Alternus Vera

Ronak Mehta   
Charles W. Davidson College of Engineering  
San Jose State UniversitySan Jose CA, USA  
ronak.mehta@sjsu.edu  
  
  
Chetan Kulkarni Charles W. Davidson College of Engineering  
San Jose State UniversitySan Jose CA, USA  
chetan.kulkarni@sjsu.edu  
  
  
 Nupur Yadav Charles W. Davidson College of Engineering  
San Jose State UniversitySan Jose CA, USA  
nupur.yadav@sjsu.edu  
  
  
 Lokesh V Charles W. Davidson College of Engineering  
San Jose State UniversitySan Jose CA, USA  
lokesh.vadlamudi@sjsu.edu

# Introduction

The increasingly wider and more commonplace dissemination of “fake news”, i.e., news of dubious quality promoting intentionally false information calls for its mitigation as it threatens to shake the foundations of trust and ethics that drive social institutions.

The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Fake news detection on social media has recently become an emerging research that is attracting tremendous attention.

“fake news” is intentionally written to mislead its readers into believing false information, accusations or extrapolate from misleading or false data.

This mal intention makes it increasingly difficult and nontrivial to detect based on mere content.  It is therefore paramount to amalgamate additional contextual information features and factors:  such as user social engagements on social media, sentiment, political stance, etc. to help make a more definitive and authoritative determination.

In addition, exploiting this additional contextual information is challenging in and of itself as users’ social engagements with fake news produce data that is big, incomplete, unstructured, and noisy.

Because the issue of fake news detection on social media is both challenging and relevant, we conducted this survey to further facilitate research on the problem. In this survey, we present a comprehensive review of detecting fake news on social media, including fake news characterizations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets. We also discuss related research areas, open problems, and future research directions for fake news detection on social media.

# Individual Factors

|  |  |
| --- | --- |
| **Team Member** | **Contribution** |
| Ronak Mehta | Network Based |
| Nupur Yadav | Political Affiliation |
| Chetan Kulkarni | Stance Factor |
| Lokesh Vadlamudi | Title Vs Body |

## Network-Based [1]: Social Media users forms different networks in terms of interests, topics, and relations. Dissemination process of Fake News forms an echo chamber cycle. To represent these types of network patterns for fake news detection we need to extract network-based features. These network-based features can be extracted by constructing specific networks among the users.

Different types of networks can be constructed:

## Stance Network: The network's nodes indicate all the tweets relevant to the news and the edges indicates the weights of similarity of stances.

## Co-occurrence Network: The network consists of inter-connection of terms based on their paired presence within a specified unit of text.

## Friendship Network: This network indicates the following or followed structure of users who post related tweets.

## Co-occurrence network [2] was used to fetch network-based features and LSTM [3] network architecture to classify the veracity of the news.

## Political Affiliation: Political Affiliation means closeness to or supporting a particular political party. It is a very important factor in determining if a news or statement is fake because party members usually stick to the party's philosophy and any digression from that is a good approximator of fake news. So, I have tried to model this factor to check the credibility of news.

## For the purpose of data enrichment, I have **used Kaggle Fake News dataset and PolitiFact News dataset** which had similar domain data to that of liar dataset (comprises mainly of political topics).

## Stance Factor: Stance is the way in which speakers position themselves in relation to the ongoing interaction, in terms of evaluation, intentionality, epistemology or social relations. Hence in the pursuit to find out about the stance of the news can actually help us determine if the news is fake or not.

## For the purpose of data enrichment I have used SemEval Dataset (which has Tweets Data and Stance associated with the tweets) and Fake News Challenge (which has dataset news and stance associated with tweets )

## The whole idea of stance detection is based on following assumptions

## Similar Statements tend to have Similar Stance

## TF-IDF of words in statements can help us identify stance

## Title Vs Body: In Title Vs Body factor, we compare how the title fares with the context of the content in the body. i.e the amount of similarity between the two is taken. Generally, the fake news contains title that is irrelevant to the actual content inside.  To detect the similarity, we use doc2vectorizer between the two.

## Datasets used are Liar Liar Dataset and Fake news Dataset.

# Datasets used for Enrichment

# Liar Liar

# Kaggle Fake News

# Kaggle Fake News Detection

## Politifact Fake News

## SemEval

## FakeNews Challenge

# Methodology

Each team member has modeled a single factor to determine the authenticity of the News.

