Fake News Detection on Social Networks

Yiyang Sun, Handong Xu

Supervisor: Ratan Dey

Abstract

Fake news has always been a problem since ancient days. However, with the help of online social media, fake news can spread more easily and quickly nowadays. Meanwhile, the authors of fake news have also adapted. The methods of reporting the real news are applied (by the reporters) in creating the fake news. It is really hard for people to differentiate fake news from real news. Therefore, we tried different machine algorithms and designed a text CNN model to detect fake political news from social media. By using the word frequency dictionary, the model outperformed traditional detection systems in terms of both accuracy and running time.

I. Introduction

People nowadays can easily retweet or repost messages on social media. Without checking the authenticity of the content, it becomes very easy for people to share fake news to their friends by mistake. Fake news on social networks like Twitter, Weibo, Wechat, etc can misguided lots of people and have caused great damage to society. For example, in the year of 2018, a piece of news wrote that a group of Russian hackers invaded the U.S. electricity grid during the winter.[1] But it turned out this article was not written based on solid facts but vague hypothesis. According to the Knight Foundation, more than 6.6 million tweets circulated across Twitter in the month leading up to the 2016 election that were tied to fake news and conspiracy news publishers[2]. One famous example is that Hillary Clinton was rumored to be a Russian spy before the day of the election[3]. Lots of people reposted this fake news which to some degree affected the election results.

After realizing the consequence of fake news, many companies nowadays start to hire people to label fake news manually. However, such a job wastes a lot of time and resources and cannot detect fake news when they first come out online. To detect fake news more effectively and quickly, many algorithms have been designed with the help of deep learning. And many states of the art results have been realized which encourage our further research into this area.

II. Related Works

Many innovative methods have been introduced in the area of fake news detection over the past few years. Right now,there are two major detection approaches to detect fake news. One is called the early stage detection and the other is the propagation detection.

A. Early stage detection

The main goal of early-stage detection is to capture the different features of fakes news compared with the true news. A recent study put its focuses on discovering the most relevant features of the fake news by implementing a simple linear regression on a labeled dataset[4]. The researchers started from selecting 45 features and then tested the significance level of each feature in the linear model. Their result showed that only 7 out of the 45 features were proved to be effective with an overall accuracy around 70%. This indicated that fake news can be detected with only a few parameters. But it needs to be noticed that these 45 features are selected by people's common sense on what a fake news could look like. Thus they are biased to researchers' subjectivity. Another intuitive way to tackle the fake news detection problem is from a syntactic stylometry perspective[5]. Only three features are included in this model which are words, shallow syntax, and deep syntax. One example of the shallow syntax is that fake news tends to use verbs and personal pronouns like 'I', 'my' more often. While true news tends to use more of nouns, adjectives, prepositions. The deep syntax is the probabilistic representation of a sentence which is calculated based on the overall dataset. The result from this method showed that after training with Support Vector Machine (SVM), it was able to increase the accuracy by 14% compared with the previous models[6]. The good side of this approach is that the result can be easily interpreted and understood by human. And it's more intuitive in the sense that fake news usually has different writing styles than the true news. However, features that can be captured by this model were very limited. More hidden features of fake news were ignored by the model. Thus some recent studies proposed a new method which use Convolutional Neural Network (CNN) to classify sentences[7].

Traditionally, CNN is used in the area of image detection. A few recent studies have proved that CNN is also efficient in detecting the features of texts. Some researchers designed a hybrid CNN model. This hybrid CNN model included not only text but also meta-data of the text speaker[8].

Another interesting hybrid model called TI-CNN mainly focused on detecting latent features of the online posts[9]. This TI-CNN model was trained on both the texts and the images from online posts. In order to reduce the dimension of the text vector, this model first apply the word2vec model to their text data which decrease the risk of model overfitting[10]. Then, by applying multiple filters to the text, more latent features are found by the CNN. Since they also add images to the dataset, there are more data available to train the model which helps to improve the result. However, the cost is that the result is more complicated to explain. It is hard to tell how much impact do the image have in the process of detection. And also the fact that they chose to combine text and image to train the model was under the assumption that fake news is always related to some specific types of images. But there is no other supporting evidence or researches to support it. This makes this combination less meaningful. So instead of putting images into the dataset, some researchers took the user comments of posts into account[11]. Based on the historical user comments to posts, they used a Conditional Variational Autoencoder (CVAE) to generate some comments for the new post that the model had never seen. And then their model used these generated comments together with the posts to do the detection. However, this method could be biased due to the fact that these automatically generated comments can be affected by the historical comments available. These comments added more variance and bias into the model.

In general, people are trying to add more information to their CNN model besides the texts themselves. But there is little solid proof or justification for these information to be added. And it is hard to tell weather the good performance of such models is actually the effect of those extra information or it's simply an overfit to the dataset.

B. Propagation detection

Different from the early stage method above, propagation detection utilizes the information from the post's distribution route online over time to detect fake news. A group of researchers from China used Sina Weibo as dataset and designed a tree structure of the propagation of the posts[12]. The assumption is that fake news has a different propagation route than true news. For instance, a post sent directly from a public figure with credibility is less likely to be fake than a post from a normal user. This provides another way to think about the fake news detection problems by taking the network effect into account. Another angle on the analysis of propagation is to think of the posts and the reposts as time series data. A model is designed using a dynamic

series-time structure for classification[13].This structure traced the grammar changes of relevant posts to the topic and decide whether this topic is fake or not. While the major concern for the propagation method is that if we can only detect fake news after it had been spread out around the internet, then how can we undo the damage that the fake news have already made to the society.

