Alternus Vera

Anushri Srinath Aithal
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Political Affiliation
anushri.srinathaithal@sjsu.edu

Harini Balakrishnan
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Sensationalism
harini.balakrishnan@sjsu.edu

Sunder Thyagarajan
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Context, Clickbait
sunder.thyagarajan@sjsu.edu

Ravi Katta
Dept. Software Engineering
San Jose State University
San Jose, CA, USA
Contribution: Context, Clickbait
ravi.katta@sjsu.edu

A novel approach to detect Fake News using NLP Distillation process. 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. 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 (0.6*Clickbait) + (0.1 *Sensationalism) + (0.2*Political Affiliation Feature) + (0.1*Context)

Keywords—Distillation, Fake News, Political Affiliation, LDA, Sensationalism, Sentiment, SenticNet5, NLTK, Tokenizer, Wordcloud, t-SNE, Count Vector, Doc2Vec, TF-IDF, Cosine Similarity, Word2Vec, lda2vec

I. INTRODUCTION

A. Political Affiliation

Media influences our everyday decisions. Our lives are constantly driven by things we see around, hear and believe. Social media and news being the biggest influence, we appreciate news that agrees with our prior believes. Having said this, with the new-found media outlets like Facebook, Twitter, the pressure to be on top has increased for journalists and media corporations. Studies have proven that, if the headline or cover photo does not catch the reader's attention they tend to scroll past the article. This has led media to incorporate lot of bias in the articles. Even though this ensures more public engagement, it can lead to the widespread of incorrect information.

The US politics is dominated by two ideologies: liberal and conservative. Both these parties have their own opinion on issues related to global warming, gay rights, foreign policies and immigration. Author's political bias would result in influencing the readers. In US 2016 elections, it is noticed that the agenda of the political parties are obviously biased over the different issues, but the bias in the different forms of public communication like news media, journals, news channels are at times quite difficult to comprehend from a cursory look. According to Felix Biessmann [22], it is observed that a positive sentiment of the political bias always tends to favor readers and result in communicating incorrect information. Considering the above, we decided to consider political affiliation as one of the primary factors in classifying the news as fake or not.

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. Sensationalism

Sensationalism is a major type of editorial bias that is existing in the current mass media in the form of events, topics and news stories that are overhyped to popularize the media and attract the reader to read in detail, share, comment and like the news in social media. Sensationalism has been a serious treat and one of the major features to contribute to fake news. This concern has been there ever since 20th century. It creates a biased opinion or impression on the events and causes the people to be manipulated to the real true story. It is, in general, considered an insignificant influence on overall society. The news story topics are trivial or contrary to the standard professional writers and real journalism. There are several tactics that are related to the sensationalism feature. They are the articles that are being very emotional, controversial, informative but intentionally omitting real facts, being loud to seek attention and expressive. In each set of news articles, using the significant misrepresented stories can be identified using the above tactics. The content that has most insignificant and subjects that are irrelevant can be identified.

This paper focuses on classifying the liar and liar dataset based on text sentiment analysis. We aim to train the document model with sentiment metrics which can be used in future to predict the test data or future document's sentiment analysis. The words are made up of alphabets. But when considering sensationalism, 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 sensationalism. Each word has sentiment polarity that need to be considered to predict sensationalism.

Sentiment Analysis is one of the most important factors in text analysis and many researches like [30] and [31] has promising proofs that sentiment analysis influences the prediction of a text being impactful. One of the main issues with text analysis is to extract the emotions inside the document and how it is used to detect fake positive news or fake negative news from a document. The sentiment feature can be classified into two types positive and negative. This

paper focuses on classifying the liar and liar dataset based on text sentiment analysis. We aim to train the document model with sentiment metrics which can be used in future to predict the test data or future document's sentiment analysis. The polarity and intensity of the document plays major role in sentiment analysis. In this study, we analyses 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), Random Forest (RF) and Stochastic Gradient Descent (SGD) for sentiment classification of the liar and liar training and test dataset.

The analysis measured with results of our machine learning experiments show that the Linear SVM algorithm accuracy is more than other algorithms. Which also confirms the reference paper [31]. The maximum accuracy is 95% using TF-IDF vectorization. The vector is then compared with the sentiment polarity corpus extracted and enriched from SenticNet 5. 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 sentiment analysis and I saved a final dataset of training and test documents with the sentiment features like polarity, intensity, sentiment vectors and cosine similarity score. Overall the contribution of the sentiment analysis includes fake news dataset which is identified with sentiment metrics like negative and positive which helps to classify the content of the article and identify the major sentiment polarity and intensity score. In addition to classifying, we have also performed vectorization and comparison of corpus TF-IDF vectors with the sentiment vector to analyses 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.

After classifying the news articles based on Sentiment analysis, I performed TF-IDF Vectorization and Cosine Similarity on the Sensational words corpus that consists of 1400+ words based on 12 different emotional categories. The news article tokens are converted as a single corpus to do cosine similarity. Thus, the article that has more similarity score is sensational. I classified the article with binary classification with 1 for sensational and 0 for non-sensational. Cosine similarity helps to find highly similar news to have more prediction rate for sensational news. Addressing the sensational news is the baseline to predict if the news is fake or not.

Based on the sensational classification I performed Doc2Vec model. Doc2Vec model helps to do fixed size vector which is used in machine learning classification algorithms. It uses distributed bag-of-words that helps to paragraph the vectors by training the neural network on the predicted probability distribution. Thus, the news articles are vectorized based on the sensationalism and classified. I performed two major classification models: Linear Support Vector Machine (SVM) and Logistic Regression (LR).

