Alternus Vera

Amrutha Singh Balaji Singh
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Controversy
amrutha.singhbalajisingh@sjsu.edu

Anand Muralidhara
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Source Credibility
anand.muralidhara@sjsu.edu

Poorva Agarwal
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Context / Venue
poorva.agarwal@sjsu.edu

Snehal Yeole
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Credibility/Fact Checks
snehal.yeole@sjsu.edu

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the wide spread of "fake news", i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. The paper presents various techniques used to identify the factors influencing fake news detection and the process of classifying the fakeness factor by using NLP techniques. Various factors are analyzed using set of supervised classification models and vectorized with values based on the document feature.

Keywords—Distillation, Fake News, Controversy Score, LDA, Lexical Feature Credibility, Sentiment analysis, Context, Source Credibility, SenticNet5, NLTK, Tokenizer, Wordcloud, t-SNE, Doc2Vec, TF-IDF, Cosine Similarity, Stemming

I. INTRODUCTION

A. Source Credibility

The criticality of accessing news in order to distinguish between true news and fake news is enormous, and it goes without saying that it is an enormous task, as the reach for them is enormous, the results of which are nothing short of disturbing. With this project, we intend to work on a fake news set to analyze news, its source, the bearer of news, the credibility of news and other key factors that influence news and come up with a solution to distinguish between fake news and actual news.

Credibility of a news source plays a huge role in accessing and analyzing fake news. In the given dataset, the "context" column has been chosen to be the area of interest. This label hold data of the news source, which gives an insight to the exact origin of the news. The tracking down of the news source first point of news analysis and help narrow down the authenticity of the information itself.

We will proceed to the pre-processing of the data source for analysis with the identification of the data source. The data set contains approximately 10,000 rows of 14 columns in the data set. We deleted the characters in order to improve the application runtime and not have to deal with the kind of mistake of Not a Number errors.

After the generic analysis of the dataset in the previous section, we dig deep into each of the selected component. Below is the initial analysis of the dataset with respect to the "context" of the news source. With the pre-processed dataset, we do visualizations to understand the dataset and the correlation between the columns against one another, better. This step plays a crucial role in getting the nuances of the data present with relation to another.

One of the key problems is that there are numerous unique values for context, if we analyses the dataset we will end up with thousands of different values, which leads to an unsupervised classification problem. We will apply LDA on context identify top ten topics and perform unsupervised learning.

LDA stands for Latent Dirichlet Allocation. LDA. We need LDA to perform topic modeling. Let's say that we have a set of News Articles which were documents. By reading those articles we will be able to classify the news into different topics such as Sports, Politics or Science It's not herculean task for a human eye to identify the topic of a news article.

This paper presents various approaches to classify news as fake or not. A distillation approach is incorporated to perform feature engineering and ensemble technique is used to combine the features and build a fake news classifier.

B. Controversy

A controversy is a dispute or argument in which people express strong opposing views. When a popular TV show kills off a well-loved character, there's bound to be a lot of controversy. Controversial content can be defined as any contentious matter or argument that may spark public debate. The approach is motivated by several factors. First, news is a significant part of our everyday lives. It shapes our beliefs and opinions on how we see the world and now more than ever people rely on a variety of online resources for their news.

Therefore, the aim is to identify the potential of using single keywords to detect controversial content or topics that will influence fake news by applying Distillation process and multi-label classifiers.

This paper focuses on classifying the liar and liar dataset based on controversial statements. We aim to train the document model with multiple metrics which can be used in future to predict the test data or future document analysis. The words are made up of alphabets. But when considering controversial statements, the reality is words are triggers that consist more than strings. Words are used to convey a message that can also be used to manipulate someone or convince someone. Words are made up of emotional polarity that contributes to the controversial factor. Each word has controversial polarity that need to be considered to predict controversy score.

In this study, we analyze the liar and liar training dataset and detect the sentiment analysis metrics as vectors and perform text classification using those vectors. More specifically we have used five supervised machine learning algorithms: Naïve Bayes (NB), Logistic Regression (LR), Linear Support Vector Machine (SVM) and Random Forest (RF) classification algorithms on the liar and liar dataset.

The analysis measured with results of our machine learning experiments show that the Random Forest and SVM algorithms accuracy is more than the other algorithms. Cosine similarity helps to identify the highly similar news to have more prediction rate to be a fake news. Addressing the fake news problem is the baseline for the entire analysis. In addition to classifying, we have also performed vectorization and comparison of corpus TF-IDF vectors to analyze the similarity score among the documents which helps to predict if an article is more negative or a positive news. All the contributions and analysis are designed to support future work on fake news detection research project.

For distillation process, editors use more sentiment-loaded language on news articles. We calculated the frequencies of negative/neutral/positive words for each article and use these as input for classifiers. The algorithm to train the emotion polarity classifier (EP), using both the annotations and sentiment measures. Useful in analyzing the degree of controversy. Post that, we built the LDA model using Bag of Words on the Data set. Then, applied TD-IDF and found that performance of LDA without TF-IDF is better than LDA with TF-IDF. The accuracy increased to 55% with Gradient boosting. Controversy score was computed by applying Doc2Vec, LDA, BOW and cosine similarity. The model was trained using Random Forest Classifier, which provided threshold as 0.27 and maximum accuracy as 0.51.

C. Context/Venue

While analyzing the dataset, we will realize number of mediums/venues at which the news can be published. To an extent, the venue in which news is published is a critical feature in determining the authenticity of the news. Context can be any of the following locations online, house floor to press release. Another important location to account for is social media

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the widespread of “fake news”, i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention.

One of the key problems is that there are numerous unique values for context, if we analyses the dataset we will end up with thousands of different values, which leads to an unsupervised classification problem. We will apply LDA on context identify top ten topics and perform unsupervised learning

LDA stands for Latent Dirichlet Allocation. LDA. We need LDA to perform topic modeling. Let's say that we have a set of News Articles which were documents. By reading those articles we will be able to classify the news into different topics such as Sports, Politics or Science It's not herculean task for a human eye to identify the topic of a news article. But humans cannot handle zillions of documents that may be presented to them by various channels. So, it is paramount that we can teach computers to understand these topics.

This is where we can leverage topic modeling. LDA is the most popular topic modeling technique Topic modeling is a machine learning algorithm that helps unsupervised datasets such as news articles. These models are great at grouping words together into topic.

Guided LDA, LDA is an algorithm for unsupervised learning but in certain cases the topic words will not make meaningful classification, where we create a seed list of topics based on existing knowledge of dataset. This model drives the

model to converge based on the seed list to have a meaningful understanding of data.

With the guided LDA we will end up with finite context labels to tag the documents, and run vectorization on my headline to identify distribution and classification of news on these features.

This context LDA model is a derived feature as it is not part of the dataset. We will run vectorization on the feature and distill it with LDA topic score on headlines and sentiment score. Extracted context/venue features for a given document need to be represented as a vector for building the polynomial equation.

D. Credibility/Fact Checks

As an increasing amount of our lives is spent interacting online through social media platforms, more and more people tend to seek out and consume news from social media rather than traditional news organizations. The reasons for this change in consumption behaviors are inherent in the nature of these social media platforms: (i) it is often more timely and less expensive to consume news on social media compared with traditional news media, such as newspapers or television; and (ii) it is easier to further share, comment on, and discuss the news with friends or other readers on social media.

For example, 62 percent of U.S. adults get news on social media in 2016, while in 2012, only 49 percent reported seeing news on social media. It was also found that social media now outperforms television as the major news source. Despite the advantages provided by social media, the quality of news on social media is lower than traditional news organizations. However, because it is cheap to provide news online and much faster and easier to disseminate through social media, large volumes of fake news, i.e., those news articles with intentionally false information, are produced online for a variety of purposes, such as financial and political gain.

The extensive spread of fake news can have a serious negative impact on individuals and society. Therefore, it is a necessity to combat the fake news problem. There are several factors that contributes towards the detection of fake news on social media and Credibility/Fact Checks is one of them. Credibility is a broad term which can include assessing the credibility of publisher, credibility of the lexical features in the dataset, and context credibility. The speaker sentiment analysis plays a major role in assessing the credibility of the given news which helps us to predict the truthfulness or fakeness in the news. The sentiment of a speaker helps us to understand the tone/feeling of a statement, whether the statement is positive or negative.

This paper presents the detection of fake news based on the credibility/Fact checks factor which has been identified as one of the important component to overcome the challenge of fake news.

II. DATASET, SCRAPING AND ENRICHMENT

A. Source Credibility

As the first step in the feature engineering for the credibility of the news source, we begin by tagging the text data with the respective party to which they are affiliated. Using the dataset (Liar, Liar), we started our exploration on fake news analysis. We started off with understanding the dataset in terms of the labels and other critical data present in it. On careful attention we were able to identify the following labels that might be useful for the project:

1: the ID; 2: the label; 3: the statement;

4: the subject(s); 5: the speaker;

6: the speaker's job title; 7: the state info;

8: the party affiliation; 9-13: the total credit history count, including the current statement; 9: barely true counts;

10: false counts; 11: half true counts;

12: mostly true counts; 13: pants on fire counts;

14: the context.