C. Our Insights

From the summary table below, it can be seen that with proper tuning of parameters, both early stage and propagation methods can achieve a high accuracy. In order to put the model into practice use, we choose to focus on the early stage detection method. And our objective is to not only achieve a high accuracy but also a shorter training time compared with current models.

Related Paper	**Method**	**SVM**	**Neural Network**	**CNN**	**RNN**	**Highest Accuracy**
Syntactic Stylometry	Early Stage	Yes				91%
TI-CNN	Early Stage		Yes	Yes	Yes	92.1%
Propagation Structure of Sina Weibo	Propagation	Yes				91.3%
Time Series of Social Context	Propagation	Yes				89.6%

TABLE I

METHODS USED AND RESULTS ACHIEVED FROM LITERATURE REVIEW.

Table 1 shows a summary of the related papers' method and their highest accuracy. Both early stage and propagation stage methods achieved a high accuracy. But the data for early stage was easier to get, so we chose to focus on early stage detection. Since TI-CNN has a accuracy of 92.1%, our goal is to beat TI-CNN and achieve a higher accuracy.

III. DATASETS

Right now, there have already been a lot of well-labeled datasets related to fake news open to public. For the purpose of our project, there are a few constraints that we need to specify for our dataset.

First of all, the dataset should be related to political topics. And in order to make sure that the dataset itself is not biased, the range of topics should be as wide as possible. In the process of looking for datasets, we noticed that a few public datasets contain news only relates to Trump

and the last US presidential election. The limitation of topics covered would potentially affect our experiment result.

The second constraint is that the dataset should contain only text content and title of the text. The purpose is that we wanted our model to achieve a good result in a practical scenario. Though some state of the art result was achieved by adding metadata or even image data to the text, it would potentially cause the problem of overfitting to some specific dataset and cannot be put into a more general use case.

Last but not least, the dataset should have a simple binary label. A piece of news can either be labeled as true or fake and no other label is allowed in our experiment. Some research paper came to their conclusion that it is sometimes difficult to label a piece of news as completely fake or true and there should be some extra labels like not confirmed, partially true, etc. However, in order to suit our purpose of designing a detection model that can be used in practice, we decided to use simple labels to train our model.

After applying these constraints to the available datasets online, we shortlisted them into two.

A. Getting Real about Fake News from Kaggle[14]

This dataset contains text and metadata collected from 244 websites and represents 12,999 post. All the data is collected by a website extension which checks the links in the web page to see if they are unreliable sources and then classify the text into fake or true. The advantage of this Kaggle dataset is that it has a clear structure and clean text contents. A few papers had also used this dataset to train their model. [9] However, the length of the text in this dataset is relatively short. Also it only covers news on US presidential election from October 2016 to November 2016.

B. General Political News Dataset

In order to overcome the disadvantages of the Kaggle dataset, we built our own dataset for model training. The main goal is to include more political topics and cover a longer time range so that the model trained under this dataset can be put into practical use.

We chose to include political news on The New York Times as main source of data. And it is reliable to label these news as true since The New York Times has a good credibility in history. Besides that, we also collected labeled news from PolitiFact.com which is also a well-known organization aiming to manually check the credibility of political news on the internet. PolitiFact started in 2007, with reporters and editors work for different newspaper and news media partners

reporting on the accuracy of statements made by elected officials, candidates and others involved in U.S. politics. In total, there are 439 entries of news in this dataset with long text messages among which 50% are fake news.

IV. Our Method

A. Data Preprocessing

In the first phase, we wanted to process all the text data as image. The reason is that both text and image have a fixed structure and can be expressed in terms of vector form. The difference is that images use pixels as the basic element while text takes the form of word or character. For image detection problem, a vector representation is used to train the model. Then the first question is how to represent the text in a similar way. After literature review, we decided to try two ways to convert text into vectors and compare their results. One is building a dictionary based on the word frequency in the text and the other is the method called word2vec.

1) Word Frequency Dictionary:

We first created an embedding method based on the frequency of the word appeared in the training set. Each word is placed in an ordered dictionary and its index is used to create a corresponding vector. The advantage is that this method is easy to implement and more efficient to run. Also it is intuitive to understand. Since if a word appears a lot in the training set, then this word deserves more attention for the model compared with the word that rarely shows up in the text. While the disadvantage of this method is as clear as its advantage. If one word in the test set never shows up in the training set, we could not assign a proper vector but have to use a randomly generated vector. And this could be a problem if the train set and test set are significantly different in terms of the text.

To have a better idea of the result of frequency dictionary, we list the top 10 words in the dictionary: 1. 'trump', 2. 'clinton' ,3.'people', 4. 'hillary', 5. 'president', 6. 'election', 7. 'campaign', 8.'time', 9. 'obama', 10. 'donald'.

Figure 1 is a bubble graph visualization where the size of the circle represents the number of frequency in the dataset. We can see from the graph that Trump are most frequently word in the dataset following by clinton. From the result we can see that the news included in the datasets are mostly related to the US presidential election.

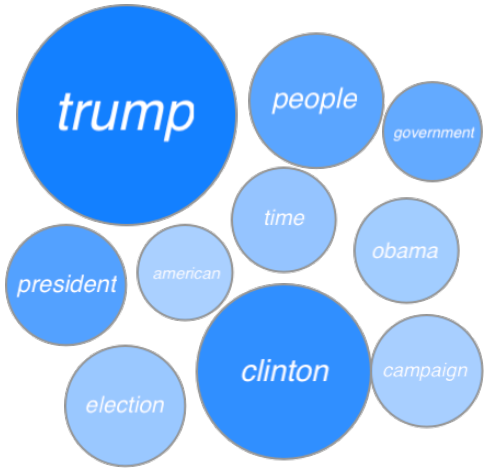


Fig. 1. Most Frequent Words

2) Word2vec:

While the dictionary based on word frequency is easy to implement and understand, it lacks the information of text context. Under frequency dictionary, words with similar meaning in the text context are not necessarily put close to each other in the vector space. This could potentially cause a bad performance in the model training process. In order to solve this problem, the word2vec method is also used in our experiment.

