Fake and Real News Article Classification

Seungyun Lee	Daniela Lopez	Brandon Yu	Wan Suk Lim	Michael Tang
Computer Science	*Computer Information Systems*	*Computer Science*	*Computer Science*	*Computer Science*
Cal Poly Pomona	*Cal Poly Pomona*	*Cal Poly Pomona*	*Cal Poly Pomona*	*Cal Poly Pomona*
Pomona, USA	Pomona, USA	Pomona, USA	Pomona, USA	Pomona, USA
seungyunlee@cpp.edu	danielalopez@cpp.edu	brandony@cpp.edu	dreamerlim@gmail.com	michaeltang@cpp.edu

***Abstract*—These days, there are often so much news being presented in the media that we often times don't know if what we're reading is true or not. It becomes a problem when it's hard to distinguish news that are true and the ones that are actually fake. Our objective in this research project and experiment is to come up with a way to classify real news and fake news. Our dataset has numerous amounts of instances, specifically 44,919 instances containing 23,502 fake and 21,417 real news. The methods that we plan on using are SVM and Naive Bayes in order to classify each instance of the news into the correct category. We will use pre-processing Python libraries in order to trim our data down into a more concise dataset and then proceed to use machine learning algorithms to train our model in order for it to be able to classify the instances correctly. Our hope is to be able to come up with a model using keywords and patterns to distinguish what's real and what's fake.**

***Index Terms*—Machine Learning, Media, News Articles, Text Classification**

I. Introduction

Just like spam emails, we live in the era of false information including fake news in various forms of media. This is dangerous, as many people in the world believe what they see online without attempting to verify if the online information is true or not. It is important to be able to distinguish real news from fake news to make better and well informed decisions in our life. Many different entities including nations, political groups, and individuals produce fake news for the sake of fulfilling their individual agendas. Luckily, we currently live in a society where technology has evolved enough, that it can help change our lives for the better, including this instance. In our paper, we will be using machine learning algorithms, such as Support Vector Machines(SVM) and Naive Bayes, to distinguish real news from fake news utilizing the data provided by kaggle. The dataset is already labeled, but one of our main objectives include having our program effectively label new data so people can avoid having to manually read through the news and creating the label themselves. This methodology is similar to a spam email filtering system with Natural Language Processing (NLP). Spam email filtering remains to be a difficult issue to resolve, even to this day, because the content within the spam email continually evolves to evade the filtering system. So naturally, we also expect the news articles to evolve in order to evade our system. We hope that our methodology and ML model/algorithm will also evolve to a point where it will be able to handle such situations.

II. Dataset

Our dataset has 44,919 instances, consisting of 23,502 fake and 21,417 real news [1]. Both fake and real news data have four features: title, text, subject, and date. Title is the title of a news article, text is the actual content of that news, subject tells what category the news belongs to, and the date tells when the news was posted. Our class labels are fake news and real news. We have a large dataset, which is sufficient to divide into 2 subsets for training and testing.

III. Methodology

Our methodology has three parts: preprocessing data, building text classification models, and evaluating results. We are preprocessing text data to feed to our machine models. We have decided to use SVM and Naive Bayes as our models. Then, we will use popular metrics to check our results.

A. Preprocessing Data

We are going to preprocess the data as described below.

a) Add Label Column: Our dataset does not have a label column. We have two csv files; one for real news and another for fake news. Each having 4 feature columns with no class label. So, we will first add the correct class label to each data point, then combine both datasets.

b) Data Preprocessing: Our features are all texts, except the date. So, we need a NLP library to process all the texts and we will use spaCy, possibly NLTK as well, to pre-process the text data. We will apply the following steps to preprocess the texts.

- **Tokenization**: Using the library, we tokenize every word, punctuation, and etc.
- **Removal of punctuation**: Then, we remove punctuations since they carry no meanings, reducing the size.
- **Removal of stop-words**: We will also remove stop-words since they carry no meanings, reducing the size.
- **Lemmatization / Stemming**: Then, we will either do lemmatization/stemming to reduce words to their root form.

c) Data Vectorization: To vectorize the texts, we will TF-IDF vectorizer. TF-IDF is a measure that tells the relevance of a word in a document in a collection of documents. So, we will create a vector of TF-IDF using the library.

d) Train and Test Set Ratio: We are planning to split our dataset into two subsets for training and testing. The ration will be 7:3, where 7 is for training and 3 is for testing. We will randomly choose to split the dataset.

