform optimally for a bag of words model.

Data Collection :

- We have considered reputed a sources like the New York Times and the Guardian as the sources of our real dataset . We have used their API's to extract data from the 2016 news cycle.
- For our fake data set we are using data from Kaggle and included several features. This data was collected using the webhose.io API which collected data from 244 websites.

Data Visualization

- The real news dataset we gathered is very comprehensive and includes a lot of information about a scraped article.
- The info fields included article id, abstract, headline, keywords, word count etc.
- Amongst all these features we decided to use the title and text of the article as they held the most semantic significance for our bag of words model.



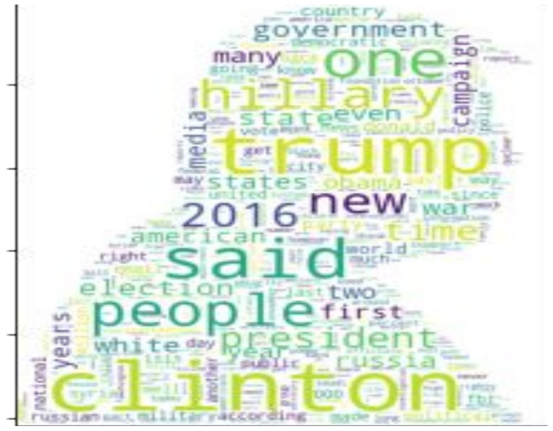
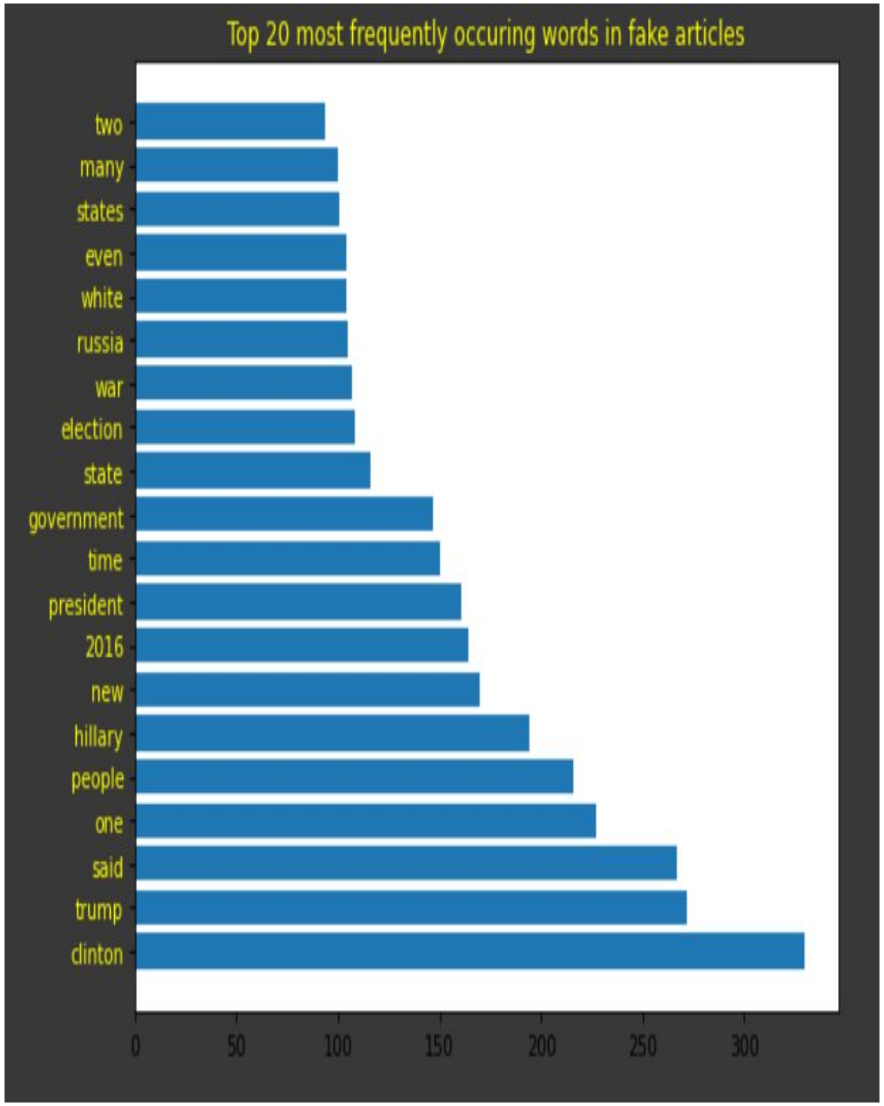
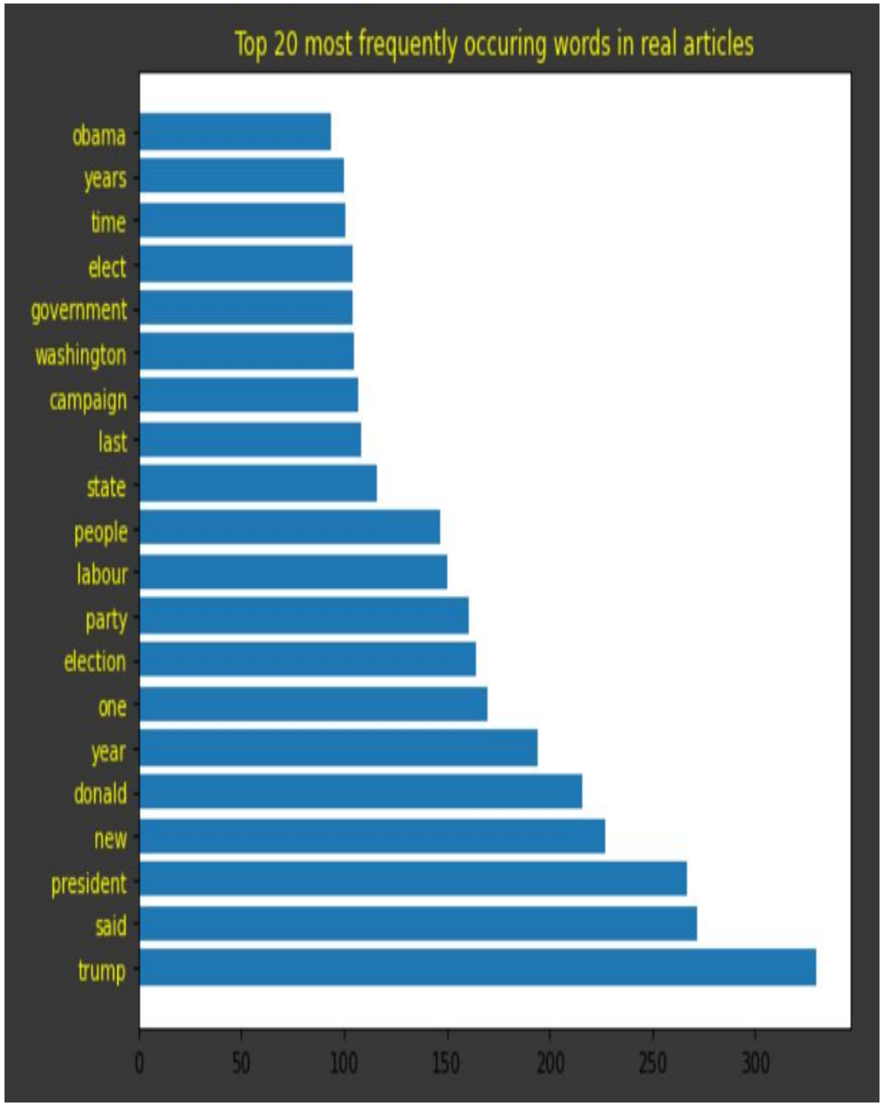