Word2vec combines group of related models which aim at producing word embedding. In general, word2vec is a shallow neural networks with two layers. It was first proposed by Tomas Mikolov and his group[15]. The main advantage of this method is that it can reconstruct the linguistic contexts of words. Word2vec takes a group of words as its input and produces a vector space. Each unique word in the group will be assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the group of words are located close to one another in the space[16].

There are two different approaches to use the word2vec model: one is Continuous Bag-of-Word (CBOW) and the other is Skip-Gram. CBOW takes the other words around the target word as input and gives the prediction of the target while Skip-Gram takes in the target word and gives the potential words around the target. We will use CBOW as an example to explain our preprocessing model in detail.

A one-hot vector of all the words around is used as input for the CBOW model. The dimension of one hot vector is defined as V, the number of words is defined as C. Then we multiply each

one-hot vector by the input weight matrix *W* to have our initialized vectors.

$$initialized\ vector = X_{1\times V} \times W_{V\times N} \tag{1}$$

Then we calculate the simple average of these initialized vectors as our input for the hidden layer.

$$input\ of\ hidden\ layer = \frac{1}{C}\sum initialized\ vector_{1\times N} \tag{2}$$

Now we can apply the output weight matrix *W* to the above result. And in order to have the probability distribution, we then use softmax function to obtain the final result.

$$result\ before\ activation = input\ of\ hidden\ layer \times W^{-1} \tag{3}$$

$$result\ after\ activation = softmax(input\ of\ hidden\ layer \times W^{-1}) \tag{4}$$

The element in the result after activation with the highest probability will be chosen as the predicted result. By comparing this prediction with the true one-hot label, we can use gradient descent algorithm to iterative update *W* and *W*'. After training, the *W* matrix will be used for the embedding of any new words in the future.

In general, applying this method of word embedding will effectively reduce the dimensions of input data and avoid the problem of sparsity. In the meantime, the generated vectors in embedding space will still keep their relative position with each other.

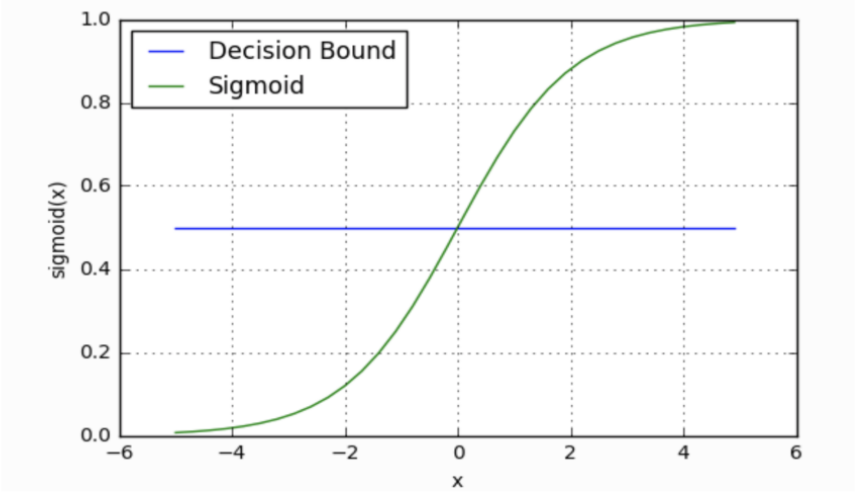
B. Proposed Methods

In the first phase, we tried a few different classification methods in order to compare the results among them. And some methods had been proved to be more efficient in this task.

In our project, we will mainly use auc-value and confusion matrix as a way to evaluate the models. A confusion matrix is very useful in a classification problem where each row represents the actual class and each column represents the predicted class. With this visualization, people can not only easily see how many records are correctly or falsely classified but they can also find out how many true records are misclassified as fake, and how many fake records are misclassified. AUC value stands for the area under ROC curve. ROC curve is a graph showing the performance of a model with different thresholds. Therefore, auc value is a more comprehensive measure than simply the accuracy.

1) Logistic Regression:

Linear regression is the first method we tried since its easy to implement and interpret the result. Considering our detection is a binary classification problem, logistic regression is an appropriate form of linear regression. The basic idea was that first we regressed all the data to obtain the regression function. Then in order to have the discrete classification result (true or fake), Sigmoid activation function is used to map the prediction[17].



$$S(z) = \frac{1}{1+e^{-z}}$$

Fig. 2. Sigmoid Function

Since we took only 0 and 1 as the result, the normally used Mean Squared Error cost function will no long fit. Instead, we used Cross Entropy function to measure the error of logistic regression[18]. The advantage of Cross Entropy cost is that it is a monotonic function which always increase or decrease.

$$if\ y = 1, Cost(S(z), y) = -log(S(z))$$
$$if\ y = 0, Cost(S(z), y) = -log(1 - S(z))$$
$$CrossEntropyLoss = 1/n * \sum_{n=1}^{N} Cost(S(z), y) \tag{5}$$

The accuracy we obtained from logistic regression with word2vec embedding was 88.98% for the short news text dataset and 78.54% for the long news text dataset which is close to the

state of arts results. Considering the short running time for this method, logistic regression is a relatively efficient method.

freq-LR-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	1995	275
	True	568	413

freq-LR-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	28	13
	True	21	26

w2v-LR-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2138	132
	True	159	822

w2v-LR-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	33	8
	True	11	36

TABLE II

LOGISTIC REGRESSION CONFUSION MATRIX.

