Evaluation of Nonparametric Bayesian Classification Methods for Predicting Fake News from Article Titles

Nathaniel Hawkins

1 Introduction

It is growing increasingly important to remain up to date with current events in our modern-day political climate. The proliferation of "fake news" as a term for slander has cast doubt on media and news sources alike, making it challenging for individuals to distinguish between the true sources of information, and borderline propaganda meant to lower the popular opinion of a particular candidate. It is, therefore, important for checks and systems to lie in place that can quickly and accurately determine whether or not the information one is consuming is legitimate. In this work, we will evaluate two nonparametric Bayesian classification methods, k-nearest neighbors (KNN) and Parzen window (PW), for their ability to distinguish between real news and "fake news." We will compare these nonparametric methods to a suite of baselines classifiers in the scikit-learn (sklearn) python library. Through these efforts, we hope to not only contribute to the field of fake news detection, but answer two major questions: 1) how accurately can this prediction task be done using the title of the news article alone?, and 2) how well do nonparametric methods perform compared to some popular and simple classifiers?

The field of fake news detection has largely been dominated by deep-learning-based methods[1]. The field largely began making progress starting in 2017 following major progress in the field of deep learning in general[2]. A wide variety of deep learning methods have been implemented for classifying fake news. These include fully connected deep networks[3, 4], long short term memory networks [5, 6], bi-directional long short term memory

networks [7, 8, 9, 6], convolutional neural networks [5, 6], graph convolutional neural networks[10], recurrent neural networks[11, 12], and sophisticated language-based models like BERT[9, 13]. These models have achieved variable success in classifying fake news with accuracies topping out well over 98%. While these methods are highly accurate and complex, the time required to train and implement these models is substantial. The hardware and time requirements for training and utilizing these models often exceeds that available to the typical user. Additionally, as more training data becomes available, retraining and reiplementing these methods quickly becomes infeasible. Therefore, these "simpler" models, while traditionally achieving lower performances comparably[1], are easier to implement, easier to retrain, and much less computationally costly. The features used in the classification task have also widely ranged. They have included continuous bag of words representations, TF-IDF vectorizations, and neural network embeddings like ELMo and GloVe. In this work, we will evaluate the effectiveness of nonparametric models on the simplest feature set: a one-hot encoding of the article titles. While a one-hot encoding of article titles is less linguistically informative compared to something like a neural network embedding, it allows us to focus the prediction task on the first thing a reader would see (i.e., the title) and further reduces computational time by making the feature space sparser. Though, we note that using more robust feature sets, including the contents of the article itself, pose an area of future work for this project.

2 Data

To carry out this work, we utilized a publicly available dataset on Kaggle. The dataset can be found by clicking this text. The features of this dataset is summarized in Table 1. The news articles range from early 2015 to mid 2018. Predominantly, the topics of these articles are political news and world news, which addresses the motivation behind this work well. There is a slight class imbalance present in this data with approximately 1,000 more fake news articles than true news articles. This class imbalance is negligible compared to the total number of samples (approximately 1-2%), but we will address this issue using class-balanced 5-fold cross validation and averaging the performance across folds. By doing so, the imbalance will be "averaged" out and we project it will not significantly impact performance.

Number of Article Titles	44,266
Number of Article Titles (True)	21,416
Number of Article Titles (Fake)	22,850
Mean Length of Titles (Words)	9.29
Median Length of Titles (Words)	9
Minimum Length of Titles (Words)	1
Maximum Length of Titles (Words)	29
Number of Features After One-Hot Encoding	18,206

Table 1: Summary of dataset used in this work. Lengths shown in this table are the texts following preprocessing.