The attributes in the data include positive, negative, strong, weak, active and passive. Each word in the dictionary is given a score indicating the percentage of the attitude. This attitude polarity is used, and documents are tagged as leaning towards positive, negative, active, passive, strong or weak.

With the identification of the data source, we proceed to the pre-processing it for the analysis. The dataset contains approximately 10000 rows in the data set with 14 columns to them. We have removed the characters in order to enhance the runtime of the code and not having to deal with Not a Number (NaN) kind of errors.

When dealing with unstructured data, the biggest challenge is text tagging. Data enrichment of two types are performed. One, the data set is tagged based on the credibility. In this process, the word overlaps between the dictionary of words and headline text is identified. Using this word overlap, the polarity of the text towards the attitude present in the dictionary is derived. Each headline is then tagged based on the polarity.

The second enrichment performed is using the cosine similarity. Cosine similarity between the dictionary of words and headline is evaluated using both Word2Vec and TF-IDF methodology. The resulting cosine similarity between the two vectors (dictionary of credibility and context) is used later for multi-class classification.

B. Controversy

"Liar, Liar Pants on Fire" dataset from PolitiFact.com is used in this paper. The data set consists of manually labelled short text. This data can be used for fact checking. The data set consists of 3 files: train.csv, test.csv and validate.csv that can be used to run the NLP and ML lifecycle. The data consists of 14 fields which includes the headline text, labels, statements etc.

As the first step in feature engineering for controversy, we begin by tagging the textual data with the respective party they are affiliated to. Several approaches are suggested to perform the same. Each word in the dictionary is given a score indicating the percentage of the attitude. This attitude polarity is used, and documents are tagged.

When dealing with unstructured data, the biggest challenge is text tagging. Data enrichment of two types are performed. One, the data set is tagged based on the controversy. In this process, the word overlaps between the dictionary of words and headline text is identified. Using this word overlap, the polarity of the text towards the attitude present in the dictionary is derived. Each headline is then tagged based on the polarity.

The other enrichments performed are by using the cosine similarity, Doc2Vec, Latent Dirichlet Allocation, Variance and Latent Semantic Analysis. Cosine similarity between the dictionary of words and headline is evaluated using both Word2Vec and TF-IDF methodology. The resulting cosine similarity between the two vectors is used later for multi-class classification.

C. Context/Venue

We start with "Liar Liar Pants on Fire" dataset from politifact.com. The initial dataset consists of train.tsv, test.tsv and valid.tsv, each of these have context as a feature. We will leverage context. However, the dataset has that can be used for topic modeling. The dataset has a label to classify the news article, which is transformed to a new feature called "encoded_label". This transformation helps determine if any of the train, test and valid datasets have a bias that need to be factored. The data understanding started with distribution of dataset against a scalar label. Then since context is our feature, analyzed the unique values of context feature. The results were alarming and proved to be difficult to use as a feature. There were totally 5143 unique context/venues.

The immediate next step was to reduce the number of contexts to finite topics or mediums. Performing LDA was the logical next step. Prerequisite to perform LDA is to clean up the data leveraging stemming, lemmatization and filtering algorithms and build the BOW vector and TF-IDF vector. In this process a dictionary is created using the preprocessed dataset leveraging Gensim libraries. The actual ldamodel using Gensim libraries and bag of words is built with a request to generate top 10 topics. The extracted context lda topics are words that live together and does not immediately map to a topic label. Guided LDA or Semi supervised modeling was required to classify context to human readable labels. Analyzing few news articles of every context topic helped relabel the dataset. This process enriched the dataset with a derived context column which can be used for labeling and vectorization lda2vec was performed against the topics and weights to form a vector representation of topic feature, at the same time sentiment analysis was performed on the headline text using Vader.

D. Credibility/Fact Checks

"Liar, Liar Pants on Fire" dataset from PolitiFact.com is used in this paper. The data set consists of manually labelled short text. This data can be used for fact checking. The data set consists of 3 files: train.csv, test.csv and validate.csv that can be used to run the NLP and ML lifecycle. The data consists of 14 fields which includes the headline text, labels, party affiliation, topics etc.

The process of implementing this factor for fake news detection commence with loading the base dataset that we are using for this project – "Liar, Liar Pants on Fire" followed by Exploratory Data Analysis, Visualizations to get the better understanding of the dataset. Once the data preparation step is performed on the base dataset, Bag of words and TF-IDF models are generated as part of Distillation process followed by performing Speaker Sentiment Analysis on the statement attribute to verify the authenticity of the given news. Since, the base dataset has three files consisting of train, test, and valid datasets, data enrichment will be performed on all these three sets. Data cleaning and data preparation step focuses on cleaning the dataset by removing numbers, punctuations, stop words, and performing stemming and lemmatization which are the most important steps in Natural Language Processing. The Beautiful Soup was used to extract the statements from the glove.6B.50d dictionary and the dataset was enriched with the newly extracted statements.

As the first step in feature engineering for Credibility and Fact Checks factor, we begin with amalgamation of Liar Liar dataset with the glove.6B.50d dictionary/dataset which generates the vocabulary and convert the features into vector using count vectorizer. The second enrichment on the dataset is done by reducing the six labels in the dataset to two – True and False representing the news is fake or not. Then, added as a separate column in the dataset and the classifier/model is trained based on these enrichment.

III. IMPLEMENTATION DETAILS

A. Source Credibility

- **What did I try?**

The brief summary on the steps followed are

1. *Pre-processing*

Headline text is cleaned using a number of NLP cleaning techniques. The text-cleaning process begins with the conversion of all text data to a lower case, followed by the removal of stop words from the NLTK library. The Spell Check is done using Word2Vec. The Google's word lists are used as the corpus for the Word2Vec model. The spell checker of Peter Norvig is used to identify the correctness and replace the words misspelt in the text of the headline. The next step is to remove the punctuations and use the string library of the NLTK. This is accompanied by the stemming. Stemming is the process of reducing a word to its word stem that is attached to suffixes and prefixes or to the roots of lemma-known words. Snowball stemmer is a language written out of all forms of a word to extract meaningful word root. Post stopping the length of the headline text is reduced by removing words that are less than 3 in length. Verbs are also used to perform lemmatization. On both the train and test data, this pre-processing is performed. We dig deep into each of the selected components after the generic analysis of the data set in the previous section. Below is the data set's initial analysis of the news source's "context."

label / context	False	True	barely-true	half-true	mostly-true	pants-fire
All	1974	1662	1642	2106	1948	831
a news release	50	50	51	52	50	24
a speech	50	56	38	60	57	12
a press release	46	63	42	57	50	15
an interview	69	39	36	53	55	14
a TV ad	35	24	47	55	42	17
a tweet	30	31	27	25	41	18

Fig 1.

2. *Visualization*

To identify the average length of text, the pre-processed text will be visualized. This helps to determine the size of the vector without much loss of data. In order to understand the distribution of different classes of text, the text categories are also visualized. For visualization, the Matlibplot library is used. Word cloud is used in the textual data to understand the importance of different words.

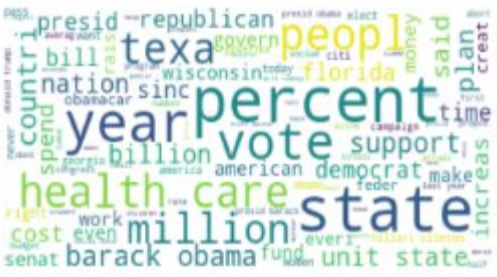

Figure 2. Word Cloud Visualization

	label	context	verdict
0	False	a mailer	0
1	half-true	a floor speech	1
2	mostly-true	Denver	1
3	False	a news release	0
4	half-true	an interview on CNN	1
5	True	a an online opinion-piece	1

Figure. Context and label data

With the above cross tabulation, we infer the total number of

- Barely True
- Mostly True
- False
- True
- Pants fire
- Half True
- Mostly True

They give clear insight into the credibility of the news sources in the dataset. This gives a fair understanding of the test data set to be used for the evaluation of this element. A bar graph on the same data used for cross tabulation led to a chart that was further drilled down to depict just one context from the column–"a campaign ad" to get a closer analysis of it over a bar graph:

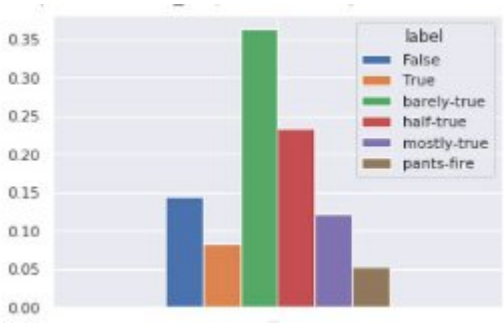

Figure 3. Bar Chart for Campaign Ad

3. *Count Vectorization*

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 sci-kit learn Count Vectorizer is used to convert headline text to vector. The library takes a vocabulary list against which the frequency of occurrence is performed to convert text to vector. In this approach, the news source of words was initialized as the vocabulary for representing headline text as count vector. Once the headline text is converted to vector representation, classification is done using various supervised learning algorithms. The train data is used to train the model on the fakeness labels. The test data is used to evaluate the model. Logistic regression, SVM, Random Forest and Multinomial Naïve Bayes algorithms are evaluated. K-Fold cross validation techniques is used to find the model accuracy. Cross validation is a technique used for evaluating model accuracy. In K-Fold, the algorithm creates a subset of the data and uses each subset to test, train and validate. SVM yields the best result of 70% accuracy.

	label	context	verdict
0	False	a mailer	0
1	half-true	a floor speech.	1
2	mostly-true	Denver	1
3	False	a news release	0
4	half-true	an interview on CNN	1
...	...	...	...
10264	mostly-true	interview on "The Colbert Report"	1
10265	mostly-true	an interview	1
10266	half-true	a Republican presidential debate	1
10267	False	a televised debate on Miami's WPLG-10 against ...	0
10268	pants-fire	a Fox News interview	0

10269 rows × 3 columns

Figure 4. Label and context for Credibility of News Source

4. *Enrichment by Tagging*

The text was marked as false or not by the count vector test. The goal is to mark every report as
present in the dictionary with a background reference. Steps involving tagging the source documents.