Table II showed the confusion matrices for the logistic regression model on dataset1 and dataset2 with different initialized vectors. The matrices showed that with w2v initialization logistic regression made fewer mistakes in labeling true as fake and labeling fake as true. The performances were better for w2v initialization for both datasets.

2) Support Vector Machine (SVM):

Even though logistic regression has its own advantages and provided a good result, considering the large dimensions of our word vectors, a deeper network is needed to discover more features of the text. Thus we implement this support vector machine model for detection.

Generally speaking, a support vector machine has more power in discovering features of the data. The reason is that it projects data to a higher dimension space and trying to find a hyper-plane which can divide all the data. Compared to the previous linear regression model which only stays in a two dimensions space, support vector machine can include more possibility for classification.

We started with a linear SVM whose hyper-plane had the following form, where w is the normal vector to the hyper-plane.

$$\overline{w} \cdot \overline{x} - b = \overline{0} \tag{6}$$

The accuracy of this linear SVM is about 80% in average which is not a very high score compared with other benchmarks. One of the reasons is that the vectors after preprocessing are

densely concentrated. The distance between the vector next to each other is the same. There is no linear plane can divide those vectors effectively.

Therefore, we tried another support vector machine with Gaussian(rbf) kernel. This Gaussian kernel enables a polynomial classifier. The result was that precision and recall increased by 6% after the use of Gaussian kernel.

freq-SVM-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2270	0
	True	976	5

freq-SVM-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	3	38
	True	0	47

w2v-SVM-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2136	134
	True	155	826

w2v-SVM-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	34	7
	True	12	35

TABLE III

SUPPORT VECTOR MACHINE CONFUSION MATRIX.

Table III shows the confusion matrices for both datasets on SVM model. For dataset1, freq-SVM labeled almost all the true news as fake. Since 2/3 of the data are fake news, the test accuracy is high. However, what the algorithm really does is just label almost all of the news fake. This showed us that the result for freq-SVM for dataset 1 can not be trusted. w2v-SVM gave us a more balanced result because it labeled most of the true news correct. Meanwhile, the matrices showed that with w2v initialization SVM made fewer mistakes in labeling true as fake and labeling fake as true. The performances were better for w2v initialization for both datasets.

3) Random Forest:

The main reason why we choose random forest is that it works well on nonlinear data and data with many features. Random forest is a collection of decision trees. Random forest model first draws different N random samples from the dataset. And when reaching a node, these samples will be divided by one of their features to see if there is information gain. From the process described above, the random forest model will significantly reduce the risk the over-fitting since it takes a random sampling approach. While also because of the randomness in the training process, parts of the text data features will be ignored. The text structure information is completely ignored.

freq-RandomForest-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2150	120
	True	530	451

freq-RandomForest-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	29	12
	True	22	22

w2v-RandomForest-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2150	120
	True	280	701

w2v-RandomForest-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	29	12
	True	9	38

TABLE IV

RANDOM FOREST CONFUSION MATRIX.

Table IV shows the confusion matrices for random forest. The matrices show that freq-randonForest labeled half of the true news as fake, but with w2v initialization random forest made fewer mistakes in labeling true as fake. The performances are better for w2v initialization for both datasets. More specifically, w2v initialization reduce the probability that true news being labeled as fake.

4) Boosted Tree:

Boosted trees model or a similar one XGBoost has shown their great capability in prediction problems recently[19]. Unlike the random forest model, each node of the boosted trees has a clear definition and objective. The main focus in this method is to improve the prediction of the residual value of the previous one.

The problem with boosted tree is that it can perform poorly with high dimensional data. Since the loss function requires an operation of Taylor expansion approximation.

freq-BoostedTree-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2247	23
	True	86	895

freq-BoostedTree-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	28	13
	True	15	32

Table V shows the confusion matrices for boosted tree algorithm. In boosted tree, the situation is not the same as random forest. W2v initialization has increase the both the false positive rate which means that the label is fake but predicted as true and false negaive rate for dataset 1. However, it decreases the false negative rate which means that the label should be true but is

w2v-BoostedTree-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	2138	132
	True	150	831

w2v-BoostedTree-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	26	15
	True	10	37

TABLE V

BOOSTED TREE CONFUSION MATRIX.

predicted as fake for dataset2. This may imply that for big datasets frequency based vectors do better for boosted tree.

5) Recurrent Neural Network - Long Short-Term Memory:

One unique feature of the news texts is that they are a sequence of data points instead of discrete data points. For the previous models, texts are treated as a group of individual word among which there are little connections. To better analyze the sequential feature of texts, we decide to train a Recurrent Neural Network (RNN).

Unlike other machine learning models, RNN consists of connections that form a directed graph following a temporal sequence. This unique structure gives RNN the ability to process sequences of inputs. In a traditional setting, the input sequence is usually time series data. In our case, we use the order of appearance of the word in the text to replace the time dimension.

In terms of the structure, we build our RNN with reference to S. Hochreiter and J. Schmidhubers work on Long Short Term Memory (LSTM) model [20]. In general, the LSTM model randomly memorizes the input it gets over arbitrary time intervals. Each LSTM unit has a cell, an input gate, an output gate and a forget gate. Further detail can be found in the original paper. Here are a graph which shows the general structure.

