Fake News Detector

Mingzhi Zhong, Yu Ji, Ziwei Chen and Bokai Li

***Abstract*— In this project, we integrated two datasets from Chegg to develop a fake news detector using three machine learning methods: logistic regression, support vector machine and fully-connected neural networks. After a series of tuning, we achieved the highest accuracy of 97% on the test set using fully-connected neural network. Data and our implementation is available at https://github.com/andrewzhong1998/Fake-News-Detector**

I. INTRODUCTION

The development of the internet has enabled convenient creation and rapid transmission of information. However, unrestricted distribution of information leads to the creation of fake news articles that can cause negative impacts on society. However, detecting fake news manually can be extremely challenging and time-consuming since it involves analyzing content, wording and even visual presentation. Thus, fake news detection is an important topic in natural language processing that benefits the society. In this project, we integrated two datasets from Kaggle into a single dataset containing 10000 real news and 10000 fake news. Three machine learning methods, including logistic regression, support vector machine and fully-connected neural networks, were applied separately to build a fake news detector.

II. PRIOR WORK

Early algorithms, such as linguistic approaches and network approaches, seek to make decisions about the authenticity of the news based on other information instead of the content of news alone. In linguistic approaches, syntax analysis, semantic analysis, rhetorical analysis, and other techniques are used to find patterns of deception in the news. In network approaches, network information, such as message metadata, hyperlinks and social network information, are aggregated to identify deceptive contents. [5]. However, both methods require extensive prior knowledge in specific domains and manual feature extraction. Thus, automated feature engineering using machine learning and deep learning methods has gained increasing attention in fake news detection.

However, automated feature engineering has its own pitfall. Many methods extract features that are event-specific, which makes them incapable of detecting deceptive contents on newly arrived events. To address this issue, Wang et al. (2018) built an Event Adversarial Neural Network (EANN) to extract event-invariant features by removing event-specific features captured by the event discriminators[6]. This method outperforms previous state-of-art techniques in fake news detection and shows the power of deep learning method in terms of feature extraction.

III. PROCEDURE

A. Data Preprocessing

We found two datasets on Chegg. The first one is Getting Real about Fake News[1], which contains contents and headers of over 12,000 fake news articles from real-world websites, identified by the BS Detector Chrome Extension. The second one is All The News[2], which contains real news articles from reliable sources, such as the New York Times, CNN, etc. We selected 10,000 examples from each dataset (20,000 in total) and created a vocabulary of size 209,429 and a mapping from each word to its unique index. Using this mapping, we created a fixed-length vector of dimension [1, 209429] for each news article by counting the frequency of each word appeared in both the header and the main content. We used 0 to label the fake example and 1 to label the real example. Finally, we randomly shuffled the whole dataset and splited the dataset into a training set, a validation set and a test set based on a ratio of 8:1:1.

B. Logistic Regression

We built a simple logistic regression model by first implementing a loglikelihood function that computes the loglikelihood of weight (w) and bias (b) given the data (X) and label(y).

The loglikelihood function we used is

$$PLL(\mathbf{w}) = \\ -\sum_i \log\{1+\exp\{-y_i(\mathbf{w}\cdot\mathbf{x}_i)\}\} - \frac{\alpha}{2}\sum_{j=1}^{p}\mathbf{w}_j^2$$

We then implemented the gradient ascent method by computing the gradient of our log-likelihood function and updating the weights and bias in the direction of gradient iteratively. We ran gradient ascent methods multiple times using different numbers of iterations, penalty values, and step sizes.

The gradient of the loglikelihood function is

$$\frac{\partial PLL(\mathbf{w})}{\partial w_j} = \sum_i \mathbf{x}_{ij}y_i\sigma(-y_i(\mathbf{x}_i\mathbf{w})) - \alpha\mathbf{w}_j, j > 0.$$

Using 500 samples, it took about 10 minutes for the logistic regression method to converge on a computer with 2.7 GHz Intel Core i5 processor, 8 GB 1867 MHz DDDR3 Memory, and Intel Iris Graphics 6100 1536 MB graphics card.

C. Support Vector Machine(SVM)

We trained our SVM model using the scikit-learn library. Multiple support vector machines with different kernels were trained on 1,000 training examples. Using kernels, we transformed the given "bag-of-words" features vectors into new features that describe the pairwise relationship between training examples. Since the size of the data matrix is sample size squared, the number of training examples we can use is restricted due to the limited amount of run-time memory. After multiple tests, we set the number of training examples to be 1,000 so that time of each run will be short enough to tune the model properly while the sample size is still large enough to prevent biased sampling.

