Alternus Vera

Vidhi Shah
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
vidhirajesh.shah@sjsu.edu

Sithara Krishna Murthy
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
sithara.krishnamurthy@sjsu.edu

Pragya Gautam
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
pragya.gautam@sjsu.edu

Reetika Goel
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
reetika.goel@sjsu.edu

***Abstract*—This paper represents a novel approach to detect Fake News using NLP Distillation process. The paper contains various techniques used to identify the factors influencing fake news detection and the process of classifying the fakeness factor by using NLP techniques. 4 different factors analyzed using set of supervised classification models and vectorization are: Stance Detection, Reliable Source, Malicious Account and Writing Style. These factors are then weighted based on their accuracy against classifying a document and a polynomial equation made of vectors is defined. The polynomial equation proposed in this paper is:**

(0.24*StanceDetection) + (0.27*Reliable Source) + (0.26*Malicious Account) + (0.23*Writing Style)

I. Introduction

A. NLP

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding.

While natural language processing isn't a new science, the technology is rapidly advancing thanks to an increased interest in human-to-machine communications, plus an availability of big data, powerful computing and enhanced algorithms.

As a human, you may speak and write in English, Spanish or Chinese. But a computer's native language – known as machine code or machine language – is largely incomprehensible to most people. At your device's lowest levels, communication occurs not with words but through millions of zeros and ones that produce logical actions.

Why is NLP important?

Large volumes of textual data

Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.

Today's machines can analyze more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that's generated every day, from medical records to social media, automation will be critical to fully analyze text and speech data efficiently.

Structuring a highly unstructured data source

Human language is astoundingly complex and diverse. We express ourselves in infinite ways, both verbally and in writing. Not only are there hundreds of languages and dialects, but within each language is a unique set of grammar and syntax rules, terms and slang. When we write, we often misspell or abbreviate words, or omit punctuation. When we speak, we have regional accents, and we mumble, stutter and borrow terms from other languages.

While supervised and unsupervised learning, and specifically deep learning, are now widely used for modeling human language, there's also a need for syntactic and semantic understanding and domain expertise that are not necessarily present in these machine learning approaches. NLP is important because it helps resolve ambiguity in language and adds useful numeric structure to the data for many downstream applications, such as speech recognition or text analytics.

Natural language processing includes many different techniques for interpreting human language, ranging from statistical and machine learning methods to rules-based and algorithmic approaches. We need a broad array of approaches because the text- and voice-based data varies widely, as do the practical applications.

Basic NLP tasks include tokenization and parsing, lemmatization/stemming, part-of-speech tagging, language detection and identification of semantic relationships. If you ever diagrammed sentences in grade school, you've done these tasks manually before.

In general terms, NLP tasks break down language into shorter, elemental pieces, try to understand relationships between the pieces and explore how the pieces work together to create meaning.

These underlying tasks are often used in higher-level NLP capabilities, such as:

1. **Content categorization.** A linguistic-based document summary, including search and indexing, content alerts and duplication detection.

2. **Topic discovery and modeling.** Accurately capture the meaning and themes in text collections, and apply advanced analytics to text, like optimization and forecasting.

3. **Contextual extraction.** Automatically pull structured information from text-based sources.

4. **Sentiment analysis.** Identifying the mood or subjective opinions within large amounts of text, including average sentiment and opinion mining.

5. **Speech-to-text and text-to-speech conversion**. Transforming voice commands into written text, and vice versa.

6. **Document summarization.** Automatically generating synopses of large bodies of text.

   Machine translation. Automatic translation of text or speech from one language to another.

In all these cases, the overarching goal is to take raw language input and use linguistics and algorithms to transform or enrich the text in such a way that it delivers greater value.

II. Alternus Vera Features

A. Malicious Account

Over the last few years, online social networks (OSNs), such as Facebook, Twitter and Tuenti, have experienced exponential growth in both profile registrations and social interactions. These networks allow people to share different information ranging from news, photos, videos, feelings, personal information or research activities. The rapid growth of OSNs has triggered a dramatic rise in malicious activities including spamming, fake accounts creation, phishing, and malware distribution. However, developing an efficient detection system that can identify malicious accounts, as well as their suspicious behaviors on the social networks, has been quite challenging. Researchers have proposed a number of features and methods to detect malicious accounts.

In this paper we will explore how to detect if an account is a Twitter bot or not. A Twitter bot is a type of bot software that controls a Twitter account via the Twitter API. The bot software may autonomously perform actions such as tweeting, re-tweeting, liking, following, unfollowing, or direct messaging other accounts. The automation of Twitter accounts is governed by a set of automation rules that outline proper and improper uses of automation.Proper usage includes broadcasting helpful information, automatically generating interesting or creative content, and automatically replying to users via direct message. Improper usage includes circumventing API rate limits, violating user privacy, or spamming.

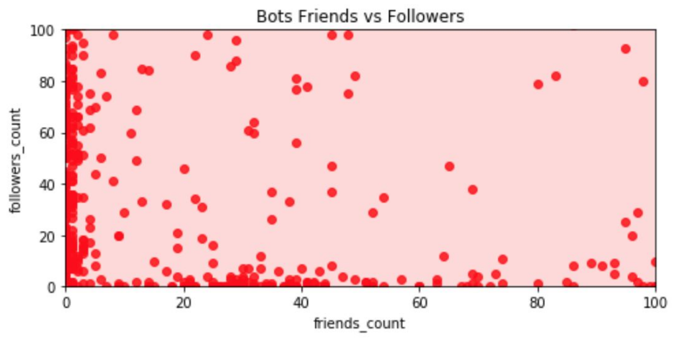
Political
A subset of Twitter bots programmed to complete social tasks played an important role in the United States 2016 Presidential Election. Researchers estimated that pro-Trump bots generated four tweets for every pro-Clinton automated account and out-tweeted pro-Clinton bots 7:1 on relevant hashtags during the final debate. Deceiving Twitter bots fooled candidates and campaign staffers into retweeting misappropriated quotes and accounts affiliated with incendiary ideals. Concerns about political Twitter bots include the promulgation of malicious content, increased polarization, and the spreading of fake news.