Due to this randomness, the execution time of this model is dramatically longer than others. While the accuracy is also higher than the previous models. The LSTM model has a 99.11% accuracy on dataset1 and 62.08% accuracy on dataset2. With a high accuracy, there are not too many false positive and false negative error.

There is an issue with the implementation of this RNN model. Since the word2vec method can only generate a probabilistic form of vectors with irrational numbers, it cannot be put into the RNN model. The LSTM model requires a rational number form of input so that it can embed the input matrix to a higher dimension. Thus word2vec is not applicable to our experiment.

Table VI shows the LSTM confusion matrices. Since w2v initailization return float numbers,

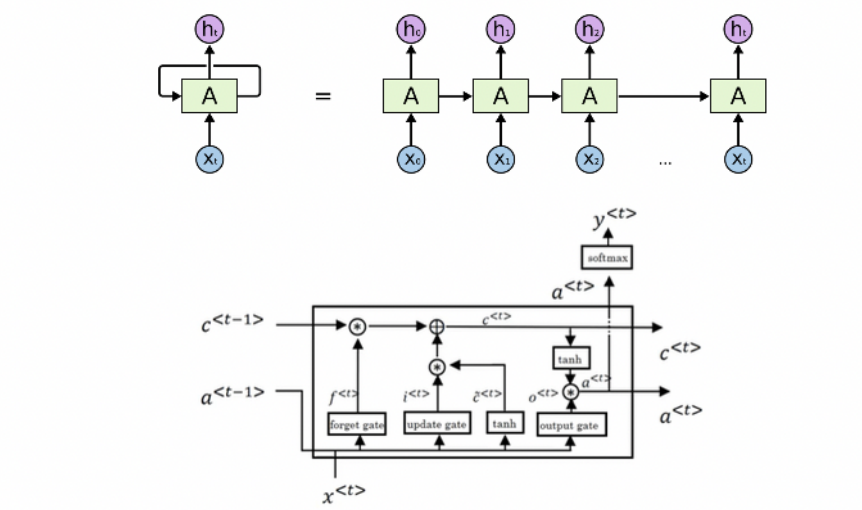


Fig. 3. LSTM Structure

freq-LSTM-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	1153	8
	True	1	463

freq-LSTM-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	15	21
	True	1	6

TABLE VI

LSTM CONFUSION MATRIX.

we cannot add embedding layer to w2v initialized vectors. Therefore, there are only two confusion matrices for frequency based vectors. The results produced by LSTM are very good. For dataset1, only 9 records were labeled falsely. While, in dataset2, more records were labeled falsely because the dataset is too small and the network wasn't trained thoroughly.

6) Text CNN:

Convolutional Neural Network (CNN) has shown its great potential in the area of computer vision during the past few years. By applying different types of convolution matrix to the image data, a CNN can detect and extract many hidden patterns in the image. Since there are many similarities in between the image data and text data, we also tried to apply the CNN approach to our detection problem.

Kim designed a CNN model which consists of one-dimensional convolutional layers and max pooling layers[7]. And this model has been proved to be very efficient in classifying short sentences into different categories. Inspired by Kim's model, we designed our own text-CNN

model which can classify news texts with many sentences into the category of true or fake. First, we chose to use the vectors from frequency dictionary instead of the embedded vectors from word2vec method. In order to make the task more similar to a image detection problem, we reshaped the vectors into 4 dimensions which are number of samples, lens of word vector, embedding dimensions and a constant. Then, we applied a two-dimensional convolution matrix instead of one which was supposed to be more sensitive to the pattern. Moreover, to avoid the loss of information during the max pooling process, we chose to use k-max-pooling instead of the traditional 1-max-pooling method. Figure 4 is the structure of our text CNN model.

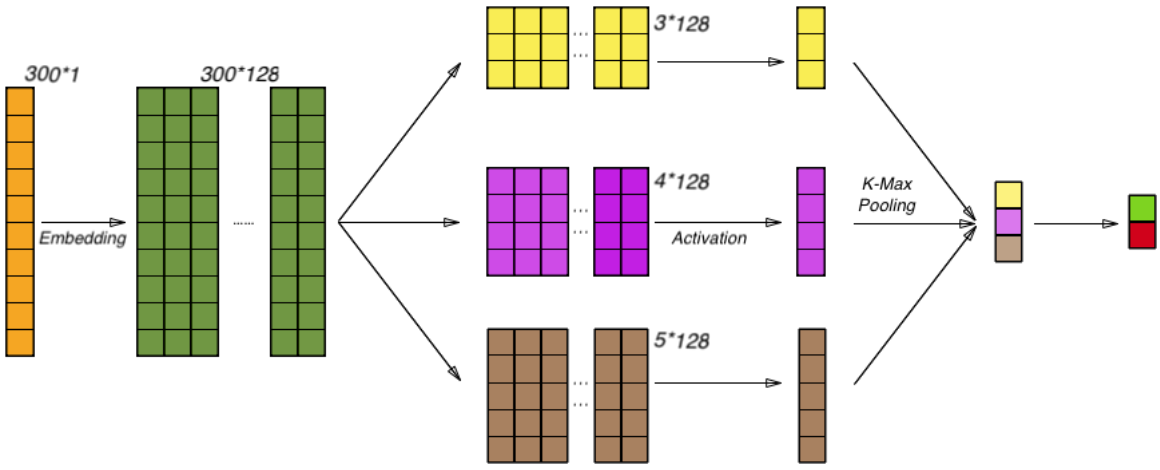


Fig. 4. text CNN Model Structure

The result from the above text-CNN model shows that with frequency dictionary embedding, it reach a accuracy of 98.97% which is very close to the RNN result (99.11%). However the RNN method takes much more time than our text-CNN model. And the same reason as above, word2vec method also cannot be applied here.