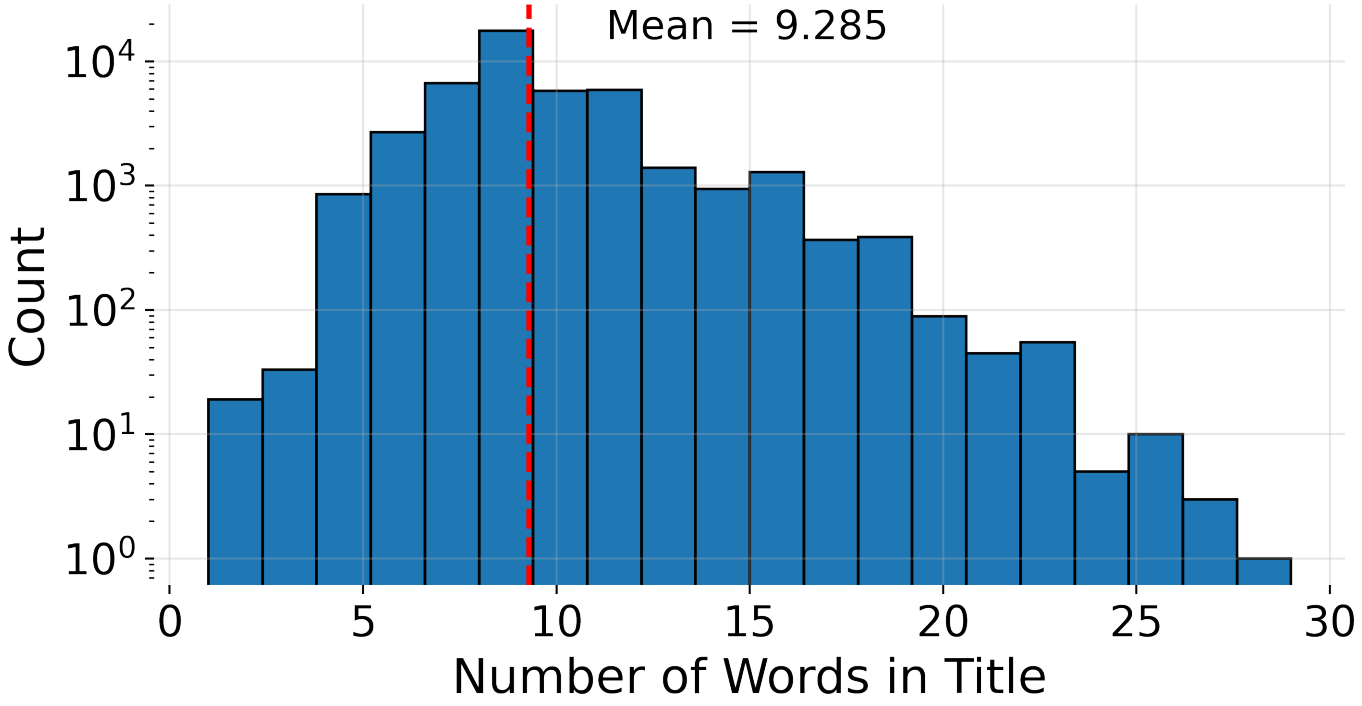

Figure 1: Histogram of length of article titles. Mean description shown with verticle red line. Y-axis is logarithmically scaled.

3 Methods

Prior to training our classifiers, we first preprocessed the titles of the articles before vectorizing them to create our feature set. The preprocessing follows most of the standard steps used in other works[1]. We first extracted all of the titles from the dataframe for fake and true news. Then, we removed non-UTF-8 enocdable characters, removed punctuation from the titles using regular expressions, filtered out stopwords (e.g., "at," "and," "the,") using the list of english stopwords in the NLTK python library, and stemmed the text using the Porter stemming algorithm in NLTK. Text preprocessing is a staple for most NLP tasks because it removes language artifacts and casts the words into a more general form. In the case of this works, the text preprocessing reduces the dimensionality of the final feature space once we vectorize it. The preprocessed article titles are cast to numerical representations by turning them to one-hot vectors. The size of the vector is equal to the total number of unique words across all titles in the corpus. Each index corresponds to a single word. To vectorize a title, the feature vector gets a value of 1 if the word is present at all in that title (no matter how many times) and a 0 if it is not. This vectorization is done using the `feature_extraction.text.CountVectorizer(binary = True)` function in sklearn.