B. Text Classification

For our models, we chose two popular classifying methods to do the task.

- **Multinomial Naive Bayes**: Naive Bayes is based on the Bayes theorem. It's a probabilistic machine learning classifier model, which is used to assign a class label to each data instance. It's called naive, because it makes an assumption that each feature is independent and does not affect other features. Since our task is classifying whether news is fake or real, Multinomial Naive Bayes is suitable for our objective.
- **Support Vector Machine**: Support Vector Machines is a supervised classifier that places data on one of two sides of a space. A hyperplane is created which is placed between the data points that are closest but also equally far away from each class. The idea is that the hyperplane will be placed in a space that will classify data the best. It's called a support vector machine because you can think of the training data supporting hyperplane placement. For our project, the SVM classifier will classify a news article as either real or fake.

C. Performance Evaluation

To evaluate how well our models works, we are going to use Accuracy, Precision, Recall, and F1 metrics to check the performance of the models. These four metrics are typical metrics that are used when evaluating classification models.

- **Accuracy**: The measure that the classifier correctly labeling true positives and true negatives.
- **Precision**: The ratio between the true positives and all the positives that are labeled by the classifier.
- **Recall**: The measure that the classifier correctly labeling true positives.
- **F1**: The harmonic mean of precision and recall.

IV. Results

A. Multinomial Naive Bayes

After performing tf-idf vectorization to the dataset and training the Naive Bayes Model with the results, we end up with an accuracy of 92%, precision of 93%, recall of 92%, and finally an F1 of 92%. Using the metric of AUC(area under curve), which is the measure of the ability for a classifier to distinguish between classes, we find that it's 0.98 for both classes. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

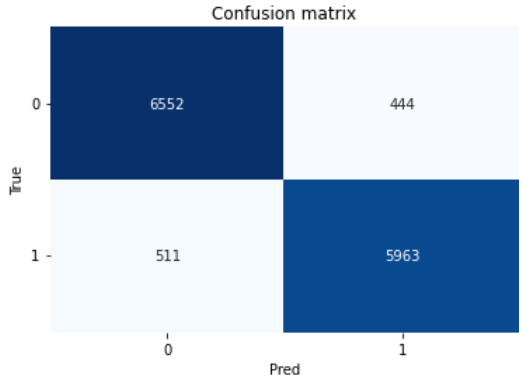

Fig. 1 - Confusion Matrix of 1-Gram

The initial model was created with 1-gram. A 1-gram is where the model uses every word as a feature. As you can see from Fig. 1, our model performs fairly well under the circumstances and predicts the True Positives and True Negatives correctly most of the time.

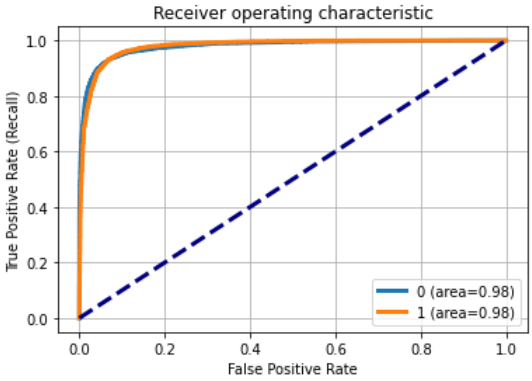

Fig. 2 - Receiver Operating Characteristic of 1-Gram

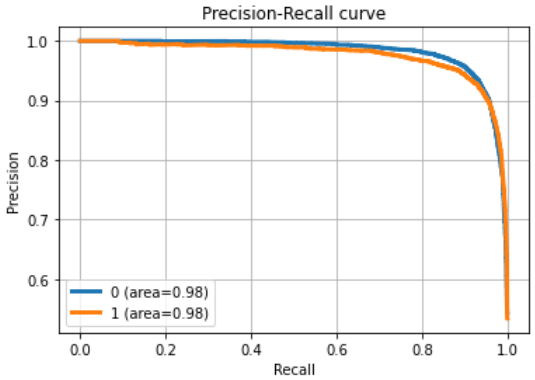

Fig. 3 - Precision-Recall curve of 1-Gram

From Fig. 2, you can see the receiver operating characteristics. The blue line from the graph represents if we were to randomly guess the label and the orange line represents the predictions made from the 1-Gram. As you can clearly see, our model for the 1-gram performs way better than if we were to randomly guess the class label. With Fig. 3, you can see that both the precision and recall of the 1-gram model are

