Automated Fake News Detection

BIA-660-C
Instructor: Prof. Rong Liu

Viveksinh Solanki
Akshay Rane
Ronald Fernandes
Gaurang Patel

Introduction

Fake news detection is the ongoing research area under Natural Language Processing(NLP) domain. In simpler terms, fake news can be defined as any information which is misleading and verifiably false. Fake news can be harmful as it can propagate conspiracy or mistrust.

Motivation and objective

Traditional approach to detect the fake news against the real news is done by human annotators who manually check all the news and decide its genuineness based on the available truthful resources. This approach is very time consuming and tedious. Hence, we are borrowing methods from Machine Learning and Natural Language Processing literature and applying those to automate the classification of fake news.

In the past couple of years, India has faced many mob lynching incidents due to the spread of fake news on one of the top social media platforms, like WhatsApp. Fake news had affected the 2016 U.S. presidential elections. Ongoing elections in India are also being influenced via fake news being spread on social media platforms.

Related work

For automatic fake news detection problem, there are mainly two types of approaches, which have been followed widely: linguistic based approaches and fact-checking based approaches. In the later approaches, fake news is compared with the news from multiple reliable resources for classification [1]. We will be following the linguistic based approach, which includes utilizing text properties as features for machine learning(ML) models.

The linguistic approach has yielded promising results in differentiating satire from real news [4]. Kai Shu [5] has proposed "tri-relationship", which is the relationship among publishers, news pieces, and users, as baseline features for machine learning models to classify fake news. Bennett Kleinberg and team [3] have used linear Support Vector Machines(SVM) classifier with ngrams, punctuations, readability, syntax and Psycholinguistic features as different feature sets. This approach [6] uses text and image based convolutional neural networks(TI-CNN) with news title, text and image as features for fake news detection. Also, there has been comprehensive study [2] done in which authors have surveyed different Natural Language Processing(NLP) based approaches on multiple publicly available datasets with different Neural network and non-neural network models.

Scrapping and Data collection

The first and foremost part of any machine learning project is data collection. For our project we needed fake news and real news data. So we first searched through Wikipedia to find the relevant sites to scrape the data.

For fake news data, we found "https://100percentfedup.com" site from [7] which is easy to scrape with less amount of time. This site is good for fake news dataset since it is questionable, and totally based on the extreme right-wing bias through story selection. The site has used poor resources and has been using loaded emotional wordings to promote the propaganda. It has also failed the fact checks from the fact checkers and has complete lack of transparency. First we tried using "Selenium" but it was taking too much time to scrape only 1000 articles, so later on we used the "Beautiful Soup" to scrape 1000 articles from the site. We only scraped the politics category news for our project.

For real news data, we used the NYtimes API "https://api.nytimes.com/" from [8]. We selected this site since NYtimes has won 127 Pulitzer Prizes which is comparatively better than any other media agencies. Also NYtimes is ranked 17th in the world by circulation and 2nd in the USA. Here also, we scraped 1000 articles of the politics category only using "Beautiful Soup" by using API and then from the responded JSON, used the web URLs of each article to scrape the full content.

So at the end, we formatted 2 datasets of fake news, one with only headlines (titles of the news) and another with only content (body of the news). Similarly, we did for real news.

Raw features

We calculated word count, sentence count and punctuation count for raw documents and added them as features in dataset. We performed this step before doing any sort of preprocessing, because we believe that formulating these features before cleaning would allows us to capture bias present in dataset.

Iterative data cleaning, preprocessing and basic analysis

We converted all documents to lowercase, removed punctuations, newline characters and standard stop words. Then we transformed binary categorical labels to binary numerical labels i.e. fake = 1 and real = 0. And finally we dropped duplicate records as well as records with missing values.

By plotting WordCloud, we were able to find repeated words such as author name, advertisements, twitter URLs etc. Also, we found some repeated texts (see Table 1). Next, we plotted word frequency distribution for each label and each dataset to understand the words occurring most of the times in both datasets. This step helped in finding more unrelated words like want, would, anything etc.

Finally, we utilized random forest feature importance to extract the top discriminating features, which helped us in finding repeated n-grams with no inference, more unrelated words and repeated texts. (see Table 1) We performed these 3 steps of cleaning, preprocessing and basic analysis iteratively to reduce the underlying differences, which can introduce bias in our final outcome.