One problem with the result of text-CNN is that for the dataset 2, there is a huge proportion of false negative results. With a small sample size, text-CNN is not able to identify the fake news very well.

The result of textCNN is similar to that of LSTM. For dataset1, since there are over 10000 records, the data are enough to train CNN. Therefore only 9 records are labeled falsely. However, dataset2 is too small in size with only 300 records as training samples, all records were labeled as true by textCNN in dataset2.

freq-textCNN-dataset 1	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	1153	6
	True	3	465

freq-textCNN-dataset 2	**Predicted Class**		
		Fake	*True*
Actual Class	*Fake*	0	0
	True	16	27

TABLE VII

TEXT-CNN CONFUSION MATRIX.

C. Summary of Result

All the experiments were carried out on our personal computer. It is a 13-inch MacBook Pro (Late 2016, Four Thunderbolt 3 Ports) with 2.9 GHz Intel Core i5 processor and 8 GB 2133 MHz LPDDR3 memory. No GPU is used for any test.

By comparing the overall accuracy and running time, we can notice that the RNN model and our textCNN model with the frequency dictionary had the best result. And the running time of textCNN is only 1/10 of the running time of RNN. Another thing is that w2v initialized algorithms outperform their corresponding frequency based algorithms. (Fig. 5)

Figure 5 shows the auc-value and run time summary for all the models that we had trained. We can see that for dataset1, textCNN and LSTM have the highest auc value at around 99%. However, for dataset2, textCNN and LSTM did not outperform other algorithms because of the small size of dataset2. W2v initialized algorithms gave a better result because w2v is a deep learning algorithm which has been trained on a lot of data.

Figure 6 shows the precision, recall and f1 statistics. In terms of the precision and recall score, both RNN and textCNN performed very well on dataset1. And the precision, recall and f1 scores for both algorithms are high for dataset1. Therefore, these two algorithms are considered reliable do not simply label the data all true or fake as SVM does. While with a limited number of data on dataset2, textCNN is not able to label the fake news very well compared with RNN. Due to the small size of the dataset, both deep learning algorithms do not perform very well. Another thing that deserves notice is that freq-SVM does not have an F1 score since it predicts most records as all true or false. Moreover, w2v initialized algorithms generally give a better result. (Fig. 6)

To sum up, textCNN performs well on a big dataset with high auc value, precision, recall, f1 score and quick run time. LSTM is accurate, but it takes more time to run for big datasets. These two algorithms do not do well on small dataset because they cannot be thoroughly trained

Accuracy	Dataset 1	Running Time(s)	Dataset 2	Running Time(s)
freq-LR	64.99%	0.97	61.81%	0.04
freq-SVM	50.25%	13.66	53.66%	0.06
freq-RandomForest	70.34%	3.91	61.96%	0.26
freq-BoostedTree	95.11%	25.31	68.19%	0.24
freq-RNN	99.11%	4065	62.08%	139
freq-textCNN	98.84%	448	50.00%	13
w2v-LR	88.98%	0.52	78.54%	0.08
w2v-SVM	89.15%	222.85	78.69%	0.11
w2v-RandomForest	83.09%	2.97	75.79%	0.19
w2v-BoostedTree	89.45%	28.87	71.06%	0.49
w2v-RNN	Not Applicable			
w2v-textCNN	Not Applicable			
*w2v = word2vector *freq = frequency dictionary				

Fig. 5. AUC-value and Run Time Summary.

and the result will be awful. In general, traditional machine learning algorithms perform better when we feed word2vec initialized vectors to it no matter the size of the dataset.

V. Second Phase Experiment

A. Problem Spotted

The result from the first phase experiment proved that our text-CNN model was effective. However, the result was obtained from training on only one dataset. When we tried to apply the model to a different dataset, the model could not classify the news correctly even though both datasets were related to political news. This may due to the fact that dataset2 is too small in size and deep learning algorithms can not be trained thoroughly. Since w2v initialized algorithms do

Accuracy	Dataset 1			Dataset 2		
	Precision	Recall	F1	Precision	Recall	F1
freq-LR	0.600	0.421	0.495	0.667	0.553	0.605
freq-SVM	1	0.005	0.010	0.553	1.0	0.712
freq-RandomForest	0.790	0.460	0.581	0.676	0.532	0.595
freq-BoostedTree	0.975	0.912	0.942	0.711	0.681	0.696
freq-RNN	0.99	0.99	0.99	0.857	0.222	0.353
freq-textCNN	0.994	0.987	0.990	0.628	1.000	0.771
w2v-LR	0.862	0.838	0.850	0.818	0.766	0.791
w2v-SVM	0.860	0.842	0.851	0.833	0.745	0.787
w2v-RandomForest	0.854	0.715	0.778	0.760	0.809	0.784
w2v-BoostedTree	0.863	0.847	0.855	0.712	0.787	0.747
w2v-RNN	Not Applicable					
w2v-textCNN	Not Applicable					
*w2v = word2vector *freq = frequency dictionary						

Fig. 6. Precision, Recall and F1 statistics.

better than frequency dictionary based algorithms for traditional machine learning algorithms, and embedding layers can not be applied to w2v initialized vectors, we want to combine the two results from frequency based deep learning algorithms and w2v initialized machine learning to see if their combination can further improve the results.