I. Based on the score in the dictionary, group the list of source words into Positive, Negative, Strong, Weak, Active and Passive.

II. Find the word overlap between each group of words for each headline text.

III. Calculate the polarity so that the attitude of the headline is the highest weight for an attitude in the overlap set.

IV. Mark and expand the records by adding a new feature to the dataset.

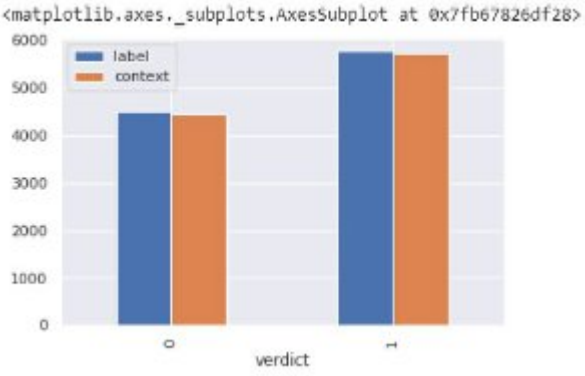

Figure 5. Bar chart of the verdict

Based on the above graph, if we are to perform cross tabulation, then this gives a clear picture on the reliability of the news source given we have the verdict column to decide on:

verdict	0	1	All
context			
All	4447	5716	10163
a news release	125	152	277
a speech	100	173	273
a press release	103	170	273
an interview	119	147	266
a TV ad	99	121	220

Fig 6

5. *Word2Vec*

Figure 7. Word2Vec Visualization using t-SNE

Another alternate approach to vectorize text and identifies the similarity is done using Word2Vec. Word2Vec model is used to convert each word in a text to vector. It then retrieves the features from this vector. The steps followed are

I. Find the word overlap between dictionary of words and the text.
II. Calculate polarity by summing the attitude weights associated with each word in the overlap set.
III. Normalize this polarity vector and enrich the original dataset with this representing the weighted credibilty vector pa_weight in the dataset.
IV. Tokenize – split the text data into words using regular expression
V. Build a Word2Vec model and derive the same number of features as the credibilty vector size.
VI. Normalize the vector derived using Word2Vec model
VII. Find the similarity between the weighted credibility factor and the Word2Vec of the headline.
VIII. This similarity can be used to tag the text as Positive, Negative, Strong, Weak, Active or Passive.

6. Natural Language Processing

Using the nltk library, data preprocessing has been performed. Topic Modelling is a statistical model for finding nonconcrete topics that are present in the dataset. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions

Tokenization Split text to sentences and sentences to words. Switch into lowercase for words and remove punctuation.

```
# create the dictionary
dictionary = gensim.corpora.Dictionary(processed_docs)
count = 0
for k, v in dictionary.iteritems():
    print(k, v)
    count += 1
    if count > 10:
        break
```

```
0 mailer
1 floor
2 speech
3 denver
4 news
5 releas
6 interview
7 onlin
8 opinion
9 piec
10 press
```

Fig 8

Remove words that have fewer than 3 characters.
Removed stop words from the dataset.

Lemmatization: Words in third person are changed to first person and verbs in past and future tenses are changed into present.

What did not work?

Count Vector approach did not work as it only considers the frequency of the occurrence of words. It does not evaluate the similarity between two documents. This did not help in tagging the text.

The cosine similarity between the dictionary of words and the headline text showed poor results and the fakeness classification using this methodology The Word2Vec model had huge loss of information as the vector sizes were small. This yielded poor classification accuracy.

TF-IDF and cosine similarity also did not help me derive how closely related the documents were to the attitude present in the political dictionary.

What worked and Alternate Approach

Naive Bayesian classifier: Based on the dataset, the below list the precision, recall for the context. The average precision is about 51% as shown below:

```
print(classification_report(class_test,predictions))
```

	precision	recall	f1-score	support
0	0.44	0.53	0.48	556
1	0.57	0.47	0.52	727
avg / total	0.51	0.50	0.50	1283

Fig 9

Polynomial equation for the factor:

Based on the NB calculated above, a polynomial equation for the credibility of the news source can be calculated using:

Credibility factor = 0.4*(prediction from NB)

A heat map can be plotted based on the outcome form the Naïve Bayesian classifier which is found below:

Figure 10. Correlation Matrix

1. *Doc2Vec*

As an alternate approach to TF-IDF Doc2Vec model is trained. Doc2vec is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents. Doc2Vec is an adaptation of word2vec model. The vectors generated by doc2vec can be used for tasks like finding similarity between sentences/paragraphs/document. The distributed bag of words method is used to vectorized each headline text. The vector size is a fixed length of 20. Accuracy is tested against various vector size and 20 yields the best result. The Doc2Vec model is trained by tagging the documents based on the party affiliation tag created in the previous step. A classifier is built to predict multi-class text based on party affiliation tag. The doc2vec model achieves a better accuracy of 45%.

2. *Distillation*

Distilling the most representative information from a text but also excluding the general background information to produce a more informative low-dimensional vector representation for the text. A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions. Combining multiple models into an ensemble by averaging their predictions is a proven strategy to improve model performance. While predicting with an ensemble is expensive at test time, we use distillation mechanism which allow us to compress expensive ensemble into smaller models and combine them to improve accuracy. The results from distillation is combined to form a single feature and later features from each member of the group is represented in a polynomial equation to build a fake news classifier. The steps followed in distillation is

I. Sentiment Analysis
II. Topic modelling using LDA
III. Ranking based on Speaker importance

Sentiment Analysis, or Opinion Mining, is a sub-field of Natural Language Processing (NLP) that tries to identify and extract opinions within a given text. The aim of sentiment analysis is to gauge the attitude, sentiments, evaluations, attitudes and emotions of a speaker/writer based on the computational treatment of subjectivity in a text. According to many researches, sentiment analysis is important for fake news identification. A research suggests that if a news is positive the probability of it being fake is more. Text Sentiment is derived using Vader Sentiment Library. VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive, negative or neutral. The sentiment polarity score is used to tag the document as Positive, Negative or Neutral based on the max polarity score from the Vader analyzer. The values are then normalized using log to achieve better accuracy. I have not used the encoded label as is but normalized the value so that the classifier does not weigh a sentiment more than the other. Hence, normalization using math.log () is performed. From sentiment analysis, it was inferred that most of the text had a neutral sentiment and hence not a very effective feature for multi-class text classification.

Latent Dirichlet Allocation (LDA) is a Probabilistic, generative model which uncovers the topics latent to a dataset by assigning weights to words in a corpus, where each topic will assign different probability weights to each

word. In LDA, the modelling process revolves around three things: the text corpus, its collection of documents, D and the words W in the documents. Therefore, the algorithm attempts to uncover K topics from this corpus. The LDA algorithm first models' documents via a mixture model of topics. From these topics, words are then assigned weights based on the probability distribution of these topics. It is this probabilistic assignment over words that allow a user of LDA to say how likely a word falls into a topic. Subsequently from the collection of words assigned to a topic, are we thus able to gain an insight as to what that topic may represent from a lexical point of view. From a standard LDA model, there are really a few key parameters that we must keep in mind and consider programmatically tuning before we invoke the model:

I. components: The number of topics that you specify to the model
II. α parameter: This is the Dirichlet parameter that can be linked to the document topic prior
III. β parameter: This is the Dirichlet parameter linked to the topic word prior

After distillation, fake news classifier is developed using two different methods

I. Scalar Vector Addition: The scalar values sentiment score, topic score, rank is added to the doc2vec source credibility feature.
II. Vector Embedding: In this technique, the doc2vec model is embedded with sentiment score, topic score and rank value.

Together with doc2vec, the model comes up against the second dataset with a 43 percent accuracy to obtain a credibility classification. The meaning of the initial data set, which was about 67 percent accurate, was not very reliable and thus reduced the accuracy while inducing a new source of data.

We understand that sensationalism alone is not enough to determine whether a news is fake, but it plays a role. We wanted to give credibility 40 percent of the four factors we use to classify data as true or not.

B. Controversy

- **What did I try?**

Initially visualized the datasets for train, test and valid.