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. Clickbait

Clickbait (headlines) make use of misleading titles that hide critical information from or exaggerate the content on the landing target pages to entice clicks. As clickbait's often use eye-catching wording to attract viewers, target contents are often of low quality. Clickbait's are especially widespread on social media such as Twitter, adversely impacting user experience by causing immense dissatisfaction. Hence, it has become increasingly important to put forward a widely applicable approach to identify and detect clickbait's

A lot of small business owners and marketing agencies like to use clickbait because it's a super-fast way of generating web traffic — and it can generate results. Industry-specific listicles can save users a lot of time and energy attempting to aggregate information for themselves. The subsequent increase in traffic this content creates can improve a site's search engine presence phenomenally. That's a win-win.

E. Spam

It is not possible to know who exactly is sending advertisements or email or information to millions and millions of people, to this kind of information, no matter what the fact of the news is, this kind of information is being spread and exploited by different organizations. When we talk about emails, millions of mail boxes are cluttered with spam. This kind of spam can impact the regular mails, as there is a chance of deleting the actual info accidentally. Also, there is a spam in pornography that should not be exposed to children.

II. Data Set, Scraping And Enrichment

A. Political Affiliation

"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.

As the first step in feature engineering for political affiliation, we begin by tagging the textual data with the respective party they are affiliated to. Several approaches are suggested to perform the same. In one of the methods, the Stanford Political Dictionary is used. This data is scraped from the Harvard Dataverse website. This data consists of 3373 words with attitudes defined on a scale of 1 to 3. The attitudes 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.

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 political affiliation. 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 political words and headline text) is used later for multi-class classification.

B. Sensationalism

The dataset has 3 CSV files: test, training and valid. Each file has 14 columns: the ID of the statement. the label, the statement, the subject(s)/news, the speaker, the speaker's job title, the state info, the party affiliation. Column 9-13 are the total credit history counts, including the current statement like barely true counts, false counts, half true counts, mostly true counts, pants on fire counts. The final column is the context (venue / location of the speech or statement).

SenticNet is a framework that provides set of tools and techniques for sentiment analysis with commonsense reasoning, linguistics, psychology, and machine learning. SenticNet is used to compute the sentiment metrics with a statistical approach to help sentiment analysis by focusing on a semantic-preserving representation of natural language concepts and on sentence structure. The data is scraped and saved it as a text file (senticnet5.txt) which consist of 100,000 words with polarity and intensity rate ranging from-1 to +1. Minus one (-1) is extreme negativity, and plus one (+1) is extreme positivity. The knowledge base is free. It can be downloaded as XML file too. There are three columns: word/ token, polarity and intensity in the senticnet.txt which is scraped.

Using this SenticNet data we enriched three columns in the training and test dataset. First column is the sentiment score for the entire document based on the aggregation of the intensity of each word in the document. Second column is the sentiment polarity based on the aggregation of the polarity of each word in the document. This column is the vector of the sentiment intensity of each word in the document. Based on the sentiment intensity of the entire document, vectorization is performed, and various classification models are compared for best results.

I performed using the NLTK library known as Sentiment Intensity Analyzer. The Valence Aware Dictionary and sentiment Reasoner has created a package that performs sentiment analysis using the polarity-based, where pieces of texts are classified as either positive or negative, or valence-based, where the intensity of the sentiment is considered. For

example, the words ‘good’ and ‘excellent’ would be treated the same in a polarity-based approach, whereas ‘excellent’ would be treated as more positive than ‘good’ in a valence-based approach. This is based on the lexicons of sentiment - related word. I considered four parameters “positive”,” negative”, “neutral” and “compound”. The metrics for each parameter is a score ranging from -1 to +1. I enriched two columns using this library. Sentiment vector made of these 4 parameters as a column for each document and Vader polarity value for each document which is computed by considering the maximum “negative” or “positive” or “neutral” metric of the document. For instance: if a news has Vader metrics higher for positive, its Vader polarity is taken as 1. 0 for negative and -1 for neutral.

Sensational dictionary was not readily available online or in any library. In order to classify an article to be sensational or not was a complicated task. The Persuasive Revolution website performed a study on various news channels like CNN-News, CNN, Sky news, ABC news and created a list of 1400+ sensational words that have been used by these channels to popularize their news. I scraped these words, preprocessed and converted them into a corpus. Which is then compared with the training news articles. By comparing these two corpora I was able to predict the similarity score which is greater for sensational articles and less for non-sensational articles.

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. Clickbait

To understand the problem better, we explored a a dataset from the clickbait challenge 2017 (clickbait-challenge.com) comprising of over 21,000 headlines/titles, each of which is annotated by at least five judgments from crowdsourcing on how clickbait it is. We attempt to build an effective computational clickbait detection model on this dataset. We first considered a total of 331 features, filtered out many features to avoid overfitting and improve the running time of learning, and eventually selected the 60 most important features for our final model.

E. Spam

There are different ways of fighting the spam one is by taking social measures, one of them is not responding to spam, other is not publishing mails on public webpages. Apart from this there are different approaches for handling this like machine learning and knowledge learning. In knowledge learning we talk about setting different set of rules. Machine learning is the area we are interested in. with machine learning technique we take set of pre-classified documents, then we make use of algorithm.

The goals of this machine learning algorithms are to get a spam filter which is also called as decision function which will tell whether the given input is spam or legitimate. When it comes to popular algorithms, naive Bayesian classifier, k-NN classifier, the neural network classifier and the support vector machine classifier are some of them.

