**Alternus Vera: A Fake News Detection Model**

|  |  |  |  |
| --- | --- | --- | --- |
| Golnaz Bidabadi  *Software Engineering*  *San Jose State University*  Santa Clara, USA  Golnaz.bidabadi@sjsu.edu | Puja Kawale  *Software Engineering*  *San Jose State University*  Santa Clara, USA  puja.kawale@sjsu.edu | Supreetha Ganapathi  *Software Engineering*  *San Jose State University*  Santa Clara, USA  supreetha.ganapathi@sjsu.edu | Swetha Chandrasekar  *Software Engineering*  *San Jose State University*  Santa Clara, USA  swetha.chandrasekar@sjsu.edu |
|  | Vijay Samuel Alexandar Vaseeharan  *Software Engineering*  *San Jose State University*  Santa Clara, USA  vijay.samuel@sjsu.edu | |  |

### ***Abstract - Fake news is one of the most critical societal concerns today, as it potentially plays a pervasive role in manipulating opinions and deceiving individuals and thus introduces a profound impact on society. The rise of social media as one of the main resources for news consumption as well as everyday use of digital devices by increasing number of people have contributed significantly to propagation of the fake news. Therefore, finding ways of detecting and assessing the truthfulness of news is one of the most-needed solutions to minimize the impact of deceptive news claim. This paper proposes a solution using natural language processing (NLP) techniques and Machine learning algorithms to help achieve maximum accuracy in fake news detection.***

**Keywords— fake news detection, veracity, Natural Language Processing, Social Media, Machine learning, Ensemble methods, sentiment, Latent Dirichlet allocation (LDA)**

1. Introduction

Today we live in a world where there are new terms that have been coined like “alternate facts” where in fact a fact is defined as a statement that holds true. Fake news has engulfed the world that we live in. Fake news is not only used to spread propaganda but it is also a term that is used to discredit sources that convey facts that a speaker doesn’t necessarily agree with.

1. FAKE NEWS DEFINITION

The first step in trying to detect fake news is to define what fake news actually is. Fake news is something that doesn’t agree with the facts. Facts can represent a certain historical occurrence, a scientific truth or a documented statement like the constitution or the law of the land. One of the biggest challenges in today’s world with so much information being circulated is how does one identify real news versus fake news. This paper tries to document an effective mechanism by which fake news can be discerned.

1. FAKE NEWS DETECTION

Fake news detection is defined as the prediction of the chances of a particular news article (news report, editorial, expose, etc.) being intentionally deceptive by Rubin, et al. (2015). The paper attempts to detect fake news in a methodological approach toward categorizing news in a spectrum of veracity, trying to measure the fakeness of news articles. It takes into account the body of the article, the headline (title) of the news as well as its sources (author[s], speaker[s] and publisher[s])

1. Methodology

The aim of the project is to assess the veracity of a particular news article. Considering Shu, et al. (2017) paper, one of the first steps that we need to take for detection of fake news is to identify the main factors that would be considered in the computation of the fake-news-likelihood scores. After identifying the most important factors, various NLP and machine learning techniques were implemented, and extensive analysis was performed to determine the impact of each factor in detection of fake news. Detailed implementation and analysis of all the considered factors are demonstrated in the section 2.2.

1. Data Enrichment and Datasets

To tackle the fake news problem, we used an ensemble method. Ensemble is a technique that creates multiple models and then combines them to produce better results than any of the single models individually.

In this project for computing different aspects and factors of the model we used alternative sources of data. Each factor was trained on different datasets based on its characteristics. The collective model which is the polynomial equation that we develop at the final stage of the project has been trained on these distinct datasets, each of which relate to a particular factor. We have then defined specific outputs for the equation. Therefore, when a new test dataset comes in as the input into the model, the polynomial equation will classify the incoming dataset as one of the output labels by looking at the aspect of the input dataset. The different datasets used for factors are introduced below.

## Liar Liar, pants on fire

The dataset is collected through the API of the PolitiFact website is a fact-checking US-based website. The work has been done by Wang, W. Y. (2017) in their benchmark dataset called “Liar Liar Pants on Fire” to come up with a data set that can help to learn how real news can be distinguished from fake news. The author defines a range of veracity ('true', 'mostly-true', 'half-true', 'barely-true', 'false', and 'pants-fire') to determine the fakeness of a statement. The study identifies fakeness by looking at the historical track record of the speakers in their statements. How many times did the person speak the truth? How many times did the person’s “pants were on fire”? How many times was the person half true? These can determine if the person would speak the truth the next time or not. The authors allude to the fact that there are many other mechanisms to detect fakeness. Some of which are discussed in this paper.

## Fake News Challenge

The goal of the Fake News Challenge is to overcome the problem of fake news by using Machine Learning and NLP techniques to assess the veracity of a news. It focuses on the problem of Stance Detection to try to automatically assess the stance of the body of a news article with respect to its headline. The assumption behind this approach is that the news is likely to be genuine if multiple credible news sources show a positive stance for similar headlines. On the other hand, the veracity of the news is questionable if sources with less credibility show a positive stance toward that news. The data provided in the fake news challenge dataset are headline, body, and stance instances, where stance is classified as one of agree (The body text agrees with the headline), disagree (The body text disagrees with the headline), discuss (The body text discuss the same topic as the headline, but does not take a position), unrelated (The body text discusses a different topic than the headline)

## Kaggle Dataset: Getting Real about Fake News

Getting Real about Fake News is a Kaggle dataset contains text and metadata from fake and biased news sources around the web. URLs are collected from BS Detector which is a plug-in used by Mozilla and Chrome browsers to detect unreliable sites and sources and to warn the user accordingly. BS Detector just states a warning message if the article is found to be fake It does not specify the percentage of error and neither does it classify news into levels of “fakeness”. The types in the Kaggle dataset are labeled as one of bias, fake, conspiracy, bs, satire, hate, junksci, state.

## d) Kaggle Dataset: Democrats vs Republicans tweets

Democrats vs Republicans tweets is a dataset on Kaggle that contains party affiliation, twitter handle and the tweet text of various politicians from democratic party and the republican party. The main purpose of this using this dataset is to train a custom document embedding to model party affiliation. This dataset was based on tweets from the members of the house of representatives of the congress.

## Data World: Crowdflower/Hate Speech

Hate Speech identification is a dataset created for academic research for the purpose of detecting hate speech and offensive content in text. It was published in the proceedings of the 11th international conference on Web and Social Media (ICWSM). It contains text labelled under three categories namely, offensive, hate and clean speech. This enrichment dataset was used for the purpose of modelling hate-offensiveness as distilled factors on LIAR LIAR dataset

.

## Kaggle Dataset: Dataset for detection of cyber trolls

Detection of cyber trolls is a Kaggle dataset with tweets and social media post from internet trolls. Internet trolls can provoke issues on social media by spreading fake news. By building custom document vectors to detect trolls, this dataset can be used as an enrichment dataset for the purpose of fake news detection on LIAR LIAR dataset. The dataset contains social media posts from troll and non-troll accounts.

1. Analysis of the factors

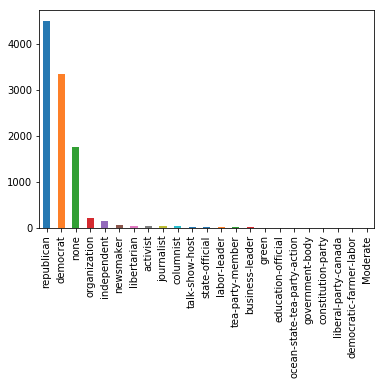
In this section we’ll talk about each factor covering the machine learning and NLP steps that was done to analyze the impact of that particular factor in fake news detection.

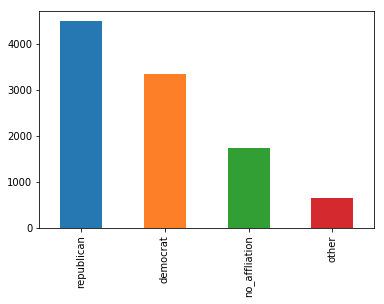
1. **Party affiliation (Swetha Chandrasekar)**

Detecting fakeness depends on several factors. Some of them include party affiliation of the newsmaker. Being affiliated to a far-right wing party can be a strong signal of a fake newsmaker.

