FakeNet: A Scalable Framework to Detect and Analyze Fake News

Shantanu Jaiswal

U1423003C
School of Computer Science and Engineering, Nanyang Technological University Singapore
SHANTANU004@e.ntu.edu.sg

Abhay George Nainan

U1423092K
School of Computer Science and Engineering, Nanyang Technological University Singapore
ABHAYGEO001@e.ntu.edu.sg

Jacob Sunny

U1421414L
School of Computer Science and Engineering, Nanyang Technological University Singapore
JACOB004@e.ntu.edu.sg

Mohammad Sharique Zaman

U1422997K
School of Computer Science and Engineering, Nanyang Technological University Singapore
SHARIQUE001@e.ntu.edu.sg

ABSTRACT

The proliferation of Fake-News is a significant problem at the intersection of politics and technology, due to the potential for malicious agents to drive socio-political agendas. Current approaches to detect fake-news involve manual fact-checking, and crowd-sourcing which are both tedious and error-prone. The complexity of the task means text mining and machine learning models can make significant contributions to fake-news detection. While the problem-statement of fake-news is simple, the number of variables describing fake-news are both numerous and complex. Corresponding approaches can range from natural-language models to social-network propagation analysis. To this end, we surveyed existing literature and explored the following three approaches: language analysis, stance analysis and website domain/social media account trustworthiness. Consequently, we developed a system that leverages on text mining and machine learning models to detect fake articles. The system currently constitutes an ensemble of three models that use text features, stance scores and domain trustworthiness for prediction. Additional models can be added to the system to incorporate more variables of fake news. The methodology of the current system as well as the performance of its constituent models are presented in this report.

KEYWORDS

Fake-News Detection, Data Mining, Natural-Language Processing, Deep Learning, Ensemble Methods, Machine Learning

1 INTRODUCTION

Social media and online publications have become the key medium of information distribution, due to its ease of both creation and consumption. This mechanism has however been exploited by malicious agents who seek to drive their social or political agendas through Information Operations. The term Information Operations is defined by Facebook in its whitepaper [1] as:

"…*Actions taken by organized actors (governments or non-state actors) to distort domestic or foreign political sentiment, most frequently to achieve a strategic and/or geopolitical outcome. These operations can use a combination of methods, such as false news, disinformation, or networks of fake accounts aimed at manipulating public opinion...*"

The need for automation, and the lack of a clear-cut algorithmic process to verify an article makes this problem an interesting target for data mining, natural language processing and machine learning based solutions.

1.1 Case Study – Russian Involvement in US Elections

In 2017, in the aftermath of the US Elections of 2016, it came to light in several investigations that Russian political actors had extensively used social media to subvert elections in many countries[1]. Paid Russian trolls as well as automatic bots created several pages and distributed messages via social media favouring candidates as well as spread fake news and propagandist articles to sow discontent between

different groups to socially engineer a favourable election result. Representatives from Facebook and Google were strongly rebuked in a Congressional hearing for their failure to identify and curb these activities. 'Fake news' played a prominent role in these activities.

2 PROBLEM DESCRIPTION

Text-Mining is an approach that uses Language-Analysis tools such as TFIDF to represent the local-information of an article. Given a corpus of articles, as seen in the Kaggle Getting Real about Fake News challenge, a TFIDF vector for the corpus is generated, and then Logistic Regression, Random Forest, and XGBoost models are trained to classify the articles.

Problem Statement: Given an *Article x*, comprising of a *Body*, *Headline*, and *Website Domain*, learn *F(x)*, where *F(x) maps x to the class set C = {1: Fake, 0: Verified}*

Information about an *Article x* can be divided into two primary components, namely *meta-information* and *local-information*. *Meta-information* pertains to external information about the article, such as publisher reputation, publication date-time, and article tags issued by the publisher. In contrast, *Local-information* is extracted solely from the *Headline* and *Body* text strings of the Article. The breadth of potential features that can be extracted from both an Article's meta-information and local-information allow for multiple approaches to build the required classifier.

Subproblem 1: Given an *Article x*, from an *Article corpus X*, develop a classification model that extracts textual-features, learns the corresponding parameters, and generates $\exists$ *F(x)*, where *Fmodel1(x) maps* x to the set *C = {1: Fake, 0: Verified}*

Another method to using *Local-information* is extracting the topic of an *article*, identifying its *stance* on the topic, and comparing this *stance* to the *stance* of other articles with the same topic. This is akin to human process of examining different articles to *fact-check* the claims of the article in question. This leads to:

Subproblem 2: Given an *Article a*, and another *Article b*, develop a *stance* prediction model that extracts textual-features, and learns *F(a, b)*, where *F(a, b) maps* the tuple (*a, b)* to *the set S= {Discusses, Agree, Disagree, Unrelated, }*

Stance Definition: A Stance is a discrete representation of the mapping between a headline and an article, i.*e. for* $\exists$ *(a,b), a,b* ϵ *ArticleSet, f(a,b)* ϵ *{Agree, Disagree, Unrelated}*. If the claim made by *Article A*, matches the claim made by Article B, *then f(a,b) = Agree*. If the claim made by *Article A*, is the opposite of the claim made by *Article B*, then *f(a,b) = Disagree*. Otherwise, *f(a,b) = Unrelated*

Meta-information of an article is actively used by humans in discerning its trustworthiness. The reputation of the journalist or the publisher is an important consideration when verifying an article. The *trustworthiness (Tw)* of an article is evaluated as follows:

$$Tw(domain) = P(real|domain) - P(domain|fake)$$

$$= \frac{Count(domain,real)}{Count(domain)} - \frac{Count(domain,fake)}{Count(fake)}$$

At the same time, two articles on the same topic, but from different domains, could have similar textual-features but differing truth-values. An ensemble model is required to firstly weight the stances of an article by its domain trustworthiness, and secondly combine the approaches in *Subproblem 1* and *Subproblem* 2 to classify an article.