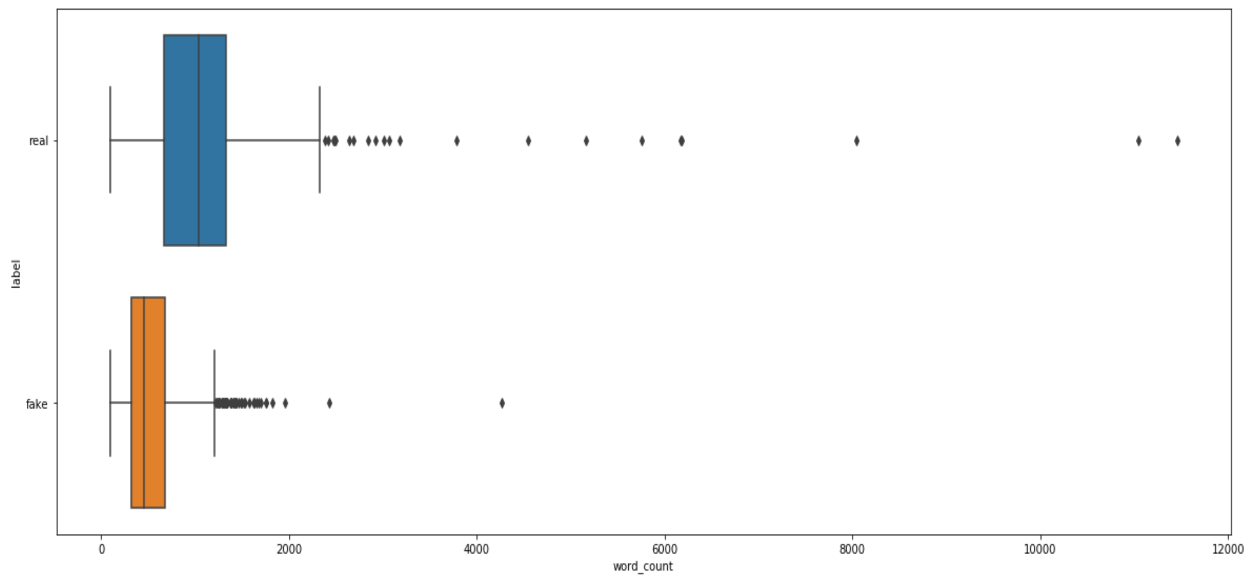
Final data cleaning and preprocessing

We plotted boxplot of word count for fake content vs real content (see figure 1). As we can notice from the boxplot, word count is skewed for real content data. Hence, we decided to keep 75th percentile of fake labels which is approximately 700 as upper bound of allowed word count for any document in out content dataset. Hence, removed documents having word count > 700 from content dataset. After performing this step, we had 757 fake content and 228 real content docs. As we can see, more documents were removed from real content set as it was skewed for word count.

Table 1: repeated texts and n-grams

Repeated Strings	**Frequency**
“welcome guide national host”	84
“busy american biggest stories might missed links read”	23
“stories making news washington today”	127
“love hear email”	241
“forwarded subscribe delivered”	157
“today briefing compiled”	141
“isabella grull”	101
“read story”	51
“trending katie hopkins warns americans grave danger ahead great country become united kingdom”	**977**

Figure 1: Word count for fake vs real content



Next, we did some transformation of words to balance wordings for both labels:
i.e. 1) we replaced all 'dem', 'dems' and 'democrat' to 'democrats'
2) we replaced all 'rep, 'reps' and 'republican' to 'republicans'

Based on the findings from iterative cleaning, preprocessing and analysis, we did further preprocess as follows:

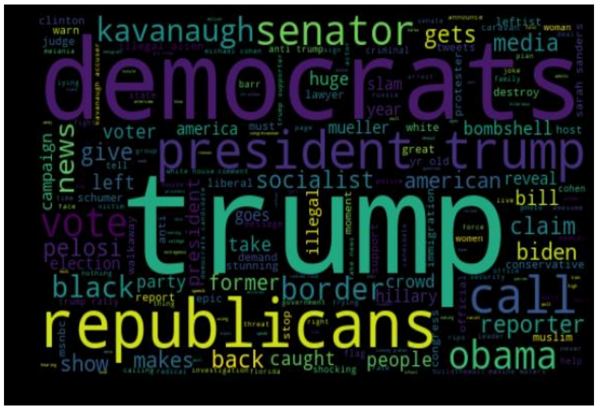
- Removed words with length < 3
- Removed repeated words by adding them to stop words list
- Removed repeated texts mentioned in table 1
- Finally, we subsampled datasets to balance number of labels:
    - Headlines dataset: fake-747 and real-747
    - Content dataset: fake-228 and real-228

Even after doing all of this pre-processing, we found real content dataset to be biased for words. Hence we counted number of words occurring in more than 15% documents, which means words occurring in more than 30 documents. We found count = 585, hence we decided to use max_features=500 for TfidfVectorizer. Here, max features will allow us to control features/words which will be feed to machine learning models. By choosing max_features=500, we only kept top 500 max occurring words. This helped us in disregarding rare/biased words for both labels and eventually resulted in less biased dataset for final models.

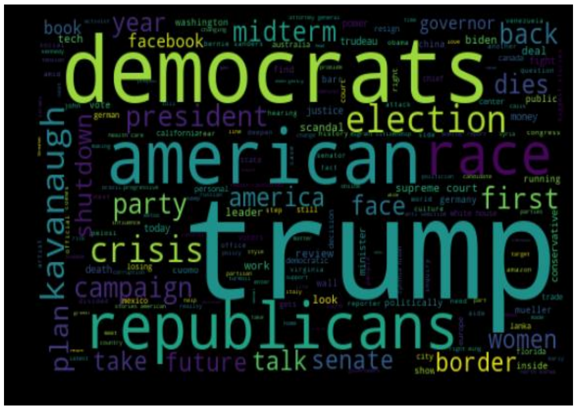
Exploratory Data Analysis

The word cloud in primary Data Analysis helped to identify the most frequently occurring words. These word clouds showed words with high frequency but having no significance i.e. neither affecting the tone nor the meaning of articles. The post EDA done after doing the preprocessing showed the following result for Fake news headlines and contents and Real news headlines and content.