We tried linear, polynomial and radial basis kernels. These three kernels are defined as below:
- Linear Kernel: $< x^T x^{'} >$
- Polynomial Kernel: $(\gamma < x^T x^{'} > + r)^d$.

d is the maximum degree; gamma and r are coefficients
- Radial Basis Kernel: $\exp\{-\gamma(\mid x - x^{'} \mid)^2\}$

γ: influence of each training sample on each other

With 1,000 training example, the training time for SVM varies from 15 minutes to 20 minutes depending on the complexity of kernel on a computer with 2.3 GHz Intel Core i5 and Intel Iris Plus Graphics 640 1536 MB.

D. Fully-Connected Neural Network

We built a fully-connected neural network and trained the model on 16,000 training examples and validated on 2,000 validation examples with a mini-batch of size 300. We tuned hyper-parameters by examining the validation accuracy at the end of each training trial. By experimenting with different choices of hyper-parameters, we achieved potentially the best model, which has 13 layers, with a test accuracy of 97%. We chose ReLU as the activation function of hidden layers to "allow effective gradient flow when the input is positive"[3] and sigmoid function as the activation function of the output layer to map the output into the interval (0, 1).

$$z_l = w_l * a_{l-1} + b_l$$
$$a_l = g_l(z_l),$$

where

$$g_l(x) = max(0, x), \text{ if } l < L$$
$$g_l(x) = \tfrac{1}{1+e^{-x}}, \text{ if } l = L.$$

During the training process, we computed the loss using the cross-entropy function after the last layer,

$$Loss = \\ -\tfrac{1}{M}\sum_{m=1}^{M} y_m log(a_m) + (1-y_m)log(1-a_m),$$

where M is the batch size. We then updated the trainable parameters of the network mini-batch by mini-batch using Adam and a step size of 10^{-4} during each epoch.

When predicting, we set 0.5 as the cutoff value so that examples with output probabilities greater than or equal to 0.5 are marked 1(TRUE), and marked 0(FALSE) otherwise.

The computational time is about 15 minutes on the Tesla V100-SXM2 GPU.

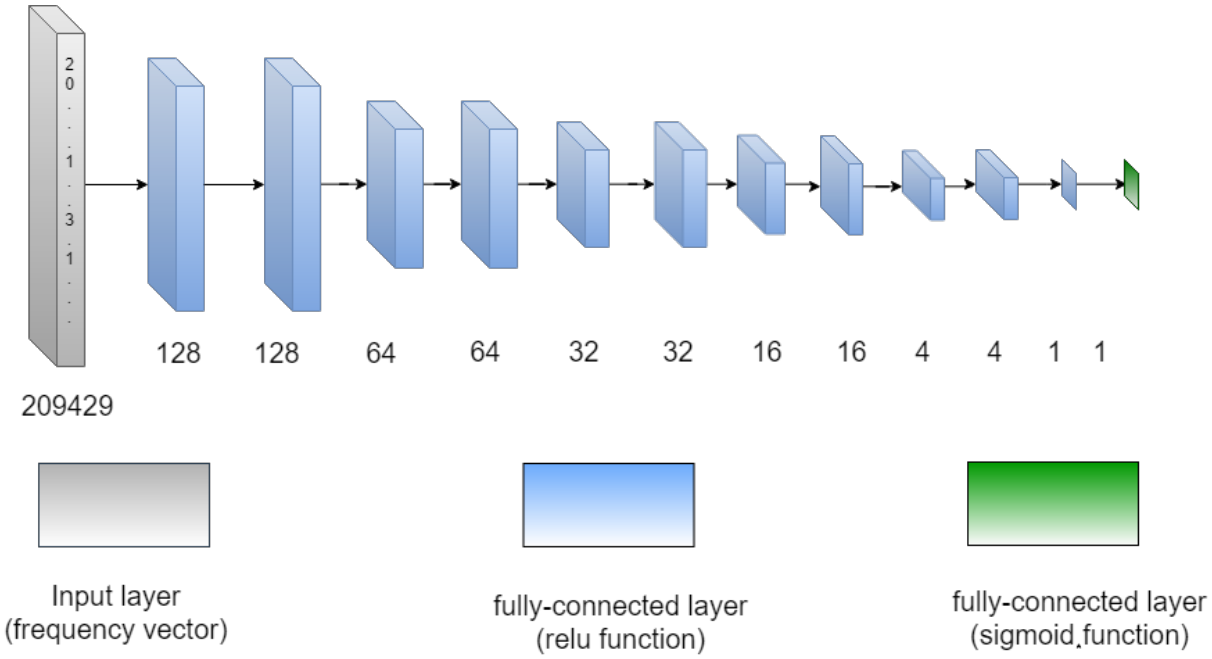

Fig. 1. The Structure of our 13-layer FC-NN

IV. RESULT

A. Logistic Regression

The overall accuracy was improved from 69.27% to 90.10% by selecting the best numbers of iterations, penalty values, and step sizes. The confusion matrix result is as follow:

Prediction \ Data	Fake	Real	Total
Fake	904	112	1016
Real	86	898	984
Total	990	1010	2000

Table 1. Confusion Matrix from Logistic regression

B. Support Vector Machine

The highest accuracy using SVM is given by linear kernel:

Prediction \ Data	Fake	Real	Total
Fake	483	40	523
Real	35	442	473
Total	518	482	1000

Table 2. Confusion Matrix from Linear SVM

We obtained an overall accuracy of 92.5%. For the purpose of fake news detection(negative sample classification), we obtained a 93.24% precision and 92.35% recall which result in an F-score of 0.9275.

RBF kernel achieved a similar F-score(0.923) while polynomial kernel has lower performance(0.863).

The issue with "bag-of-words" representation that is elaborated in "DISCUSSION" section is the most obvious in SVM model because features are formulated by its "similarity" to other training examples:

- Linear Kernel creates a projection of one example on the other
- Polynomial Kernel creates a polynomial transformation of the linear projection
- Radial Basis Function creates the Euclidean distance between two training examples

This means that fake news detection relies on the assumption that positive and negative examples should have very different frequency vectors, and a violation to this rule can lead to wrong classification.

C. Fully-Connected Neural Network

The best test accuracy was achieved by using a 13-layer fully-connected neural network specified

in Figure 1.

Prediction \ Data	Fake	Real	Total
Fake	1002	45	1047
Real	15	938	953
Total	1017	983	2000

Table 3. Confusion Matrix from FC-NN

At the end, we achieved the test accuracy of 97%, false-positive rate of 0.015, and false-negative rate of 0.048. We found that the false-negative rate is much higher than (about 3 times) the false-positive rate.

V. DISCUSSION

As we expected, logistic regression converged fastest since it had the fewest parameters while the other two methods took much more time to converge even with a better CPU and GPU configuration. The neural network had the best performance in test-time fake news detection. This is within our expectation as neural network is more complex than logistic regression and SVM, thus able to capture more variances that predict whether a given news article is fake or not.

We compared several pieces of news articles that are classified correctly with those that are not. After examining these two sets respectively, we found dominant words in successfully classified news are more meaningful (e.g. “shooting”, “office”, “stocks”, “Trump”, etc.), while those in the news that we incorrectly classified are less meaningful (e.g. “a”, “that”, “and”). Intuitively, it implies that incorrectly classified news have few distinguishable features since usual and meaningless words (whose frequencies are about the same among all articles) stand out in the feature vectors.

Failing to classify news that is dominated by usual and meaningless words is probably caused by not standardizing feature vectors. In addition, the use of “bag-of-words” feature representation limited our model’s ability to analyze and extract important features, such as sentiment or phrasing, which are poorly represented by word frequency. In order to address this problem, future models should consider standardizing feature vectors and incorporating more features that captures linguistic patterns and sentiments which are significant in deciding the authenticity of a piece of news.

VI. CONCLUSION

In conclusion, we developed a fake news detector using fully-connected neural networks after comparing the performance of three machine learning methods mentioned above. The test performance of our detector achieved an accuracy of 97%. At the same time, we discovered a shortcoming of our current “bag-of-words” feature representation, which failed to capture linguistic patterns and the tone of the speech. We propose that future fake news detector models should consider standardizing features and incorporating more features such as linguistic patterns and sentiments in order to better represent news articles.

REFERENCES

[1] Risdal, M. (2016, November 25). Getting Real about Fake News. Retrieved from https://www.kaggle.com/mrisdal/fake-news

[2] Thompson, A. (2017, August 20). All the news. Retrieved from https://www.kaggle.com/snapcrack/all-the-news

[3] Ramachandran, Zoph, Barret, V., Q. (2017, October 27). Searching for Activation Functions. Retrieved from https://arxiv.org/abs/1710.05941

[4] Murphy, K. P. *Machine Learning - A Probabilistic Perspective*. Cambridge, Massachusetts, The MIT Press.

[5] Conroy, N. J., Rubin, V. L., Chen, Y. (2016, February 24). Automatic deception detection: Methods for finding fake news.

[6] Wang, Y., Ma, F., Jin, Z., Yuan, Y., Xun, G., Jha, K., Gao, J. (2018). Eann. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining - KDD 18. doi:10.1145/3219819.3219903ll