Subproblem 3: Given an *Article a:*

- compute *Tw(a) for domain trustworthiness*
- *Compute* $F_{model\text{-}1}$*(a) for dataset specific score*
- use *Subproblem 2* to compute *P (a == True)*
- generates *F(Tw(a)*P(a==True),* $F_{model\text{-}1}$*(a))*, where *F (a, b) maps* the tuple (*a, b)* to *class set C = {1: Fake, 0: Verified}*

3 LITERATURE REVIEW

While fake-news detection might be a relatively nascent field, text-mining of news articles is an established body of research, and several techniques from these fields are borrowed to build our classifiers.

3.1 Feature Extraction Review

Feature extraction for Fake News Detection is split into the two general fields of News Content Features and Social Context Features.

News Content Features: News Content Features are extracted from attributes such as Article Source, Headlines, Body Text, and Images/Videos. These features can be language-based such as presence of clickbait titles, hate-speech, and long-winded rant-like sentence structures, among others. Text-mining techniques such as *n-grams*, *bag-of-words*, and *Parts-of-Speech* (POS) tagging can help identify such syntactical features [6]. Domain-specific language structures, i.e. features that are specific to News

Articles, such as quotations, number of citations, quality of external links can also be used to capture hints of deception or at least poor-quality journalism, giving us, an idea of which articles are well-researched and substantiated, and which articles are fake-news simply masquerading to be professional journalism [2].

Social Context Features: Such features depict how the article has propagated across the media, from publication to social-engagements such as sharing, liking, and re-tweeting. User-related Social Context Features include frequency of posting, posting time-of-day, and sentence templating to check whether the original publisher is a bot, or an actual human [2]. Group-related Social Context Features create *Sharing-Graphs* to verify whether a group of members constantly share or re-tweet between themselves to make the article more popular, or whether the article's dissemination is more organic.

3.2 Machine Learning Model Review

There is no specific Machine Learning model that consistently outperforms other models, or is inherently better suited to Fake News detection. Support Vector Machines are conventionally used as the baseline model, and its performance (not tuned) ranges from 50% to 70% across different fake-news datasets [4].

A key concern in Fake News detection is that models are overfitting on certain textual parameters such as some specific *n-gram* or *TFIDF* vector column. Ensemble models such as XGBoost and Random Forests are shown to learn more nuanced rules about the parameters, and thus reduce overfitting [5].

Lastly, Deep Learning has been used extensively, especially Recurrent Neural Networks, in hopes that the model is able to extract more information from temporal sequences of textual features than static models. While RNNs, especially LSTMs are shown to outperform Ensemble Decision-Tree based models, the performance increase is reported to not be significant [4].

Lastly, it is shown that models trained on one fake-news corpus are not generalizable to articles from a topically distinct corpus. This is especially true for textual-features based machine-learning models, which rely on the presence of corpus-specific *n-grams* and sentence structures to classify an article [5].

4 EVALUATION METRICS

To evaluate our model that we developed for the discussed problem, various evaluation metrics have been used. In this subsection, we define the following terms:

- *True Positive (TP):* when predicted fake news is fake news
- *True Negative (TN):* when predicted true news is true news
- *False Negative (FN):* when predicted true news is fake news
- *False Positive (FP):* when predicted fake news is true news

Using these definitions, we compute the following metrics:

$$Precision = \frac{|TP|}{|TP| + |FP|}$$

$$Recall = \frac{|TP|}{|TP| + |FN|}$$

$$F1 = \frac{Precision - Recall}{Precision + Recall}$$

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

These metrics are frequently used in machine learning for showing the performance of classifiers. In our specific context, accuracy depicts the similarity between the predicted and real fake news. Precision measure the subset of all detected fake news that were classified as fake news. But it is relatively easy to get a high precision, which is possible by making very few positive predictions. This is where recall comes in. Recall is used to measure the sensitivity, or the true positive rate which is the fraction of annotated fake news articles that are predicted to be fake news. F1 is used to combine precision and recall, which can provide an overall prediction performance for each our models.

5 DATASET SPECIFIC FEATURE MINING (Subproblem 1)

5.1 Introduction

The prominence of hate-speech, and extreme-bias in Fake-News articles makes language-analysis based methods a promising approach. This is because of the relatively higher frequency of derogatory phrases such as 'Muslim-Invasion' and 'Crooked-Hillary', which can be captured by n-grams. TFIDF vectors extract such prominent n-grams by calculating their frequency relative to the corpus, and can be

passed as input vectors to machine-learning models to build a classifier [8]. Given a dataset consisting of news articles (*A)*, let A_{text} represent the textual content of the article. A_{text} is a tuple of $A_{headline}$ and A_{body}, each of which contain a *String* object storing the text data of the respective article section. $A_{headline}$ and A_{body} can be represented as a TFIDF vector [4,12].