Dataset cleaning:

Standard NLP practices such as tokenization, stemming, lemmatization was done to the textual data to clean the datasets and make it ready. There were 23 different party affiliations in the LIAR LIAR dataset. For the purpose of containing the problem, 23 different party affiliations were bucketed to 4 major party affiliations.





Dataset enrichment and preprocessing

For the purpose of adding more signal to model a particular factor, open datasets from the internet had been used to enrich the existing dataset. Please find the various enrichment datasets used below for various factors by Swetha Chandrasekar.

|  |  |  |
| --- | --- | --- |
| Factor | Enrichment dataset | Distillation process followed |
| 1. Party affiliation | Democrat Vs. Republican Tweets [link](https://www.kaggle.com/kapastor/democratvsrepublicantweets) | 1. LDA to identify topics 2. Gensim Doc2vec to represent each tweet |

Algorithm implemented to model the factor

Since modeling the factor is a multi-class classification problem, several algorithms like Random Forest, Support Vector machine (SVM), one-vs-all, and Multinomial Naive Bayes were tried. After several iterations, Neural-network with Softmax as the algorithm gave the best accuracy for party affiliation and Hate speech. Also, NN with Softmax outputs a vector where each class probability sums up to one.

Analysis of the impact in the final polynomial equation

Logistic regression as an algorithm outputs coefficient that represents the importance of each feature and its direction in correlation to the label. Therefore, LR was trained on party affiliation factor to predict fake vs factual label on LIAR LIAR dataset and that coefficient is min-max normalized with some manual intervention.

|  |  |  |
| --- | --- | --- |
| factor | Accuracy (multi class classification accuracy achieved) and algorithm used | Logistic regression coefficient to predict fake vs factual news using these factors |
| Party affiliation | 0.771113 (Neural network with softmax) | republican: -0.171  democrat : 0.408  no\_affliation 0.1403 other \_party: 0.1494 |
|  | | |

Generative model

In the above sections the aim was to try to predict the impact of the party affiliation in detection of fake news. To predict a potential politician response a generative model was used. The LIAR LIAR dataset was enriched with tweets from republican vs democrat Kaggle dataset to be used for the model.

A Neural Network model on LIAR LIAR dataset was developed using a generative language model learning with Wikipedia dataset to predict the next N-words given a topic. The model had an accuracy of 0.27 in predicting the next word.

1. **Hate speech motive (Swetha Chandrasekar)**

Fake news could be politically motivated content with hate speech and offensive statements from the newsmaker.

Dataset cleaning

Standard NLP preprocessing such as tokenization, stemming, lemmatization was applied to the textual data to clean the datasets.

Dataset enrichment and preprocessing

|  |  |  |
| --- | --- | --- |
| Factor | Enrichment dataset | Distillation process followed |
| 2. Hate and offensive speech | Crowdflower dataset on hate and offensive speech [link](https://data.world/crowdflower/hate-speech-identification%20https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665) | 1. LDA to identify topics 2. Gensim Doc2vec to represent each text. |

Algorithms implemented to model the factor

To model the factor different algorithms like Random Forest, Support Vector machine (SVM), and Multinomial Naive Bayes were tried. After several iterations, the best accuracy for hate speech was provided by the Neural-network with Softmax. NN with Softmax also gives a vector as an output where each class probability sums up to one.

Analysis of the impact in the final polynomial equation

Logistic regression is an algorithm that gives the coefficients as an output. This represents the importance of each feature and its correlation to the label. As a result, LR was trained to predict fake vs factual label on the LIAR LIAR dataset and the coefficient is min-max normalized with some manual intervention.

|  |  |  |
| --- | --- | --- |
| factor | Accuracy (multi class classification accuracy achieved) and algorithm used | Logistic regression coefficient to predict fake vs factual news using these factors |
| Hate-ness and offensive content in the speech | 0.68 (NN with softmax) | hate\_speech -0.55 (more the hate speech, it's a fake news) offensive\_speech: 0.484 clean\_speech :0.56 |
|  | | |

Generative model

In the previous sections predicting the impact of hate speech motive in detection of fake news was tried. To predict a potential politician response a generative model was used. The LIAR LIAR dataset was enriched with tweets from republican vs democrat Kaggle dataset to be used for the model. A Neural Network model on LIAR LIAR dataset was developed using a generative language model learning with Wikipedia dataset to predict the next N-words given a topic. The model had an accuracy of 0.27 in predicting the next word.

1. **Troll account (Swetha Chandrasekar)**

Also, whether the statement is from a troll account can help decide fakeness detection.

Dataset cleaning

Standard NLP preprocessing such as tokenization, stemming, lemmatization was applied to the textual data to clean the datasets.

Dataset enrichment and preprocessing

|  |  |  |
| --- | --- | --- |
| Factor | Enrichment dataset | Distillation process followed |
| 3.Troll detection | Cyber trolls on Kaggle [link](https://www.kaggle.com/alisaeidi92/a-very-simple-nlp-model-0-75-accuracy) | Dataset was small. Hence Doc2vec was directly applied on the tweet after standard NLP cleaning |

Algorithm implemented to model the factor:

Several algorithms like Random Forest, Support Vector machine (SVM), and Multinomial Naive Bayes were implemented in order to model the factor different. After several iterations, the best accuracy for the factor of Troll account was concluded by the Neural-network with Softmax. NN with Softmax also gives a vector as an output where each class probability sums up to one.

Analysis of the impact in the final polynomial equation

Logistic regression is an algorithm that gives the coefficients as an output. This represents the importance of each feature and its correlation to the label. As a result, LR was trained to predict fake vs factual label on the LIAR LIAR dataset and the coefficient is min-max normalized with some manual intervention.

|  |  |  |
| --- | --- | --- |
| factor | Accuracy (multi class classification accuracy achieved) and algorithm used | Logistic regression coefficient to predict fake vs factual news using these factors |
| Troll detection | 0.7546490701859628 (multinomial naive bayes) | -0.07 (more the trolls, it's mostly fake news) |

Generative model

The above algorithms help in predicting the amount of fake news. In order to predict a potential politician response a generative model was used using LIAR LIAR dataset and enriched with tweets from republican vs democrat Kaggle dataset. Using a generative language model learning using Wikipedia dataset, an NN model on LIAR LIAR dataset was developed to predict the next N words given a topic. This model had an accuracy of 0.27 in predicting the next word. It is planned to improve this in subsequent iterations.

## **Sensationalism: (Puja Kawale)**

Research was done first to understand how sensationalism contributes in classifying Fake news from the Real news. Numerous studies have shown that fake news - often more sensational than genuine information - spreads faster online because of how social media has prioritized "virality". Often breaking news or the viral headline would gain more shares and likes on social media. This fosters the existence of such fake media. Other factors found – Intent, controversy, propaganda, conspiracy, satire news, reputation, Tone, social credibility, Style.

Analysis and Idea

As per the knowledge gain on this factor, a dictionary of words related to sensationalism was collected. Then all the statements were compared to that dictionary to calculate the similarity score between them. The Higher the similarity, the higher the chances that statement is similar to the sensationalism dictionary and should be a sensational or viral.

Implementation and results

It started with importing the libraries necessary for the implementation like pandas, numpy, matplotlib. pyplot, seaborn, nltk etc...and loading of the .tsv files. Performing of several data quality check like – null values, noise, corrupted data, case, white spaces were done. The alphabets were converted to lowercase, white spaces were removed. Observation was made with respect to shape and information of the columns. Py plots were used for exploration in the dataset. Next was creation of the dictionary of sensational words example: Signed, Terrorist, Stock, Prices, Deal, War, Declare, Nuclear, Abused or Assassinated. Almost 600+ words like these were scraped from online sources and the dictionary was created. Example: my\_dictionary = "(All words)”

Data preprocessing

Below process was done in the pre-processing stage: Stemming which means cutting off the ends of the word, and in some cases also the beginning while looking for the root. For example: carrying -> carry, carried -> carrie. Porter stemmer was used to achieve this.