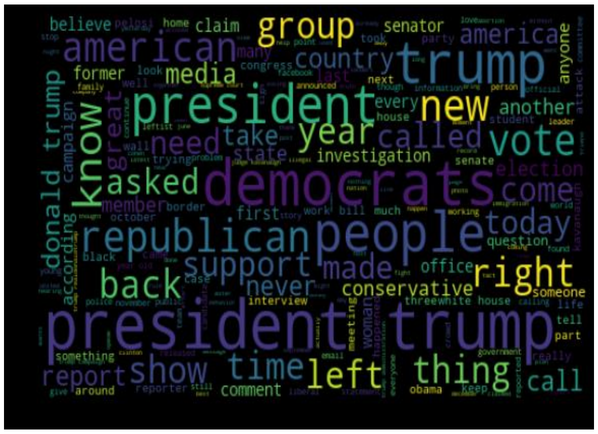
Word Clouds



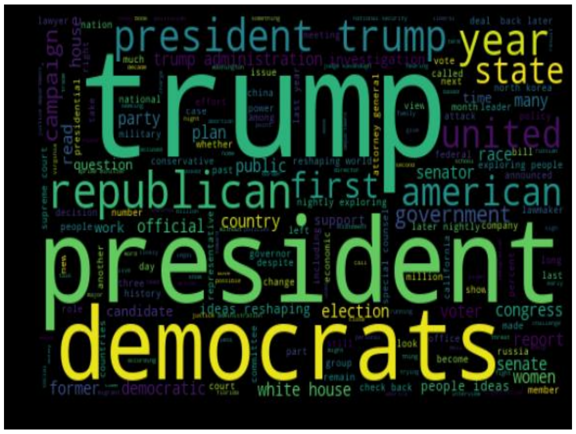
Fake news headlines



Real news headlines



Fake news content



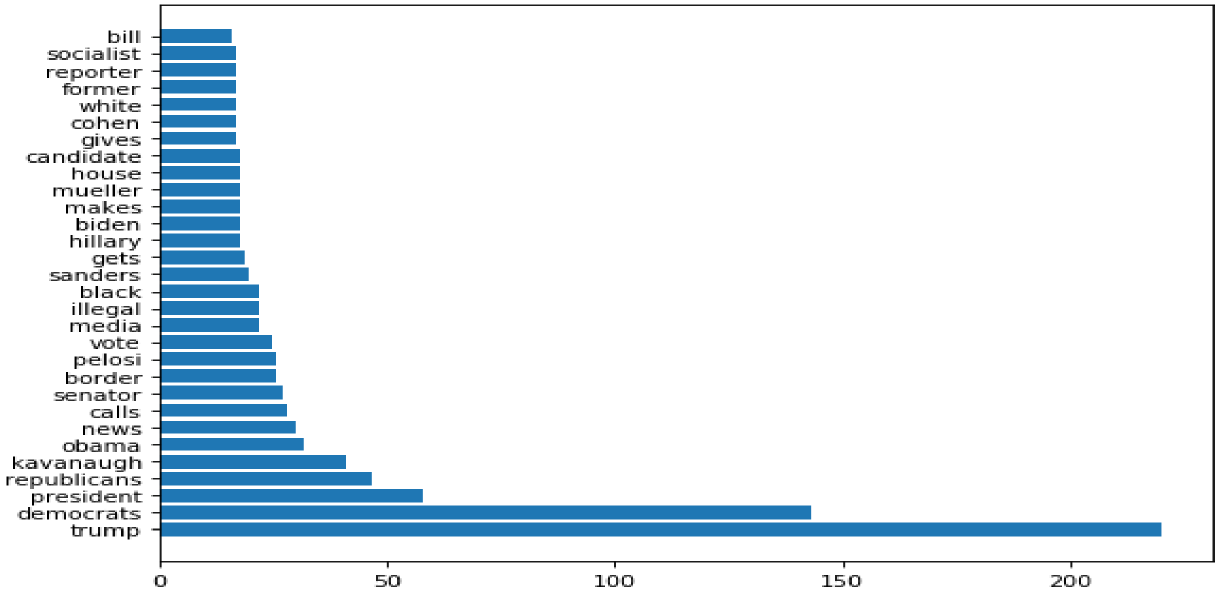
Real news content

The word cloud for fake and real news headlines showed similar high frequency words like trump, democrats, republicans etc. The word cloud for fake and real news content showed similar high frequency words like trump, president, republicans, democrats, year etc.