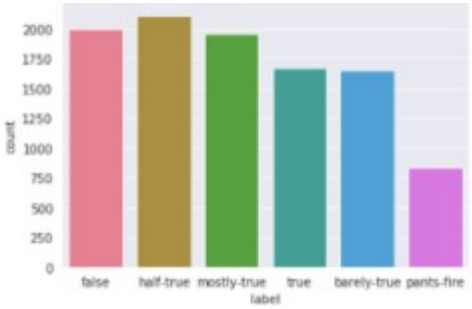

Figure 11. Initial Dataset

From the plot, it can be inferred that 'false' 'mostly true' and 'true' have much longer texts, but there are many outliers, which can be seen as points above the boxes.

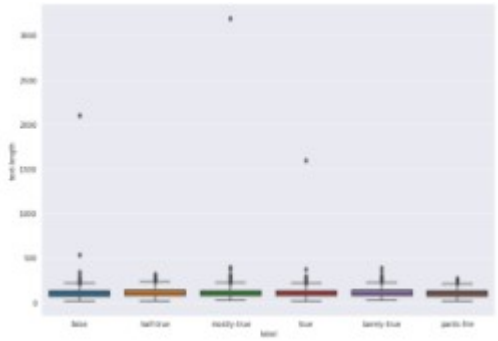

Figure 12. Violin Plot

Plotted a correlation matrix, it was determined that false is strongly correlated with pants_on_fire, and mostly true seems strongly correlated with half true. There is also a negative correlation between false counts and the other features.

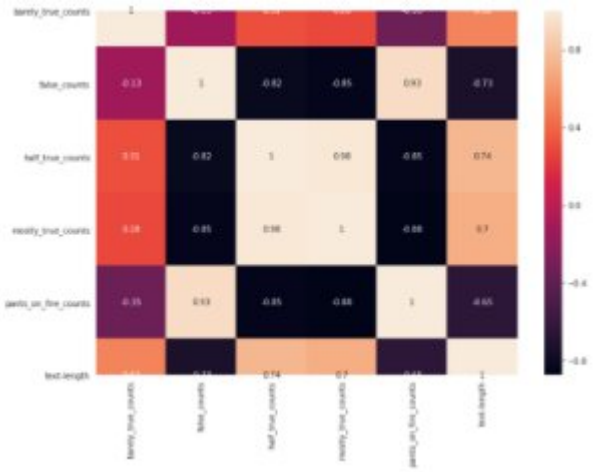

Figure 13. Correlation Matrix

o *Distillation*

After loading the dataset from TSV file, the initial step is to process the complex dataset, ensemble, feature extraction, detect the soft target and perform vectorization or classification models. Distillation is one of the most important process in deep learning and neural networks. It converts a large model into a small model by performing list of preprocessing process. Distillation combines data pre-processing, sentiment analysis, LDA topic modeling and Classification model.

The pre-processing consists of three parts: Tokenization, Normalization and Noise Removal. The sentiment is detected using Vader Sentiment Intensity Analyzer NLTK library.

- *Pre-processing*

The data preprocessing of Liar and Liar pants on fire dataset includes 4 steps: Cleaning, Tokenization, removing stop words and Stemming/Lemmatization. The first cleaning process involves removing the punctuation marks, special symbols from each article. Used the 're' regular expression library to remove punctuation marks and convert all words into lowercase. Next step is to tokenize the article into a list of words. Tokenization is the process of splitting a single sentence or paragraph into a list of basic units of words. The next step in preprocessing is to remove the stop words like this, that, and, etc. I used the NLTK library nltk. corpus which provides a list of stop words. Followed by the stemming or lemmatization process. 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. For example: studying -> study, studied -> studi. There are three types of stemming: Porter (Least aggressive), Snowball and Lancaster (most aggressive). In general, most aggressive algorithms trim faster but removes most part of the word. Whereas a least regressive algorithm like Porter takes a lot of time and keeps most of the words with less or no change. Snowball stemmer is highly used and faster than Porter and does not trim down too much information as Lancaster does. This pre-processing step are performed on both the training data and the test data.

- *Visualization*

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. Below is the representation of the word-cloud of the dataset. Visualized word cloud for each label for example for all the six labels.

Figure 14. Word Cloud

Frequency of words can be as show in the figure below.

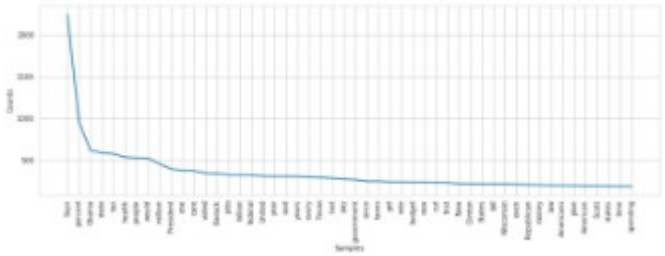

Figure 15. Frequency of Words

- *Computation of Sentiment polarity and Intensity*

Sentiment Analysis was computed using NLTK Vader. VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive, negative or neutral. The sentiment polarity score is used to tag the document as Positive, Negative or Neutral based on the max polarity score from the Vader analyzer.

- Editors use more sentiment-loaded language on news articles. I calculated the frequencies of negative/neutral/positive words for each article and use these as input for classifiers.
- The algorithm to train the emotion polarity classifier (EP), using both the annotations and sentiment measures.

- Useful in analyzing the degree of controversy.

	neg	neu	pos	compound	statement	vari
0	0.115	0.682	0.192	0.2500	Says the Annies List political group supports ...	0.07
1	0.000	0.902	0.098	0.3612	When did the decline of coal start? It started...	0.06
2	0.107	0.687	0.206	0.3182	Hillary Clinton agrees with John McCain "by vo...	0.06
3	0.000	0.606	0.394	0.7579	Health care reform legislation is likely to ma...	0.36
4	0.000	1.000	0.000	0.0000	The economic turnaround started at the end of ...	0.00

Figure 16. Variance min and max range for Controversy detection

o *LDA Topic Modeling*

The dataset consists of the news headlines. To classify these articles into meaningful topics, I performed LDA on the headlines after performing cleaning process in the dataset. Create a dictionary from 'vocabulary' containing the number of times a word appears in the training set. LDA returns a vector of topic probabilities for each document.

o *Counter Vectorization*

Using various feature extraction of text libraries like CountVectorizerto vectorize the articles in the trading and test dataset with sentiment analysis metrics and performed various classification model. Bag of Words is simply the matrix that counts how many each word appears in article. The vectors of each article are classified using the Pipeline with five supervised classification models:

Naïve Bayes (NB), Random Forest (RF), all of the model's performance is validated using the Classification Model Evaluation Technique K-Fold.

o *TF-IDF Vectorization*

TF-IDF is another vectorization technique used to vectorize text using word frequency. TF-IDF assigns lower weights for common words. I tried to vectorize using TfidfVectorizer() with ngram range of 1 to 2. In order to get better results, each word is Vectorized in the article. Then classify the vectored articles against the vocabulary dictionary that I scraped and enriched from the Persuasive Revolution website. TF-IDF provides the feature names and the vocabulary vector values which can be used to vectorize each news article. I also performed TF-IDF using the sentiment analysis. I classified sentiment polarity as negative, neutral and positive for each article. Performed all the five classification models and validated using KFold. The accuracy of the article being predicted with sentiment vector of values positive, negative and neutral is 94% by SVM model.

o *Cosine Similarity*

Cosine similarity is highly used when two vectors are compared. It is the cosine of the angle between couple of n-dimensional vectors in an n-dimensional space. It helps to perform a dot product of the two vectors. After we vectorized the document using TF-IDF, the controversy vector formed from the controversial word's corpus can be compared together using the cosine similarity library. The similarity score helps to predict the controversy of the document. High similarity score means the document is highly likely to be controversial. Because the words that I chose form the dictionary are highly controversial words. Hence, if the similarity of a document is high with this dictionary vector then the document is classified as controversial.

As part of data enrichment, the calculated values from different steps mentioned above were added to the dataset.

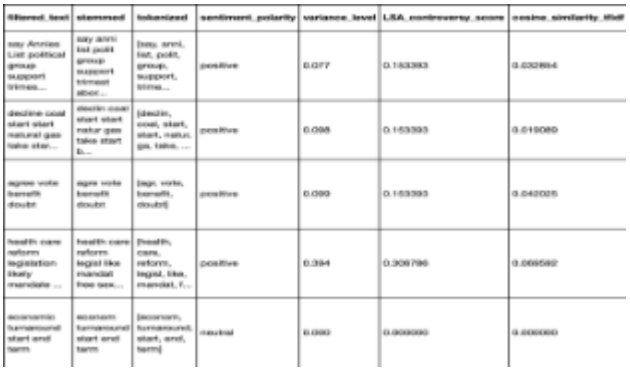

Figure 17. Data Enrichment to detect fake news

Figure 18. Data Enrichment with all the scores

- *Word2Vecand Vector Averaging*

Word2Vec is used to generate a vector for each word. This algorithm provides a vector for each word which has values that are in same range for similar words.

For instance: In the below figure, the word "Hillary" is closer to "Clinton" which makes sense as most of the articles have these two words occurring together. Similarly, the word "democrat" also plotted near to "Hillary". But the dimension of each word having a vector creates a problem when we wanted to perform multi-class classification. Specially to perform cosine similarly the dimensions need to be identical.