## Network-Based

Data Pre-processing:

* Null values were removed from all the datasets.
* Visualized the features and text.
* Encode labels into categorical values
* Removal of unwanted features.
* Removal of Special Characters and Punctuations, Lower casing the text, Tokenization, Removal of Stop Words, Lemmatization, Stemming

The labels were categorized from **1-6, 1 being “true’ and 6 being ‘pants-fire’.**

Generation of the Embedding Matrix:

* GloVe word vectors were used.
* Tokenizer was created and fitted on text.
* Sequences & Padding was generated for efficient training.
* Splitted the dataset into train, test, and valid.

Modeling and training:

The model was defined, compiled, fit, and saved. This model was trained until the validation accuracy didn’t improve.

**The final tuned model gave an accuracy of 95% which was used in our final combined model.**

## Political Affiliation

I have used the following approach to model political affiliation factor.

1. Get data from liar dataset, Kaggle fake news dataset and PolitiFact news dataset.
2. Initial data exploration
3. Remove unwanted columns
4. Clean the datasets using standard NLP practices such as stop word removal, lemmatization, stemming, removing punctuations etc.
5. Update the Party affiliation of liar dataset to include only three types to handle class imbalance in party affiliation labels

* Democrat
* Republican
* No-affiliation and Other as Other

1. Perform LDA on all the three datasets and added LDA score to liar dataset.
2. Create corpus of text by combining both the datasets
3. Create Doc2Vec embeddings (length = 10) using genism for each tagged document
4. Feature Engineering: Use the generated doc2vec embeddings directly as features along with other categorical features such as state\_info, speaker\_job\_title, context
5. Train a neural network on the above features which is using softmax and predicting a vector, which comprises the probability scores for each party affiliation.
6. Use the probability scores generated along with the LDA score to train several multiclass classification models (such as Logistic Regression, Decision Tree Classifier, Gaussian NB, XG Boost Classifier, Random Forest Classifier, SVC and K-Neighbors Classifier) to predict the fakeness of the given statement **between 1-6 with 1 being true and 6 being pants-fire.**
7. Use the best model to predict fakeness of a statement or news.

## **What worked?**

1. Doc2Vec embeddings resulted in more powerful representation of the text than the other embeddings tried.
2. Neural Network model outperformed other models for predicting a probability vector for party affiliation which comprises probability scores for each party.
3. Tried various multiclass classification models to predict fakeness between 1-6 but SVM performed the best among all the multiclass classification models trained to predict fakeness of a statement

## **What did not work?**

1. LDA as part of distillation helped in getting insights of the topics different datasets contained but didn’t help in generating any useful features.
2. Word2Vec didn’t perform as good as Doc2Vec so went ahead with Doc2Vec model.
3. Tried several models to predict the probability vector for party affiliation such as Logistic Regression, SVM, Naïve Bayes, Random Forest but Neural Network model outperformed others.
4. Tried various multiclass classification models to predict fakeness between 1-6 but SVM performed the best.
5. Adding LDA score to liar dataset didn’t help improve the accuracy of the final model.

## **What alternatives did you try?**

1. Tried tuning the hyperparameters for training Doc2Vec and Neural Network models such as learning rate, no of iterations to see what is working better.
2. Tried increasing and decreasing the number of embeddings generated by doc2vec model and which

are directly used as features along with other categorical features to predict the probability vector for party affiliation.

|  |  |
| --- | --- |
| **Technique** | **Accuracy** |
| Without Amalgamation, using word2vec and binary classification | 45% |
| Without Amalgamation, using Doc2Vec and binary classification | 52% |
| With Amalgamation, using Doc2Vec and binary classification | 56% |
| With Amalgamation, using Doc2Vec and multiclass classification | 22% |
| With Amalgamation, using Doc2Vec, adding LDA score and multiclass classification | 22% |

Following figure shows the different multiclass classifications models trained to model *Political Affiliation* factor and predict fakeness of statement using that factor. As, you can see **SVM gave the highest accuracy of approx. 22% which is used in the final combined model.**

A close up of a map

Description automatically generated

Figure 1: Models and their accuracies for political affiliation factor

## Stance Factor

Following is the approach I have used to model my factor.

1. Get All the data from both the datasets
2. Remove the unwanted columns
3. Remove Digits
4. Perform Stemming the words
5. Remove Stop Words
6. Remove Punctuation marks
7. Update the Stance of SemEval Dataset to the following values

* 'AGAINST' to 1
* 'FAVOR' to 2
* 'NONE' to 0

1. Update the Stance of FakeNewsChallenge Datset to the following values

* 'unrelated' to 0
* 'disagree' to 1
* 'agree' to 2
* discuss' to 3

1. Combine Both the dataset
2. Create Tagged Document
3. Create a doc2Vec Model
4. Check the accuracy
5. Run other Models like Naïve Bayes, LR, KNN, Dtree, Rforest, SVM, KNN
6. Compare the model performances
7. Use the best model to classify the stance in the liar liar dataset

## **What worked?**

Basic TF-IDF vectorizer with logistic regression, Decision Tree Algorithm and Random forest performed better giving 75% accuracy.