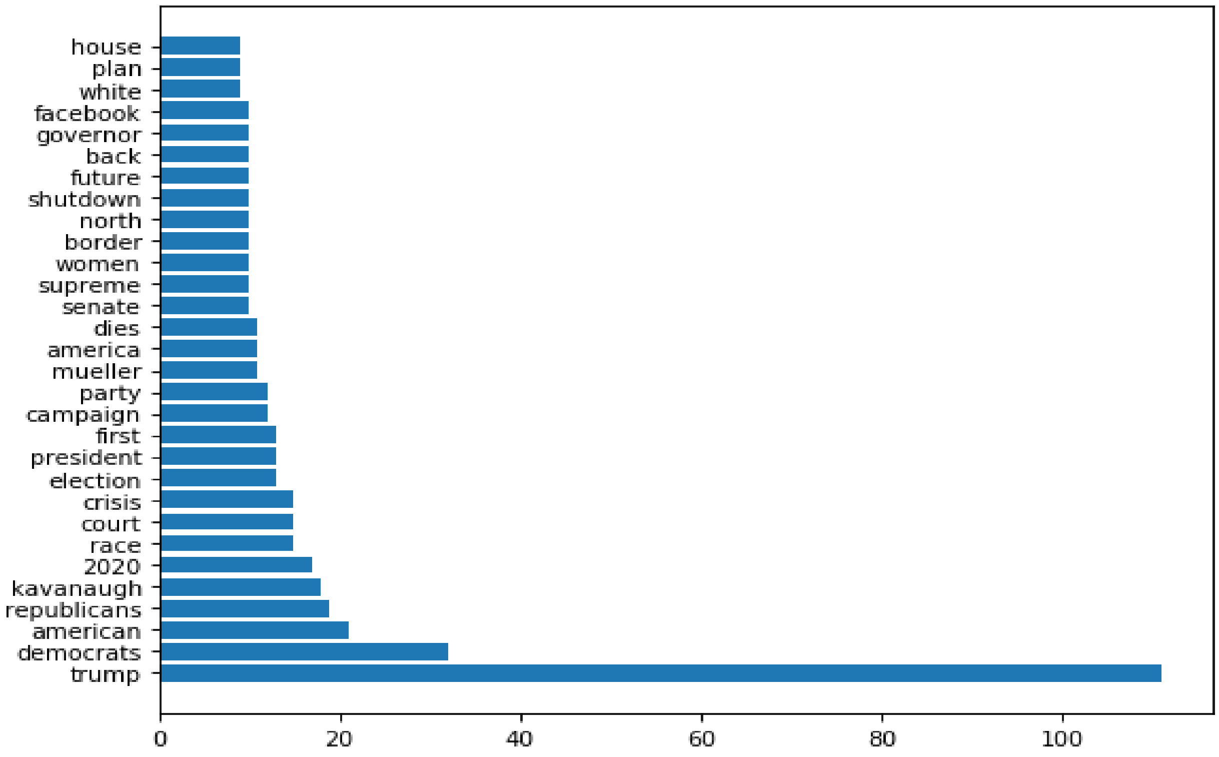
These word clouds demonstrate that the fake and real news contain similar words and helps the model to reduce bias. Dissimilar word having high frequency would introduce bias as the model will classify the articles based only on these words. Having similar words with high frequency lets the model to classify real and fake news based on the context of words.