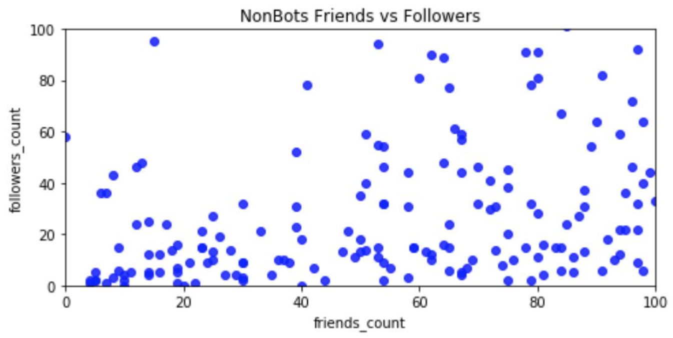
Implementation
The popularity of Twitter has attracted spammers to disseminate large amount of spam messages. Preliminary studies had shown that most spam messages were produced automatically by bot. Therefore bot spammer detection can reduce the number of spam messages in Twitter significantly..
In this project I have tried to identify if the Twitter account is malicious or real. The two categories that have been considered for the result bins are bots or non bots. Here, Bots

are nothing but a program used to make automated posts or follow users.

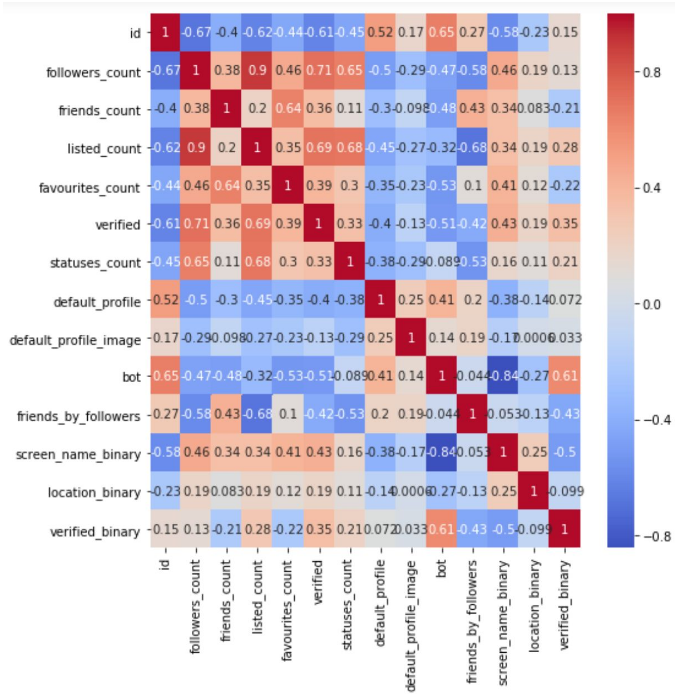
Dataset(s), Scraping, Enrichment:
Dataset was taken from Kaggle (https://www.kaggle.com/charvijain27/detecting-twitter-bot-data) The dataset consists of two CSV files - test_data_4_students.csv and training_data_2_csv_UTF.csv. The columns

The columns used are id, id_str, screen_name, location, description, url, followers_count, friends_count, listed_count, created_at, favorites_count, verified, statuses_count, lang, status, default_profile, default_profile_image, has_extended_profile, name, and bot. These are all features within a Twitter account. By evaluating these features we will be able to judge if the account belongs to a person or a bot.





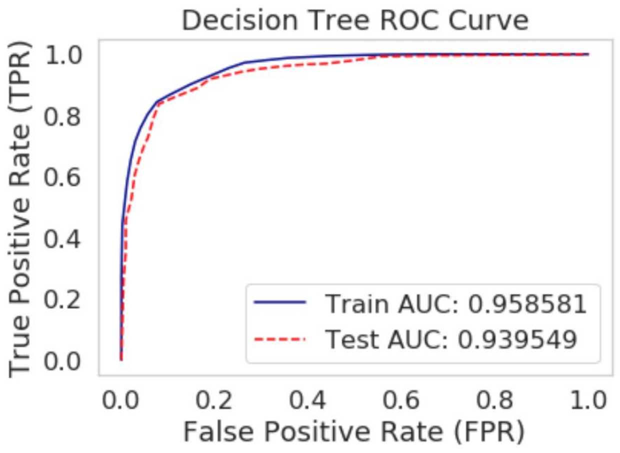
What did you try
I have implemented four different algorithms on the Detecting Twitter bot Dataset from Kaggle. The four algorithms are Decision Tree, Random Forest, Multinomial Naive Bayes and Classifier using bag of words. The first step was to use Spearman correlation to visualize if there were any correlations between the features.

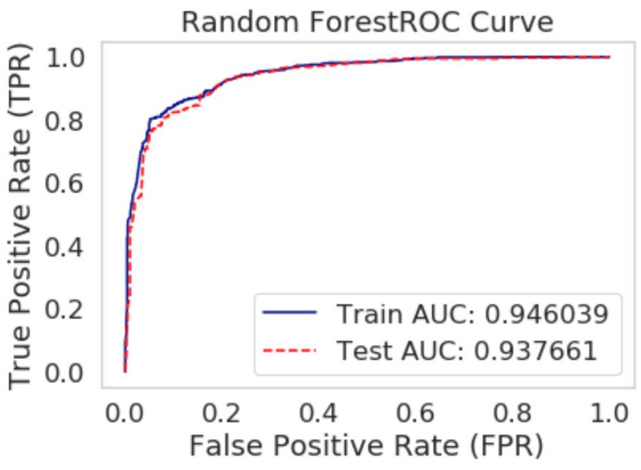


Result:

- There is no correlation between id, default_profile, default_profile_image and target variable.
- There is strong correlation between followers_count, friends_count, listed_count, favorites_count, statuses_count, verified and target variable.
- We cannot perform correlation for categorical attributes. So we will take screen_name, name, description, status into feature engineering. While use verified, listed_count for feature extraction.

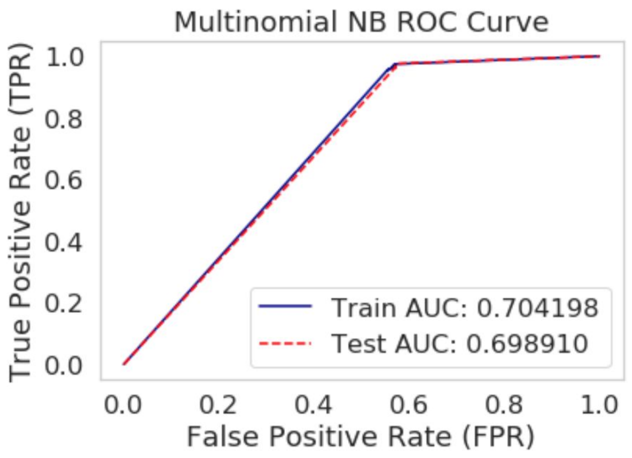
What worked
Decision Tree, Random Forest gave high accuracy however after implementing specific functions relevant to the twitter account details the Classifier using bag of words showed the best accuracy results for both test and train accuracy.