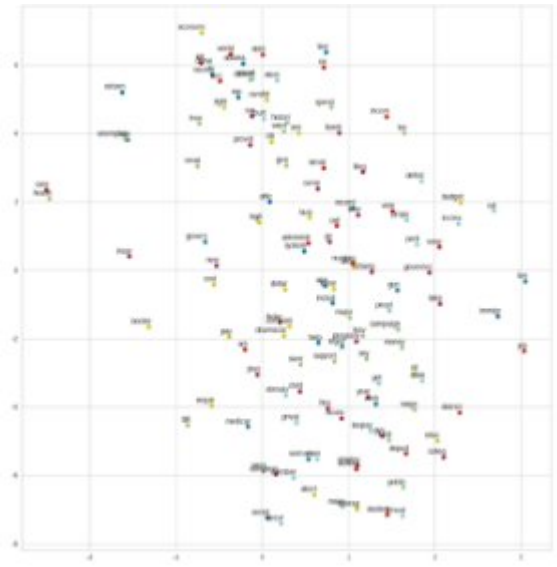

Figure 19. Visualize Word2Vecusing TSNE

- *Doc2Vec*

Doc2Vec provides a vector representation of the entire document or text based on the number of features. Gensim Doc2Vec model is used to convert the document into a vector. Distributed Bag of Words algorithm is used to get the vector representation of the text. The tagged documents based on the controversial classification are vectorized and then used for training and testing the text classification. Here the vector is formed for the entire document. But CountVectorizerand TF-IDF provides vector based on each word in the document.

- **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 sentiment factors of the document. So, I decided to enrich the data with sentiment metric before vectorizing and performing classification models. I enriched 5 columns related to sentiment analysis which is used to classify the document based on its sentiment level. I initially tried to classify the document as 'positive' and 'negative'. I formed a metrics that provides the polarity and the intensity of each word. I aggregated it for each document and vectorized using CountVectorizer.I wanted to use this vector to classify the document and predict a future document if it is a positive or a negative sentiment.

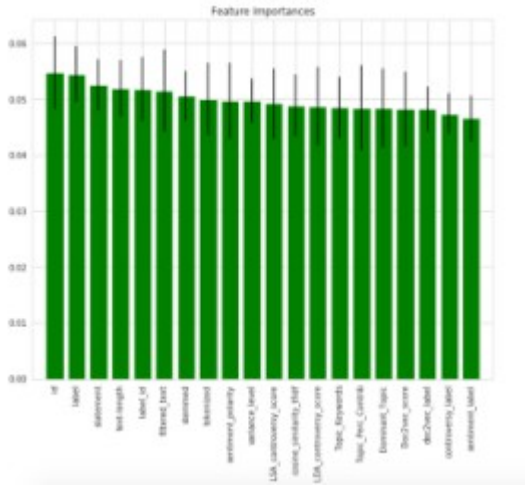

Figure 20. Feature importance using Random Forest Classifier

Few documents were found to have neutral polarity which means either they have same amount of positive and negative sentiment or none. In that case the accuracy of a document being either positive or negative will be wrong. Then classified the document based on three vector parameters "positive", "negative" and "neutral". All the five supervised classification models are performed as earlier.

In order to vectorize the entire document and classify it based on the sentiment vector we decided to use the Doc2Vec - Distributed Bag of Words (DBOW). DBOW is the doc2vec model analogous to Skip-gram model in word2vec. The paragraph vectors are obtained by training a neural network on the task of predicting a probability distribution of words in a paragraph given a randomly-sampled word from the paragraph. We used Tagged Document and created a model using the training dataset with the vector of sentiment intensity as the label. We formed a vector for each document and classified the model using classification models.

Algorithms	Accuracy
Multinomial Naive Bayes	76.83%
Linear Support Vector Classification	95.03%
Support Vector Machine	95.07%
Random Forest	96.67%
K Nearest Neighbor	51.09%

Figure 21. Accuracy scores

The metrics using Random Forest Classification algorithm is shown in the following figure.

	precision	recall	f1-score	support
1	0.89	1.00	0.94	734
3	1.00	0.95	0.98	1826
accuracy			0.97	2560
macro avg	0.95	0.98	0.96	2560
weighted avg	0.97	0.97	0.97	2560

Figure 22. Metrics – Precision, Recall, F1 Score

We used the preprocessed data and created a corpus. Scraped 1400+ sensational words from a popular website and created a dictionary. vectorized the corpus using TF-IDF vectorizer. Performed cosine similarity and computed the similarity score. The documents that has more similarity score is classified as controversial.

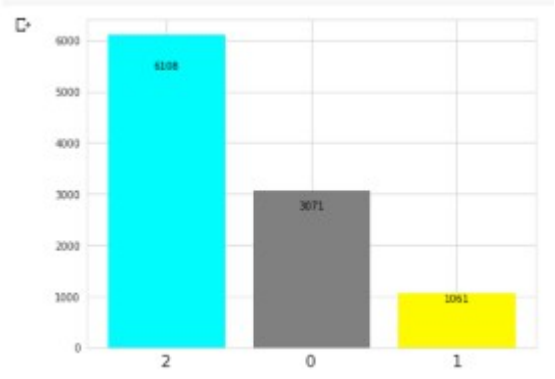

Figure 23. Bar Chart

0 - Non-controversial
1 - Weakly- controversial
2 - Controversial

The number of controversial statements is higher than non-controversial statements.

- **What did not work?**

As I was taking the sentiment intensity of each word and aggregating the words to get single vector for each document, I tried to get a vector for each word. I tried Word2Vec library.

LDA TF-IDF model did not perform better, since it calculates score based on frequency of terms appearing in my vocab document.

The Word2Vec provides vectorization for each word, which causes dimensional issues with the 'senti_word_vector' column of each document. I performed Vector Averaging which gave me single vector for each document but with different length of the vector.

- **What worked and Alternate Approach**

The TF-IDF Vectorizer with n-grams worked way better in when we applied K-Fold cross validation. The n-grams considers the words surrounding a given word to understand the context which gave better scores. By performing sentiment analysis, the mapping of the sentiments to labels really helped in generating controversy score. We performed Random Forest on the data. It gave 96% accuracy score. Support Vector Machine also gave good accuracy on 95%. Semi supervised learning or seeding topics, provided better results and clear classification of news articles, we identified the top words for every topic.

C. Context/ Venue

- **What did I try?**

The brief summary on the steps followed are

1. *Pre-processing*

   Label feature of the dataset is transformed to a scalar for observing the distribution of data. The transformation helped visualization of the datasets. Further headline text is cleaned using the techniques such as remover stop words, tokenizing, stemming and lemmatization. For removing stop words nltk. corpus library provided with the list that helped clean up the data. Stemming was done making use of Porter Stemmer and Snowball stemmer respectively for word2vec and LDA respectively. Performed spell check using the Google News spell vectors dictionary.

2. *LDA for context*

   Context feature in the dataset explains the venue were the headline was made. There are over 5000 unique context values, in the dataset. To classify

them into meaningful avenues for publishing new, performed LDA on context after performing cleaning process on context column of the dataset.

Performed Guided LDA for the top 10 mediums to identify where a new is published. This process also helped “Ranking” the context. If a news is published in political forum such as senate or house it is trustworthy, over an election campaign or social media

3. *Visualization*

The pre-processed document is distributed against encoded label and visualized to identify the average text length. This helps in deciding the vector size without much data loss.

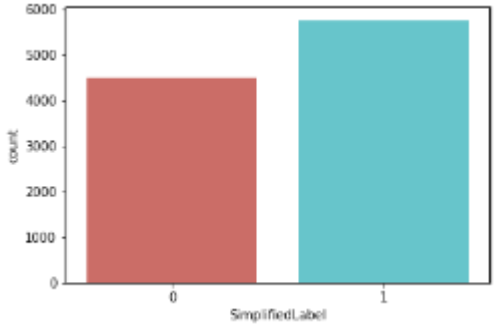

Fig. Distribution of preprocessed dataset

The context categories are also visualized to understand the distribution of various classes of text. Matlibplot library is used for visualization. Word cloud is used to understand the importance of various words in the textual data. This process helped perform Guided LDA

Fig. Wordcloud of context feature

After performing Guided LDA, each headline was updated with context features. Ran distribution to plot number of headlines for each context.

Ranking of semi supervised context were done based on authenticity of context, listed the context as per ranking.