III. IMPLEMENTATION DETAILS

A. Political Affiliation

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

  1. *Pre-processing*

     Headline text is cleaned using several NLP cleaning methodologies. The text cleaning process begins with converting all the textual data to lower case, followed by removal of stop words using NLTK library. Spell check is performed using Word2Vec. The Google’s word lists are used as the corpus for the Word2Vec model. Peter Norvig's spell checker is used to identify the correctness and replace the misspelt words in the headline 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. Post stemming the headline text length is reduced by removing words whose length is less than 3. Lemmatization is also performed on verbs. This pre-processing is performed on both the train and test data.

2. *Visualization*

The pre-processed text is visualized to identify the average text length. This helps in deciding the vector size without much data loss. The text 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.

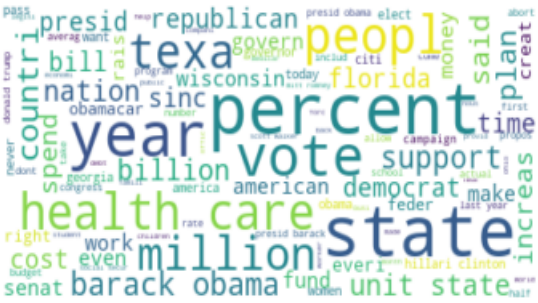

Fig. Word Cloud Visualization

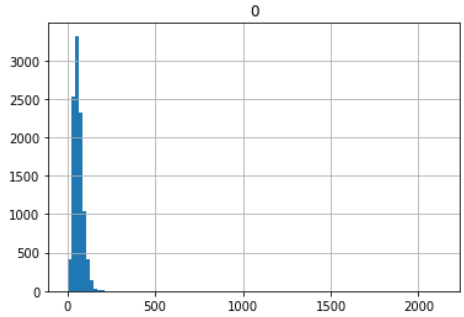

Fig. Word Length Distribution in Train Data Set

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 political dictionary 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.

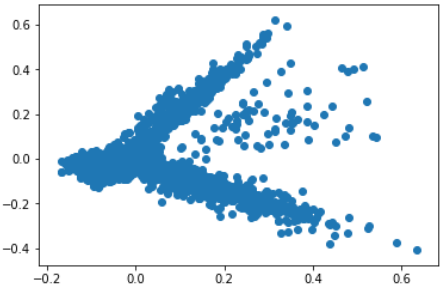

Fig. Count Vector Visualization using PCA

4. *Enrichment by Tagging*

Through the count vector method, the text was classified as fake or not. However, the end goal is to tag each of the document with a political attitude as present in the dictionary. Steps involved in tagging the documents for political attitude

I. Group the list of political dictionary words into Positive, Negative, Strong, Weak, Active and Passive based on the score present in the dictionary.
II. For each headline text, find the word overlap between each group of words.
III. Calculate the polarity such that, the highest weight for an attitude in the overlap set represents the attitude of the headline.
IV. Tag the documents and enrich by adding a new feature to the dataset.

5. *TF-IDF*

TF-IDF is another technique to vectorize text using word frequency. TF-IDF works by penalizing the common words by assigning them lower weights while giving importance to words like Messi in a document. In this approach, the entire dictionary of words is treated as a single document and a TF-IDF vector of length 500 is created. Each headline document is then vectorized using TF-IDF to represent as a vector of size 500. The cosine similarity between the vectors is calculated. The data set is enriched using this cosine similarity which indicates how politically affiliated the headline is.

The TF-IDF approach was best suited at this point as it provides a similarity score that determines how close to political bias the article is. The cosine similarity value ranges from -1 to 1

where 1 indicates the two documents are very closely related to each other.

6. *Word2Vec*

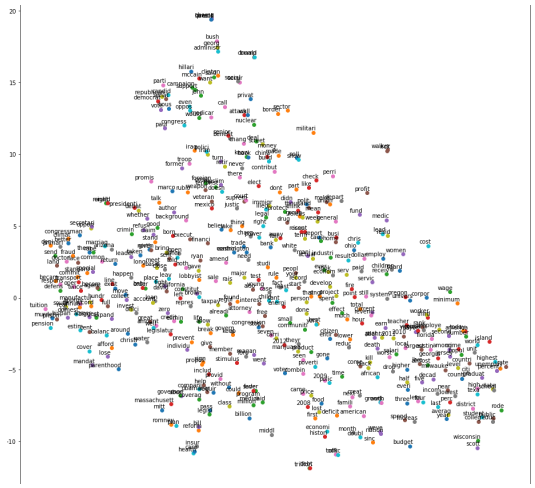

Fig. Word2Vec Visualization using t-SNE

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 political 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 political polarity vector size.
VI. Normalize the vector derived using Word2Vec model
VII. Find the similarity between the weighted political vector and the Word2Vec of the headline.
VIII. This similarity can be used to tag the text as Positive, Negative, Strong, Weak, Active or Passive.

- **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.

Tagging the documents based on the Stanford political dictionary did not yield good results. 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**

1. *TF-IDF with custom built dictionary*

As an alternate approach for feature engineering political affiliation, a dictionary is created by analyzing the "Liar Liar" data set. The party affiliation tag distribution is analyzed. The data set has a high concentration of republican and democratic party headlines compared to others. The party affiliation tags are encoded and grouped into four sub categories for further processing and to reduce the complexity of multi-label classification. The party affiliation considered are republican, democrat, independent and all the others are grouped into a single category named other. The party affiliation distribution can be visualized as below.