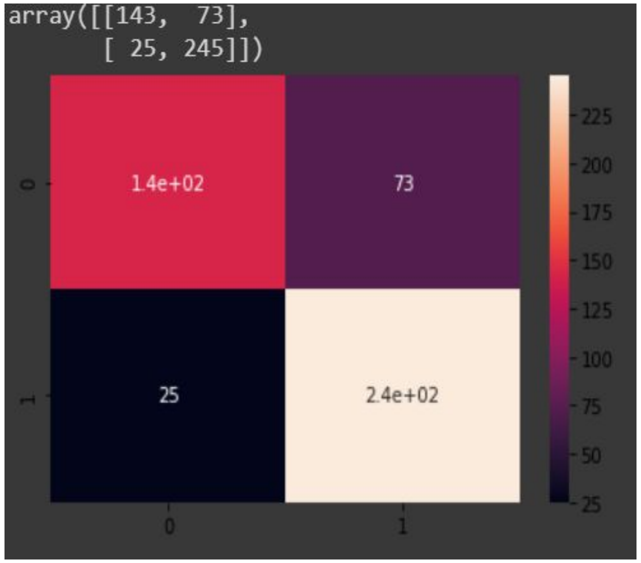
fig1.1. Scatter of the bag of words model.

fig1.2. Word Cloud for fake news.


Top 20 most frequently occuring words in fake articles
two
many
states
even
white
russia
war
election
state
government
time
president
2016
new
hillary
people
one
said
trump
clinton
0
50
100
150
200
250
300

Top 20 most frequently occuring words in real articles
obama
years
time
elect
government
washington
campaign
last
state
people
labour
party
election
one
year
donald
new
president
said
trump
0
50
100
150
200
250
300

Method 1 – Naive Bayes

- For our initial implementation we have used a Naïve Bayes classifier.
- Methodology:
  - We first created a bag of words as a dictionary for our whole dataset and along with which we also created the set of common words for the real and fake dataset.
  - From the labels inside our dataframe, we got the counts of the common words within articles labelled Real and the ones labelled Fake.
  - Using the count of common words from individual articles and their total count for a given label, we calculated the logarithmic probabilities of an article being real or fake
  - The expression for log probabilities:
  - $$P(Y|X) = log(P(Y) * \prod_{i}^{N} P(X_i)^{n_i})$$



```
array([[143,  73],
       [ 25, 245]])
```

fig. Confusion Matrix for Naive Bayes Classifier.

Results : The Confusion Matrix classification accuracy of the Naïve Bayes classifier with an accuracy of 79.83%

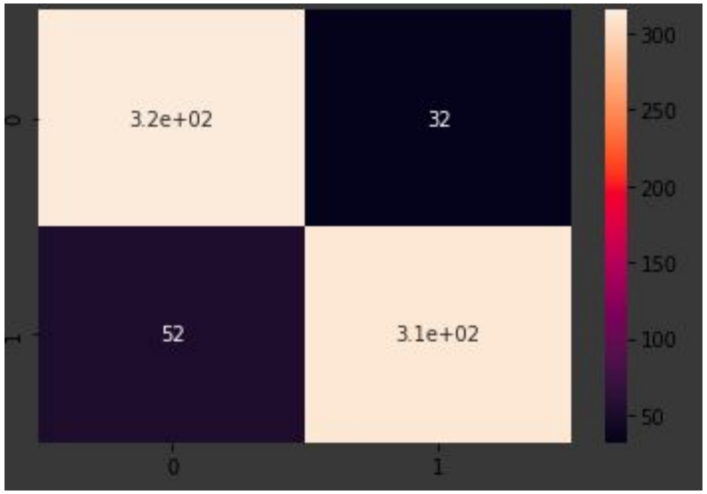
Method 2 – Logistic Regression

Results :

Validation Accuracy :90.77%

Test Accuracy : 88.88%

Confusion Matrix:



Description:

- We have used binomial logistic regression method for our training.
- The training involves using gradient descent of the cross entropy loss function.
- We have used 5-fold cross validation to asses the best regularization term .
- The split between test data and training data is 75-25.
- We have evaluated l1, l2 norms as the regularization parameters and considered a range of values in our evaluation:
- Finally we decide on the following hyperparameters :
- Regularization Term: L2 Norm
- Regularization Value : C=5

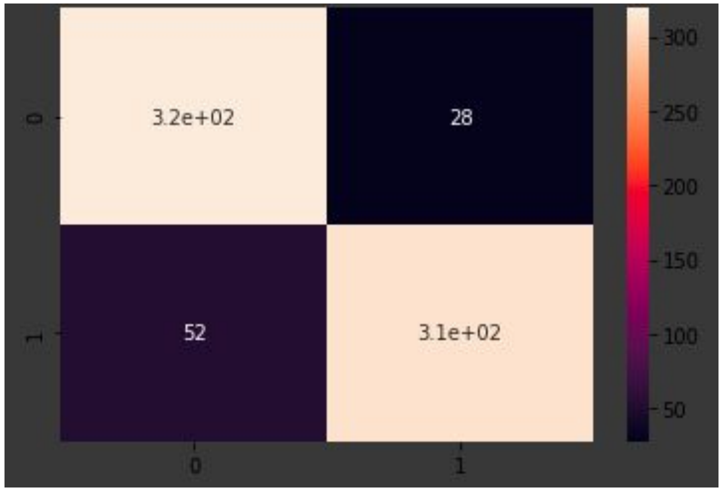
Method 3 : Random Forests

Results :

Test Accuracy : 88.68%

Validation Accuracy:89.02%

Confusion Matrix:



Description:

- For our final implementation we are using ensemble methods which is a random forest model.
- Random forests is like bagging but takes only a subset of features at random from the total.
- The best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered.
- The parameters for selecting the best tree were :
  - splitting criterion- Information gain or gini impurity,
  - The depth of the tree
  - The number of estimators gives how many trees in a random forest.

Discussion

- We learned how essential data collection and feature selection can be.
- One instance of this was the drop in performance our model when we considered just the title compared to the the full body of the article.
- We also realized that various models perform differently for the same classification problem with differing trade-offs.
- A reason why Naive bayes might not have performed well could be due to the fact it relies on independent features as there is correlation, the model does poorly in comparison to Random Forests and LR.
- We believe the library models additionally can capture much more abstraction of the these models and allows them to perform better than our implementation from scratch.

Model	Test Accuracy(%)
Naïve Bayes	79.83
Random Forests	88.88
Logistic Regression	88.68

Q&A

Any Questions?

Vectorization Method

TF - IDF :

- TF denotes the Term Frequency. This is the number of times a word occurs in an article.
- IDF denotes the Inverse Term Document Frequency. This refers to the total number of times a word occurs in the entire documents.
- Both these terms comprise our vectorized format for our bag of words model.
- This vectorized format is used in all the Methods implemented as the input for the classification purpose.

Naive Bayes

The Key Idea

- Represent a document X as a set of (w, a frequency of w) pairs.
- For each label y, build a probabilistic model P(X| Y = y) of documents in class y.
- To classify, select label y which is most likely to generate X:

$$y' = \underset{y}{\text{argmax}} P(X|y) * P(y)$$

The probability of a given word belonging to a class is then given in terms of Bayes rule.

The probabilities are found by raising the IDF probabilities by the TF values and then taking a log of it.

$$P(w_1, \cdots, w_n | Y = y) = \prod_{i=1}^{n} P(w_i | Y = y)$$

$$P(W = w_i | Y = y) = \frac{count(W = w_i \ \& \ Y = y)}{count(Y = y)}$$

Naive Bayes(Contd.)