These word clouds verify that the preprocessing of words, removal of strings was a right decision in moving forward for model evaluation.

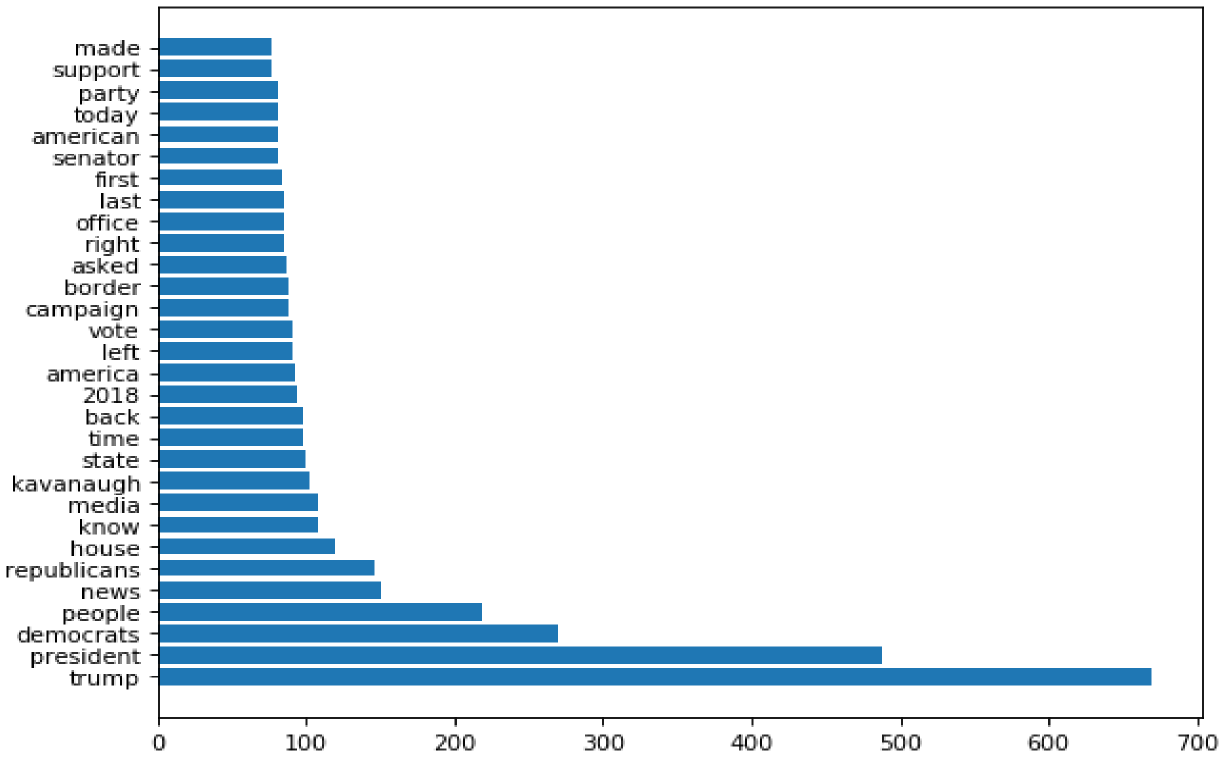
Word Frequency Distribution



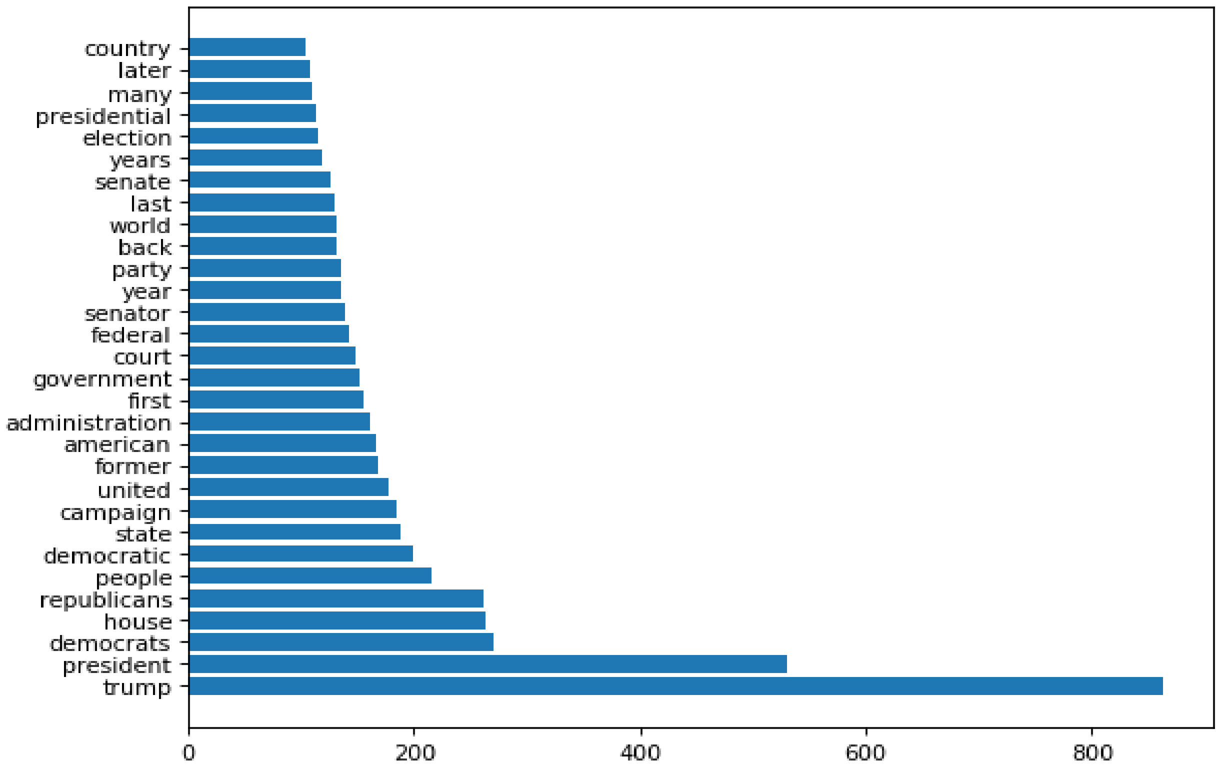
Top 30 words in fake news headlines



Top 30 words in real news headlines



Top 30 words in fake news content



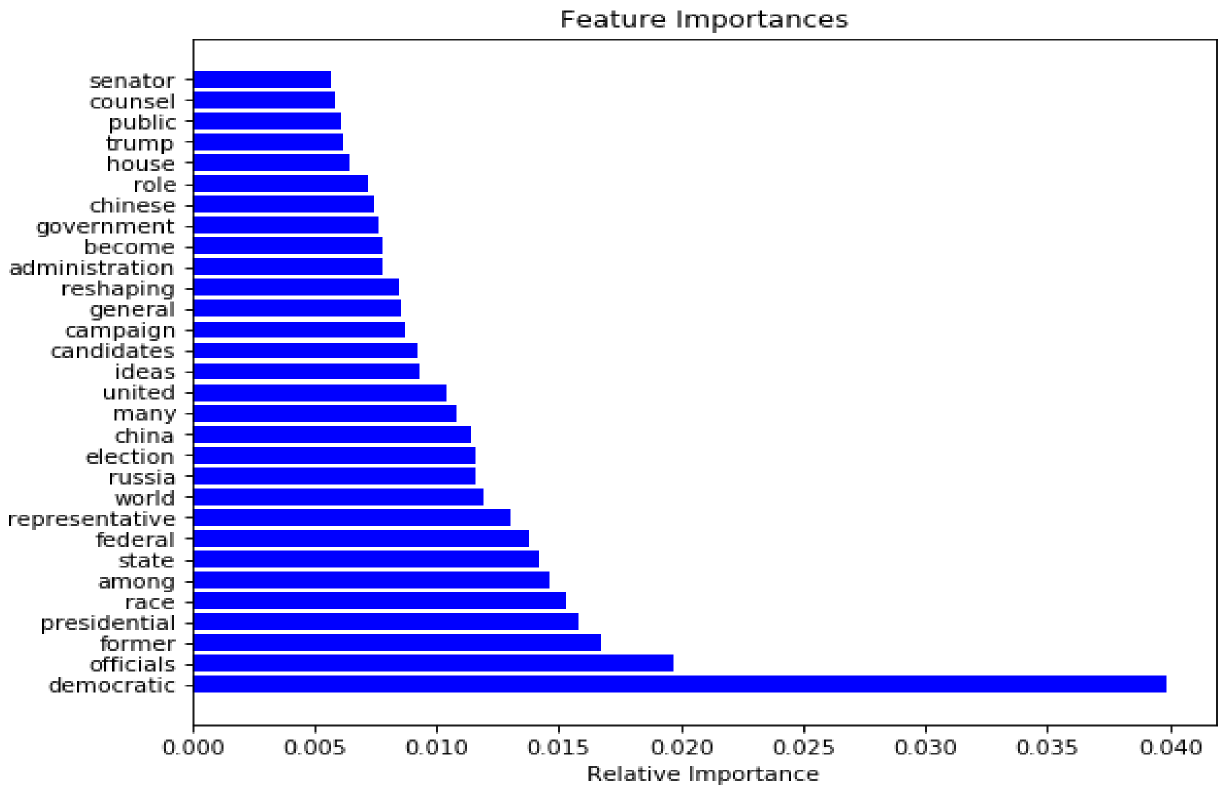
Fake Top 30 words in real news content

These graphs show the words on Y-axis against its word count in documents on X-axis.

The top 30 words in the real and fake news headlines and content are similar. These words are "trump"," president"," democrats"," people"," republicans". This explains that the fake news and real news have same context and the authors use minor dissimilar words to actually make the fake news sound like real. This reduces the bias even further, because in the same context the dissimilar words will help in classifying the real and fake news.

The word count of these similar words in different but they are in proportion for all the words. This approximately constant ratio reduces the bias as based the higher count only of certain the model wont classifies it.

Random Forest Feature Importance



Top 30 features using random forest importance- content dataset

The top 30 features from random forest feature importance for content dataset are displayed in the above figure. Random forests are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. They also provide two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy. [9]

Random forests are a popular method for feature ranking, since they are so easy to apply in general, they require very little feature engineering and parameter tuning and mean decrease impurity is exposed in most random forest libraries. With correlated features, strong features can end up with low scores and the method can be biased towards variables with many categories. [9]

This step helps to identify the top important features for the model. As the context of both the news are same, such features maintain the correlations and will only classify the least important ones.

Models

Refer table 2 for parameters description.

We applied 4 different models for both the datasets: Support Vector Machine (SVM), Multinomial Naïve Bayes (MNB), Random Forest and Convolutional Neural Network (CNN).