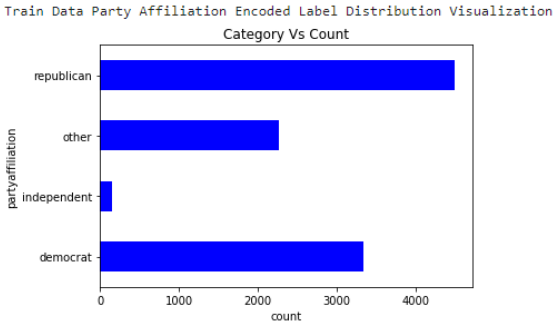

Fig. Party Affiliation Distribution

As a next step, for each of these party affiliations tag the top occurring words are picked. This forms the new political affiliation dictionary against which TF-IDF vectorization is performed. A manual approach to TF-IDF is done by taking the frequency matrix with respect to the words in the political dictionary. A classifier is built to predict the party affiliation tags using the TF-IDF model. This achieves an accuracy of 43%.

2. *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%.

3. 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

I have chosen topic score, which is a probability of how much percentage the headline is close to the topic instead of the topic number. The topic numbers are in the range of 1 to 10. If I use the topic range, then the classifier adds more weightage to topic 10 even though this is not the right form of ranking. Topic score is a value between 0 and 1 which is more appropriate.

Ranking is performed by assigning a speaker importance to top frequently spoken leader in the data set. Rank is assigned as a mathematical logarithmic value of the frequency of occurrence of the speaker. As there are more than 10 speakers, all the other speakers are awarded the least score of 1.

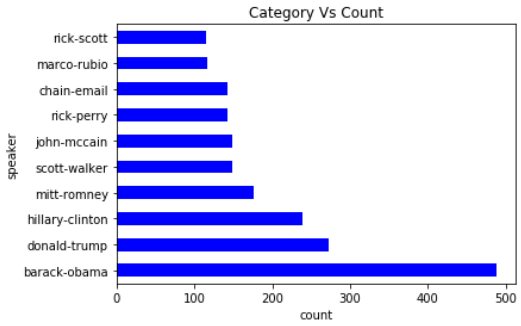

Fig. Speaker Importance

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 political affiliation feature.
II. Vector Embedding: In this technique, the doc2vec model is embedded with sentiment score, topic score and rank value.

From the above approach of distillation, the accuracy achieved was 56% which is evidently more (without distillation 37%) than what was achieved without distillation process.

B. Sensationalism

- **What did I try?**

  - *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. In addition to this, we also performed spell check using Google News vector. The Google News vector consist of 3 million words based on 300 features that is vectorized and compared as 1.5 GB data. By comparing and detecting the correct spelling for each word in the news article does take more than 5 minutes. But by doing this the distillation process is improvised.

  - *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 in to al 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's 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 both training dataset and test dataset.

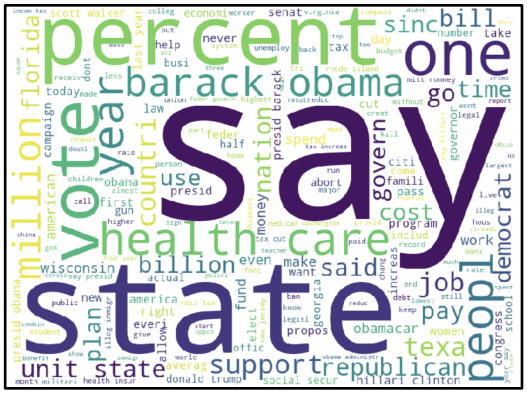

Fig. Word Cloud Visualization of Train Dataset

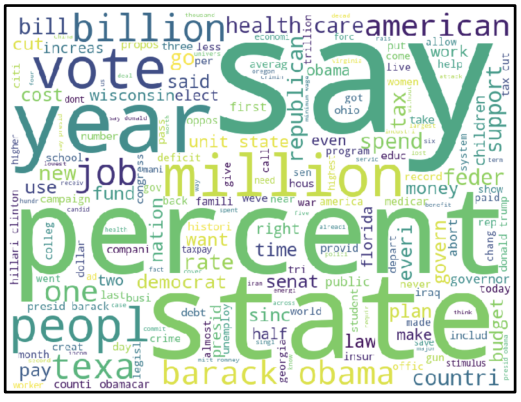

Fig. Word Cloud Visualization of Test Dataset

  - *Computation of Sentiment polarity and Intensity*

SenticNet is an initiative of MIT Media Laboratory in 2009. It is designed based on bag-of-concepts model. It is highly used for emotional aware intelligent applications. The SenticNet5 provides over 100,000 words based on the commonsense concepts using recurrent neural

networks to infer primitives by lexical substitution. The dictionary consists of three essential columns: the word token, polarity and sentiment intensity.

1. For each article the intensity of each word is computed and formed a vector for sentiment intensity which can be used for multi-class classification later.
2. Calculate the polarity such that, if the document had maximum words based on positive sentiment, then he document is rated as positive. This a column of 0 and 1 is used for TF-IDF vectorization.
3. Third column enriched using this dictionary is sentiment score which is the aggregation of sentiment intensity of all words in an article/document.