***What did not work?***

Neural network model with sophisticated linguistic application like doc2Vec and word2vec did not perform as expected. They gave 27% accuracy which is very low.

Following figure shows the different multiclass classifications models trained to model *stance* factor and predict fakeness of statement using that factor. As, you can see **XG boost gave the highest accuracy of approx. 43% which is used in the final combined model.**

A close up of a map

Description automatically generated

Figure 2: Models and their accuracies for stance factor

## Title Vs Body

The following approach has been used to model this factor.

Data Cleaning:

Initially we load the data, the data set we had is in ‘.tsv’ format. All the train, test and valid datasets are loaded. We assign the column names as there were no names. The output was assigned as label column. Loaded everything into respective data frames. We checked for null values, there were none.

Data Preprocessing:

The output column had multiple labels. So, combined ['mostly-true','half-true','original','true'] labels as ‘True’ labels i.e ‘ 1 ‘. Other classes as stored as ‘0’ i.e ‘False’ labels. We convert the original label into 1’s or 0’s by passing the function for every row in the dataset. These news labels are stored in a new column known as ‘labelEncoded’ column.

Now, we install all the necessary libraries needed for our model to run from NLTK. A custom function is written to clean the data. In this function, the stop words are removed from the data, stemming is done and finally punctuations are removed. We apply this function to body column containing the context.  The new cleaned data is stored in a new column. We now take the columns that are needed for our factor in a separate dataframe, [‘subject’, ‘finalHeaderBody’, ‘labelEncoded’]

In subject column, the rows generally consist of multiple different words with a comma in between them. So, in order to separate these, we use a custom function named ‘break\_subject’ that takes the subject data row and returns a list containing separate words that make up the subject.

A screenshot of a cell phone

Description automatically generated

Doc2Vectorizor model:

In order to use the doc2vectorizer model to find the similarity, we first need to generate tagged words in our body context. So, we use Tagged Document module to generate a list of words from our context rows and store all the tagged word rows in a variable. We pass this variable to build the vocabulary of our model. We train on top of this.

A screenshot of a cell phone

Description automatically generated

Now, in order to compare the subject with the body, we could have easily compared with subject in its original form but, sometimes the context can be seemed relevant to the title if we compared with different forms of words. So, a custom function to generate a list of synonyms to our title is needed. It uses wordnet which in turn uses ‘synset’ to generate the synonyms. This function is applied to all the rows of subject column and the generated synonyms are stored in a separate column. Finally, the doc similarity function is applied on top of this generated synonym column to check the similarity between title and body.

Dataset:  Fake news dataset

We tried to apply same procedure to generate the similarity column, but the title and body will have 100 % similarity score because in this dataset, the title is the first sentence of the body. There is no point in running the model in this dataset.

Following figure shows the different multiclass classifications models trained to model *title vs body* factor and predict fakeness of statement using that factor. As, **you can see XG boost gave the highest accuracy of approx. 43% which is used in the final combined model.**

A close up of a map

Description automatically generated

Figure 3: Models and their accuracies for title vs body factor

# Final combined model

Since, **we are doing a multiclass classification, each of our individual models are generating a list of six probability scores for 1 to 6 (1 being true and 6 being pants-fire) class labels** and then we are doing a weighted average for every probability and at last returning the max probability with the corresponding index denoting a label from 1-6 (here indices from 0 to 5).

Following is the snapshot of the final combined model and the polynomial equations we formed.

A screenshot of a cell phone

Description automatically generated

Figure 4: Polynomial equations in final combined model

##### References

1. Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu, “Fake News Detection on Social Media: A Data Mining Perspective,” ACM SIGKDD Explorations Newsletter, 2017
2. https://hal.archives-ouvertes.fr/hal-02334445/document
3. https://keras.io/layers/recurrent/#lstm
4. <https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184>
5. <https://www.datacamp.com/community/tutorials/simplifying-sentiment-analysis-python>
6. <https://github.com/vikaspro90/Tweet-Stance-Detection>
7. <https://www.programcreek.com/python/example/103013/gensim.models.doc2vec.TaggedDocument>
8. https://github.com/swethac18/AlternusVera/blob/master/factor\_Party\_Affliation/Party\_Affliation\_RNN\_GENERATION2.ipynb
9. <https://towardsdatascience.com/how-to-gain-state-of-the-art-result-on-tabular-data-with-deep-learning-and-embedding-layers-d1eb6b83c52c>