Using Machine Learning for Fake News Detection

Yipin Lu yl886@georgetown.edu
Prerna Kaul pk676@georgetown.edu
Ratnadeep Mitra rm1773@georgetown.edu

Abstract

This paper explores the application of natural language processing techniques for the detection of `Fake News', that is, misleading news stories that come from non-reputable sources. Using a dataset obtained from Kaggle, we apply term frequency-inverse document frequency (TF-IDF) and bag of words (BoW) to a corpus of about 4,000 articles. We test our dataset on multiple classification algorithms – Naïve Bayes, Recurrent Neural Networks, Support Vector Machines with Stochastic Gradient Descent. We find that bag of words fed into Support Vector Machine with Stochastic Gradient Descent model identifies non-credible sources with an accuracy of 98.1%.

1 Introduction

With the advent of social media and online news websites, we are exposed to news from all corners of the world. Unfortunately, only some of the news that we read are authentic while some are fake - intended to mislead. The problem of fake news is extremely pervasive nowadays and therefore in our project, we attempt to build models that will supplement the human ability to differentiate legitimate news from falsified news with the use of natural language processing. We have used multiple methods of classification in our analysis - Naïve-Bayes classification, recursive neural networks with long-short term memory and support vector machines.

2 Dataset and Preprocessing

The dataset used is Kaggle's 'Fake News Detection' dataset, which contained 4009 news articles collected from various sources. The dataset contained the following fields - URLs, Headline, Body and Label. Out of all the fields, only Body and Label were kept, and the rest were discarded.

The pre-processing tasks were performed to prepare the data for analysis included removal of punctuations and special characters from the articles using regular expressions, conversion of the articles to lowercase and removal of stop-words. Since the articles were not collected from Twitter, there were no hashtags in the articles. Before converting the articles to lowercase, the text was tokenized and then joined back after the processing. Stop-words were removed because they constituted the unimportant part of the news articles and not probably not contributing towards indicating whether the news article is legitimate or falsified. After that, we used TF-IDF and Bag of Words to conduct text vectorizations towards the cleaned text data.

3 Methodology

3.1 Naïve-Bayes

Naïve Bayes is a straightforward machine learning model that can be applied for classification. It adopts the Bayes Theorem and it is a collection of estimated probabilities.

$$P(c|x) = \frac{P(x|c) \times P(c)}{P(x)}$$

Equation 1: The Bayes Theorem

As Equation 1 shows, the Posterior Probability $P(c|x)$ can be calculated by the product of likelihood $P(x|c)$ and class prior probability $P(c)$ over the predictor prior probability.

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \ldots \times P(x_n|c) \times P(c)$$

Equation 2: The Methodology of Naïve Bayes

Hence, as Equation 2 illustrates, we can express the $P(c|x)$ as the product of $P(x_i|c)$ times class prior probability. In our case, the $P(x_i|c)$ is the probability of feature i when condition c holds. From this calculation, we can conclude that Naïve Bayes can handle an arbitrary number of independent features, and it is very fast. However, it also has some weakness. For example, it assumes all the features are independent, which might not be true in text classification. Moreover, it is sensitive to outliers. In general, Naïve Bayes works well for text classification problems. Hence, in our case, we used the Naïve Bayes to train a baseline model for comparison. It achieved an accuracy of 93.9% for the testing dataset. And for the training dataset, the accuracy is 97.4%. Although it reveals a slight overfitting problem, the overall performance of Naïve Bayes is sound.

3.2 Recurrent Neural Network

Recurrent neural network (RNN) is a sequence of neural network blocks that are linked to each other like a chain. Each one is passing a message to a successor. They overcome traditional neural networks' shortcoming when dealing with sequential data, be it text, time-series or even DNA and are very capable of handling "long-term dependencies". RNN uses its reasoning about previous events to inform the later ones. They are networks with loops in them, allowing information to persist, as shown in Fig.1.

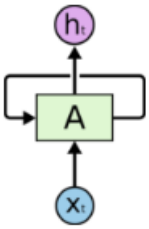

Figure 1: Recurrent Neural Network Node

One drawback of the standard RNNs is the vanishing gradient problem, which affects the performance of the neural network because it cannot be trained properly. This happens with deeply layered neural networks, which are used to complex data, for example long bodies of text. Standard RNNs use a gradient-based learning which degrades with increasing complexity.

Long-Short-Term-Memory (LSTM) networks are a special kind of RNN, capable of learning long-term dependencies. RNNs built with LSTM units categorize data into short-term and long-term memory cells. Doing so enables RNNs to figure out what data is important and should be remembered and looped back into the network, and what data can be forgotten. LSTM networks manage to keep contextual information of inputs by integrating a loop that allows information to flow from one step to the next. LSTM predictions are always conditioned by the previous experience of the network's inputs.