Lemmatization was considered. Lemmatization is converting the words of a phrase to its dictionary state, example, given the words amusement, amusing, and amused, the lemma for each and all would be used. ex) carrying -> carry, carried. Tokenization was also done. It means splitting a document to sentence or paragraph or may be into the most basic units’ ‘word’, This generated a corpus of all statements and the dictionary vocabulary. Part of speech was Implemented by signs “POS” to each word (and other token), such as noun, verb, adjective was also made.

Distillation

The text in the corpus was vectorized in different ways like Word2Vec, Doc2BOW, TFiDF. For Document Similarity, TFiDf was required to calculate the cosine angle between the two documents. The TFiDF vectorizer was Applied and generated the weight for every word in the corpus. Topic modelling algorithms like LDA and LSA were also implemented and their scores were compared. Based on the comparison LDA was best at distinguishing the topic categories. After that LDA using TFIDF and Bag of Words were also implemented to see if the scores would improve. To compare the scores performance evaluation and testing was done on new statement as the input. LDA using BOW had a better accuracy score of 0.58 in comparison to LDA using TFIDF. Then cosine similarity was considered as a vector towards the polynomial equation.

Ranking

Every document in the corpus was compared with the statements using cosine similarity as shown below.



Later, this was added to a dictionary with key-value pair and was sorted in descending order to get the top sensational news headlines. This was a step towards Ranking of dataset with respect to sensationalism.

## **Controversy: (Puja Kawale)**

Several factors were found to understand what all can contribute to decide an article to be fake, some of them were- Intent, controversy, sensationalism, propaganda, conspiracy, satire news, reputation, Tone, social credibility, Style. Controversy seemed interesting as it is very common for some person, topic, opinion or matter to become controversial if it is not completely true or honest. Research was made and news statements with more controversy in them were sorted out.

Problems solved:

* Data Observation: Checked the data quality and integrity like attribute information, description and null values [isnull ()].
* Data Cleaning and pre-processing – Wrote python function to remove stop words, punctuations. Applied stemming and lemmatization.
* Distillation –Implemented Sentiment Analysis, POS Tagging, Tokenization, Vectorization (TFIDF, Word2Vec, Doc2BOW), Topic Modelling (LSA, LDA), Topic modelling comparison and Ranking.
* LDA – After finding out that LDA was better model than LSA, she executed performance evaluation, ran LDA using bag of words, ran LDA using TF-IDF and Verified the models on unseen document.
* Visualization – Her visualization tools were - Word Cloud, tSNE (distribution of topics), Py Plot, BokehJS 0.12.16
* Classifiers – She applied classifiers like Logistic Regression, Decision Tree, Random Forest Classifier, Naïve Bayes, SVM to the model to check the accuracy and performance.
* Document Similarity: Cosine Similarity Matrix using TFIDF t (Vector of my factor to input in team polynomial equation)
* Creation of Factor vocabulary
* Comparison of news statements with the dictionary words
* Conclusion – higher the cosine similarity higher the similarity between news headlines and the dictionary.
* Ranking: Sorted the news in Liar-Liar dataset with highest sensation stuff to lowest.

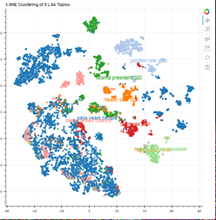


Figure. LSA

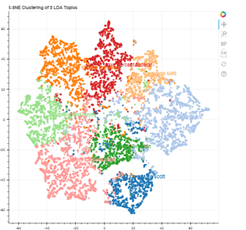


Figure. LDA

Visualization

Several visualization techniques like Pyplot, Histograms, tSNE, Bokeh, Word Cloud was done during data analysis and analyzing the sensationalism factor.

Classifiers accuracy for Sensationalism:

|  |  |
| --- | --- |
| **Classifier** | **Accuracy** |
| Logistic Regression | 0.15 |
| Naïve Bayes | 0.03 |
| Decision Tree | 0.15 |
| Random Forest | 0.19 |

## **Stance Detection (Supreetha Ganapathi)**

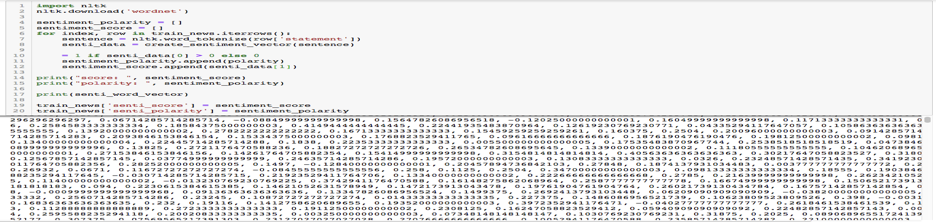
Stance detection typically is a stage of combating fake news and classifies the stance of news headline and body as: agree, disagree, discuss, or unrelated. The values are self-explanatory.

For the stance detection factor, the paper “Fake News Stance Detection” by Chai et al. was used as the baseline and starting point. The LIAR LIAR dataset was chosen to work on. Unfortunately, this dataset doesn’t not have the news body.

Data Enrichment

In the Liar Liar dataset, the news headlines along with the label field are provided. The values for the label are: 'half-true', 'mostly-true', 'false', 'true', 'barely-true’, ‘pants-fire'. For Classification of news as either fake news or not the label values were tagged in a way that half-true and mostly true are considered “true” while the rest are tagged as “false”.

The “senticnet” pre-defined vocabulary which already has classification of whether word is a positive or a negative word and its corresponding intensity score was used.



For the purpose of stance detection, the dataset was enriched by adding the following columns:

* Sentiment vector: for each word in the headline, replace the word with the intensity score and store it in an array.
* Senti\_score: based on the intensity of each word in headline, calculate the overall sentiment score of the news headline
* Senti\_polarity: based on the overall polarity score of words in headline, if its positive, tag the sentence as 1, if it is negative tag it with -1 else 0 being neutral

Approach to analyze the impact of factor in fake news detection was as follows:

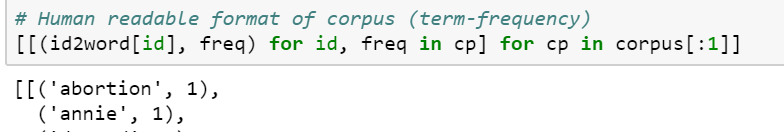
* Number of overlapped words
* Cosine similarity, similarity function (spacy library)
* Number of overlapped n-grams
* Bow (Bag of words)

1. Number of positive/negative words in the headline
2. Word sentiment
3. Polarity of the headlines
   1. Positive
   2. Negative
   3. Neutral

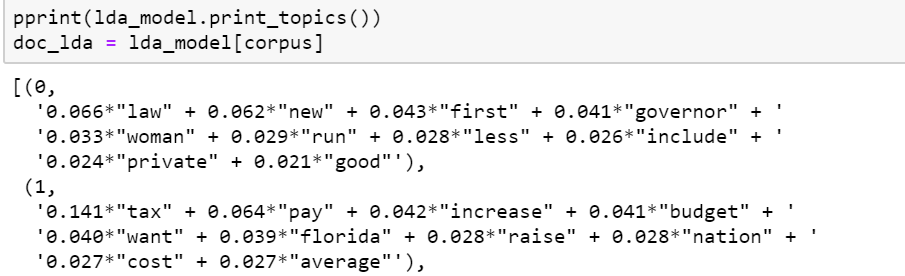
Distillation

To extract the hidden topics from corpus (collection of documents) we use topic modeling. LDA is one of the popular algorithms for achieving this. Python’s “Gensim” package provides an easy implementation of LDA.

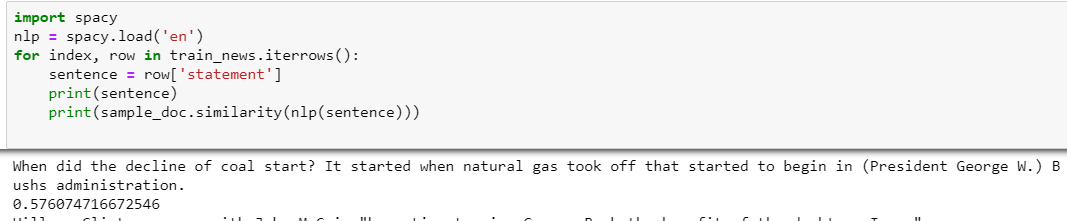
As first step, performed data pre-processing, that includes converting to lowercase, removal of stop words, removal of punctuation, and tokenizing the words from the headline. Gensim package provides a function “id2word” which can be used to determine the term document frequency and this dictionary mapped dataset is fed to the LDA model.