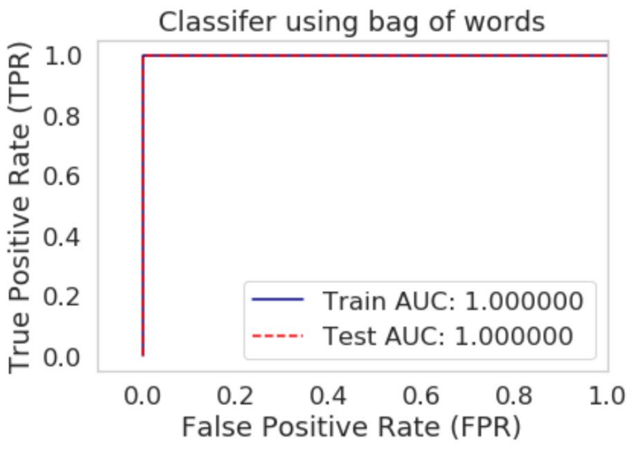


What did not work

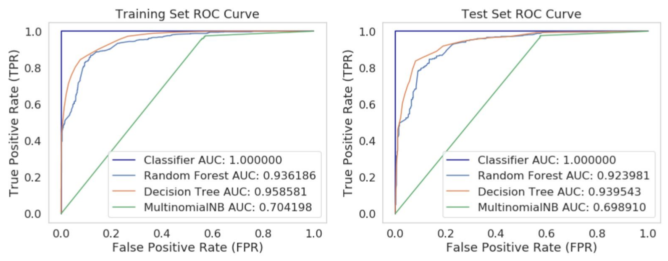
The Multinomial Naive Bayes gave the lowest accuracy for both the Training and Test accuracy.



I also performed feature engineering using bag of words to more accurately identify if the account is a bot or non-bot. I have checked twitter account specific features, such as checking if the name or screen name contains bot, checking if the user is verified, or checking if the description contains buzzfeed to help with the prediction model.



Summary



Multinomial Naive Bayes performs poorly and is not a good choice. It gives Training Accuracy: 70% and Test Accuracy: 69%

Decision Tree and Random Forest give very good performance and generalizes well. Decision Tree gives Training Accuracy: 95% and Test Accuracy: 93% Random Forest gives Training Accuracy: 94%and Test Accuracy: 93%

Classifier using the selected bag of words and uniquely defined functions to help predict if the account is bot or not gives Train Accuracy: 100% and Test Accuracy: 100%

B. Stance Detection

The idea of fake news is often referred to as click-bait in social trends and is defined as a "made up story with an intention to deceive, geared towards getting clicks”. Some news articles have titles which grab a reader’s interest. Yet, the author only emphasizes a specific part of the article in the title. If the article itself does not focus on or give much truth to what the title had written, the news may be misleading.

The goal of this project is to use natural language processing techniques to automate stance detection, since it is not practical for humans to fact check every piece of information produced by the media.

Stance detection is a method used to determine the quality of a news article by taking into consideration what other organisations write about the same headline. A body of text is claimed to agree, disagree, discuss, or be unrelated to a headline, Fake News Challenge (2016) Stance detection is the method that will be used to determine the quality of a news source.

With this system, for a set of news headlines, statistics can be gathered with respect to the stances. With these statistics, a user can come to their own conclusion of whether a new organisation has reputable news sources. To achieve these stances, this system will train on the data supplied by the fake news challenge. This data will provide the stance along with the headline and body to allow the system to learn which word combinations lead to which stance. For testing, data will be provided without the stances.

Dataset(s), Scraping, Enrichment:
From the FakeChallenge.org a dataset has been provided which consists of a headline and a body of text. This body of text may be from a different article. Allowing bodies of text from different articles allows this system to take into account what the other organisations are saying about he same headline. The output of the system will be the stance of the body of text related to the title. 2 csvs that I have used to implement the system are:

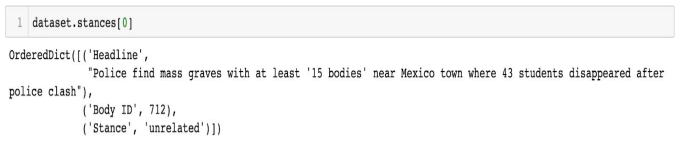
“Train_bodies.csv”: Contains Body ID and Body Text - no of records : 2532

“Train_stances.csv”: Contains Headline, Body ID and Stance - no of records : 49972

The system will support support the following stance types:

- Agrees
- Disagrees
- Discusses
- Unrelated

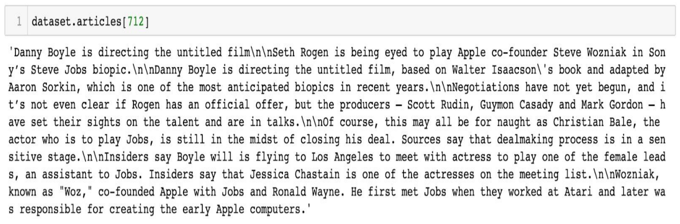
A sample example from the dataset:



```
dataset.stances[0]
```

```
OrderedDict([('Headline',
              "Police find mass graves with at least '15 bodies' near Mexico town where 43 students disappeared after
police clash"),
             ('Body ID', 712),
             ('Stance', 'unrelated')])
```

Above image is from the “train_stances” file that shows a headline along with its body ID. Actual body text associated with body ID = 712 is shown below. It is clearly seen that the two of them are totally unrelated and thus states the stance type for the same.



```
dataset.articles[712]
```

```
'Danny Boyle is directing the untitled film\n\nSeth Rogen is being eyed to play Apple co-founder Steve Wozniak in Son
y’s Steve Jobs biopic.\n\nDanny Boyle is directing the untitled film, based on Walter Isaacson\'s book and adapted by
Aaron Sorkin, which is one of the most anticipated biopics in recent years.\n\nNegotiations have not yet begun, and i
t’s not even clear if Rogen has an official offer, but the producers – Scott Rudin, Guymon Casady and Mark Gordon – h
ave set their sights on the talent and are in talks.\n\nOf course, this may all be for naught as Christian Bale, the
actor who is to play Jobs, is still in the midst of closing his deal. Sources say that dealmaking process is in a sen
sitive stage.\n\nInsiders say Boyle will is flying to Los Angeles to meet with actress to play one of the female lead
s, an assistant to Jobs. Insiders say that Jessica Chastain is one of the actresses on the meeting list.\n\nWozniak,
known as "Woz," co-founded Apple with Jobs and Ronald Wayne. He first met Jobs when they worked at Atari and later wa
s responsible for creating the early Apple computers.'
```

Implementation:
The brief summary on the steps followed are -

What did you try:

- Pre-processing: As part of pre-processing, I started reading both csv data files. Then from the dataset, randomly body ids were extracted and divided into training, dev and test ids so that it can be used further while training the model. After dividing, below are the sizes:

- Training size:	40106
- Dev size:	4835
- Test data:	5031

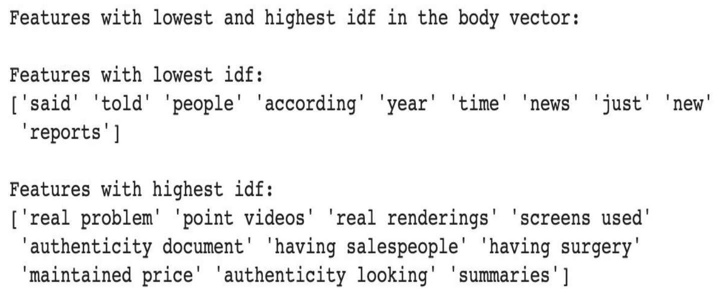
- Tokenization/Vectorization: The text must be parsed to remove words, called Tokenization. Then the words need to be encoded as integers or floating-point values for use as input to a machine learning algorithm, called feature extraction (or Vectorization)

- Bag-of-Words Model: A simple and effective model for thinking about text documents in machine learning is called the Bag-of-Words Model, or BoW. The model is simple in that it throws away all of the order information in the words and focuses on the occurrence of words in a document. This can be done by assigning each word a unique number. Then any document we see can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector could be filled with a count or frequency of each word in the encoded document. This is the bag of words model, where we are only concerned with encoding schemes that represent what words are present or the degree to which they are present in encoded documents without any information about order. There are many ways to extend this simple method, both by better clarifying what a "word" is and in defining what to encode about each word in the vector.

- N-grams (sets of consecutive words): With ngram_range = (1,2) it will take below features:
  Every feature:
  ['big', 'going', 'grass', 'great', 'having', 'park', 'player', 'time', 'unlike', 'women']
  For ngram_range = (1,2) means the algorithm will consider 2 words together while doing vectorization
  ['big', 'big player', 'going', 'grass', 'great', 'great time', 'having', 'having great', 'park', 'park grass', 'park time', 'player', 'player park', 'time', 'time park', 'unlike', 'unlike women', 'women', 'women big']

- CountVectorizer: CountVectorizer can lowercase letters, disregard punctuation and stopwords, but it can't LEMMATIZE or STEM

- TfidfVectorizer: The goal of using term frequency – inverse term frequency is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus. It learns vocabulary and idf, return term-document matrix. Convert a collection of raw documents to a matrix of TF-IDF features. Equivalent to CountVectorizer followed by TfidfTransformer. The token which appears maximum times, but it is also in all documents, has its idf the lowest. The tokens can have the most idf weight because they are the only tokens that appear in one document only. the more times a token appears in a document, the more weight it will have. However, the more documents the token appears in, it is 'penalized' and the weight is diminished. Some of the features with lowest and highest idf in the datasets are shown in below image:



```
Features with lowest and highest idf in the body vector:

Features with lowest idf:
['said' 'told' 'people' 'according' 'year' 'time' 'news' 'just' 'new'
 'reports']

Features with highest idf:
['real problem' 'point videos' 'real renderings' 'screens used'
 'authenticity document' 'having salespeople' 'having surgery'
 'maintained price' 'authenticity looking' 'summaries']
```

- Stemming: The idea of stemming is a sort of normalizing method. Many variations of words carry the same meaning, other than when tense is involved.

- Lemmatizing: A very similar operation to stemming is called lemmatizing. The major difference between these is, as you saw earlier, stemming can often create non-existent words, whereas lemmas are actual words. So, your root stem, meaning the word you end up with, is not something you can just look up in a dictionary, but you can look up a lemma.

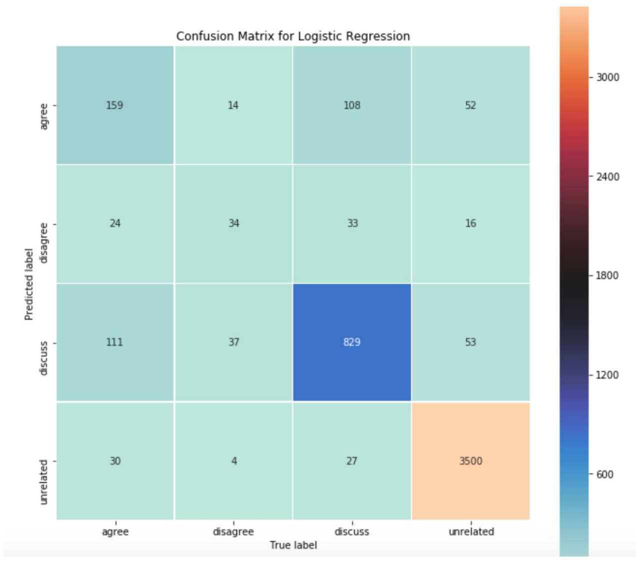
What Worked:

Doing above pre-processing helped me identify latent variable described below:

- Latent Variable/Rank: I calculated the intersection of body and headline alongwith union of body and headline. Then took the length of common words b/w body and headline divided by length of all the words of body & headline and labeled it as word overlap feature. This is the process of distillation where i have identified the rank/latent variable which is used further for model training and prediction.

- Model Algorithms: I ran four algorithms to train the model and predict the stance type. Those were: Logistic regression, Multinomial Naive Bayes, Random Forest and XGBoost Classifier. Logistic regression and Random Forest did show good results on the dataset. Figure-1 shows the true vs predicted values for these 2 algorithms. Figure-2 shows the summary of accuracies for all the algorithms combined. Figure-3 shows the Learning Curves for baseline system and improved system for Random Forest & Logistic Regression algos. Baseline system is basically a system with TF-IDF vectors as its features. Improved system is run with TF-IDF Vectors, Cosine similarity and Word Overlap and we can clearly see how drastically the accuracy scores have improved.

Figure-4 shows ROC Curve of Random forest model for all the 4 stance detection types outlined with different colors.