also extremely good, which means the model is effective at predicting if the data is real or fake news. We also went a step further to test out the different results with different n-grams. An n-gram is where we substitute different values of n, and we tried all the values from 1-10. The results indicated that the performance of the model would peak at 2-gram at 96% accuracy and following of 3-Gram with a 94% accuracy. The accuracy continues with a steady decline as you increase the number of the gram with the worst accuracy ending at 10-gram, which has an accuracy of 54%.

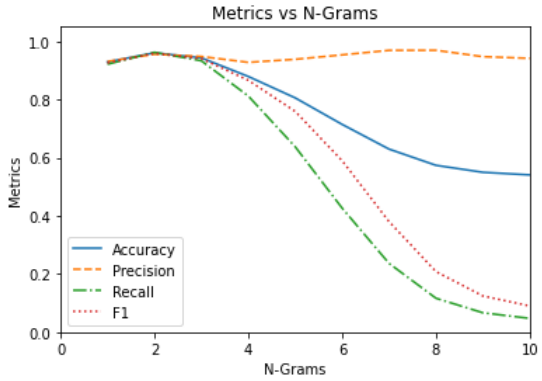

Fig. 4 - Metrics vs N-grams

From Fig. 4, you can see the accuracy, precision, recall, as well as the F1 for our n-gram experimentation results. You can see the metrics for the model peaks at 2-gram as stated in the previously and continues with a steady decline, reaching the worst result as the number of grams goes up. Overall, the Multinomial NB model is fairly effective at classifying real and fake news with a 2-gram reaching peak accuracy at 96%.

Max Features	Accuracy	Precision	Recall	F1
5,000	0.96035	0.95566	0.96215	0.95889
6,000	0.96057	0.95694	0.96122	0.95908
7,000	0.96132	0.95728	0.96246	0.95987
8,000	0.96198	0.95819	0.96292	0.96055
9,000	0.96206	0.95791	0.96339	0.96064
10,000	0.96273	0.95980	0.96277	0.96128

Table I. 2-Gram Model with Different Max Features Values

With the results from Table I, further experimentation with multiple max feature values were used to drive and validate how our model would perform. One thing to note is that a 2-gram model was used for this experimentation. In this case, we can see that the more features that we have, the better the model will perform in all measures of metrics. The lowest accuracy in the table is 96.035% when we utilize 5,000 as the max features and the highest accuracy in the table is 96.273% when we utilize 10,000 as the max features.

B. SVM

The SVM model was tested with various parameters to find the best possible results. We first tried different feature sizes (5, 25, 50, 100, 250) for each kernel type (linear, poly, rbf). For these tests we kept the parameters (C = 1, degree = 3, and gamma = auto) the same. For the linear kernel we found that a low amount of features produced the worst results overall, while a high number of features produced better results. For the poly kernel we found that a lower number of features produced better results. A higher number of features, specifically after 25, produced poor results. The rbf kernel performance peaked at 25 features and continued on a downward trend after that. Overall, the highest performing combination of parameters when looking at the kernel and features was the linear kernel with 250 features at a 99.2% accuracy. The lowest combination was the poly kernel with any feature value after 25 at 52.8% accuracy. We only tested up until 250 so features beyond that value could vary.