Built the LDA model with 20 topics ( each topic is a combination of words with corresponding weightage/importance). Printed the topics as we can see for each topic the important words and their weights



Based on Count Vectorizer and topics from LDA (taking those words having the highest weights) a sample document was built. Then a document “SIMILARITY” comparison of this sample document (which has the most repeated words that are of both positive and negative sentiment) against each of the news headline was applied and the similarity score was printed.



For the Word Sentiment (News headline polarity) the headline were tagged as positive sentiment or negative sentiment using the Vader sentiment analyzer as shown below. Vader is a lexicon rule-based sentiment analysis tool that runs sentiment analysis on words and then labels the semantic orientation of the sentence as positive or negative.



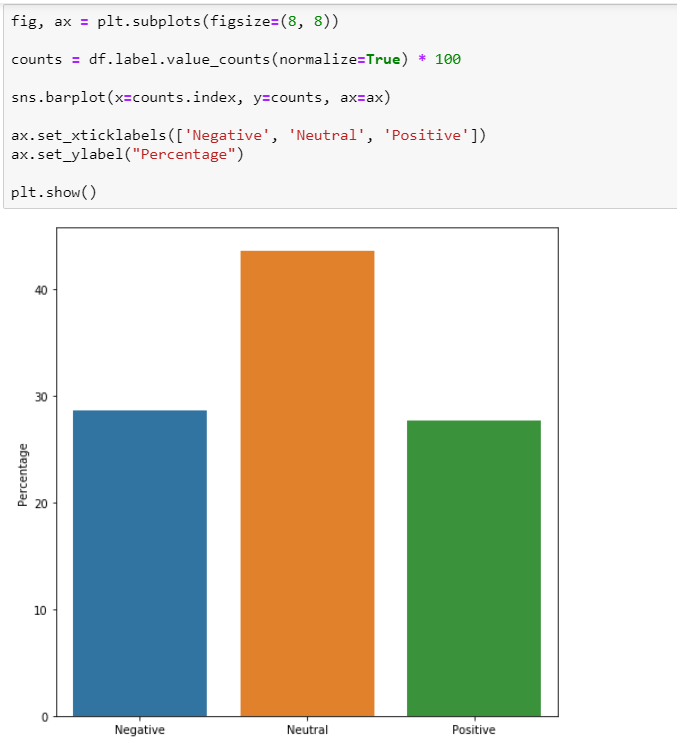


Fig. Bar plot showing the percentage distribution of headlines based on polarity

Algorithms discuss

The term document frequency was calculated using TF-IDF to vectorize the words from the headlines. The sentiment polarity which was calculated earlier was used as the target variable. Also, the Multinomial Naïve Bayes, Logistic Regression, SVM and Random Forest classifier were implemented. Of all the classification algorithms, Random Forest classifier showed the highest accuracy at 95.18%.

Analysis and results

For the factor “Stance detection” one of the things is to determine the similarity of headline against the body of the news. Since in the Liar Liar dataset, the body of news was not available tagging the stance whether the headline “agrees” with the body or “disagrees” or “is neutral” or “Unrelated” was not possible. Therefore, as an alternative a sample document with most frequent words was created and was checked against each of the headline for similarity.

1. **Lexical Features Credibility (Golnaz Bidabadi)**

One of the well-known principles of design is KISS ("keep it simple, stupid") which emphasizes on simplicity rather than making the system complicated. Motivated by the KISS principle and following the paper published by Shu et al. (2017), which recommends the linguistic factors as one of the features that can be used for fake news detection, we studied to determine if the lexical features could be useful for detecting the truthfulness or fakeness of news.

Data Preparation and Exploratory Data Analysis (EDA)

The Liar Liar dataset is divided into three sub-datasets: training dataset, test dataset and validation dataset. All datasets are in tsv format. The step taken for data preparation and data analysis are as follows:

* Adding the column names to each the sub-dataset: The name of columns was extracted from the readme files and were added to all sub-datasets
* Data analysis based on the factor, credibility of lexical features: After a basic data analysis, two columns were identified that needed to be processed during the experiment:
* The “label” column, which has been label by human annotators and includes the level of the veracity of the data records.
* The “statement” column, which consists of the short publish news (claim) written in natural language (i.e. English).
* Exploration of the quality of the dataset: To make sure there are no missing values in the dataset the target columns were Explored.
* Data Analyzation of values in the target columns: Since the statement column includes no missing values and also include natural language sentences, no further value analysis on this column seemed to be necessary. However, A data value distribution analysis on the label column was needed. The distribution of classes based on the label column is demonstrated in the below table.

Table - Distribution of label-classes in the three sub-datasets

|  |  |  |
| --- | --- | --- |
| Distribution of label-classes in the training dataset | Distribution of label-classes in the test dataset | Distribution of label-classes in the validation dataset |
|  |  |  |

As shown in the table, all datasets contain of different types of label-classes. Therefore, all classes should be considered in the classification process and none of them should be dropped.

Data Enrichment

Considering the best practice of fake news detection, discussed by Shu, et. al (2017), the following lexical features were chosen to be extracted from the statement values:

* Total number of words,
* Average number of characters per word (average word-length)
* Frequency of large words
* Frequency of unique words

As part of the data enrichment, the total number of words was added as a new column to the sub-datasets. Next the average number of characters per word in each statement was added to the sub-datasets. Then then the frequency of large words in each statement was added. Based on the average length of the words that was observed, large words were defined as a word which has 9 or more than 9 characters. Finally, the frequency of unique words in each statement was added to the sub-datasets. A word has been defined as unique, if it has been used in one and only one of the statements.

Feature engineering and Data Preprocessing

The data we have is all in text format. Therefore, preprocessing was done because some sort of numerical feature vector is needed to perform the classification task.

* Removing numbers and punctuations
* Stemming
* Creating the Bag-of-words model

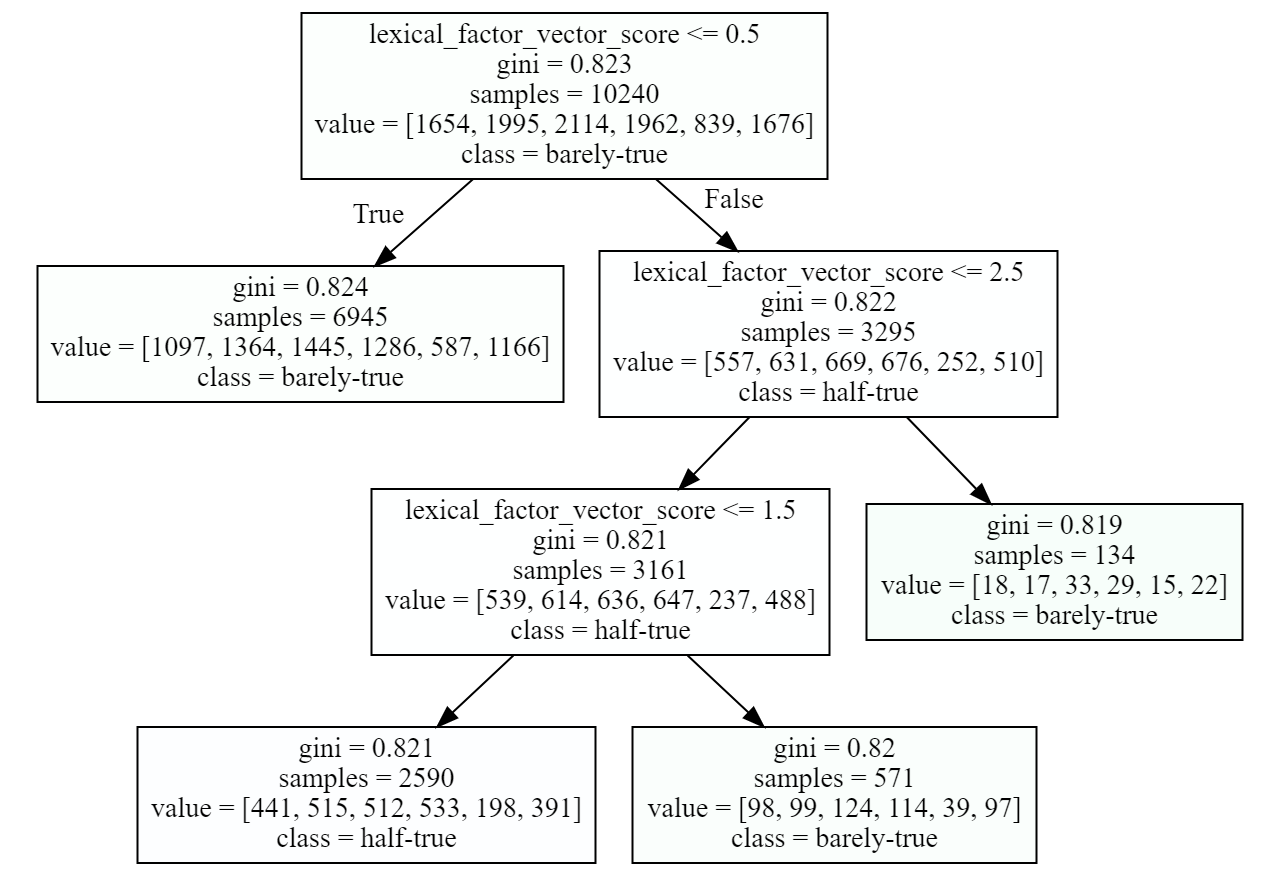
For the first three lexical features (out of four) stemming should have been done as the goal was to extract those features from the text as it was. Feature four (Frequency of unique words per statement) needed preprocessing, including removing numbers and punctuations, conversion to string, stemming as well as the bag-of-word model. A word has been defined as unique, if it has been used in only one of the statements.