Semi Supervised Contexts

- senate_house_floor
- news_release_nation
- pressrelease_media
- presidential_debate
- book_newsletter
- townhall_meeting
- public_comments
- interview_media
- twitter_online_campaign
- interview_campaign_appearance

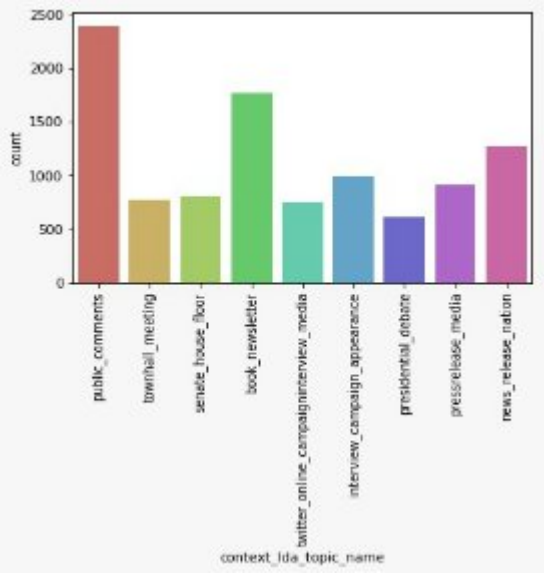

Fig. context LDA topic distribution

4. *Bag of Words - Count Vectorization*

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. The sci-kit learn CountVectorizer is used to convert headline text to vector. The library takes a vocabulary list against which the frequency of occurrence is performed to convert text to vector.

Classification algorithms

- Naives Bayes regression
- Logistic regression
- SVM Stochastic Gradient Descent
- Linear SVM Classifier
- RandomForestClassifier

K-fold cross validation algorithm is performed to build the confusion matrix to analyse

classification algorithms. Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problembecause it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

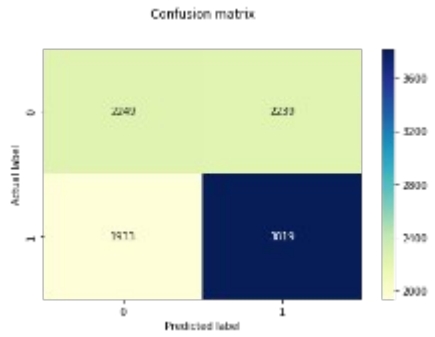

Fig. Sample K-Fold Confusion matrix

5. *NGrams - TF-IDF Vectorization*

So far, we have used bag of words technique to extract the features and passed those features into classifiers. We have also seen the f1 scores of these classifiers. now let's enhance these features using term frequency weights with various n-grams

TF-IDF stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

TF-IDF Vectorizer converts a collection of raw documents to a matrix of TF-IDF features. The approach was done to see if the accuracy scores improved by bringing context to the mix and we did see significant improvements in accuracy

Out of all the models fitted, we would take 2 best performing model. We would call them candidate models from the confusion matrix, we can see that random forest and logistic regression are best performing in terms of precision and recall (look into false positive and true negative counts which appears to be low compared to rest of the models)

Logistic Regression

	precision	recall	f1-score	support
0	0.69	0.37	0.46	553
1	0.62	0.81	0.71	714
micro avg	0.62	0.62	0.62	1267
macro avg	0.61	0.59	0.58	1267
weighted avg	0.62	0.62	0.60	1267

Random Forest

	precision	recall	f1-score	support
0	0.56	0.46	0.50	553
1	0.63	0.72	0.67	714
micro avg	0.61	0.61	0.61	1267
macro avg	0.59	0.59	0.59	1267
weighted avg	0.60	0.61	0.60	1267

Fig. Classification Report

6. *LDA2Vec*

Lda2vec is obtained by modifying the skip-gram word2vec variant. In the original skip-gram method, the model is trained to predict context words based on a pivot word. In lda2vec, the pivot word vector and a document vector are added to obtain a context vector. This context vector is then used to predict context words. The idea of integrating context vectors in the word2vec model is not a new idea. Paragraph vectors, for example, also explored this idea in order to learn fixed-length representations of variable-length text fragments. In their work, for each text fragment (size of a paragraph) a dense vector representation is learned, like the learned word vectors.

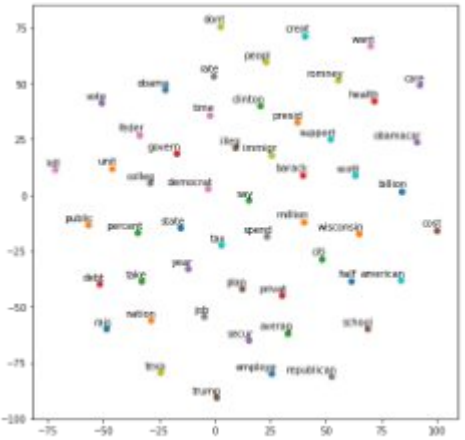

Fig - Visualization of lda2vec model using t-SNE

7. *DOC2VEC*

The algorithm is broader adaptation of word2vec which can generate vectors for words. For sentence similarity tasks, doc2vec vectors may perform reasonably well. While Word2Vec computes a feature vector for every word in the corpus, Doc2Vec computes a feature vector for every document in the corpus.

Since the tagged documents were headlines and our goal were to create a vector for every headline, doc2vec was the way to go and yielded better results than Mean Embedding Vectorization

- **What did not work?**
  I. Count Vectorization of the headline text and applying classification did not yield great accuracy results.

  II. Bag of Words model did not work as much as n-grams did as bag of words was not taking context into the equation. The classification performed on vectors created by bag of words and counter vector did not yield great results Word2Vec did not work directly on our topic labels to convert enrich topic feature as a vector. word2vec works on a word instead of the whole document thus resulting in vector of vectors and leading to matrix multiplication errors on cross products

  III. Unsupervised learning for topic models, when the topic labels were retrieved from LDA model, the topics did not give a definitive meaning or classification of the news article.

- **What worked and Alternate Approach**

  I. The TF-IDF Vectorizer with n-grams worked way better in when we applied K-Fold cross validation. The n-grams considers the words surrounding a given word to understand the context which gave better scores

  II. lda2vec is an extension of word2vec and LDA that jointly learns word, document, and topic vectors. It is specifically building on top of the skip-gram model of word2vec to generate word vectors. With lda2vec, instead of using the word vector directly to predict context words, we leverage a context vector to make the predictions. This context vector is created as the sum of two other vectors: the word vector and the document vector

  III. Semi supervised learning or seeding topics, provided better results and clear classification of news articles, we identified the top words for every topic.

D. Credibility/Fact Checks

- **What did I try?**

The brief summary on the steps followed are:

1. *Pre-processing*

The text cleaning process begins with adding the required column names to the dataset, followed by converting label classes to numeric. Then all the textual data is converted to lower case, followed by removal of stop words using Gensim and NLTK libraries and the punctuations are also removed from the text. The next step is to remove punctuations and is done using the NLTK's string library. Followed by this is, 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. Snowball stemmer is a language written to extract meaningful word root from all forms of a word. Snowball Stemmer and Porter Stemmer was used for the stemming of words.

Stop Words Removal: Removing stop words, there are some words in english language which may not contribute meaning to the phrase words such as before, had, when are called as stop words and they are filtered using nltk library stop words = nltk. corpus.stopwords.words('english') is used for doing the same.

Stemming: Stemming is about getting the words by removing suffixes, the words we get may not be the dictionary word. I used the library called port stemmer for this. porter = nltk.PorterStemmer() is used for doing the same.

Tokenization: Tokenization is the process of splitting the words or complete sentences into the phrases from the body of the text.

Lemmetization: Lemmetization usually refers to the process of using a vocabulary and morphological analysis of words, normally aiming to remove the endings or suffixes of the words and return the root/base form the given statement.

2. *Visualization*

Exploratory Data Analysis is performed on the base dataset in order to visualize the distribution of data in training set using the Pair Plot.

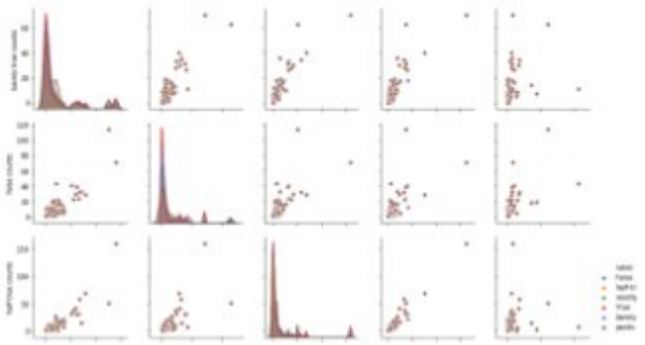

Fig – Pair Plot

Also, the distribution of classes in train, test, and validation data is visualized based on the labels in dataset. These visualizations helped to identify the total number of classes belonging to a particular category/label.

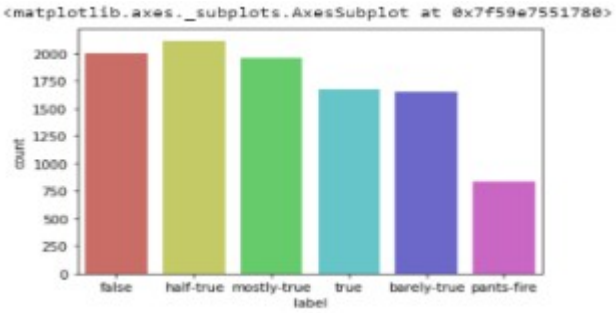

Fig – class distribution

Matplotlib library is used for visualization. Word cloud is used to understand the importance of various words in the textual data.

The below figure depicts the visualization of Pie chart which show the distribution of data in training set.