5.2 Dataset Description

The data-set is unique, and has been constructed from existing fake-news data-sets from the Fake News Challenge, and Kaggle's *Getting Real about Fake News Challenge*. It has then been enhanced by web-scraping news articles from verified sources such as The Guardian, New York Times, and Bloomberg, using a custom-built web-scraper through the libraries *Scrapy* and *BeautifulSoup*.

Table 1: Dataset Statistics

	Count	Fake article Count	Verified article Count
Overall	27473	18409	9064
Train	18406	12299	6107
Test	8160	5514	2646
Validate	907	596	311

5.3 Data Exploration

We observed from the visualizations that the words that were commonly occurring in fake news articles fall into three main categories:

Strong Adjectives: Words like 'brutal' and 'massive' are the most frequently used in fake news articles as clickbait material for attracting the attention of internet surfers.

Politically and Religiously Charged: Common words that came up during the mining showed that words instigating political and religious bias were popularly read by the respective sectors.

Name-Calling: These appeared in fewer articles and tries to instigate hate and violence against sectors of the society.

Table 2: Commonly Occurring Word Groups

Adjectives	Political Religious	Name-Calling
Brutal	Brexit	Crooked-Hillary
Aggressive	Mosul	Islamic Pests
Massive	Obama	
Destructive	Trump	

The prominence of hate-speech, and extreme-bias in Fake-News articles makes language-analysis a promising approach.

5.4 Feature Extraction

One interesting point which emerges from our preliminary observations is that the text provides a valuable insight into determining the fakeness of the given article. Hence, we decided to extract an array of well-known text features, few of which we have described below.

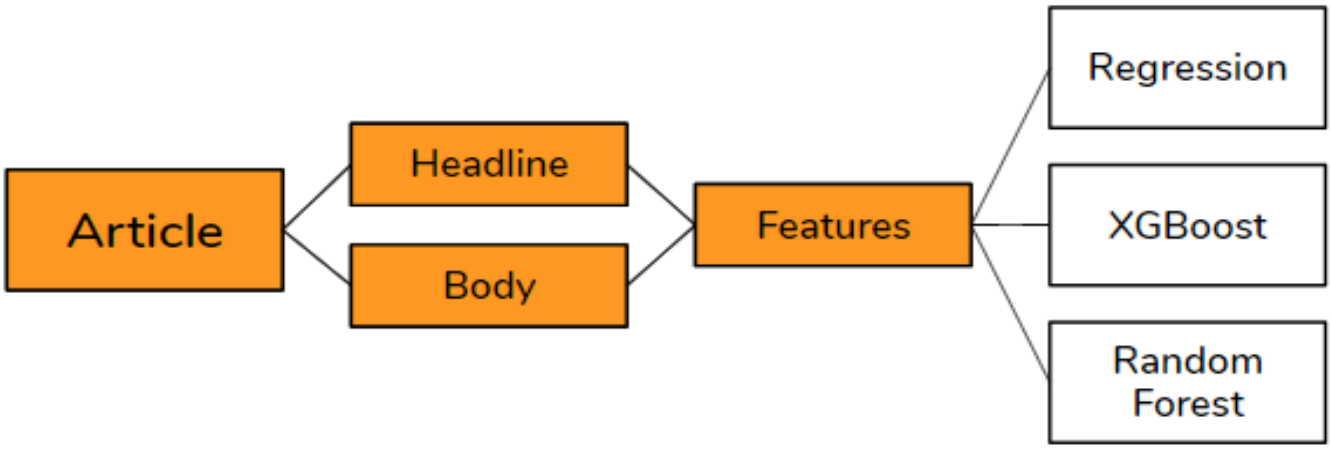
TF-IDF: Term Frequency / Inverse Document Frequency is a statistical measure that reflects how important a term is in a document or a collection of documents. It is used as a weighting factor in information retrieval, data mining, etc. i.e. it is used to reflect the relative contribution of different words to a document. Term Frequency is simply a measure of a term in the document, which approximates the importance of the term. However, it does not consider the fact that certain words just appear more frequently in the language or that particular corpus. Therefore, Inverse Document Frequency, a measure of how frequently a word appears in the corpus is used to offset Term Frequency to provide a more accurate approximation of the term's importance.

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

For example, if we wished to rank a set of documents based on which is most relevant to the query 'the nuclear war', we should prioritize occurrence of the term 'nuclear' more than



the term 'the' 'the' occurs way too frequently and does not necessarily indicate that the given document is very relevant to the query. That is the information TF-IDF provides us.

N-Grams: An n-gram is a contiguous sequence of n items that forms a given sequence of text or speech. These items could be syllables, words, letters or anything else based on the application it is being used for. An n-gram of size 1 is called 'unigram', of size 2 is called 'bigram', of size 3 is called 'trigram' and so on [13].
From the sentence '*The US President Donald Trump tweeted …*', unigrams will be '*The'*, '*US'*, '*President'*, '*Donald'*, … and the bigrams will be '*The US'*, '*US President'*, '*President Donald*', '*Donald Trump'*, and so on. We can use n-gram frequency to match similar articles. For example, matching frequency of the n-gram 'Trump tweeted' in different articles is a much stronger indicator that they are similar compared to just matching frequency of the words 'Trump' and 'tweeted'[10].

Figure 1: Data Pipeline Schema for Text-Mining

Bag of Words: Bag of words is a simplifying representation of a text or document solely as a multiset of its words, disregarding their relative order or grammar, but preserving their frequency. Bag of word representation is a prerequisite for extracting useful features from the document for its classification, for instance, term frequency. This model is commonly used in natural language processing and computer vision[10].