Classifications and Evaluation of Classifiers

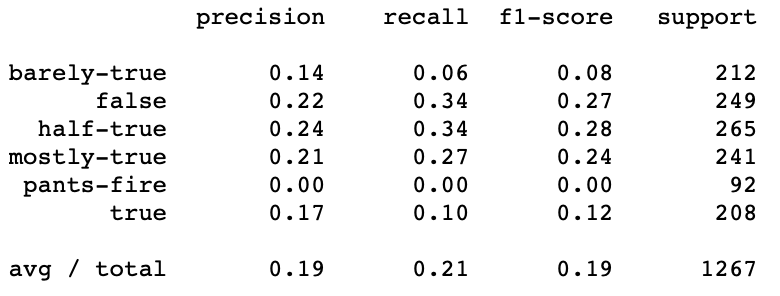
The lexical feature credibility factor is a vector of four extracted lexical features as a its dimensions. Below approaches were applied on the factor.

Decision Tree Classifier based on the lexical features

Decision tree classification was implemented on the lexical features as a vector of four features.



The accuracy was 21.468034727703238. The evaluation results of the decision tree fitted classifier is shown Below.



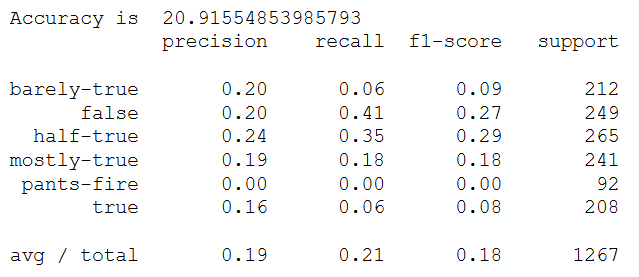
Different max\_depth and min\_samples\_leaf was tested and evaluated in the process of setting up the classifier. The above setting which was reported in this document was one of the best possible variants. Moreover, the classifier was also evaluated by subsets (the four features) of the extracted lexical features. However, none of the subsets resulted in a better accuracy than the above reported one.

Mapping the lexical features vector to a scalar feature and running the Decision Tree classifier –

Four lexical features were extracted, each of them was added as a separate column to the dataset and the classifier was trained based on that feature. The lexical features were considered as a vector consisting of 4 features. The assumption, however, was to let the classifier infer the importance of each feature. In the final polynomial equation for calculation of fake-news-likelihood a single score of the factor lexical-feature is needed. Therefore, a function was calculated and developed to map the lexical features vector to a scalar factor. The developed function is as follows:

Def lexical\_factor\_vector\_score (word number, average\_word\_length, large\_words\_frequency, unique\_words\_frequency): return(word\_number/10+average\_word\_length/5+large\_words\_frequency+9\*unique\_words\_frequency)

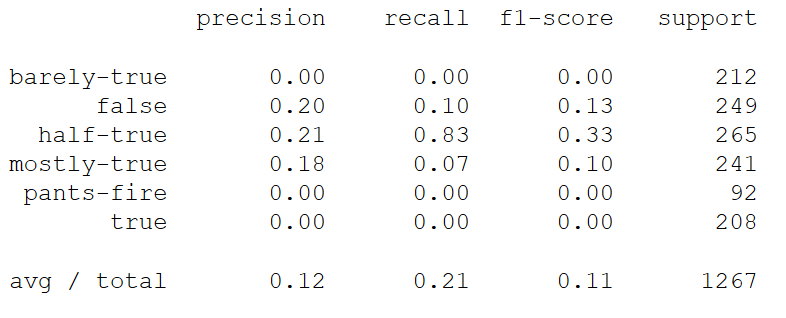
The above function shows the polynomial equation that was used for calculation of the lexical features factor. word number and average\_word\_lenge are normalized. All features except unique\_words\_frequency has given a weight of 1, while the unique\_words\_frequency feature has a weight of 9. The weightings were identified by more than 100 times different try-and-error training of a decision tree classifier. In each try different weights were associated to each factor to find out the optimal weightings. One important observed result here was that out of the four features created for lexical features “frequency of unique words” has the most impact on the detection of fake news. As a result, the highest weight was assigned to this feature in the equation. The results of the classifier evaluation is presented below.



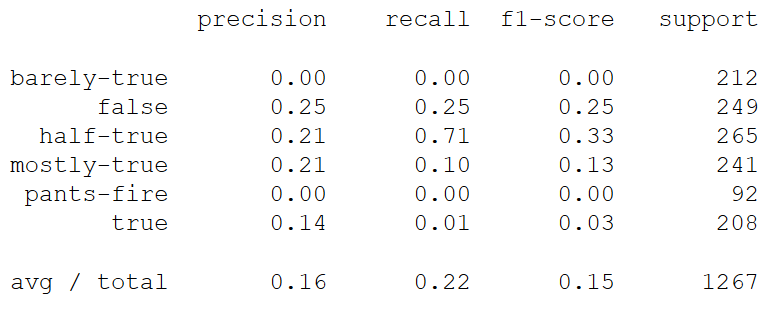
Naïve Bayes Classifier based on the lexical features

The results of the two previously discussed approaches were not satisfactory. As such the naïve Bayes classifier was also tested to see if it would improve the result. Again, both lexical features as a vector and lexical features as a single scalar factor were used in training the classifier.

The evaluation results of the Naïve Bayes classifier on the scaler-feature-based factor is as follows:



The evaluation results of Naïve Bayes classifier on the vector-feature-based factor is demonstrated below.



As you can see the general results of the naïve Bayes classifier on the lexical features are not satisfactory and are similar to the result of the decision tree classifier. One interesting observation was that the recall score of the half-true category is very high in this classifier. However, considering the low precision, it would not be very helpful in general.

Results and Analysis

The evaluation results show that precision and recall score of the classifiers (decision tree and Naive Bayes) for the factor of “lexical features credibility is not very high. Therefore, in the final polynomial equation of fake news detection a low weight should be assigned to this factor. But this study showed that basic lexical features could be as useful as some other high-level factors in detection of fake news when it is considered in aggregation with other factors. This can be seen as an important finding of the research done on this factor, which once again shows that KISS principle should always be considered in design.

1. **Speaker reputation (Golnaz Bidabadi)**

While the history of humankind is the history of lies and truths, the concept of “fake news”, in the way that is used nowadays, is a very new concept. The fake news entry in Wikipedia has been only created in January 2017[[1]](#footnote-1). There is no doubt that the champion of the socio-political discourses surrounding the concept of “fake news” is Donald Trump. Considering this example, it is fair to assume that the speaker of a news (or the person who is referred in the news) can influence the chance of fakeness. Based on this assumption, we studied whether the prior reputation of the speaker could be used as a factor for predicting the fakeness of a news. The assumption here is that if many fake (or true) news has been published about a speaker, the probability of having the same type of news about that speaker in the unseen news of dataset is higher.