The advantages of RNN and LSTM when it comes to processing sequential data made it the model of choice. The deep learning model shown in Fig.2 put together embedding layer, followed by LSTM layer with 2 dropout layers and finally a fully connected dense output layer. The loss function used was binary cross-entropy.

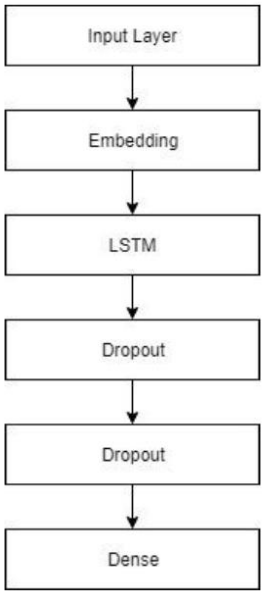

Figure 2: RNN-LSTM Model

The model was trained for 100 epochs. It was found that the keeps increasing but almost stabilizes around 78% - 80% after approximately 80 epochs. It was observed that the loss decreased consistently and rapidly except a sharp around the 30th epoch. The model had two dropout layers, which introduces some sort of randomness in the network and results in training a slightly different model every epoch. That might be responsible for the sharp spike in loss around the 30th epoch. We stopped training the model after 100 epochs, because the consistent pattern of increasing accuracy and decreasing loss might have indicated overfitting. Fig.3 and Fig. 4 show the accuracy and loss while training and validating the RNN-LSTM model for 100 epochs.

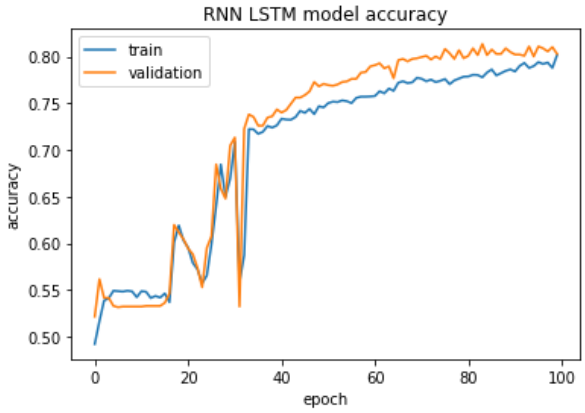

Figure 3: RNN-LSTM model accuracy

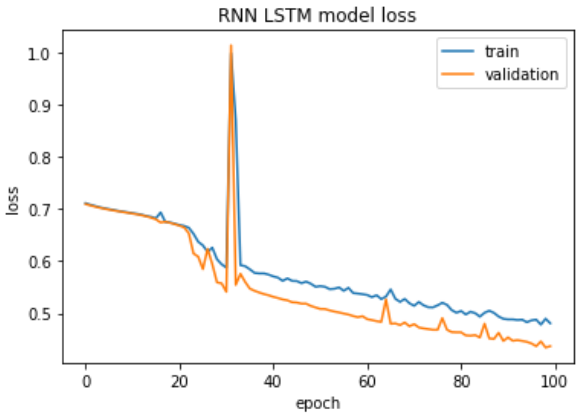

Figure 4: RNN-LSTM model loss

The model performed well on the validation set and it was evaluated on the test set. The loss on the test set was 0.438 and the accuracy achieved was 80.39%.

3.3 Support Vector Machines

Support-vector machines are supervised learning models that analyze data used for classification. Given a set of training observations, each marked as belonging to one or the other of two classes, an SVM training algorithm builds a model that assigns new observations of the test set to one class or the other. An SVM model is a representation of the observations as points in space, mapped so that the points of the separate classes are divided by a clear gap (margin) that is as wide as possible, as can be seen from Fig 5.

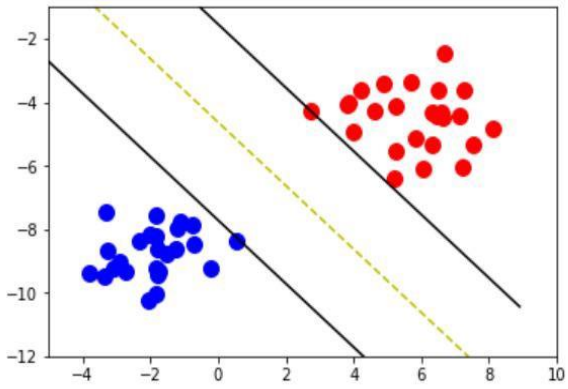

Figure 5: Support Vector Classifier with a hyperplane (yellow-dashed line)

A support-vector machine constructs a hyperplane in the space, which is used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since larger the margin lowers the generalization error of the classifier.