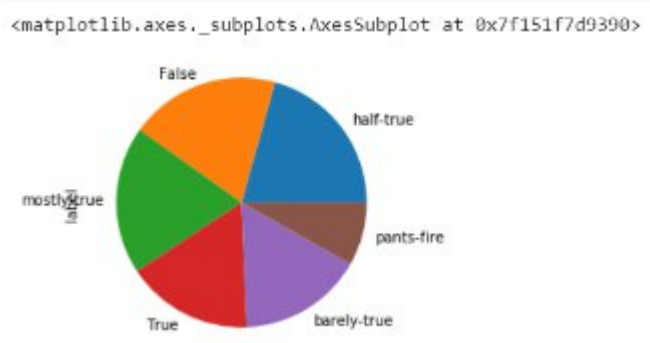

Fig – Pie Chart

The below figure represents the heatmap that shows the distribution of six different labels in the training dataset

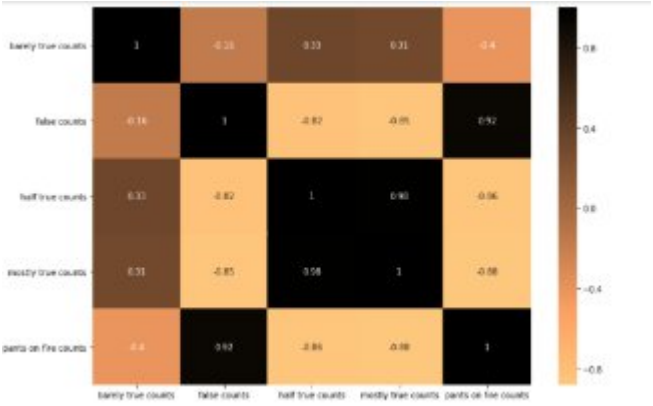

Fig – HeatMap

WordCloud visualizations helps in getting better understanding of the statements by providing the top most words occurring in the dataset. Below is the figure depicting the same:

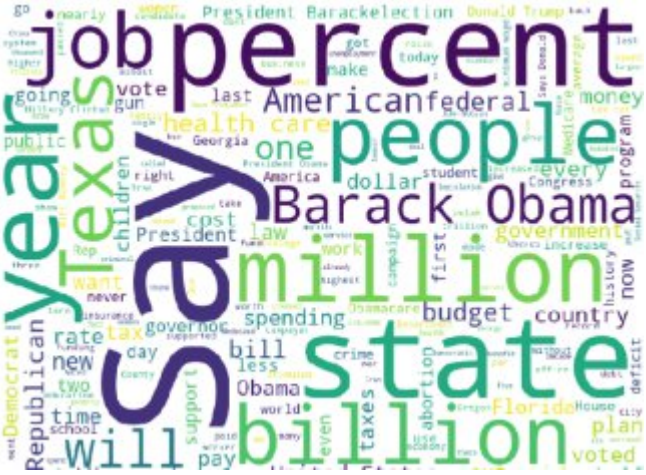

Figure #: WordCloud of topmost statements

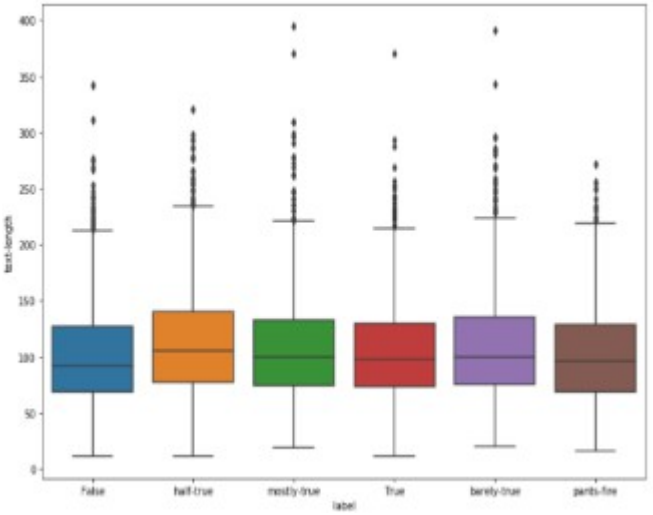

Figure #: Violin Plot for label distribution

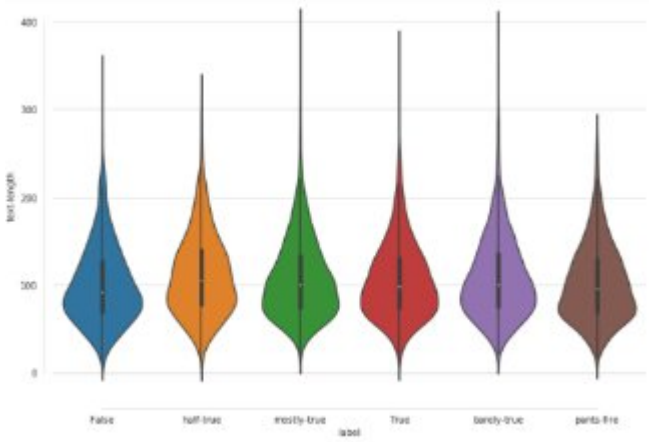

Figure #: Violin Plot for label vs text length

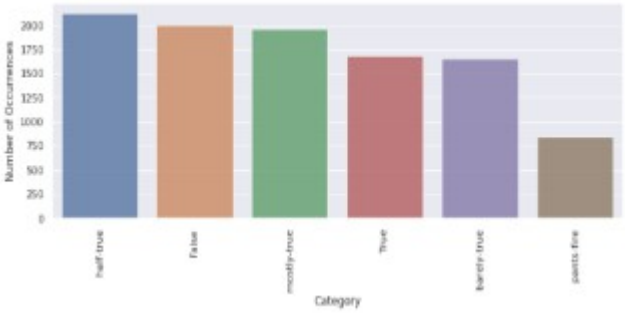

Figure #: Number of occurrences vs category

3. *Speaker Sentiment Analysis*

Considering the best practice of Fake News detection as discussed in the Fake News article, the Speaker sentiment analysis and statements analysis was performed in order to assess the credibility of the given news. Sentiment Analysis is a linguistic analysis technique that identifies opinion in a piece of text. It is used to verify the credibility of the speaker for a particular statement. The total contribution of words from each speaker was identified and out of those total contributions, the positive and negative words distribution was also analyzed.

As part of data enrichment, a separate column 'speaker_sentiment_score' was added to the dataset. Below is the pie chart for positive words distribution in the dataset.

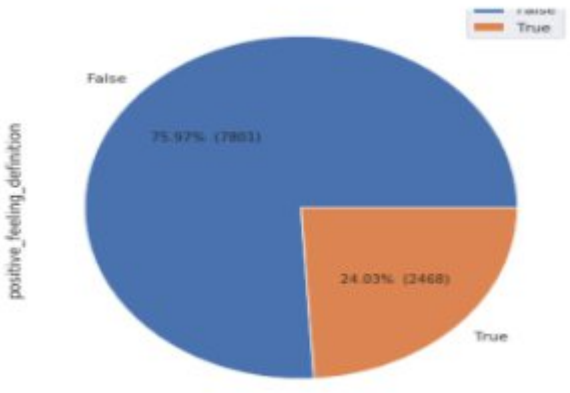

Fig – Pie chart for positive words distribution

4. *Word2Vec*

Word2Vec id a model that provides the direct access to vector representations of word, which can help achieve decent performance across a variety of tasks machines are not well at. There are two versions of Doc2Vec: skip-gram and CBOW.

skip-gram attempts to predict the immediate neighbors of a given word. It looks one word ahead and one behind, or two ahead, or some other variation. We use the word we want to model as our input X, and the surrounding words as target output Y. Once we can predict the surrounding words with a fair degree of accuracy, we remove the output layer and use the hidden layer to get our word vectors.

The Word2Vec model is implemented in order to create the feature vector – document term matrix using the count vectorizer technique. The vocabulary is generated using the glove.6B.50d dictionary for the purpose of generating the word embeddings on which the classification algorithms are implemented.

5. *Doc2Vec*

Doc2vec is an NLP tool for representing documents as a vector and is a generalizing of the word2vec method. The process in generating the Doc2Vec model involves instantiating the Doc2Vec model – Distributed Bag of Words (DBOW). Similar to the Word2Vec approach, the Doc2Vec has two models – Distributed Memory (DM) and Distributed Bag of Words (DBOW).

DBOW is the doc2vec model analogous to Skip-gram model in word2vec. The paragraph vectors are obtained by training a neural network on the task of

predicting a probability distribution of words in a paragraph given a randomly-sampled word from the paragraph.

6. *Bag of Words*

The Bag of words model in NLP is a way to represent text data for Machine Learning algorithm and it is easy to understand and implement. Bag of Words model is basically a process of extracting features from the text for use in machine learning algorithms.

In this approach, the tokenized words and the frequency of each token is calculated. For eg, the frequency of words for the given sentence might look like below:
"it" = 1
"the" = 1
"worst" = 0

In this approach, each word or a token is called as an n-gram. The dictionary is created from the processed documents containing the number of times a particular word appears in the dataset which is called as a corpus. Gensim Filter extremes is applied that filter out the tokens which appear in less than 15 documents or more than 0.5 documents.