Data Analysis and preparation

The Liar Liar dataset was used for this factor. To work on the speaker reputation as a factor in detection of fake news the available information provided by the Liar Liar dataset was explored. The following columns could be used for investigating the speaker reputation in the dataset:

'speaker', 'speaker job', party, barely\_true\_counts, false\_counts, half\_true\_counts, mostly\_true\_counts.

Features that were left out in analysis of speaker credibility in this study are as follows:

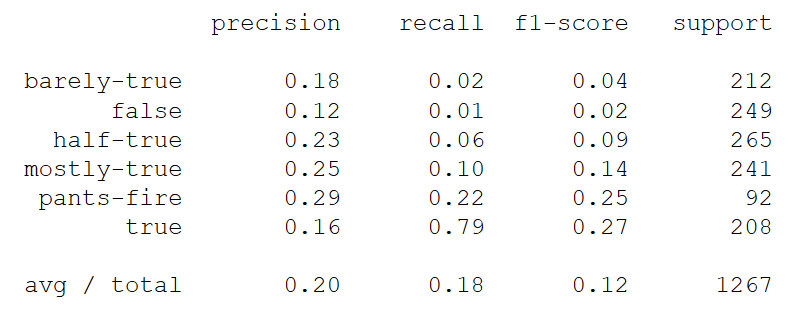
‘speaker’: In general, the speaker’s name could be an interesting feature for determination of speaker credibility. However, in Liar Liar dataset, most of the speakers appear only once or a very few times in the dataset. As a result, it does not seem that the speaker’s name would be a very suitable candidate, considering the dataset.

The same argument regarding the speaker also hold for the speaker job. ‘party’, or political affiliation, is a very good candidate to be explored However, since the political affiliation factor was analyzed as a separate, ‘party’ as one of the features was not included here.

As a result, only the history of the speaker as provided by the dataset was considered to infer the veracity of news and fake news detection. The barely\_true\_counts, false\_counts, half\_true\_counts, mostly\_true\_counts columns were used to explore the factor. The hypothesis here is that, it might be possible to infer fake news based on the number of fake news that have been assigned to the speakers previously. Clearly, this does not mean that a speaker, is a person who produces the fake news or that every single news about a particular speaker is fake. However, what could be inferred from the assumption is that the possibility that a fake news is produced about a particular speaker could be higher than others. For example, it might be the case that the number of fake news that are generated about Obama are more than Trump. Therefore, the history of (the number of) fake- or non-fake news about a speaker is considered in determining the impact of the factor in fake news detection.

Classification and Evaluation of the classifier

Naïve Bayes classifier was used to classify fake news based on the provided columns that was explained previously. The result of the classifier to detect fake news based on a vector-feature definition of the speaker credibility are shown below. The vector includes the barely\_true\_counts, false\_counts, half\_true\_counts, mostly\_true\_counts columns.



As we can see the results are not satisfactory, which is not surprising. The main reason as discussed earlier is that not all statements of a speaker, e.g. Barack Obama, are fake or non-true. But still this factor could be considered in the final polynomial equation, as the probability of a news to be fake could be influence by the speaker of a statement.

Deep Learning/Neural Networks classifier was also applied on the factor. It was done as an experiment to test whether it could improve the result in classification of news based on the speaker reputation features. Based on the tested settings, the achieved result was not satisfactory as shown below.

# 

Analysis and comparison between lexical feature credibility and speaker reputation

None of the factors that were tested so far could be individually used to detect all classes. As a result, some comparison between lexical features credibility and speaker reputation was done. Other approaches were also tested in a try to improve the results.

Comparing predictability of single label-classes based on the lexical features and speaker-reputation (history) features using one-hot-encoding: A comparison between every single class of labels was conducted considering it might provide novel insights. Moreover, it was assumed that it could be the case that if the classifier was trained only for predicting one of the label categories, this could improve the results. For this purpose, one-hot-coding was applied on the labels, and naïve Bayes classifier was trained to predict every single class that we have in the dataset. In the following, the results of this process for each of the classes based on lexical features and speaker-history features is compared.

|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: true | Evaluation results based on the speaker-history features => Label: true |
|  |  |

Reflection: Lexical features is not suitable to be used for detecting the ‘true’ news. But speaker-history produces a 29 percent precision for detecting the ‘true’ news.

|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: mostly-true | Evaluation results based on the speaker-history features => Label: mostly-true |
|  |  |

Reflection: Both Lexical features and speaker-history features produce a very bad recall for the mostly-true news. The precision of the lexical features is better.

|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: half-true | Evaluation results based on the speaker-history features => Label: mostly-true |
|  |  |

Reflection: Both Lexical features and speaker-history features produce very bad recall for the half-true news. The precision of the lexical features is better.

|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: barely-true | Evaluation results based on the speaker-history features => Label: barely-true |
|  |  |

Reflection: Lexical features is very disappointing on the barely-true labels, and speaker-history features seems to be a better feature for this class.

|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: false | Evaluation results based on the speaker-history features => Label: false |
|  |  |

Reflection: Both Lexical features and speaker-history features produce a very bad recall for the false news. The precision is almost the same.

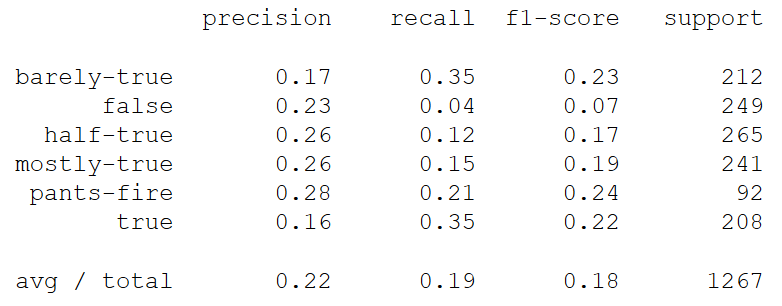
|  |  |
| --- | --- |
| Evaluation results based on the lexical features => Label: pants-fire | Evaluation results based on the speaker-history features => Label: pants-fire |
|  |  |

Reflection: The lexical-features cannot predict the pants-fire news. Although, the speaker-history features also has a very bad recall.

It seems that if we train the classifiers only based on single target labels for the lexical feature’s credibility factor and speaker reputation factor it can only be useful for detecting ‘true’ news. This is not a surprising result, since enough information was not provided for the classifier in this scenario. However, the importance of this finding is that it clearly shows that if we consider the truthfulness, falseness, or fakeness of news as a relative concept which cannot be dealt with as a binary classification problem, then solving the problem in a binary way would not be helpful to understand the gradual changes in the classes.

Combining the two factors of lexical features and speaker reputation

Both lexical features and speaker reputation factors will be applying separately in the final polynomial equation that will be used to detect fake news. But as an experiment, a naïve Bayes classifier was trained and evaluated on the combination of the vector-features of both factors. The result is as follows:



The result shows that aggregating two factors is not far better than the results of each factors separately.

1. **Title-Body-Correlation (Golnaz Bidabadi)**

The stance correlation between body and headline of a news has been shown to be an appropriate factor to detect the fakeness level of a news (e.g. see Thota 2018). However, there have not been much study to test whether the correlation of other features of titles and bodies, could be helpful factors for predicting the type (level of fakeness/truthfulness) of news. Therefore, it was tested, whether Doc2Vec, LDA, and sentiment correlation between titles and bodies of news could be useful factors for detection of fake news.

Data Enrichment

The two “Getting Real about Fake News" Dataset and "FakeNewsChallenge" dataset were combined to enrich the dataset to see the impact of the title-body-correlation factor in fake news detection.