Senticnet Vocab Size: 39891

	Token	Polarity	Intensity
0	abandon	negative	-0.84
1	abandoned	negative	-0.85
2	abandonment	negative	-0.82
3	abase	negative	-0.90
4	abasement	negative	-0.90
5	abash	negative	-0.77
6	abashed	negative	-0.92
7	abashment	negative	-0.76
8	abasia	negative	-0.67
9	abate	negative	-0.86

Fig. Sentic Net5 Data Enrichment

Sentiment Intensity Analyzer is a new tool provided by NLTK to implement features based on sentiment extracted from the document. It provides four sentiment values for each document. The vector consists of negative score, neutral score, positive score, and compound metrics. Below is a representation of sentiment intensity of one document. The document is installing cleaned and tokenized. So, it consists of only valid words.

```
say anni list polit group support third trimest abort demand
{'neg': 0.123, 'neu': 0.656, 'pos': 0.221, 'compound': 0.296}
```

Fig. Sentiment Intensity Analysis Vector

- *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. Performed LDA predicting the top 10 topics and classifies the news articles based on these top 10 topics. This process also helped to categorize the news articles under top 10 topic. Compared the frequency of the news article words with their respective topic and computed similarity score.

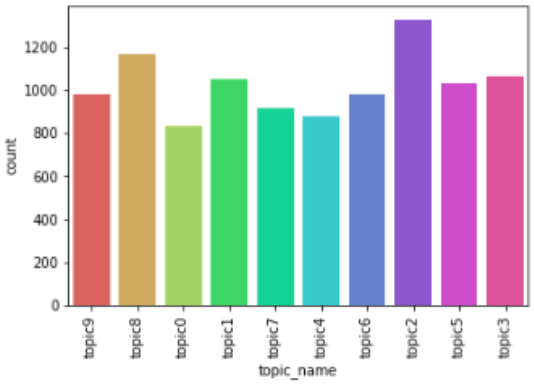

Fig. LDA Top 10 Topic distribution

- *Counter Vectorization*

Using various feature extraction of text libraries like CountVectorizer to 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), Logistic Regression (LR), Linear Support Vector Machine (SVM), Random Forest (RF) and Stochastic Gradient Descent (SGD). Out of these models the LinearSVC predicted the document sentiment Vectorized using CountVectorizer () with 95% accuracy. All the model's performance is validated using the Classification Model Evaluation Technique KFold. It has a single parameter called k that refers to the number of groups the given dataset is split into.

- *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 3. In order to get better results, each word is Vectorized in the article. Then classify the vectored articles against the sensational 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 97% by Linear SVM model.

- *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 sensationalism vector formed from the sensational word's corpus can be compared together using the cosine similarity library. The similarity score helps to predict the sensationalism of the document. High similarity score means the document is highly likely to be sensational. Because the words that I chose form the dictionary are highly sensational word. Hence, if the similarity of a document is high with this dictionary vector then the document is classified as sensational.

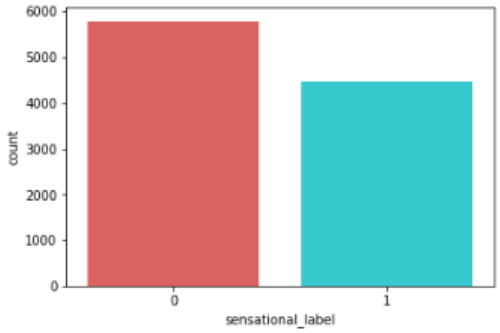

Fig. Sensational classification on Training data

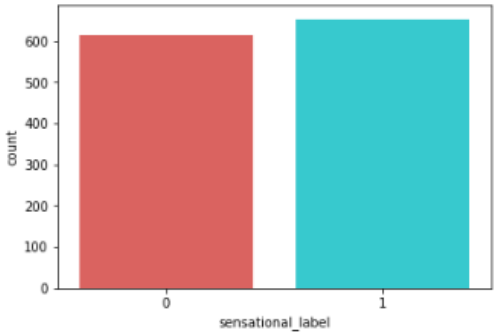

Fig. Sensational classification on Test data

- *Word2Vec and 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.

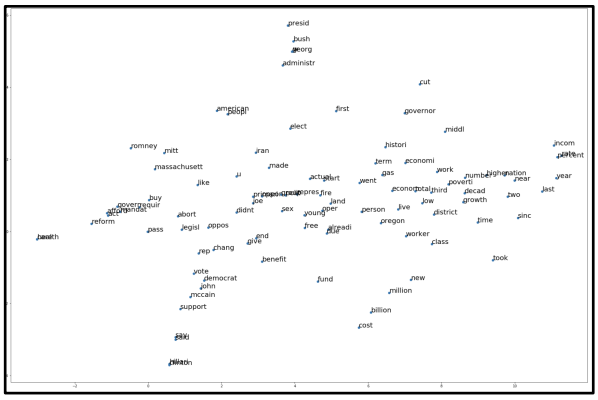

Fig. Visualize Word2Vec using TSNE

- *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 sensational classification are vectorized and then used for training and testing the text classification. Four supervised classification models were performed and the accuracy of a document with sensationalism classification is 56.4%. Here the vector is formed for the entire document. But CountVectorizer and 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 used two essential enrichment sources: SenticNet5 and Sentiment Intensity Analyzer. 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. Out of 5 classification models LinearSVC gave the highest prediction score of 95%.

Models	Accuracy
Naive Bayes	92.1%
Logistic Regression	94.8%
Linear SVM classifier	95%
Stochastic Gradient Descent	94.1%
Randome Forest Classifier	92.0%

Fig. Classification model results for Counter Vectorizer

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. Using the Sentiment Intensity Analyzer, the article is analyzed and then vectorized using the TF-IDF vectorizer. Then classified the document based on three vector parameters “positive”, “negative” and “neutral”. All the five supervised classification models are performed as earlier. When compared with Counter vectorization method the prediction score of linear SVM classifier increased from 95% to 97%.