- The criteria on which our Naive Bayes Model classifies the dataset is based on the log probabilities of the labels for a given article, i.e the probability if its real or fake.
- For Xn features (X1….Xn) which are the words, the log probabilities for their label can be calculated as follows.

$$P(Y|X) = log(P(Y) * \prod_{i}^{N} P(X_i)^{n_i})$$

- In this expression, we are calculating the conditional probability of the labels given the data. P(Y) represents the prior of a given label observed from our dataset. ‘ni’ represents the frequency of a given feature in any selected article.
- So we find P(Y=real|X) and P(Y=fake|X). We assign the label of the greater probability to the article under consideration.

Logistic Regression

1. The key idea is to decide how important(using weights) a word is to a real news article and how important it is to a fake news article.
2. The cross entropy loss if given by the following equation for y being the true value and y cap. $\log p(y|x) = \log \left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$
3. To compute the optimal weights use a gradient descent approach where the gradient is where L is the loss function and is parameterised by theta.

$$\nabla_{\theta} L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix}$$

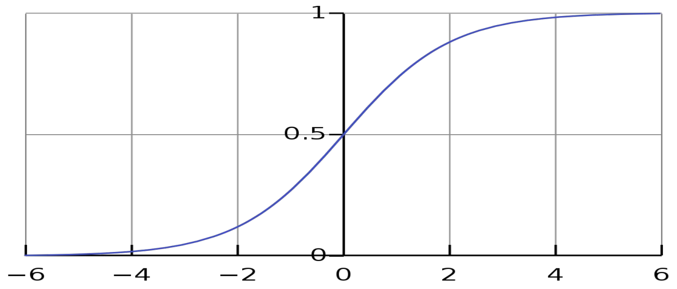
Logistic Regression(contd.)

The final equation for gradient descent becomes with respect to our sigmoid cross validation error function is

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$$

The magnitude of the amount to move in gradient descent is the value of the weighted slope by a learning rate η . A higher (faster) learning rate means that there is a faster probability of convergence but the drawback is it could lead to the model never fitting correctly.

The sigmoid function outputs a probability allowing use to give a weighted score to the function.



Logistic Regression(Working Details)

1. The accuracy when we use simply the headline, we have less context is 76.13% a significant drop when we are trying to predict just using the headline.
2. We have used 5 fold classification and used Sklearn GridSearchCv to perform this validation and to obtain the optimal parameters.
3. We have used the same dataset to compare the respective models.
4. Cross validation accuracy is : 90.77%
5. Hyperparameter: C= 5
6. Regularization Type : L2 Norm

Random Forests

Random Forests are based on the key principles of Decision trees.

For choosing the splitting criterion of a decision tree we are use gini impurity.

The Gini impurity is just used to measure how often a randomly chosen point would be incorrectly labeled.

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

Here C is the total number of classes and p(.) is the probability of picking a datapoint from a given class ‘i’.

We use the feature with the higher gini gain to finally split the features

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Random Forests(contd.)

Bagging: Random forests works on the principle of bagging.

This means we select a random sample with replacement from the training sample and fit trees for these samples.

Given that we bag the terms B times and we have unseen samples x' then to predict the class of this new article we have the predicted class as by taking the majority vote. Here the f_b function is the trained decision tree from the bag.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

By using random forests we are able to reduce the correlation among the trees in the bag without increasing variance too much.

We sample some of all the features d, as m such that m ≤ d. Typically m is selected to be the square root of d.

Random Forests(contd.)

The implementation details are as follows:

HyperParameters:

Max Depth of a tree :20

Number of Estimators : 100

Splitting Criterion: Gini Impurity

Validation Accuracy: 89.02%

Test Accuracy: 88.68%

Software Used

We have used the following software and tools:

1. Python - Google Collab & Jupytr Notebook
2. NYT API & Guardian API
3. Various Python libraries :
   a. Pandas
   b. Scikit Learn
   c. NLTK
   d. Matplotlib
4. References:
   a. https://developer.nytimes.com/docs/articlesearch-product/1/overview
   b. https://open-platform.theguardian.com/documentation/
   c. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
   d. https://www.kdnuggets.com/2017/04/machine-learning-fake-news-accuracy.html