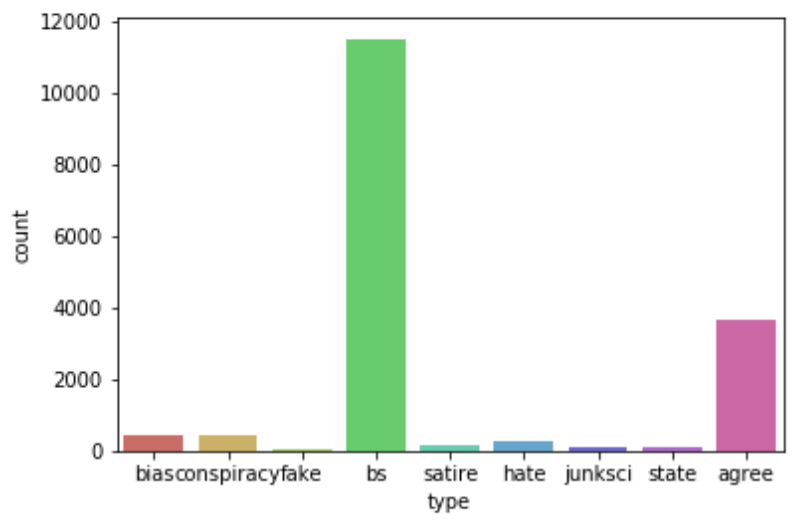
The “Getting Real about Fake News dataset includes "Text and metadata from fake and biased news sources around the web". Since the dataset does not include "unbiased" or "true" news, the "Fake News Challenge" dataset was used to enrich its data. Only he "agree”-labeled news data of this dataset was used to enrich the "Getting Real about Fake News" dataset. The work was done under the hypotheses that the label "agrees" in the "Fake News Challenge" dataset was considered as "True". Based on the fake news challenge definition “agrees” is define as "the body text agrees with the headline" (<http://www.fakenewschallenge.org/>)

It is worth mentioning that since main objective of this study was not to evaluate stance-correlation between headlines and text-bodies but to test other features, the “Fake News Challenge” dataset (which labeled only based on the stance-correlations) were not used in this experiment.

Which is available on GitHub (<https://github.com/FakeNewsChallenge/fnc-1>) and has been published by <http://www.fakenewschallenge.org/>. We only use the "true"-labeled news of this dataset to enrich the "Getting Real about Fake News" dataset.

Exploratory Data Analysis

To explore the data a value distribution analysis of the data was performed on the type column which consists of the label-classes.



As it is clear based on the above class distribution analysis. The classes in the dataset are not evenly distributed and many of the news are labeled as “bs”. The “agree” class that was added to the dataset in the data enrichment process holds the second rank in the class distribution matrix. While the distribution of classes is not ideal, however none of the classes were removed. The main motivation for this was to prevent the dataset from becoming a binary dataset, only including “bs” and “agree” news. Fakeness and trueness are not binary concepts and the datasets that are used for fake news detection should also not be binary, if one aims to perceive fake news detection as a real-world and non-abstract problem.

Data Preprocessing

After the data cleaning the following data preprocessing methods where applied on titles and text-bodies:

* Spell check and removing stop words
* Stemming
* Lemmatization
* Removing Punctuations

Using word cloud visualization, the following table represents how the preprocessing has changed the dynamics of the most frequent words in the dataset:

|  |  |
| --- | --- |
| Word Cloud on titles before preprocessing | Word Cloud on titles after preprocessing |
|  |  |
| Word Cloud on text-bodies before preprocessing | Word Cloud on text-bodies after preprocessing |
|  |  |

Table - The impact of preprocessing represented using word cloud visualization

Distillation and Feature Engineering

Beyond basic features such as bag of words (BOW) and TF-IDF, the below features were also extracted for titles and bodies. The aim was to look at sub-areas and embedding (latent variables) within the factor topics, sentiments, and others so additional features and parameters were extracted to assess the factor better. This would potentially provide the ability to get a higher accuracy score for the factor.

* Doc2Vec Embedding
* Cosine similarity between title and body of every news
* LDA using Bow
* LDA using TF-IDF
* Sentiment
* Sentiment correlation between titles and bodies
* Cosine similarity between titles and bodies based on 10-dimensional LDA vectors

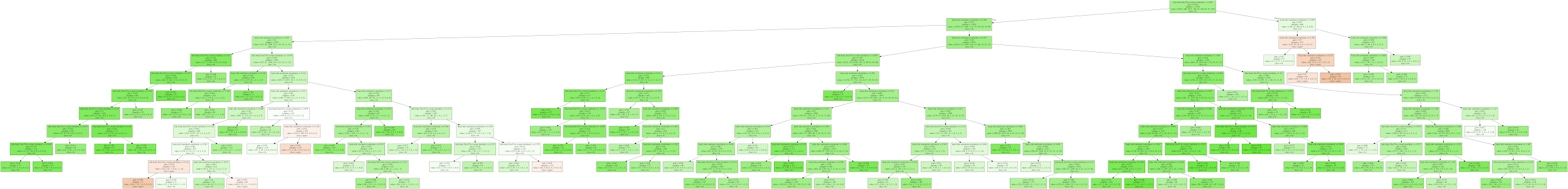
Classifiers

The following classifiers were trained and evaluated on the single features that were mentioned above and also the combination of different features and the vectors that could be constructed based on the extracted features:

* Decision tree classifier
* Random forest classifier
* Naive Bayes classifier
* Neural network

The following table summarizes the evaluation results of some of the conducted experiments. More results are available in the attached notebook.

|  |  |
| --- | --- |
| Naive Bayes on Title-Body Doc2Vec-Cosine-Similarity and Sentiment  Accuracy: 0.7141961767471012 |  |
| Decision Tree on Title-Body Doc2Vec-Cosine-Similarity  Accuracy is 68.6305233469132 |  |
| Naive Baise on lda-bow title-body-correlation and lda-tfidf title-body-correlation  Accuracy: 0.7141961767471012 |  |
| Neural Network on lda-bow title-body-correlation and lda-tfidf title-body-correlation  No satisfactory result |  |
| RandomForestClassifier on all extracted features  Accuracy: 0.7141961767471012 |  |



Representation of decision tree on title-body Do2Vec cosine similarity feature

Testing the model on unseen document

As a proof of concept, beyond constructing different vectors for describing the correlation between titles and body of a single document and evaluated different classifiers to predict the type of news based on this factor, the cosine similarity between the title-body correlation of an iconic “agree” news and a randomly chosen (unseen) document was calculated. Later, based on the calculated cosine similarity the label of the randomly chosen document was correctly predicted. This can be seen as an application of prototype theory of cognitive science[[2]](#footnote-2) in machine learning (fake-news detection). Similar to humans, here, an iconic exemplar (a prototype or stereotype) has been used for categorization of an unseen instance. See the attached notebook for more reflection and details on this topic.

1. **Spell Checks and Grammar Checks (Vijay Samuel)**

June Casagrande in an article in the LA Times describes one of the mechanisms to identify if news is real or fake is to identify if there are spelling and grammatical errors. The reason the author provides to this claim is that genuine companies spend a lot of money on proof reading that is not done by small time Twitter scammers. To that extent, the factor of spell checks and grammar checks where attempted to be determined for the detection of fake news on the Liar Liar dataset. For this purpose, the statements documented in the dataset was used.

Data Enrichment

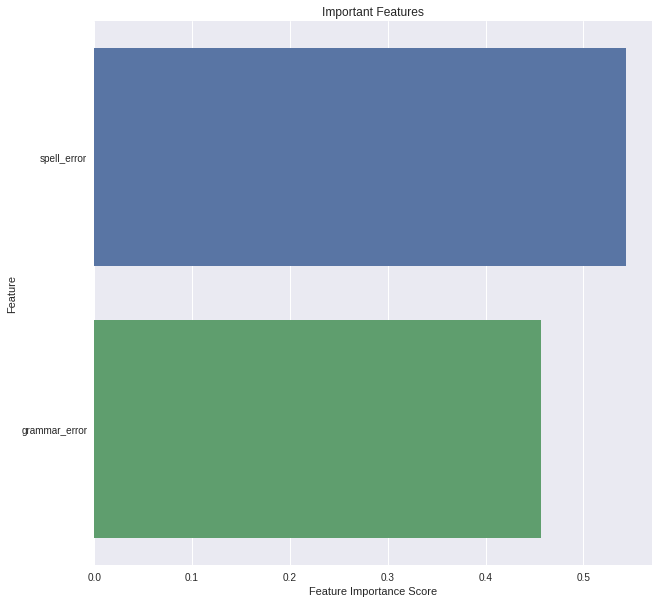
To do grammar checks, “language-check” which can determine both spelling and grammar errors and return the count of them was needed. A new feature was added to document the number of errors in the statement.