5.5 Models Experimented

On extracting a preliminary set of features, we fed them into three different models to train the prediction of the news articles.

XGBoost: XGBoost stands for extreme Gradient Boosting. It is an open source software library which provides a gradient boosting framework for C++, Java, Python and many other languages. It is used for supervised learning problems, where we use the training data (with multiple features) Xi to predict a target variable Yi. It is developed with both deep considerations in terms of systems optimization and principles in machine learning. Over the years it has gained a lot of popularity in Data Science circles and it is the algorithm of choice in Kaggle competitions.

Linear Regression: Linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables. It is now increasingly used as an algorithm in machine learning applications. Linear regression assumes a linear relationship between the input variable and a simple output variable. Based on the training data, it allows us to form a model which allows future predictions. Linear regression model's simplicity makes it a very attractive model.

Random Forest: Random Forest is one of the most popular and most powerful machine learning algorithms. It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging. Random Forests are an improvement over bagged decision trees. A problem with

decision trees like CART is that they are greedy. They choose which variable to split on using a greedy algorithm that minimizes error. As such, even with Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions[10]. Random forest limits the learning algorithm to a random subset of features to search and therefore avoids this problem.

5.6 Result Summary

Below are a summary of the results we have obtained from the different models that we tested:

Table 3: Model Performance Statistics

Models	F1 Score	Accuracy
Logistic Regression – Headlines	69.58	75.83
Logistic Regression – Body	74.87	75.49
Random Forest – Headline	70.54	76.74
Random Forest – Body	89.07	88.71
XGBoost – Headline	95.33	91.28
XGBoost – Body	92.49	93.75

Both Random Forests, and XGBoost offer a marked improvement in accuracy and F1-Scores over the baseline model of Logistic Regression. The key take-away is the success of ensemble models. We suspect there are three reasons for this is:

1. Decision-Tree based models are able to unearth more complex rules than Regression models, and are able to identify singular features or feature-combinations that are dominant.
2. Ensemble Decision-Trees in general reduce variance because the average opinion of many models would be less noisy than that of one model [9].
3. Ensemble Decision-Trees are less likely to over-fit than regression models. This can be clearly seen in the case of F1-Score, and Accuracy charts against number of samples during cross-validation. While the F1 and Accuracy Scores suddenly rise for Linear Regression after a critical mass of samples during training, this trend is not supported during testing.

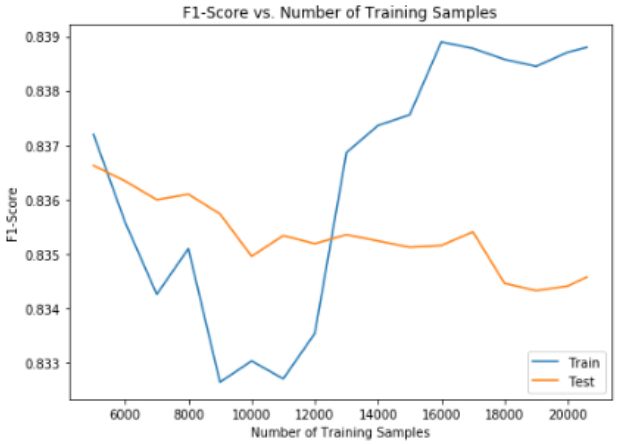


***Fig 2: F1 Score of Logistic Regression* - Headline TFIDF**

***Fig 3: F1 Score of XGBoost using* Body TFIDF**

As can be seen, XGBoost seems to be actually learning complex decision rules, since both testing and training scores are steadily increasing with the number of samples. Whereas, for Logistic Regression, the testing scores are decreasing even though the training scores are increasing.

Ultimately, the XGBoost algorithm was finalized, because it best incorporated the advantages of Ensemble Models over Logistic Regression, had the highest accuracy, and overfit the least.

6 STANCE DETECTION MODEL (Subproblem 2)

6.1 Motivation

In the previous section we described the functioning of our model which uses text features from an existing dataset to predict classify articles as fake or real. While the model achieves a decent accuracy on the testing set, it only uses mined text occurrences in the form of TFIDF to classify fake news. However, articles with similarity in TFIDF vectors can have opposite veracity. As a result, a Stance-Detection model is developed, such that if some *Article T*, has an opposing stance to a verified *Article V*, then it is likely to be *Fake*. Consider the following two headlines:

H1: "Kremlin Denies Meddling in US Elections" - Fake
H2: "Kremlin Officials Confess US Manipulation" - Fake

As shown above, H1 and H2 are contradictory statements yet both are classified as being fake. This is due to the heavy reliance on dataset specific text features. The previous model may have learned to associate the term "Kremlin" with fake articles based on the training dataset. Thus, any new article with the term "Kremlin" and other less relevant terms may be classified as fake.
We aim to make our model more '*symmetric*' and '*global*' by incorporating a domain weighted stance computation model. We define the above two terms as follows:

Symmetric: A model is symmetric if for two contradictory articles A1and A2,
*fake_score (A1) = 1 - similarity (A1, A2) * fake_score (A2)*

Global: A model is global if while computing fakeness of an article *A1*, it uses information from all articles related to *A1*.