Models	Accuracy
Naive Bayes	95.8%
Logistic Regression	95.8%
Linear SVM classifier	96.7%
Stochastic Gradient Descent	95.8%
Randome Forest Classifier	96.1%

Fig. Classification model results for TF-IDF Vectorizer

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 4 classification models. As we took the entire document instead of each word the accuracy of the model reduced to 91% which is less than the two earlier vectorization methods.

Models	Accuracy
Logistic Regression	91.1%
Linear SVM classifier	91.1%
Stochastic Gradient Descent	91.1%
Randome Forest Classifier	83.9%

Fig. Classification model results for Doc2Vec

I 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 gave more similarity score is classified as sensational.

- **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. There are two types of architecture options in Word2Vec: skip-gram (default) and CBOW (continuous bag of words). Most of time, skip-gram is little bit slower but has more accuracy than CBOW. CBOW is the method to predict one word by whole text; therefore, small set of data is more favorable. On the other hand, skip-gram is totally opposite to CBOW. With the target word, skip-gram is the method to predict the words found in the target words. The more data we have, the better it performs. As the architecture, there are two training algorithms for Word2Vec: Hierarchical SoftMax (default) and negative sampling. I used the default. 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.

- **Alternate Approach**

I couldn’t perform classification model as the dimension of the vector varies for each document. My solution is to try Doc2Vec which will provide a vector of fixed size for the entire document instead of each word in the entire corpus. In the word2vec architecture, the two algorithm names are “continuous bag of words” (CBOW) and “skip-gram” (SG); in the doc2vec architecture, the corresponding algorithms are “distributed memory” (DM) and “distributed bag of words” (DBOW). Based on this sensational classification all the five supervised classification models are performed as earlier. The accuracy for sensationalism classification is 56%.

I also performed Doc2Vec using the sentiment metrics as the label to form vectors based on “negative”,” positive”, “neutral” and “compound”. The dm defines the training algorithm. dm=1 means ‘distributed memory’ (PV-DM) and dm =0 means

'distributed bag of words' (PV-DBOW). Distributed Memory model preserves the word order in a document whereas Distributed Bag of words just uses the bag of words approach, which doesn't preserve any word order. I kept dm as 1. I formed vectors and trained the model. Performed 4 different classification models and predicted the sentiment on the test dataset. LinearSVC and Logistic Regression gave 91% accuracy.

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.

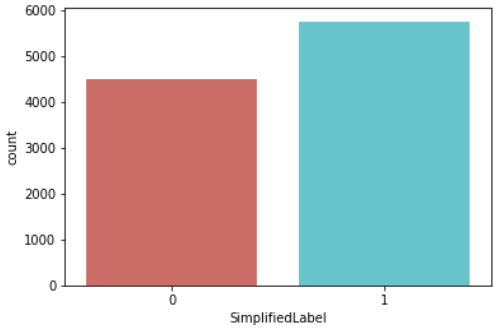

Fig. Distribution of preprocessed dataset

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

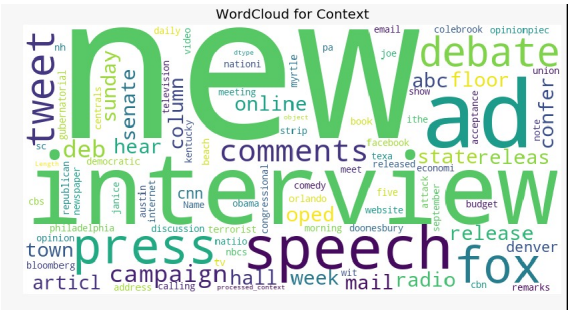

Fig. Wordcloud of context feature

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

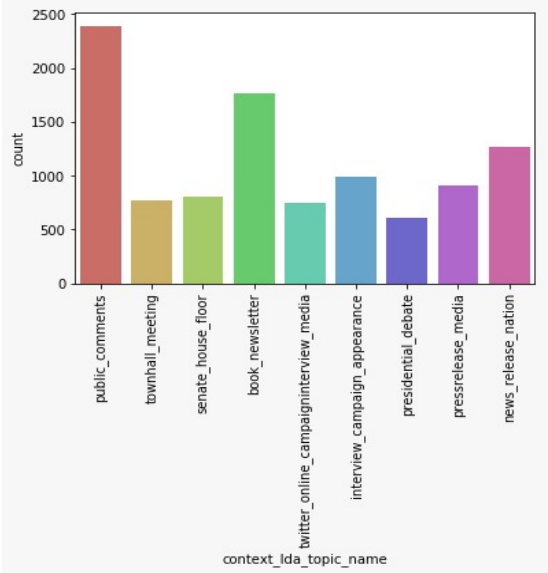

Fig. context LDA topic distribution

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 problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

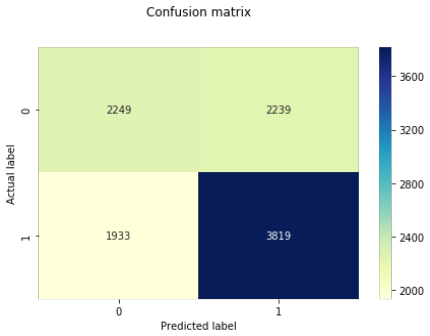

Fig. Sample K-Fold Confusion matrix

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	0.60	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.