B. Potential Causes to the Problem

After taking a closer look at our method, there are two potential causes to the overfitting problem. First is that the CNN model is too sensitive to the text features in one dataset thus it lack the ability to detect general text. Since we build our text CNN model in the first phase with

two-dimensional convolution matrix, it is very likely for it to catch too many features than we expected.

And another reason is that we did not feed the model enough data set. The two datasets we choose have very different properties although they are all on political news. One major difference is that Kaggle dataset have relatively short text than the general political news dataset. The consequence is that the word dictionaries after preprocessing are different. For example, for the word election, it would be projected into two completely different space in the two different dictionaries. And since we use the same model to do the classification, the two different word vectors for election will give us inconsistent result.

C. New Method Proposed

To get a better result, we tried to ensemble the results from different models and pre-processing methods. We chose two different machine learning methods which were support vector machine and logistic regression. Support vector machine had a lower auc value and it directly predicted the news label which gave either 0 or 1, so its weight was set to 0.1 and textCNNs weight was set to 0.9. For logistic regression, it predicted the label by giving a probability of the news being true. Therefore, the weight of logistic regression was given higher than that of SVM.

The ensemble results failed to outperformed the result of the original textCNN method after we tried different pairs of weights. In dataset1, we had enough data to train both models. For logistic regression and textCNN, if we increased the weight of logistic regression, the auc value increased dramatically. This was not surprising because logistic regression was actually the simplest form of neural networks. Since textCNN already had a good accuracy, the ensemble result was decreased by the lower accuracy logistic regression.

However, in dataset2, there were only 400 records which were far less than enough to train the textCNN model and the word-frequency method function. But the word2vector model had already been pre-trained by Google with a large amount of data. Therefore, for dataset2, pre-processing with word2vector actually achieved a higher accuracy.

For support vector machine, the decision boundaries were only decided by the support vectors. And the result from SVM was only decided by these support vectors. Meanwhile, support vector machines predicted results were only 0 or 1 instead of the probability of the records being true/fake. As a result, when we increased the weight of support vector machine, the auc

value decreased dramatically. The best result we could get is 0.9*freq-textCNN+0.1*freq-SVM. However, the result was still worse than freq-textCNN.

Accuracy	Without Ensemble-DataSet 1	With Ensemble-DataSet 1	Without Ensemble-DataSet 2	With Ensemble-DataSet 2
0.9*freq-textCNN + 0.1*freq-SVM	98.84%/50.25%	98.63%	50.00%/53.66%	51.21%
0.9*freq-textCNN + 0.1*w2v-SVM	98.84%/89.15%	98.68%	50.00%/78.69%	50.00%
0.6*freq-textCNN + 0.4*w2v-LR	98.84%/88.98%	98.58%	50.00%/78.53%	77.16%
0.6*freq-textCNN + 0.4*freq-LR	98.84%/64.99%	98.63%	50.00%/61.81%	61.80%

TABLE VIII

ENSEMBLE RESULTS.

Table VIII shows the best ensemble results when we assign different weights to two different algorithms. We could see that after ensemble, the auc value was lower than one of the algorithms in the ensemble. It indicated that ensemble could not improve our result. Although w2v-SVM did way better than freq-SVM, the combination between w2v-SVM and freq-textCNN acutally hurted the result from textCNN. Another possible explanation is that textCNN itself only falsely labeled 9 records, and w2v-SVM labeled 289 records wrong. The 9 records could be included in the 289 records, and more correctly predicted labels were affected by the 289 records from SVM.

VI. CONCLUSION AND FUTURE WORK

The problem of fake news has drawn lots of people's attention. During the last election, Clinton Hillary was falsely reported as a Russian spy which somehow affects the election results. The identification of fake news becomes more and more important. In this paper, we proposed a method where we trained frequency based vectors through a text Convolutional Neural Network. During the experiment, we found out that text CNN gave a higher accuracy and auc value than traditional machine learning algorithms for both the word2vec or frequency dictionary methods. Meanwhile, textCNN and LSTM have both returned good results. However, the run time of textCNN was 10 times faster than that of LSTM. The experiment results showed that textCNN

could accurately and efficiently predict the US president election dataset based on only the tile and text of the news.