6.2 Stance Computation Model

In this section, we describe the methodology of developing the stance detection model. The stance model computes for any given pair of headline and body, the following:

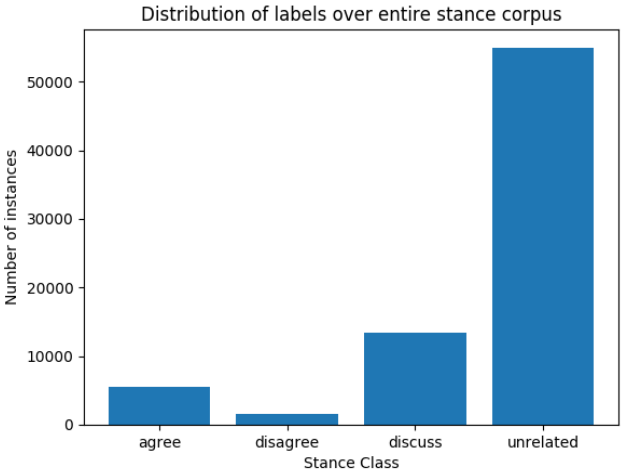
$$stance(headline, body) = argmax_x P(x|headline, body, \theta)$$

$$where\ x \in \{Unrelated, Discusses, Agree, Disagree\}$$

As shown above, the model outputs stance by learning a set of parameters θ on the training data to identify the highest probability stance class.

6.3 Dataset

To develop the stance detector model, we obtained data from the http://www.fakenewschallenge.org/ . Each row in the training and validation data consists of three columns: headline, body and stance. The testing data consists of just rows of headline and body. The dataset statistics are summarized as below:



	Total Articles	Unrelated	Discuss	Agree	Disagree
Overall	75115	54894	13373	5581	1537
Train	40350	29647	7109	2916	678
Validation	9622	6898	1800	762	162
Test	25143	18349	4464	1903	697

Table 4: Stance Dataset Statistics

As can be seen from the bar plot in Figure 2, majority of instances are unrelated. This is to ensure that the classifier model is robust to only extract key features that signify whether an article body discusses a given headline.

6.4 Feature Extraction

Given the dataset for stance only comprises of text headline and articles, we derive more text features.

1) *TFIDF for headline and body:* We again made use of TFIDF as described in section _. The TFIDF is obtained jointly for headlines and bodies. Thus, given related headlines and bodies, the TFIDF representation should be similar.

2) *Cosine similarity of TFIDF representations:* The cosine similarity is a popular similarity measure between two vectors by using the angle between them. It is computed as follows:

$$cosine\ similarity(a, b) = \frac{A.B}{|A|.|B|}$$

We use the cosine similarity between the TFIDF representation of headline, and TFIDF representation of body as a feature in our model.

3) *Word Embeddings (Word2vec vector representations):* Word2vec is a neural network based language modelling technique which maps words to vectors of a fixed dimension length. The vectors capture context dependent usage and co-occurrence of words. Using word vectors as features is extremely useful in our task since it encompasses natural language processing features such as synonyms, plurals, verb tenses and related corpus words. For example, since 'UFO' and 'aliens' are likely to occur in the same context frequently in a corpus, they will be situated close to each other in the vector space. This property is shown in the image below [13]:

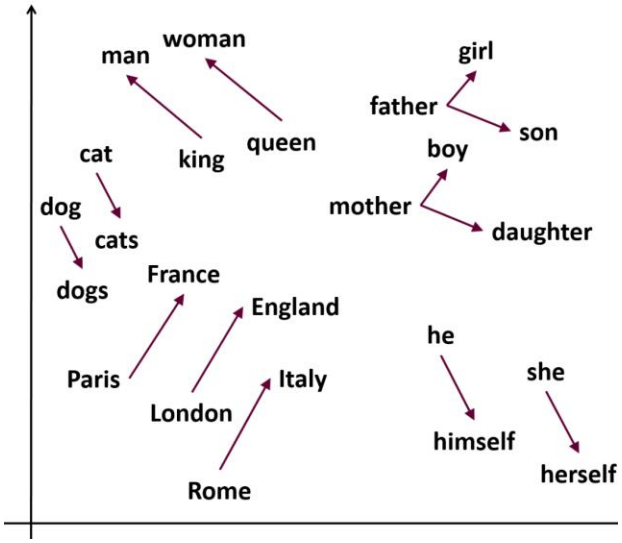


Figure 5: Word2Vec Representations

- We obtained pre-trained word vectors from https://github.com/mmihaltz/word2vec-GoogleNews-vectors . It consists of 300 dimensional representations of 3 billion English running word tokens.

4) *Singular Value Decomposition:* Singular value decomposition is a matrix factorization approach which decomposes a given matrix A into the three matrices U V D given by the following:

$$A = U.D.V^T$$

The obtained values in *D* are positive entries known as singular values. Thus, SVD can be used as a dimensionality reduction technique. We make use of it as a feature by using the singular values obtained from performing SVD on the concatenated TFIDF representation of headlines and articles.

6.5 Experimented Models

The table below lists the performance of the models we trained for stance detection on the previously mentioned dataset.

Table 5: Model Performance Statistics

Models	Competition Scores	Validation Scores
Baseline	75.20	79.53
SVM	-	65.94
BiLSTM	75.80	80.25
BiLSTM+ Concat TFIDF	**80.80**	**85.52**

Note: Since the competition ended on Jun 2nd 2017, we only obtain scores by running the competition_scorer.py provided by the organizers.

6.5.1: *Baseline Model:* The baseline model used by the Fake News Challenge makes use of Gradient Boosting Classifier. The features used are: normalized word tokens, overlap of tokens in headline and body, specific refuting words and counts of *n-grams* and *char-grams*.