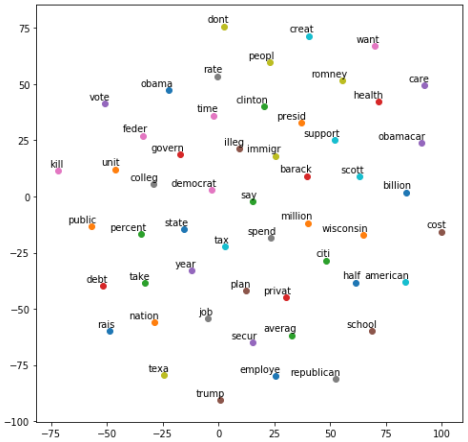

Fig - Visualization of lda2vec model using t-SNE

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 MeanEmbeddingVectorization

- **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

  III. 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

  IV. 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. CLICKBAIT

- **What did I try?**

The brief summary on the steps followed are

1. *Pre-processing*

As part of preprocessing I did tokenization of text using keras as from keras.preprocessing.text import Tokenizer also did normalization instead of eliminating the columns. Also collapsed all the white spaces into single spaces along with elimination of leading and trailing spaces, also converted the entire corpus to lowercase.

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.

As part of preprocessing deliberately decided to retain punctuations which includes question marks, exclamation, capital letters to identify clickbait ratio

2. *Visualization*

Visualized the data using word cloud library using the package , from wordcloud import WordCloud

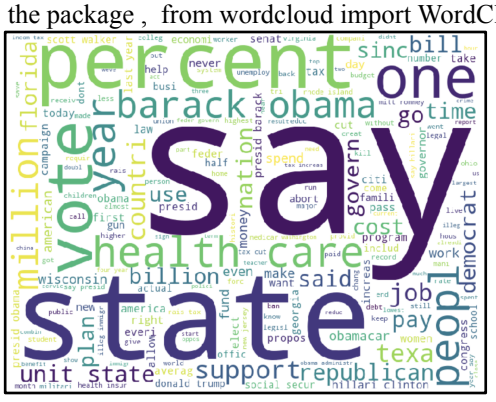

Fig - world cloud visualization

3. *Feature Extraction*

Exclamation is a key way to attract users and leads to increase in clickbait. In our example we

will check for availability of exclamation and add it to the dataset for identifying clickbait ratio

Question mark is a key way to sensationalize a news and attract users. In our example we will check for availability of question mark and add it to the dataset for identifying clickbait ratio

Caps ratio is another key feature we extract from preprocessed text. Usually users get attracted to capital letters and would increase clicks, in our case we are identifying the ratio of capital letters to lower letters which is expected to be low. We add this feature to the dataset for analyzing clickbait ratio

4. DOC2VEC

The main purpose of Doc2Vec is associating arbitrary documents with labels, so labels are required. Doc2vec is an extension of word2vec that learns to correlate labels and words, rather than words with other words.

The doc2vec models may be used in the following way: for training, a set of documents is required. A word vector W is generated for each word, and a document vector D is generated for each document. The model also trains weights for a SoftMax hidden layer. In the inference stage, a new document may be presented, and all weights are fixed to calculate the document vector.

We tag the documents using Gensim tagging, we will use headline as column to be tagged and clickbait as label. We will apply doc2vec on the tagged documents.

5. DISTILLATION

Distillation is a key step to enhance the feature vector. In this process we perform purification of text and determine tangible information. Key steps involve preprocessing of the headline, followed by

Sentiment analysis of the headline, will yield a sentiment compound score, which will be concatenated to the vector

Topic modeling, we performed LDA on headline text which resulted in top 10 topics, and that yielded us topic score, which will be concatenated to the vector. This LDA based topic score, we can perform using other algorithms such as DOC2VEC, BiLSTM, LDA2VEC

Ranking, we will rank the clickbait features based on the below rules

- caps ratio
- exclamation
- question mark
- text length

New vector formed will be part of my polynomial equation

- **What did not work?**

When including text length to evaluate clickbait score, we had fake news across the spectrum.

The dataset had limited clickbait news, even though this yielded high accuracy, the number of clickbait's in the dataset were limited compared to non-clickbait fake news

- **What worked and Alternate Approach**

We decided to drop text length from the list of features, as it did not contribute to the clickbait. We employed the following steps

1. Identify key features contributing to clickbait
    a. Question mark
    b. Exclamation
    c. Caps Ratio
2. We will enrich dataset with new labels as defined above
3. If any of the above is present, then in the headline we will mark it as clickbait
4. Identify clickbait ratio and determine whether the combination of above features led to clickbait

E. SPAM

- **What did I try?**

The brief summary on the steps followed are

1. Pre-processing

As part of preprocessing I did tokenization of text using keras as from keras. preprocessing.text import Tokenizer also did normalization instead of eliminating the columns. Also removed all the punctuation and collapsed all the white spaces into single spaces along with elimination leading and trailing spaces, also converted the entire corpse to lowercase

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.