Stochastic Gradient Descent (SGD) is an iterative method for optimizing a differentiable objective function. It aims at optimizing the problem by minimizing the objective function. SDG algorithm is used for training a wide variety of models in machine learning. Here, it is used for training an SVM. Advantages of choosing such a model is that it is efficient and is easy to implement.

The model is built by applying Grid Search Cross Validation along with different values of parameters: ‘loss’, ‘penalty’ and ‘alpha’ on Stochastic Gradient Descent Classifier which uses SVM as the loss function to find the optimum

values for the parameters to build the model. SVM with SGD training is implemented in a manner that the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (i.e. learning rate). After getting the best parameters from cross-validation the model is applied on the testing set and it performs well with an accuracy of 98.1%.

4 Discussion of Results

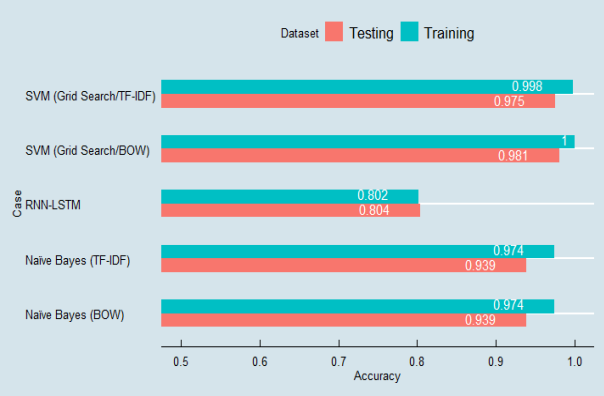

Figure 6: Training and Testing accuracy for models used in analysis

Model	Training Accuracy	Test Accuracy
SVM (Grid Search/TF-IDF)	99.8%	97.5%
SVM (Grid Search/BoW)	100%	97.5%
RNN-LSTM	80.2%	80.4%
Naïve Bayes (TF-IDF)	97.4%	93.9%
Naïve Bayes (BoW)	97.4%	93.9%

Table 1: Training and Testing accuracy for models used in analysis

Fig. 6 and Table 1 show a comparison of the performances of the different models we have used in our analysis. Naïve-Bayes and Support Vector Machines are the two better performing machine learning techniques with Support Vector Machines giving very high accuracy on both training and test dataset with hinge loss, L2 regularization and a learning rate of 0.001. With a training accuracy of 100%, SVM with Bag-of-Words might appearing to be overfitting but it gives the best accuracy on the test dataset among all the other models used. The difference in accuracy between SVM with TF-IDF and SVM with Bag-of-Words is not significant. However, Bag-of-Words technique gives a slightly higher accuracy than TF-IDF revealing that some specific words do not have much impact on the result of the classification.

One of the reasons for the unusually high accuracy of SVM and Naïve Bayes could be the manner of data collection. If there is not a lot variety in the sources of real and fake news collection, the model will be trained to identify one or two very specify and possibly distinct writing styles and when it comes to evaluating the test dataset, the model identifies the writing style very easily giving a very high accuracy rate.

RNN-LSTM usually works very well with text data but interestingly the performance of RNN-LSTM in this case is not as good as the traditional machine learning techniques we used. A possible reason for that could be size of the dataset. With only about 4000 records, the dataset is a relatively small one and the data might not be enough for the artificial neural network to learn.

6 Conclusion

In this paper, we provide general solutions for analytical modeling towards fake news detection. As discussed in detail in the previous section, after training various models on the data, the important things to notice are that although it appears Support Vector Machines overfits the training dataset, it gives better accuracy on the test dataset than the other models – Support Vector Machines with Bag-of-Words performs the best. A deep learning-based solution is provided to detect fake news which does not perform as good as traditional machine learning model which might be due to the relatively smaller size of the dataset. We have used linguistic features for the detection task. There are several interesting options for future work. One is to use a greater dataset with more features to get a

better insight on the problem of fake news. Also, building a framework to detect fake news in real time by implementing an application that runs in streaming manner will be a breakthrough in this area.

References

[1] Bajaj, S. (2017). "The Pope Has a New Baby!" Fake News Detection Using Deep Learning.

[2] Pham, L. (2019). Transferring, Transforming, Ensembling: The Novel Formula of Identifying Fake News.

[3] Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & de Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*.

[4] Rubin, V., Conroy, N., Chen, Y., & Cornwell, S. (2016). Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection* (pp. 7-17).

[5] Bourgonje, P., Schneider, J. M., & Rehm, G. (2017). From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism* (pp. 84-89).