6.5.2: *SVM Model with TFIDF representation:* Our first model consisted of computing the TFIDF representation of headline and body and feeding the concatenated representation it into a Linear Support Vector Machine.

Support Vector Machines

Support vector machines are supervised learning models with associated learning algorithms that try to determine an optimal hyperplane to maximize the distance of hyperplanes between given classes. We make use of the LinearSVC package [8].

Model Architecture

The model is represented in the figure below:

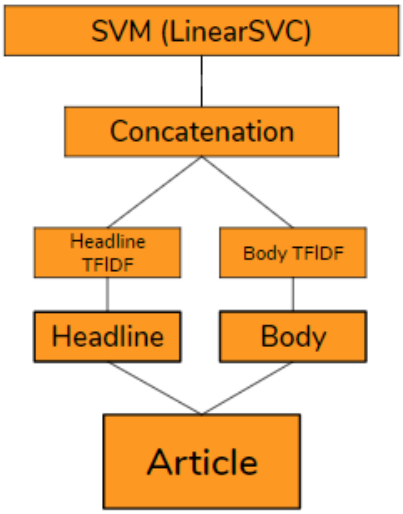


Figure 6: SVM based stance model

Model Performance

It achieves a poor validation score of 65.94% which is much lesser than the competition baseline, hence we do not score this model from the competition testing set. This may be because the data is not linearly separable and hence the SVM is not able to find a suitable decision boundary.

6.5.3: *Bidirectional LSTM with word2vec embeddings:*

Our second model consisted of using a deep learning based bi-directional LSTM model initialized with word2vec embedding representations. We first display the model architecture and then elaborate on the deep learning methods that we used for our experiment.

Model Architecture

In our experiments we make use of the Keras package with Tensorflow backend to perform experiments. Additionally, the following models are run on a GPU processor given the high dimensionality of input data.

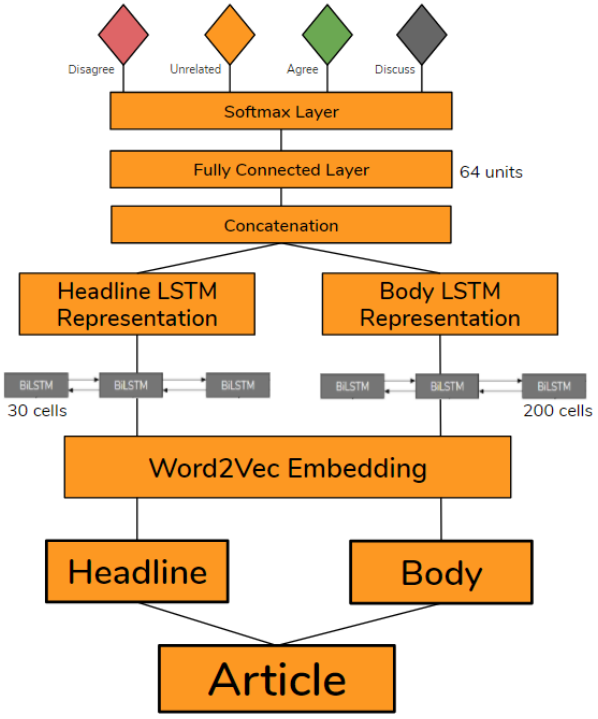


Figure 7: Bidirectional LSTM based stance model

Fully Connected Artificial Neural Network

A fully-connected Neural Network uses a non-linear activation function to learn complex non-linear decision rules. For example, using the standard *sigm*oid activation function the output *h1* is:

$$h1 = \text{sigmoid}(\mathbf{w}.\boldsymbol{x})$$

Where the *sigmoid* function is defined as:

$$sigmoid(x) = 1/(1 + e^{-x})$$

Other non-linear activation functions include *Tanh*, *Rectified Linear Unit*, and *Softmax*.

Long Short-Term Memory Network (LSTM)

Recurrent Neural networks (RNNs) is specific type of artificial neural network where connections between hidden units form a directed cycle. Hence, it can capture sequential dependencies for input data. This is extremely useful in natural language processing tasks, since written text often follows a sequential relation.

However, traditional RNNs suffer from vanishing gradients, due to which they lose information over long sequences. This severely inhibits their performance in

capturing long sequences. Hence, LSTMs are introduced, which contain of memory cells, and more parameters to capture long term dependencies.

A Unidirectional LSTM only preserves information of the past because the only inputs it has seen are from the past. In Bidirectional LSTMs, inputs are run once from past to future and then from future to past, and in this way it is able to preserve both those sets of information in its hidden states.

Softmax Output

The Softmax function is used to represent a categorical distribution, i.e. the probability distribution function of K different events. Softmax is designed to assigns the largest probability to the most frequently appearing category, and suppress the values that are significantly out of line with the average. Given an input vector x, the probability that the output y = j given by:

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

Mean Cross-Entropy Cost Function

The cross-entropy measure has been used as an alternative to squared error. Cross-entropy can be used as an error measure when a network's outputs can be thought of as representing independent hypotheses (e.g. each node stands for a different concept), and the node activations can be understood as representing the probability (or confidence) that each hypothesis might be true. In that case, the output vector represents a probability distribution, and our error measure - cross-entropy - indicates the distance the network's output distribution for stance labels against actual one hot encoded labels. The cross-entropy between a "true" distribution p and an estimated distribution q is defined as:

$$H(p, q) = -\sum x\, P(x) \log(q(x))$$

RMSProp Function

RMSProp is an improvement of Stochastic Gradient Descent (SGD). It keeps running average of its recent gradient magnitudes and divides the next gradient by this average so that loosely gradient values are normalized.