7. *TF-IDF*

TF-IDF stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

TF-IDF Vectorizer converts a collection of raw documents to a matrix of TF-IDF features. The approach was done to see if the accuracy scores improved by bringing context to the mix and we did see significant improvements in accuracy

Out of all the models fitted, we would take 2 best performing model. We would call them candidate models from the confusion matrix, we can see that random forest and logistic regression are best performing in terms of precision and recall.

Classification Report

```
[ ] print(classification_report(DataPrep.test_news['Label'], predicted_rf))
```

	precision	recall	f1-score	support
False	0.65	0.42	0.51	1169
True	0.62	0.81	0.70	1382
accuracy			0.63	2551
macro avg	0.63	0.61	0.61	2551
weighted avg	0.63	0.63	0.61	2551

Fig – Classification Report for Random Forest

8. *LDA Topic Modeling*

Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. LDA (Latent Dirichlet Allocation) is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

The LDA is applied on the Liar Liar dataset from Politifact which involved cleaning the dataset by following the NLP steps followed by applying LDA model on the dataset to find N number of topics. The Latent Dirichlet Allocation topic modeling is performed using both Bag of words and TF-IDF models. The results achieved from applying LDA with both the approaches was the topmost topics in the document along with the weight or importance of each word in that particular topic.

PyLDAvis is a python library which is used to visualize the top N topics distribution in the dataset. It also plots the keywords which are prevalent in a particular topic. After observing the distance between the two topics based on the radius, we can predict the similarity between the given documents. Below is the visualization:

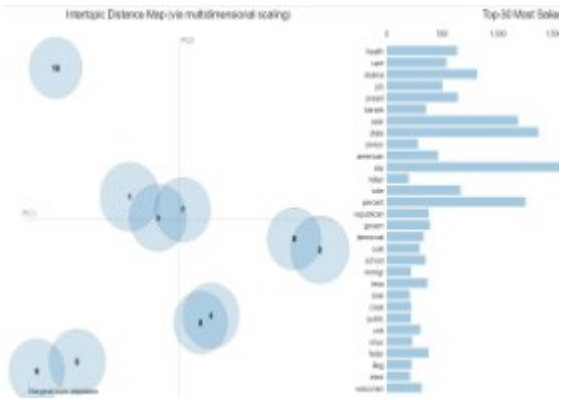

Fig – Pyldavis visualization

- **What did not work?**

Enriching the "Liar Liar" base dataset with the outside data sources for implementation of Credibility and Fact Checks factor did not work as the dataset columns was not compatible with the existing dataset.

- **What worked and Alternate Approach**

After applying the Doc2Vec model, the accuracy achieved was too low so data enrichment was performed using the glove.6B.50d dataset/dictionary and then, the six labels in the dataset was reduced to two labels – True and False for the sake of simplicity. The unwanted columns and stop words were removed from the datasets as part of Data cleaning and preprocessing as it did not have meaningful contribution to the data. I employed the following steps:

1. Speaker sentiment analysis contributing to fake news detection
    a. Identified the contribution of words from individual speaker
    b. Added the statement_sentiment_score to the dataset as part of data enrichment
    c. Visualized the total number of positive and negative words contribution from the speaker to assess the credibility of the provided news.
2. The dataset was enriched using the glove.6B.5d dataset/dictionary as the first enrichment.
3. After distillation, the labels in the dataset was reduced to two labels representing the news to be fake or non-fake and a separate column was added to the dataset as part of second enrichment.
4. Train the classification algorithms on the Word2Vec model. Once the model is properly trained, it is pickled and the given news is tested on that pickled model for its truthfulness or fakeness.
5. The Liar Liar dataset was initially amalgamated with the glove.6B.50d dictionary/dataset to generate the word embeddings and second amalgamation was performed with the Kaggle Fake News dataset.

IV. CONCLUSION

What we did as a team?

As a team, we decided on the importance of the factors presented in this paper. We brainstormed on the general pre-processing techniques we did want to use. We also had common visualization methods and similar techniques for evaluating the classification model accuracy. Each of us enriched the dataset with individual features and persisted it in a csv file. Each feature vector is persisted on csv (which is distilled with LDA, sentiment scores). 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 **0.19 * (Credibility Fact Checks) + 0.31 * (Controversy Score) + 0.30 * (Context) + 0.2 * (Source Credibility)**. The final model that we built is a variation of the stack ensemble technique. Stacked generalization is an ensemble method where the models are combined using another machine learning algorithm. The basic idea is to train machine learning algorithms with training dataset and then generate a new dataset with these models. Then this new dataset is used as input for the combined machine learning algorithm. The combined model is then used to predict the fakeness in the corpus. We as a team were able to achieve an accuracy of 57% using the various supervised learning techniques specified in this paper.

REFERENCES

A. SOURCE CREDIBILITY

[1] Data topic modelling using nltk library: https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

[2] https://www.kaggle.com/vukglisovic/classification-combining-lda-and-word2vec

[3] https://github.com/clips/news-audit/blob/master/SensationalismClassifier/SensationalismClassifier.py

[4] https://rare-technologies.com/word2vec-tutorial/

[5] https://www.kaggle.com/currie32/predicting-similarity-tfidfvectorizer-doc2vec

[6] https://www.kaggle.com/sgunjan05/document-clustering-using-doc2vec-word2vec

[7] http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/

[8] https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[9] https://www.kaggle.com/anucool007/multi-class-text-classification-bag-of-words/notebook

[10] https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne
[11] https://github.com/KnowledgeLab/doc2vec/blob/master/visualization.py
[12] https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da
[13] https://machinelearningmastery.com/k-fold-cross-validation/
[14] https://www.datacamp.com/community/tutorials/scikit-learn-fake-news
[15] https://snap.stanford.edu/mis2/files/MIS2_paper_17.pdf https://towardsdatascience.com/ranking-news-bias-in-python-e9bb5d1ba93f
[16] https://mediabiasfactcheck.com/methodology/
[17] http://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.W_zq9ehKiUl
[18] https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa

B. CONTROVERSY

[19] https://msuweb.montclair.edu/~feldmana/publications/paper_64.pdf
[20] https://fzr72725.github.io/2018/01/14/genism-guide.html
[21] https://www.ijcai.org/proceedings/2017/0583.pdf
[22] https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e
[23] http://maroo.cs.umass.edu/getpdf.php?id=1118
[24] http://ijarcsse.com/Before_August_2017/docs/papers/Volume_7/4_April2017/V7I4-0124.pdf
[25] http://aclweb.org/anthology/N18-1171
[26] https://www.onlinejournal.in/IJIRV3I2/230.pdf
[27] https://petsymposium.org/2017/papers/issue1/paper06-2017-1-source.pdf
[28] https://nlp.stanford.edu/pubs/prabhakaran2016wiki.pdf
[29] https://dataskunkworks.com/2018/06/06/extracting-topics-from-11000-newsgroups-posts-with-python-gensim-and-lda/

C. CONTEXT

[30] https://arxiv.org/pdf/1708.01967.pdf
[31] https://towardsdatascience.com/2-latent-methods-for-dimension-reduction-and-topic-modeling-20ff6d7d547
[32] https://www.datacamp.com/community/tutorials/lda2vec-topic-model
[33] http://www.scikit-yb.org/en/latest/api/text/freqdist.html
[34] https://machinelearningmastery.com/k-fold-cross-validation/
[35] https://www.kaggle.com/nhrade/text-classification-using-word-embeddings/notebook
[36] https://datascience.stackexchange.com/questions/19160/why-word2vec-performs-much-worst-than-both-countvectorizer-and-tfidfvectorizer
[37] https://www.groundai.com/project/liar-liar-pants-on-fire-a-new-benchmark-dataset-for-fake-news-detection/

D. CREDIBILITY / FACT CHECKS

[38] https://arxiv.org/pdf/1708.01967.pdf
[39] https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
[40] https://www.kaggle.com/mrisdal/fake-news
[41] https://www.politifact.com/
[42] https://www.kaggle.com/watts2/glove6b50dtxt#glove.6B.50d.txt
[43] https://towardsdatascience.com/creating-word-clouds-with-python-f2077c8de5cc
[44] https://stackoverflow.com/questions/31175140/get-a-classification-report-stating-the-class-wise-precision-and-recall-for-mult
[45] https://drive.google.com/drive/folders/1IVl4Qt92LZwvMlnJGRcEZVKFdpawhdz
[46] https://github.com/anuksebastian/AlternusVera/blob/master/AnuSebastian_012496276_PoliticalAffiliation_SocialMedia/AlternusVera_012496276_political_affiliation_AnuSebastian.ipynb
[47] https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
[48] https://medium.com/@bsangramsing/building-a-natural-language-processing-pipeline-9d93fa8ebdfb

[49] https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne

[50] https://machinelearningmastery.com/develop-word-embeddings-python-gensim/

[51] https://towardsdatascience.com/using-word2vec-to-analyze-news-headlines-and-predict-article-success-cdeda5f14751

[52] https://medium.com/@sherryqixuan/topic-modeling-and-pyldavis-visualization-86a543e21f58

[53] https://towardsdatascience.com/the-theory-you-need-to-know-before-you-start-an-nlp-project-1890f5bbb793