Linear Kernel, C-value = 1, degree = 3, gamma = auto				
Features	Accuracy	Precision	Recall	F1
5	0.75523	0.71806	0.79354	0.75391
10	0.80445	0.79060	0.79731	0.79394
25	0.99109	0.98941	0.99175	0.99058
50	0.99008	0.98962	0.98939	0.98951
100	0.99131	0.98965	0.99198	0.99081
250	0.99187	0.98920	0.99363	0.99141

Table II. Model parameters containing the linear Kernel function and varying feature values

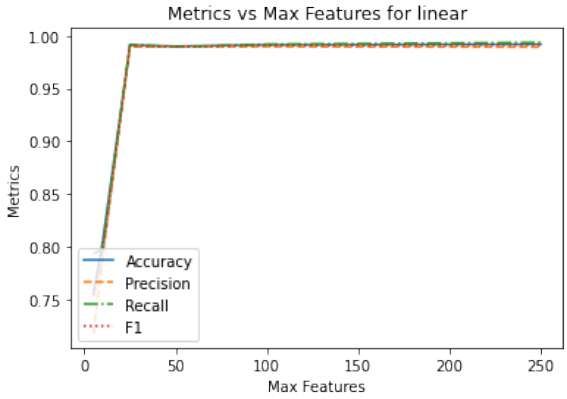

Fig. 5 - Metrics vs Max Features for Linear Kernel

Table II and figure 5 both show that as the number of features increases past 5 it stays generally high.

poly Kernel, C-value = 1, degree = 3, gamma = auto				
Features	Accuracy	Precision	Recall	F1
5	0.75979	0.72116	0.80155	0.75923
10	0.77216	0.81757	0.66650	0.73435
25	0.52750	0.0	0.0	0.0
50	0.52750	0.0	0.0	0.0
100	0.52750	0.0	0.0	0.0
250	0.52750	0.0	0.0	0.0

Table III. Model parameters containing the poly kernel function and varying feature values

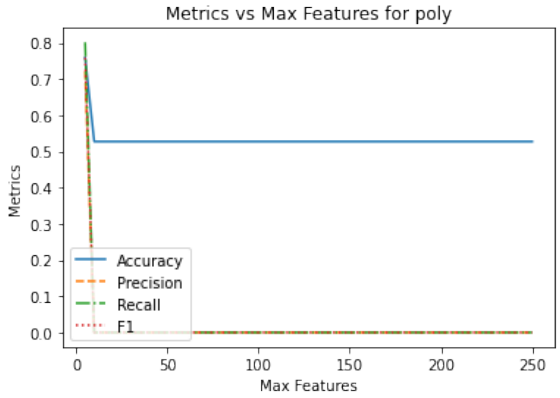

Fig. 6 - Metrics vs Max Features for poly Kernel

Table III and figure 6 interestingly show that at about 25 features the performance drops drastically and continues to perform poorly.

rbf Kernel, C-value = 1, degree = 3, gamma = auto				
Features	Accuracy	Precision	Recall	F1
5	0.76370	0.70588	0.85694	0.77411
10	0.80991	0.78004	0.83243	0.80538
25	0.98151	0.98593	0.97478	0.98033
50	0.96860	0.96854	0.96488	0.96671
100	0.95991	0.94849	0.96771	0.95800
250	0.95289	0.94029	0.96134	0.95070

Table IV. Model parameters containing the rbf kernel function and varying feature values

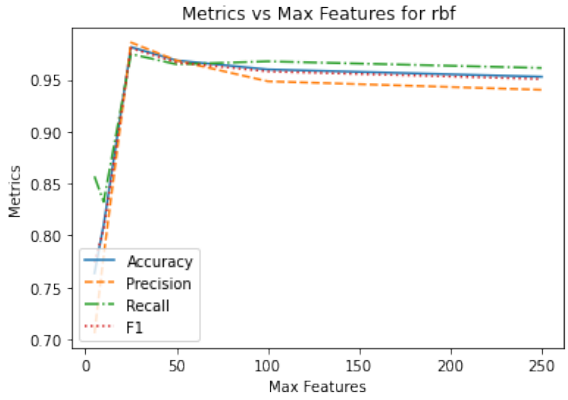

Fig. 7 - Metrics vs Max Features for rbf Kernel

Table IV and figure 7 show that performance peaks at around 25 features for the rbf kernel.