Dropout Regularization

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks.

Model Performance

As shown in Table 5, our bidirectional LSTM performs much better than the previous SVM based model and also beats the baseline during validation as well as competition testing set. However, we realized by adding more features to the deep learning model we could improve its performance.

***6.5.4:** Bidirectional LSTM with word2vec embeddings:*

Our final model consisted of using a deep learning based bi-directional LSTM model initialized with word2vec embedding representations. In addition, we concatenated TFIDF representations of headline and body. Finally, we also use the cosine similarity between both the LSTM representations as well as the TFIDF representations.

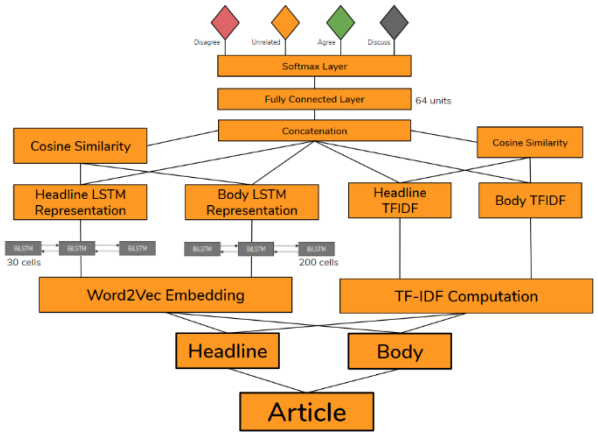


Figure 8: Deep Learning Model Pipeline

As shown above, our model now incorporates TFIDF features as well as cosine similarities between representations. The final layer input is a concatenated representation of [Headline and Body LSTM, Headline and Body TFIDF, Cosine similarity between Headline and Body LSTM, Cosine similarity between Headline and Body TFIDF]. This is fed into a fully connected neural network

layer of 64 units. The hyper-parameters for our final model are given below:

HyperParameter	**Definition**
LSTM hidden units	64
Headline LSTM length	30
Body LSTM length	200
Fully connected layer units	64
Dropout	0.2
Cost function	Mean Cross Entropy
Optimizer	RMSProp
Batch size	32

Table 6: Deep Learning Model Hyper-Parameters

Model Performance:

As shown in Table 5, the model achieves a validation accuracy of 85.52% but a competition testing score of 80.8%. This is shown in the plot below:

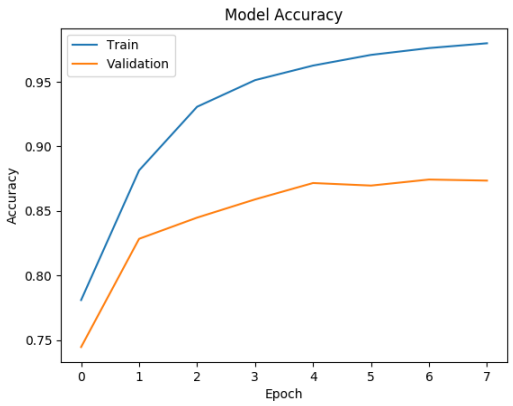


Figure 9: Stance Model Accuracy over Epochs

As can be seen from the plots, our model overfit on the training data after epoch 8. To counter this more number of epochs are run. Given GPU restrictions, we limited to GPU running time of 2 hours (40 epochs). Typically winning models are run over 100 epochs.

Additionally, the model's score on the competition testing set, when run using the competition scorer resulted in a competition score of 9148.500. This corresponds to a private rank of 13th out of 50 competitors in the competition leaderboard although the competition has now ended [11].

The final testing confusion matrix is given below:

	Agree	Disagree	Discuss	Unrelated
Agree	1046	36	256	565
Disagree	17	339	185	156
Discuss	401	209	1778	2076
Unrelated	130	88	750	17381

The model accuracy came to 80.8%. The maximum possible score in the fake-news challenge is 11651.25. The model scored 9147.0. This would rank at 13th corresponding to 50 historical participants.

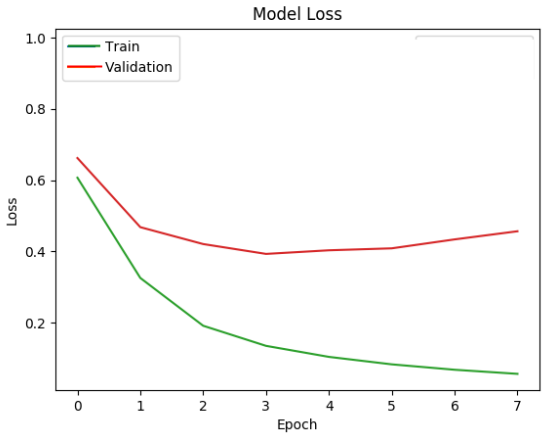


Figure 10: Stance Model Loss over Epochs

7 COMBINED MODEL

7.1 Motivation

From our observations in section 5 and 6, we see that both the Dataset-specific text Feature Mining and the Stance Detection Models provide us with valuable insights into determining the fakeness of a given news article. Our final model proposes a novel combination of the above two models to determine the fakeness of the article.