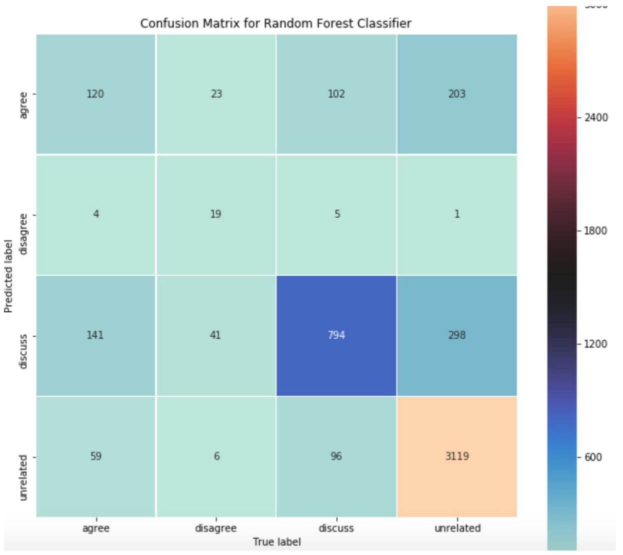


Figure-1: Confusion Matrix

- Model Predictions With Liar Liar Dataset: Even thought this dataset is different from the liar-liar dataset, what they have in common is the body and the headline. In liar-liar dataset it is named as text aka body and headline as headline. Hence, the inputs to predict the stance and decide whether the news is fake or not completely depends on 2 input parameters: Body/Text and Headline. The implementation for the model is done in such a way that it is able to predict 80% accurately for any body-headline combination whether the news is fake or not.

What Did Not Work:

- Model Algorithms: From the below results of all the algorithms it is clearly seen that Multinomial Naive Bayes and XGBoost are not a good fit for this particular dataset. Their accuracy went below 50% and hence if applied, they will fail to predict the correct results more accurately.

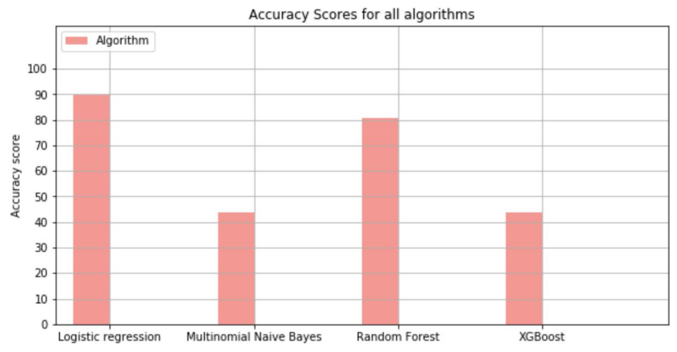


Figure-2: Accuracy Comparison Bar Plot

- TFIDFVectorizer: Although tfidfvectorizer is a great way for text data to convert into matrix form, it is very time consuming. Hence, for any new text data that we have to make the predictions, it has to first pass through the vectorizer which takes a lot time for processing.

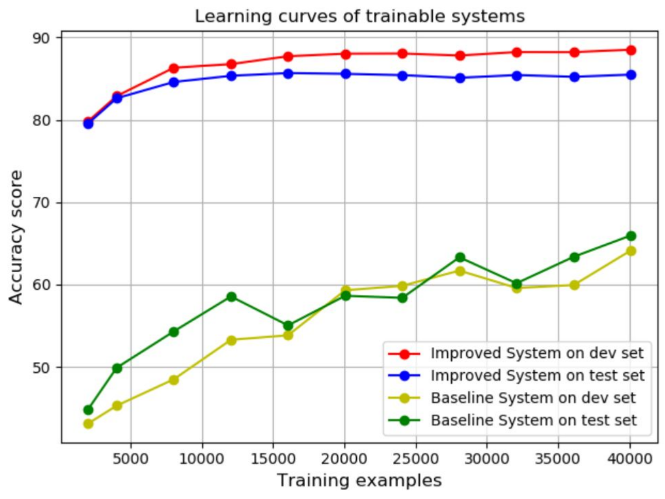


Figure-3: Learning Curves (Random Forest & Logistic Regression)

What Alternatives Did You Try:

Initially, I created the notebook with just one classifier: Random Forest and one Features: TF-IDF Vectors but the accuracy was not turning up good. Hence, as part of improved system, I added multiple Classifiers: Logistic Regression and multinomial naive bayes along with Features: TF-IDF Vectors, Cosine similarity and Word Overlap

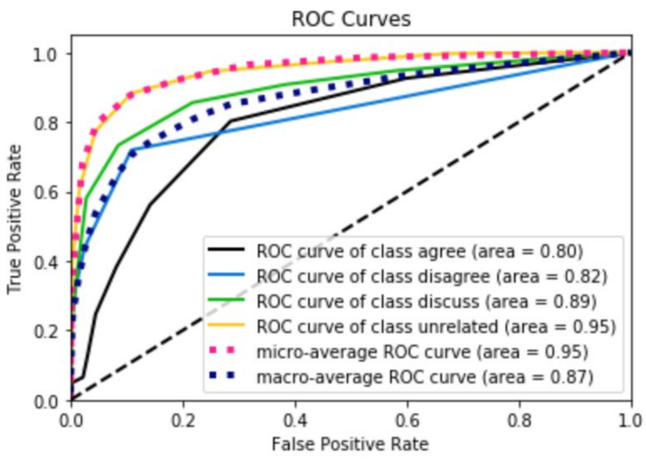


Figure-4: ROC Curve Random Forest for all 4 classes

What did you research, what references (eg code) did you study or leverage (code):

As my dataset is from the website FakeChallenge.org, I did studied the architecture given by them along with solutions, suggestions and examples provided over there. I also did go through various papers on stance detection for fake news. References studied are mentioned in the last section of the paper.

C. Reliable Source

Fake news is news articles that are intentionally and verifiably false and could mislead readers. The source of the news article plays a very important role in trusting and identifying if the news is real or fake. With the widespread availability of news online, it is very easy for people to misuse the internet and post misleading news. Thus it is very important to identify the source of the article to know if it is reliable or not. A reliable source is one that provides a thorough, well-reasoned theory, argument, discussion, etc. based on strong evidence.

There are different ways to tell if a website is credible. It can be challenging to determine whether a website we are using is credible, but here are a few things to look for:

* Author – Information on the internet with a listed author is one indication of a credible site. The fact that the author is willing to stand behind the information presented (and in some cases, include his or her contact information) is a good indication that the information is reliable.
* Date – The date of any research information is important, including information found on the Internet. By including a date, the website allows readers to make decisions about whether that information is recent enough for their purposes.
* Sources – Credible websites, like books and scholarly articles, should cite the source of the information presented.
* Domain – Some domains such as .com, .org, and .net can be purchased and used by any individual. However, the domain .edu is reserved for colleges and universities, while .gov denotes a government website. These two are usually credible sources for information (though occasionally a university will assign a .edu address to each of its students for personal use, in which case use caution when citing). Be careful with the domain .org, because .org is usually used by non-profit organizations which may have an agenda of persuasion rather than education.
* Site Design – This can be very subjective, but a well-designed site can be an indication of more reliable information. Good design helps make information more easily accessible.
* Writing Style – Poor spelling and grammar are an indication that the site may not be credible. In an effort to make the information presented easy to understand, credible sites watch writing style closely.