Table 2: Parameters

Parameter	Class	Description
char_wb	TF-IDF	Creates character n-grams only from text inside word boundaries; n-grams at the edges of words are padded with space
min_df		When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold
ngram_range		The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that $min\_n \le n \le max\_n$ will be used
max_features		Build a vocabulary that only consider the top max_features ordered by term frequency across the corpus
n_estimators	Random Forest	The number of trees in the forest
kernel	SVM	The type of the kernel to pre-compute the kernel matrix
alpha	MNB	Smoothing parameter
vocab_size	CNN	Similar to max_features, it considers the top words
max_sequence_length		Total limit of the size of the document. Words with lesser size are padded and greater size are cut
embedding_dim		The size of the embedding vector
filters		Number of filters
kernel_size		Window size of each filter
batch_size		Size of an individual batch to train one at a time
patience		The number of epochs to wait till there is no significant reduction in the loss

As we noted in our EDA part, our content dataset was little biased. And we drew conclusion to use top 500 words as features to reduce this bias. Hence, for content dataset we used max-features=500 for TF-IDFs and vocab_size=500 for CNN. For headlines dataset, we used GridSearch to find best parameters for TF-IDF. Models with their corresponding features and parameters are mentioned in Table 3.

As we can see model comparison in table 3, SVM performed the best for the headlines dataset and Random Forest performed the best for the content dataset.

For features of CNN, we utilized pre-trained GloVe to create word embeddings. CNN couldn't perform well for either dataset. We believe this happened due to smaller dataset size. As we know, neural networks require larger datasets.

Table 3: Models comparison

Model	Dataset	Features	Params	F1_macro(%)
SVM	Headlines	TF-IDF	Min_df = 3, ngram_range=(1,3), analyzer='char_wb'	86.34
	Content		Max features=500	89.20
MNB	Headlines		Alpha=0.5, ngram_range = (1,4)	73.65
	Content		Max features=500	85.58
Random Forest	Headlines		Min_df = 5, ngram_range=(1,3), analyzer='char_wb'	85.58
	Content		Max features=500	83.09
CNN	Headline	Word embeddings using pre-trained GloVe.6B.100D	Did not work	-
	Content		Vocab_size=500, Max_sequence_length =700 Embedding_dims=100 Filters=128 Kernel_size=7 Batch_size=50 Early stopping on validation loss with patience=7	79.50

Findings

We performed experiments not just using our scrapped dataset but using some already available datasets as well. After doing all the experiments, we couldn't make generalize model for fake news detection. We understood that fake news detection solely based on machine learning or natural language processing methods is not robust at all. Because, any model, we apply, will only learn the underlying patterns present in given dataset. So, if we feed new content having different pattern, our model will perform poorly.

Future work

We believe that fake news documents will have exaggerated tone/emotion. Hence, we have thought to extract tone/emotion from the given text document and utilize it as new feature. As, our dataset was biased for words, we can try different fake or real news sources and combine them in single dataset. Also, we only scrapped for 'politics' category. So, as a future work more categories can be included, and models can be more generalized.

As a research work, we propose that if somehow, we can include some sort of evidence checking for facts among fake and real news, then models could be more robust. For automated evidence checking, current natural language processing might not be adequate and we might need more advanced text comparison techniques.

References

1. Niall J. Conroy, Victoria L. Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology (2015), 1-4.
2. Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A Survey on Natural Language Processing for Fake News Detection. CoRR abs/1811.00770 (2018).
3. Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic Detection of Fake News. COLING (2018).
4. Victoria L. Rubin, Niall J. Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. Proceedings of NAACL-HLT (2016), 7–17.
5. Kai Shu, Suhang Wang, and Huan Liu. 2017. Exploiting Tri-Relationship for Fake News Detection. arXiv preprint arXiv:1712.07709 (2017).
6. Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. 2018. TI-CNN: Convolutional Neural Networks for Fake News Detection. CoRR abs/1806.00749 (2018).
7. https://en.wikipedia.org/wiki/Wikipedia:Zimdars%27_fake_news_list
8. https://en.wikipedia.org/wiki/News_media_in_the_United_States
9. https://blog.datadive.net/selecting-good-features-part-iii-random-forests/