After testing various kernel and feature value combinations we decided to take the lowest performing model (poly kernel with 25 features) and see if changing the C-value brought up the performance. The models parameters we kept constant were (Kernel = poly, degree = 3, and gamma = auto). The different C values we used were (1, 10, 25, 50, 100, 500, 1000). We found that changing the C-value did improve the accuracy. The performance was the highest with the C-value at 1000. The accuracy achieved here was 97.1%.

linear Kernel, features = 25, degree = 3, gamma = auto				
C-value	Accuracy	Precision	Recall	F1
1	0.75979	0.72116	0.80155	0.75923
10	0.75757	0.92440	0.53029	0.67396
25	0.85512	0.91320	0.76620	0.83327
50	0.88385	0.91451	0.83196	0.87128
100	0.90512	0.92590	0.86872	0.89640
500	0.95690	0.96910	0.93872	0.95367
1000	0.97127	0.98023	0.95852	0.96926

Table VI. Model parameters containing the poly kernel function and varying C values

When looking at all of the models the highest performing was the linear kernel with 250 features, a C value of 1, degree of 3, gamma as auto with an accuracy of 99.2%. The confusion matrix below will show these results.

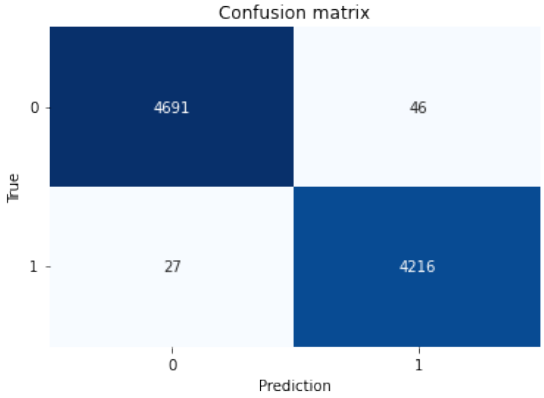

Fig. 7 - Confusion Matrix for 250 features and linear kernel (highest performing model)

V. Related Work

There is a different model other people used for the same data. The author of [2] used neural network LTSM (Long Short-Term Memory). It is a recurrent neural network capable of learning order dependence in sequence prediction problems. It is important in complex problems like machine translation, speech recognition, and etc. The author got 98% accuracy using the LSTM model and precision score is 98%. LSTM recall is 98% and f1 score is 98%. The author used Bi-LSTM model and got 99% accuracy, 99% precision, 99% Recall and 99% f1 score. Our Naive Bayes Model with the results, we end up with an Accuracy of 92%, Precision of 93%, Recall of 92%, and finally an F1 of 92%. With the same test data, the author of the blog with Bi-LSTM got much better results than our model. It is important to try different models to find the best model.

VI. Conclusion

In this work, we address the fake news problem using the machine learning algorithms like Multinomial Naive Bayes and SVM. We used fake news and real news data from [1]. First, we preprocessed the data by combining title and text. Then, we used that with a label column. We tokenized the

text and removed stop words and non-alpha text also applying stemming. After that we used tf-idf to do word vectorization. Lastly, we split the data into two sets - train and test sets.

We trained and tested the data with Multinomial Naive Bayes and SVM models. We got a better result with SVM. There are other related works using other machine learning models such as LTSM (Long Short-Term Memory) and they got a good result using the Bi-LTSM model.

Natural language processing is a growing field with machine learning techniques. There are many similar problems such as the e-mail spam filtering problem. The technology to generate fake news and spam e-mails also evolves with machine learning techniques. This field will become more important as we rely more on programs to determine which information is important.

REFERENCES

[1] C. Bisaillon, "Fake and real news dataset." Kaggle. https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset?select=Fake.csv (Accessed Oct. 7, 2021).

[2] B. Roshan, "Fake news classification: Build fake news classifier using LSTM," Analytics Vidhya, 31-Dec-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/12/fake-news-classifier-on-us-election-news%F0%9F%93%B0-lstm-%F0%9F%88%9A/. (Accessed: 20-Nov-2021).

VII. SUPPLEMENTARY MATERIAL

Check the links below to see the latex files and source codes of the project.

- **Latex:** https://www.overleaf.com/read/fxcjztpjgjfq
- **Pre-processing Data:** https://colab.research.google.com/drive/17ordBYeCIbUakjnS6u4-cWKOpodUGxDr?usp=sharing
- **Multinomial NB:** https://colab.research.google.com/drive/1rj5F-4bjml06S6WAeFP-Cl668VnsfJHB?usp=sharing
- **SVM:** https://drive.google.com/file/d/1lDL9Yro7V8rsiLHiJ5EKjl98zfx9yBW0/view?usp=sharing