Dataset(s), Scraping, Enrichment:

There are different types of data sources for an article. To identify a reliable source, the dataset was taken from a kaggle competition(https://www.kaggle.com/c/fake-news/data). It consists of a train.csv and test.csv. The testing dataset has the same attributes as train.csv without the label. They have 4 columns ie. id, title, author and text. The id is the unique id for a news article, title is the title of a news article, author is author of the news article. The author here can be a person/s, an organization , a website or a group. Text is the text of the article; could be incomplete. The train.csv has an additional class/label column which states if the data in that row is reliable or unreliable. An article which is reliable is labelled as 0 and one that is unreliable is labelled as 1. The train.csv consists of 20800 records and test.csv consists of 5200 records.

The Liar-liar dataset has columns statement, subject and speaker which are the same as author, title and text. These are the same columns which we have considered to identify a reliable source.

Implementation Details:

What did you try:

Based on the data that are available in the dataset, it can be found that the title, author and text are the features that are available and can be used in determining if the source ie. the author is reliable or not. I have tried to check if the relationship of title and body composed by the author will help in understanding how reliable the author is. For this I created a new feature in the dataset by combining the Title, body and the Author. This is a combined feature with complete information for the article. I have concatenated the title with the author and the text to form a single column. Next I performed Count Vectorization and Tf-idf on the new feature and used that data to train my models.

- Visualization:

Fig 5 shows the visualization of the Authors column that is available in the train.csv. Matlibplot library is used for visualization. Word cloud is used to understand the importance of various words in the textual data.

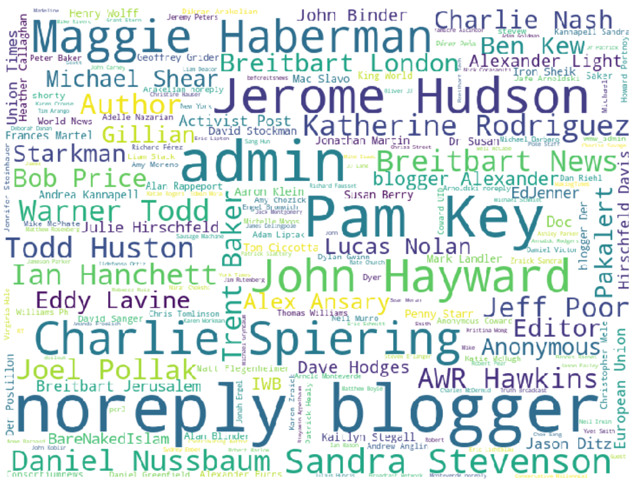


Figure 5 : Word Cloud of the Authors

- Countvectorizer

Word embeddings are mechanism by which text data can be converted to vector representation. Count Vectorization is a technique where word frequency is used to convert text to vector. The scikit learn CountVectorizer is used to convert the combined text to vector. The library takes a vocabulary list against which the frequency of occurrence is performed to convert text to vector. Countvectorizer counts the number of times a token shows up in the document and uses this value as its weight.

- NGrams - TF-IDF Vectorizer

TF-IDF stands for term frequency-inverse document frequency. TF-IDF weight is a measure used to evaluate how important a word is to a document in a corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

The training data is this new column along with the available target classes in the dataset. Using this information, I trained the models using Extra Trees Classifier, AdaBoost Classifier, Random Forest Classifier, Naive bayes and Logistic regression algorithms. Then evaluated the model by plotting the confusion matrix and the ROC Curve.

What worked:

Based the above mentioned processing and implementation steps, I was able to train the models.

What did not work:

The models were giving good results but I was looking to improve their accuracy more and started working towards improving them. For this I planned to pre-processed the data before training the models.

What alternatives did you try:

To improve my models I added the additional steps of cleaning and preparing the data before performing Count Vectorization and Tfidf.

The following cleaning was performed on the data using the different scikit learn libraries :

- Remove Special Characters and Punctuations - This was done using regular expression.
- Lower case the text
- Tokenization - Tokenization is the process of splitting a single sentence or paragraph into a list of basic units of words.
- Remove Stop Words - Stop words are words like this, that, and, etc. For removing this I used the NLTK library for English corpus. This corpus provides a list of stop words.
- Lemmatization and Stemming - Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.

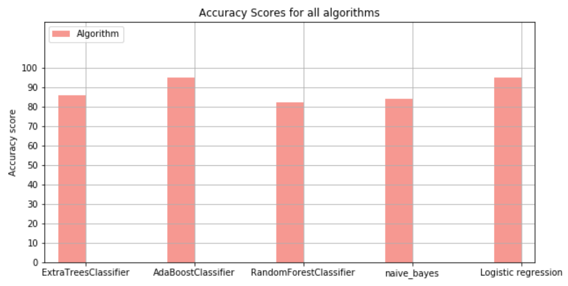


Figure 6 : Accuracy Comparison Bar Plot

Summary:
On doing the above preprocessing, it was found that the data took some time to process but then, on training the models, the accuracy of some of them improved. It is found that Logistic Regression and AdaBoost classifiers are good performers with almost 95% accuracy, to identify reliability of the source in fake news detection. Fig 7 shows the comparison of accuracies of the different classifiers.

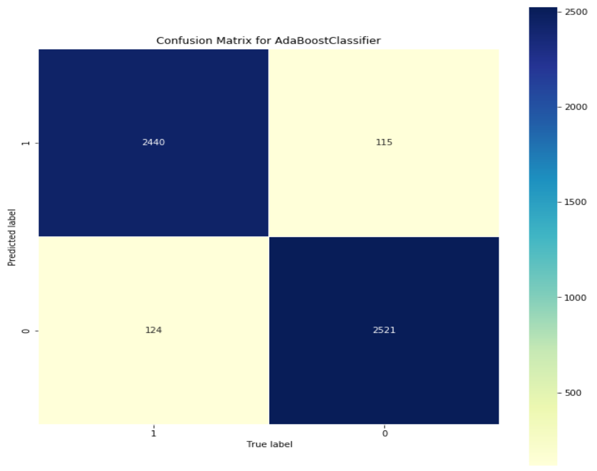


Figure 7 : Confusion Matrix for AdaBoost Classifier

What did you research, what references (eg code) did you study or leverage (code):
Reliable source is a very important feature in identifying the fakeness of the news article. To understand this feature I referenced several papers and looked up to find a labelled dataset which was available on kaggle and trained the models.For further improvements I would like to enhance the dataset with more features and train the model.