2. *Visualization*

Visualized the data using word cloud library using the package, from wordcloud import Wordcloud

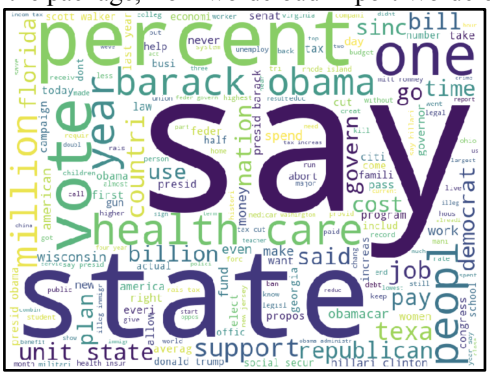

Fig - world cloud visualization

3. *Mean embedding Vectorizer*

Using word2vec we convert every word in a document to a vector, however if we need to construct a vector on the document without ending up with a vector of vectors or tensor, we apply mean embedding vectorizer, this is a common approach used to convert topics to vectors. In this scenario we will do a mean embedding vector of headline text to perform dot product against spam vectors

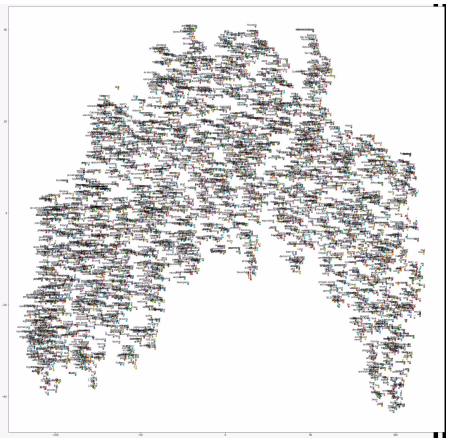

Fig - Visualization of word2vec model using t-SNE

- **What did not work?**

When exploring for a spam dataset, we encountered a lot of datasets that explained email-spam. Those spam dictionaries were not correctly reflecting the spam that we see in news articles.

When exploring other datasets, the context of the spam news was totally unrelated to politics news, which lead to bad vectorization and cosine similarity

- **What worked and Alternate Approach**

To solve the above I changed the dictionary by trying to use glove and used additive smoothing where we added a number say x for the numerator and then add the same number of x times over the probability. Still the spam dictionary needs to be refined.

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 (0.6*Clickbait) + (0.1 *Sensationalism) + (0.2*Political Affiliation Feature) + (0.1*Context). 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. Political Affiliation

1. https://www.kaggle.com/vukglisovic/classification-combining-lda-and-word2vec
2. https://rare-technologies.com/word2vec-tutorial/
3. https://www.kaggle.com/currie32/predicting-similarity-tfidfvectorizer-doc2vec
4. https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4
5. https://www.kaggle.com/sgunjan05/document-clustering-using-doc2vec-word2vec

6. https://github.com/rahul003/stance-detection/blob/master/src/featureExtractor.py
7. https://datascience.stackexchange.com/questions/8426/cosine-distance-1-in-scipy
8. https://www.kaggle.com/anucool007/multi-class-text-classification-bag-of-words/notebook
9. https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne
10. https://github.com/KnowledgeLab/doc2vec/blob/master/visualization.py
11. http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/
12. https://medium.com/@amarbudhiraja/understanding-document-embeddings-of-doc2vec-bfe7237a26da
13. https://machinelearningmastery.com/k-fold-cross-validation/
14. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
15. https://www.datacamp.com/community/tutorials/scikit-learn-fake-news
16. https://ntguardian.wordpress.com/2016/11/07/deceit-politics-analysis-politifact-data/
17. https://towardsdatascience.com/ranking-news-bias-in-python-e9bb5d1ba93f
18. https://mediabiasfactcheck.com/methodology/
19. https://medium.com/linalgo/predict-political-bias-using-python-b8575eedef13
20. https://github.com/kasramsh/Feature-Engineering/blob/master/Text/NSF%20Awards.ipynb
21. https://arxiv.org/pdf/1103.0398.pdf
22. https://arxiv.org/pdf/1608.02195.pdf
23. http://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.W_zq9ehKiUl
24. https://cs224d.stanford.edu/reports/MisraBasak.pdf
25. https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa

B. SENSATIONALISM

26. https://sentic.net/
27. https://sentic.net/
28. http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-withsocial-media-text.html
29. https://www.kaggle.com/kyen89/2-sentiment-analysis-word2vec
30. http://www.fakenewschallenge.org/
31. https://www.researchgate.net/publication/325973731_Detecting_Fake_Reviews_through_Sentiment_Analysis_Using_Machine_Learning_Techniques
32. http://web.stanford.edu/~mattm401/docs/2018-Golbeck-WebSci-FakeNewsVsSatire.pdf
33. https://www.researchgate.net/publication/325212005_Fake_News_vs_Satire_A_Dataset_and_Analysis
34. https://towardsdatascience.com/i-trained-fake-news-detection-ai-with-95-accuracy-and-almost-went-crazy-d10589aa57c
35. https://ec.europa.eu/jrc/communities/sites/jrccties/files/dewp_201802_digital_transformation_of_news_media_and_the_rise_of_fake_news_final_180418.pdf
36. https://www.researchgate.net/publication/304140282_Sensationalism_in_the_media_the_right_to_sell_or_the_right_to_tell
37. https://www.researchgate.net/publication/306022464_Sensationalism_in_News_Coverage_A_Comparative_Study_in_14_Television_Systems

C. CONTEXT

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

D. CLICKBAIT

46. https://get.simplymeasured.com/rs/135-YGJ-288/images/SM_StateOfSocial-2017.pdf
47. https://ieeexplore.ieee.org/document/7877426
48. https://www.researchgate.net/publication/320241720_Machine_Learning_Based_Detection_of_Clickbait_Posts_in_Social_Media
49. https://www.linkedin.com/pulse/identifying-clickbaits-using-machine-learning-abhishek-thakur/

E. SPAM

50. https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html
51. https://www.youtube.com/watch?v=PrkiRVcrxOs
52. https://www.booleanworld.com/building-spam-filter-using-machine-learning/
53. https://towardsdatascience.com/spam-detection-with-logistic-regression-23e3709e522