Data preprocessing

It was observed that well written news statements typically start in a shorter form which isn’t grammatically perfect. For example, “Man kills the wife and kills self”. Hence, the additional level of checks was needed to determine fakeness with better accuracy. For this purpose, “PyEnchant” was used to determine the spelling errors in the statements. To be able to detect the errors words were tokenized firs using NLTK library. Then the number of spelling errors in the statement were determined. With the error count, a new feature was added to count the number of spelling errors.

Algorithms Implementation and analysis

With the grammatical and spelling errors as the features available for training, Logistic Regression and Random Forest Classifier were applied to the dataset to determine coefficients and feature importance for the factor of grammatical and spelling errors. Once the additional level of spell checks was added, the accuracy improved but was not accurate enough to be used as a primary feature in the final model of the polynomial equation. Following were the feature weights given by Random Forest Classifier for spelling errors and grammatical errors:



It was observed that spelling errors had higher weightage than grammatical errors. On observing closer into the data, it was seen that text from news sources had typical news-like statements which can be construed as grammatical errors. This caused falser positives than expected. Spelling errors are pretty common in sources of low credibility that don’t give importance to proofreading.

1. **Religious Sentiments (Vijay Samuel)**

A point of contention during many debates is what a certain religious text conveys about day-to-day occurrences and laws that are being passed. Things like tolerance to homosexuality, terrorism, abortion etc. have gone back to quotes from religious texts. The liar liar data set with the help of validations from Politico has classified news based on the news article. This feature attempts to classify the news as real or fake with the use of religious sentiments as a criterion.

Data Enrichment

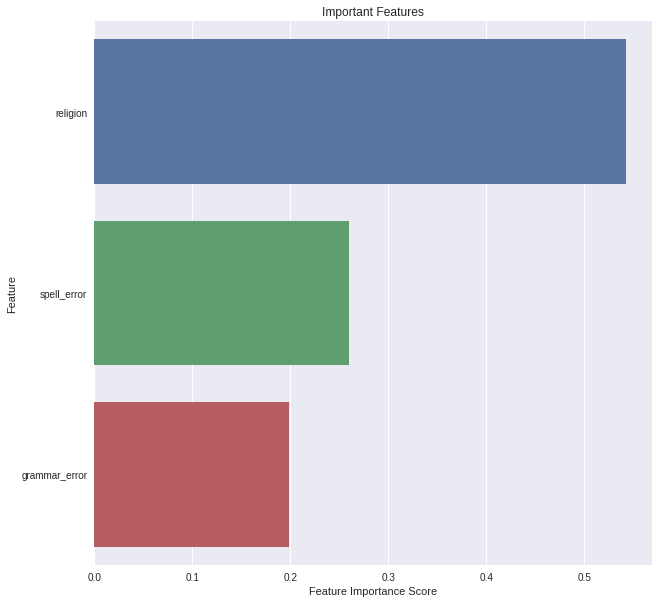
To classify documents that have religious sentiment associated with them we have come up with a corpus of most commonly used words in context to religion. Some of the most common words would be “Christ”, “Islam”, “Allah”, “Bible”, “Quran”.

Data preprocessing

Similar work as done as compared to Puja’s sensationalism study. The documents were tokenized and lemmatized. Point of Speech (POS) was also done on the statements in the dataset.

Algorithms Implementation and analysis

After preparing the data, TF-IDF was done on the data followed by cosine similarity to come up with a new feature called “religion”. Random Forest classifier was applied on the data set with “religion” as a feature. The Random Forest Classifier was able to classify with an accuracy of 0.56. Hence religion was a considerable feature to use. On training the data with spelling and grammar errors, the feature importance score seemed to be substantial. The weightage can be seen in the figure below.



However, when adding in other features discussed above, the weightage of religion as a feature reduced substantially.

1. The collective model - Polynomial Equation (Vijay Samuel)

To combat the problem of fake news we developed a collective model which is a polynomial equation consisting of all the factors discussed in section 2.2 for detecting fake news. All the factors were taken and added into a single data frame. Then Logistic Regression and Random Forest Regression were applied to all the features combined. The Random Forest Regression provided a collective feature importance for each factor. We observed that theoretically, the feature importance given by Random Forest Classifier doesn't convey direction, i.e., if the statement was fake or not. The importance scores that are generated by the classifier also are not to be used in computing feature weights as they only convey the weights used to create regions and can’t be used for creating feature weights. Given that our Linear Regression Classifier gave us a good score. We used the coefficients obtained by Logistic Regression along with the intercept to get the following logit equation.



Based on the computed intercept and coefficients the polynomial equation would look like:

|  |
| --- |
| l = -1.0512754519948995 + (grammar\_error\*-0.06782130729525217) + (spell\_error\*0.00973070799969336) + (religion\*-0.37821596367443827) + (hate\_speech\_confidence\*0.22179065390343028) + (offsensive\_speech\_confidence\*-0.6954199400672489) + (clean\_speech\_confidence\*-0.5776482575765258) + (troll\*3.7430628129810866) + (lexical\_factor\_vector\_score\*0.005937309210522443) + (barely true\*-0.015907728063848298) + (false\*0.0002847153983823605) + (half true\*0.012141837617036922) + (mostly true\*0.0015617124451878545) + (pants on fire\*-0.02149300354765071) + (compound\_stance\*-0.1109818712393586) + (neg\*0.057962420066082536) + (neu\*-0.09188996822596882) + (pos\*-1.0290077360922947) + (headline\_polarity\*0.06679666716746976) + (confidence\_republican\*-0.4681475311309757) + (confidence\_democrat\*-0.08525806808368233) + (confidence\_no\_affliation\*-0.40466847152750324) + (confidence\_other\*-0.09320493693705045) + (sensationalism\*0.16211412607271472) + (Controversial\*0.15829558515811565) + (title-body Doc2Vec-cosine-similarity\*0.19083921147340285) + (body-title sentiment-correlation\*0.06958365095431432) + (lda-bow title-body-correlation\*-0.17635035135856886) + (lda-tfidf title-body-correlation\*0.09158508934574408) |

1. Conclusion

The problem of fake news is growing in a rapid pace as social media has become one of the major sources for generating and consuming news. People start believing that their perceptions about a particular topic are true without questioning the veracity of news. In this paper, we presented a systematic approach toward solving the problem of fake news detection. Various NLP and machine learning techniques were implemented in an attempt to identify the impact of each factor in detecting weather the news is truthful or fabricated. The goal was to define an equation consisting of multiple factors with different weightage to decide the fakeness-likelihood-score of an article. We demonstrate using the coefficients for each factor in the equation.

1. ACKNOWLEDGMENTS

We would like to thank Professor Ali Arsanjani for giving us the opportunity to work on such important societal matter and for his invaluable guidance in preparation for this project.

1. REFERENCES

Rubin, V. L., Chen, Y., & Conroy, N. J. (2015, November). Deception detection for news: three types of fakes. In Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community (p. 83). American Society for Information Science.

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22-36.

Wang, W. Y. (2017). " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.

Chai, X. W. S. C. Z. Fake News Stance Detection

Thomas Davidson, Dana warmsley, Michael May and Ingma Weber. 2017 “Automated Hate speech detection and the problem of offensive language” Proceedings of the 11th international conference on Web and Social Media (ICWSM)

Thota, Aswini, et al. "Fake News Detection: A Deep Learning Approach." SMU Data Science Review 1.3 (2018): 10.

<https://dushyantadhyani.github.io/files/fnc-1-stance.pdf>

<https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview>

<https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/gensim%20Quick%20Start.ipynb>

<https://spacy.io/usage/vectors-similarity>

<https://www.ijcai.org/proceedings/2017/0557.pdf>

<https://www.learndatasci.com/tutorials/sentiment-analysis-reddit-headlines-pythons-nltk/>

<https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk>

<https://towardsdatascience.com/hacking-scikit-learns-vectorizers-9ef26a7170af>

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>

<https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24>

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)