D. Writing Style

Fake news is intentionally written to mislead readers, which makes it non-trivial to detect simply based on news content. The content of fake news is rather diverse in terms of topics, styles and media platforms, and fake news attempts to distort truth with diverse linguistic styles while simultaneously mocking true news. For example, fake news may cite true evidence within the incorrect context to support a non-factual claim.

Most liars use their language strategically to avoid being caught, in spite of their attempt to control what they are saying, language "leakage" occurs with certain verbal aspects that are hard to monitor manually such as frequencies and patterns of pronoun, conjunction, and negative emotion word usage. For detecting these linguistic patterns, it uses shallow and deep syntax analysis which use POS(parts-of-speech) tags.

There are two methods for Fake News Detection. Linguistic features capture the different writing styles and sensational headlines to detect fake news. Linguistic based features are extracted from the text content in terms of document organizations from different levels, such as characters, words, sentences, and documents. In order to capture the different aspects of fake news and real news, existing work utilized both common linguistic features and domain-specific linguistic features. Common linguistic features are often used to represent documents for various tasks in natural language processing. Typical common linguistic features are:

(i) lexical features, including character level and word-level features, such as total words, characters per word, frequency of large words, and unique words;
(ii) syntactic features, including sentence-level features, such as frequency of function words and phrases or punctuation and parts-of-speech (POS) tagging.

Domain-specific linguistic features, which are specifically aligned to news domain, such as quoted words, external links, number of graphs, and the average length of graphs, etc. Moreover, other features can be specifically designed to capture the deceptive cues in writing styles to differentiate fake news, such as lying detection features.

Dataset(s), Scraping, Enrichment:

We have used 2 datasets for this feature of fake news detection and we will use the required attributes of these datasets to train our model.
(i) Real News Dataset:
https://www.kaggle.com/snapcrack/all-the-news/home

(ii) Fake News Dataset:
https://www.kaggle.com/mrisdal/fake-news

The Real News Dataset consists of 3 csv files. After concatenating these files, the dataset have about 15712 data points and 8 columns(id, author, title, text, date, month, url, News(i.e real or fake)) after preprocessing the data, (i.e) removing duplicates and removing data points which are NULL etc. We store this preprocess data into a new file real_news.csv

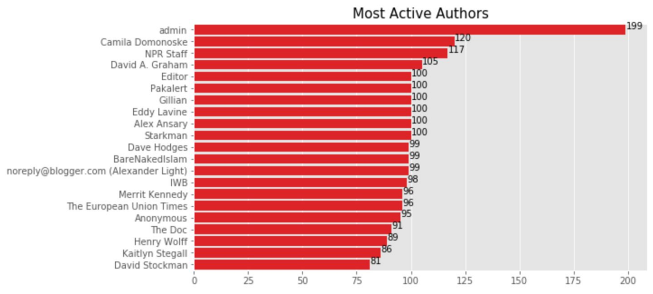
The Fake dataset have about 10900 data points and 8 columns(id, author, title, text, date, month, url, News(i.e real or fake)) after preprocessing the data, (i.e) removing duplicates and removing data points which are NULL etc. Here,we only take those columns which are common in both the datasets. We store this preprocess data into a new file fake_news.csv
Then we concatenate both the datasets into a new file named real_fake_news.csv for further analysis and implementation.

Implementation:

What did I try:
The brief summary on the steps followed are -

1.) *Pre-processing*

Data preprocessing is done to convert the raw data into a required format. Data pre- processing can be done by various methods like data cleaning, data reduction, data integration etc. In this project, the datasets are collected from different resources which have different formats and attributes. Hence, the data can be duplicate and they may contain some attributes which are not useful. So, we convert the data into our required format with required attributes which are used to train our model.



2.) *Data Analysis*

After preprocessing, it is necessary to visualize the dataset to identify the maximum used word or the word that has high frequency rate. This helps in deciding the vector size with less data loss. Word-cloud is used to visualize the dataset. It provides more information and insights of texts by analyzing correlations and similarities between words rather than analyzing texts only by the frequency of words appeared.

3.) *Generating News Feature Vector*

The most important part of detecting if a given news is fake or not is to convert the news article into a news vector which contains the important features which are used to determine the nature of the news. There are several ways to generate feature vector . We tried different approaches for the same to determine which method gives the best accuracy. Some of the methods are:

- *Bag of Words*:

Bag of words is a way of representing text in a format which can be easily processed by the machine learning algorithms. BoW is one of the ways of extracting features from text.

- *TF-IDF*:

TF-IDF stands for term frequency-inverse document frequency.TF-IDF is a method used to represent text in a format which can be easily processed by the machine learning algorithms. It is a numerical statistic that shows how important a word is to a document in a word corpus. The importance of a word is proportional to the number of times the word appears in the document but inversely proportional to the the number of times the word appears in the corpus.

Syntactical Analysis:

Syntactic analysis is the process of analysing a string of symbols in natural language, conforming to the rules of a formal grammar. We generated POS(part-of-speech) tags using the Spacy library. Our POS features will be encoded as tf-idf values for each for these tags. We strengthen POS features with unigram/bigram features.

- *Semantic Analysis*:

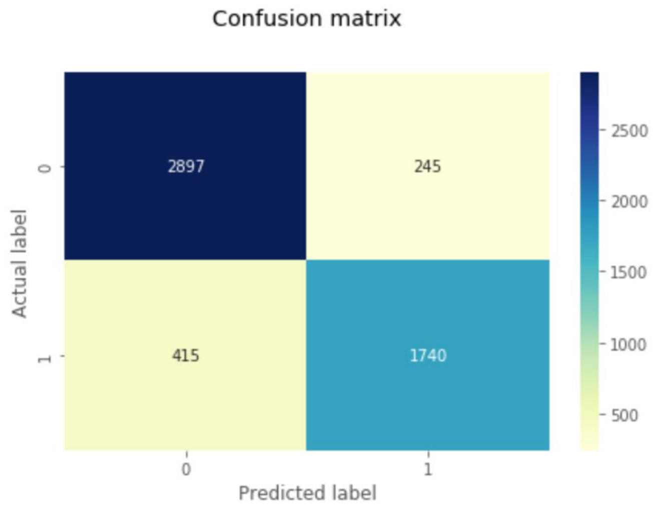
A widely used open-source resource for incorporating semantic information is Empath. Empath is a lexicon of words grouped into semantic categories relevant to psychological processes. Empath has 194 semantic categories, some of these semantic classes are emotional tone(positive or negative), anger, nervousness.