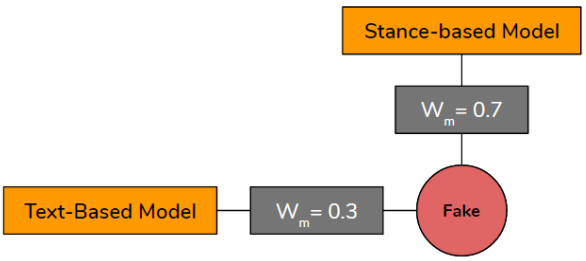


Figure 11: Combined Model Architecture

We combine the final predictions made by the model described in Section 5 and in Section 6 in our final model using a weighted average. The weights can be learned using different methods. The major limitation that we faced while was that the Text Based model was trained on a different dataset as compared to the Stance Based model. As a result, applying the combined model to either data-set would result in poor performance, since the features of both models are specific to the corpus they were trained on. A solution to this problem would be to either label the news articles in the stance data-set as '*fake'* or '*verified*', or assign stance labels to tuples of news articles in the fake-news Data-set.

8 FUTURE RESEARCH

The project is still in its nascent stages, and future research and development will focus on the following three sections:

8.1 Enhancement of Textual Features

Apart from *TFIDF* vectors, sentiment-extraction from an article body is shown to be a promising feature in classification of news articles [4, 6]. This is because of the extreme and imbalanced sentiments expressed in some fake-news articles, especially in hate-speech.

Another relevant technique is known as *Claim Extraction.* This technique is a subset of the broader technique of text - summarization [11]. Explicitly extracting and comparing the claims made by an article could yield better performance than implicitly extracting claims through vector representation of an article, and this could make Stance Recognition more effective.

8.2 Optimization of LSTM Hyperparameters

Due to limited access to a GPU workstation, it was difficult to fine-tune the hyperparameters of the LSTM such as the number of dropout, and pooling layers, the number of neurons in each layer, and other parameters. Stacking multiple layers of RNNs would bring temporal hierarchy to the model and allow it better to understand the changes in textual-features over time.

8.3 Framework Packaging and API:

Ultimately, the goal for FakeNet is to become a general library for fake-news detection, where end-users can plug in their corpus, train the models, and begin article classification. To this end, there is already a function that lets users input their article in a String format and returns a classification based on the existing corpus. However, providing an API to re-train on a supplied corpus is the next step. Speed enhancements to the training and testing process are critical as well.

9 CONCLUSION

This paper successfully demonstrates the validity of textual-feature, article-stance, and domain-weighted ensemble approaches to fake news classification. In the traditional machine-learning category, XGBoost delivers the best performance due to the low variance, and reduce overfitting that is typical of ensemble models. The Stance Bi-directional LSTM is ranked 13th out of 50th using the Scorer Module of the Fake News Challenge. The key innovation is combining a stance-based prediction model with tradition textual-features based models, thereby leveraging on a separate area of research to enhance fake-news detection algorithms. The implementation details along with the source code can be found at: https://github.com/shantanuj/Fake_Net

Model performances however, were sensitive to the percentage distribution of fake-news to real-news in the corpus. Since the real-world distribution of fake to real-news is unknown, it was difficult to design an ideal corpus. Additionally, the fake-news models presented here are

corpus-dependent and not generalizable. For example, the Bloomberg scraping focuses heavily on Trump articles, and thus it fails to recognize fake-news articles in domains outside of Politics.

REFERENCES

[1] J. Weedon, W. Nuland, A. Stamos, “Information Operations and Facebook”, ACM SIGKDD Explorations Newsletter, 2017

[2] S. Kai, S. Amy, W. Suhang, T. Jiliang, L. Huan Fake News Detection on Social Media

[3] T. Kanan, E. A. Fox, "Automated Arabic text classification with P-Stemmer machine learning and a tailored news article taxonomy", J. Assoc. Inf Sci. Technol., 2016.

[4] C. C. Aggarwal, C. Zhai, "A Survey of Text Classification Algorithms", Mining Text Data, pp. 163-222, 2012.

[5] M. Kompan, M. Bieliková, News Article Classification Based on a Vector Representation Including Words' Collocations.

[6] D. Y. Liliana, A. Hardianto, M. Ridok, "Indonesian News Classification using Support Vector Machine", Int. J. Comput. Electr. Autom. Control. Inf. Eng., vol. 5, no. 9, pp. 1015-1018, 2011.

[7] D. M. Blei, A. Y. Ng, M. I. Jordan, "Latent Dirichlet Allocation", J. Mach. Learn. Res., vol. 3, pp. 993-1022, 2003.

[8] G. Heinrich, Parameter Estimation for Text Analysis, 2009.

[9] D. X. Zhou, P. Resnick, Q. Mei, Classifying the Political Leaning of News Articles and Users from User Votes Semi-Supervised Learning Algorithms, pp. 417-424.

[10] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, Lucia Special, SemEval-2017 Task 1: Semantic Textual Similarity - Multilingual and Cross-lingual Focused Evaluation.

[11] Régis Riveret, Pietro Baroni, Yang Gao, Guido Governatori, Antonino Rotolo, Giovanni Sartor: A Labelling Framework for Probabilistic Argumentation.

[12] Johannes Furnkranz. A study using n-gram features for text categorization. Austrian Research Institute for Artificial Intelligence, 3(1998):1–10, 1998.

[13] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. arXiv preprint arXiv:1702.05638, 2017.