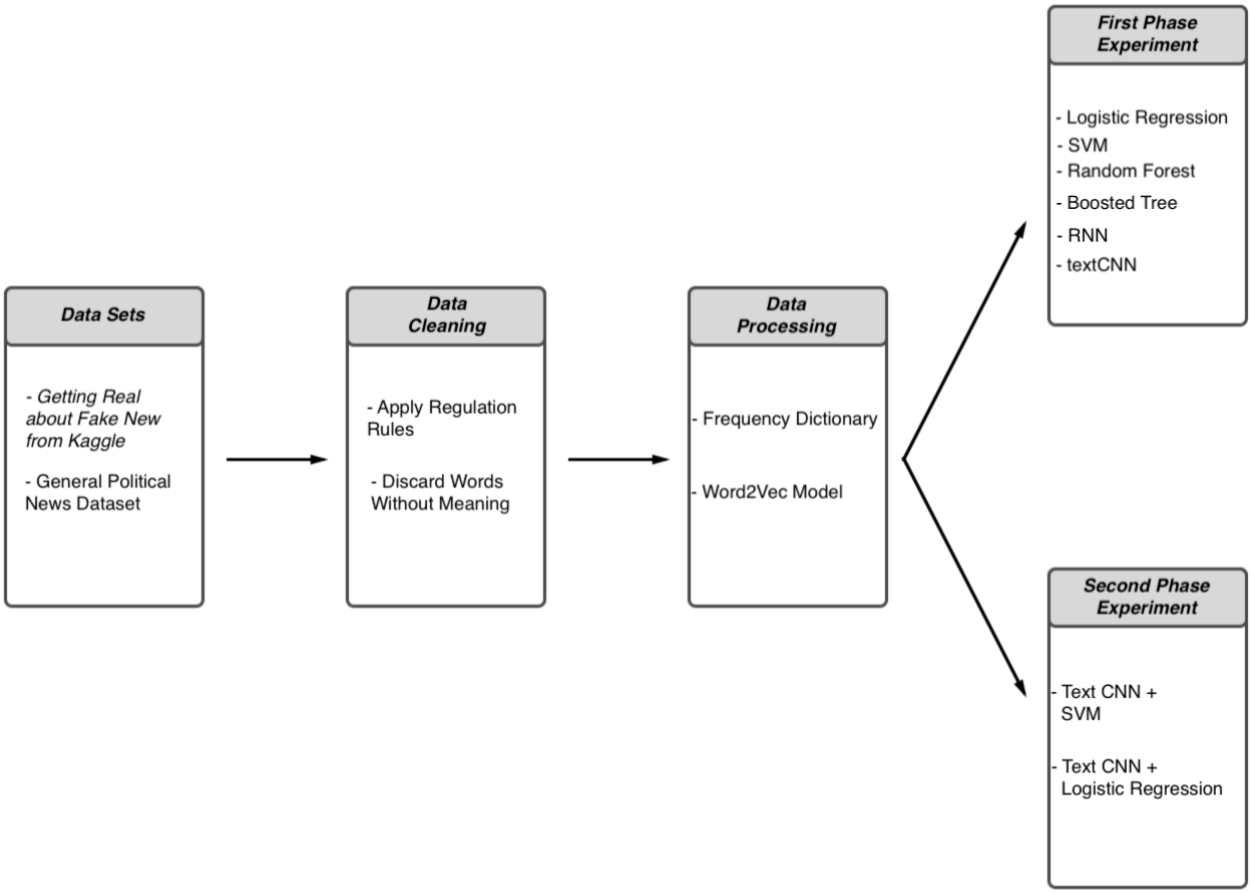


Fig. 7. Summary of Overall Process

The dataset we used contains mostly US election news with its title and content. With the new round of US election approaching, we plan to obtain more data on the new US president election. Meanwhile, we want to get the propagation structure of the news to perfect the model. So far, researchers only focus on one of the two stages to detect fake news. By running models on both stages, hopefully, the accuracy will increase because more information is added to the model.

ACKNOWLEDGEMENT

We would like to express our very great appreciation to Professor Ratan Dey for his valuable and constructive suggestions during the planning and development of this research work. Also we are particularly grateful for the assistance given by Professor Olivier Marin.

REFERENCES

[1] D. S. Correll, "Russian hackers invaded control rooms of hundreds of us electric utilities: Report," Jul 2018. [Online]. Available: https://www.washingtonexaminer.com/news/russian-hackers-invaded-control-rooms-of-hundreds-of-us-electric-utilities-report

[2] N. Mele, "Two things you should know about the media and the 2016 presidential election." [Online]. Available: https://knightfoundation.org/articles/two-things-you-should-know-about-the-media-and-the-2016-presidential-election

[3] M. Doran and M. Doran, "The real collusion story," Mar 2018. [Online]. Available: https://www.nationalreview.com/2018/03/russia-collusion-real-story-hillary-clinton-dnc-fbi-media/

[4] C. Buntain and J. Golbeck, "Automatically identifying fake news in popular twitter threads," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, Nov 2017, pp. 208–215.

[5] S. Feng, R. Banerjee, and Y. Choi, "Syntactic stylometry for deception detection," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 171–175. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390665.2390708

[6] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=944790.944813

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://www.aclweb.org/anthology/D14-1181

[8] W. Y. Wang, ""liar, liar pants on fire": A new benchmark dataset for fake news detection," *CoRR*, vol. abs/1705.00648, 2017. [Online]. Available: http://arxiv.org/abs/1705.00648

[9] Y. Yang, L. Zheng, J. Zhang, Q. Cui, Z. Li, and P. S. Yu, "TI-CNN: convolutional neural networks for fake news detection," *CoRR*, vol. abs/1806.00749, 2018. [Online]. Available: http://arxiv.org/abs/1806.00749

[10] X. Rong, "word2vec parameter learning explained," *CoRR*, vol. abs/1411.2738, 2014. [Online]. Available: http://arxiv.org/abs/1411.2738

[11] F. Qian, C. Gong, K. Sharma, and Y. Liu, "Neural user response generator: Fake news detection with collective user intelligence," in *IJCAI*, 2018.

[12] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sina weibo by propagation structures," in *2015 IEEE 31st International Conference on Data Engineering*, April 2015, pp. 651–662.

[13] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, "Detect rumors using time series of social context information on microblogging websites," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1751–1754. [Online]. Available: http://doi.acm.org/10.1145/2806416.2806607

[14] M. Risdal, "Text & metadata from fake & biased news sources around the web," Nov 2016.

[15] T. Mikolov, K. Chen, and G. Corrado, "Efficient estimation of word representations in vector space," Sep 2013.

[16] X. Rong, "word2vec parameter learning explained," Nov 2014.

[17] "Logistic regression." [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

[18] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, Feb 2005. [Online]. Available: https://doi.org/10.1007/s10479-005-5724-z

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*, 2016.

[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, 12 1997.