- *Classification*

After generating the news feature vector, now we classify the vector to whether it is fake or real. We aim to use the following classification algorithms for the purpose of classification:

- *Naive Bayes*:

Naive bayes is a supervised learning algorithm which is used for classification. It is based on bayes theorem assuming that features are independent of each other. It calculates the probability of every class, the class with maximum probability is chosen as the output.



- *SVM Classifier*:

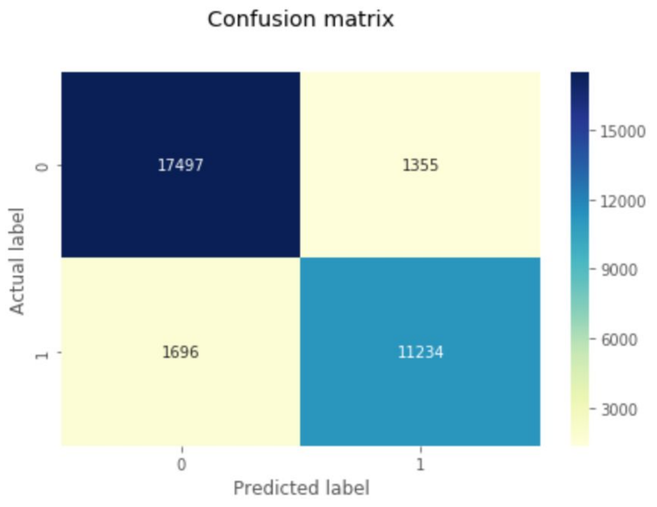
A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

- *Logistic Regression*:

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

- *Random Forest*:

Random Forests is a bagging type of ensemble model in which the base model for bagging is decision tree. In addition to bagging (i.e taking random samples of total data and train those samples independently and then take the majority voting of numerous samples of the total dataset to find the output of the total dataset), there is feature bagging (i.e column sampling in which not all the columns/features are taken into consideration while training but rather random samples of features are considered while training different samples.



What worked:

Data preprocessing and cleaning using NLTK inbuilt libraries worked well. I save the cleaned text as a column to utilize that for future analysis. There were not columns related to the syntactic and semantic factors of the document. So, I created these columns before vectorizing and performing classification models. Normalization using the count vectorizer worked very well for me. Then I perform various classification models on it and got the highest accuracy of 93% using Logistic and SGD model.

What did not work:

(i) When exploring for a writing style dataset, we encountered a lot of datasets. But those datasets were not correctly reflecting the writing style that we see in news articles. When exploring other datasets, the context of the writing style in news was totally unrelated to politics news, which lead to bad vectorization.
(ii) Naive Bayes classifier didn't work well in my case, as it gave the lowest accuracy i.e not a good predictor whether the news is real or fake.

What alternatives did I try:

Earlier, I tried fake news detection on just a single statement or line. But it is difficult to detect linguistic cues in single(or few) statement news. Therefore, now I tried to focus only on the daily news articles which have on average around 1000 words.

What did I research, what references (eg code) did I study or leverage (code):

For this project, I have to go through various papers to know how to implement syntactic and semantic analysis to detect the writing style of the person, How to apply grammar rules while analysing the text etc. During this process, I found these two papers very helpful:

- https://arxiv.org/pdf/1708.01967.pdf
- Paper: Syntactic Stylometry for Deception Detection

III. Conclusion & Future Work

What we did as a team?

Classifying "fake news" provides a novel challenge to the data science community. In many machine learning projects, the distinction between the different classes you want to predict is clear, whereas it's a lot murkier in this case. The goal of this project has been to comprehensively and extensively review, summarize, compare and evaluate the our research on fake news, which includes 1) Stance Detection 2) Writing Style 3) Reliable Source and 4) Malicious Account.

As a team, we decided on the importance of the factors presented in this paper. We brainstormed on the general pre-processing techniques and tried to use common model algorithms to classify the news. We also came up with a polynomial equation based on the factors and the accuracy scores we received by classification. The polynomial equation is then used to build a model for fake news classification. The polynomial equation that we have used is (0.24*StanceDetection) + (0.27*Reliable Source) + (0.26*Malicious Account) + (0.23*Writing Style).

There are several interesting options for future work. One is to make use of other features available in the dataset like favorites, retweets and social network and learn features for the fake news detection. A more powerful deep-learning tool should be employed to combat fake news in a professional setting.

IV. References

1. "What Is Natural Language Processing?" SAS, www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html.
2. Adewole, Kayode Sakariyah, et al. "Malicious Accounts." Journal of Network and Computer Applications, Academic Press Ltd., dl.acm.org/citation.cfm?id=3030088.

Bessi, Alessandro; Ferrara, Emilio "Social bots distort the 2016 U.S. Presidential election online discussion"

4. Shao, Chengcheng; Giovanni Luca Ciampaglia; Onur Varol; Kaicheng Yang; Alessandro Flammini; Filippo Menczer (2018). "The spread of low-credibility content by social bots".
5. "As Twitter moves to purge fake accounts, conservatives say they are being targeted - The Boston Globe"
6. McGill, Andrew (2 June 2016). "Have Twitter Bots Infiltrated the 2016 Election?".
7. Gmyrianthous. "Gmyrianthous/Fakenewschallenge." GitHub, 14 May 2017, github.com/gmyrianthous/fakenewschallenge.
8. "Fake News Challenge Stage 1 (FNC-I): Stance Detection." Fake News Challenge, fakenewschallenge.org/.
9. Reiinakano. "Reiinakano/Scikit-Plot." GitHub, 19 Aug. 2018, github.com/reiinakano/scikit-plot.
10. "CountVectorizer, TfidfVectorizer, Predict Comments." Kaggle, www.kaggle.com/adamschroeder/countvectorizer-tfidfvectorizer-predict-comments.
11. "'Liar, Liar Pants on Fire': A New Benchmark Dataset for Fake News Detection." GroundAI, www.groundai.com/project/liar-liar-pants-on-fire-a-new-benchmark-dataset-for-fake-news-detection/.
12. "KDnuggets." KDnuggets Analytics Big Data Data Mining and Data Science, www.kdnuggets.com/2017/04/machine-learning-fake-news-accuracy.html.
13. "UKnowIT (Self Service)." How Can I Tell If a Website Is Credible?, uknowit.uwgb.edu/page.php?id=30276.
14. Laizheng. "Laizheng/ML1010_GROUP_PROJECT." GitHub, github.com/laizheng/ML1010_GROUP_PROJECT.
15. Lefed. "Lefed/nlp_trash_news_classification." GitHub, 6 Nov. 2017, github.com/lefed/nlp_trash_news_classification.
16. Saiakhilaloor. "Saiakhilaloor/Fake-News-Detection." GitHub, github.com/saiakhilaloor/fake-news-detection.