The baseline models used in this work are all implemented in sklearn. The baseline models are described in Table 2. For this work, we chose some of the most widely used classifiers in the sklearn library. We implemented two supper vector machine classifiers, one using a linear kernel and one using the radial basis function, logistic regression, multiplayer perceptron, and naive bayes as our baselines. All classifiers use the default parameters used by sklearn. In this way our baseline models are not optimized classifiers with finely tuned hyperparameters.

We used the sklearn implementation of KNN as it is more computationally efficient than what we could implement. For PW, we created a simple implementation (see `parzen.py`). Our implementation of PW follows the algorithm outlined in class. To evaluate these nonparametric methods, we wanted to test a wide variety of hyperparameters. For KNN, tested with k =3,5,7,9,11,13,15,17,19, and 21, and for PW, we tested $h = 0.01$,0.05,0.1,0.5,1,5,10, and 50.

For evaluation metrics, we report the accuracy score, area under the precision recall curve (auPRC), area under the receiver operator curve (auROC),

Method	Accuracy	auPRC	F1	auROC	Time
Linear SVM	0.95	0.92	0.95	0.95	19752.63
Log Reg	0.95	0.92	0.95	0.95	126.59
MLP	0.94	0.90	0.93	0.94	1506.45
NB	0.80	0.71	0.82	0.80	21.52
RBF SVM	0.95	0.93	0.95	0.95	25225.82

Table 2: Baseline model results averaged over 5-fold cross validation. Times shown are in seconds. Linear SVM: sklearn implementation of support vector classifier (SVC) with default parameters using a linear kernel. Log Reg: sklearn implementation with default parameters. MLP: sklearn implementation of multilayer perceptron with default parameters. NB: sklearn implementation of naive bayes classifier with default parameters. RBF SVM: sklearn implementation of SVC with default parameters using radial basis function as the kernel.

KNN K-Value	Accuracy	auPRC	F1	auROC	Time

Table 3:

F1 score (F1) and the computation time. We chose to report all of these metrics because each details a difference aspect of the classifiers. Accuracy and auROC give a broad description of the classifier and are the most widely used classifier metrics, but they are insensitive metrics when class imbalances are present. As noted, the imblanace here is comparatively minimal, but we want to include metrics which are more sensitive to imbalances. Therefore, we also include auPRC and F1, which take into account both precision and recall. Computation time is reported to give an idea of the cost associated with using these classifiers. Reported metrics are averaged over 5-fold cross validation. Class balance is maintained across 5-fold cross validation to ensure representative training.

4 Results

The results for our baselines are shown in Table 2. The KNN and PW results are shown in Table 3 and 4, respectively. A plot showing performance versus computation time is shown in Figure **??**.

Parzen h	**Accuracy**	**auPRC**	**F1**	**auROC**	**Time**

Table 4:

5 Discussion and Conclusions

References

[1] R. Oshikawa, J. Qian, and W. Y. Wang, "A survey on natural language processing for fake news detection," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, p. 6086–6093, 2020.

[2] G. Bhatt, A. Sharma, S. Sharma, A. Nagpal, B. Raman, and A. Mittal, "On the benefit of combining neural, statistical and external features for fake news identification," 2017.

[3] A. Thota, P. Tilak, S. Ahluwalia, and N. Lohia, "Fake news detection: A deep learning approach," *SMU Data Science Review*, vol. 1, no. 3, 2018.

[4] T. Saikh, A. De, A. Ekbal, and P. Bhattacharyya, "A deep learning approach for automatic detection of fake news," 2020.

[5] Álvaro Ibrain Rodríguez and L. L. Iglesias, "Fake news detection using deep learning," 2019.

[6] S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, and M. Akbar, "Fake news detection using deep learning models: A novel approach," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3767, 2020. e3767 ETT-19-0216.R1.

[7] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, "Declare: Debunking fake news and false claims using evidence-aware deep learning," 2018.

[8] E. Qawasmeh, M. Tawalbeh, and M. Abdullah, "Automatic identification of fake news using deep learning," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pp. 383–388, 2019.

[9] M. A. Ayat Abedalla, Aisha Al-Sadi, "A closer look at fake news detection: A deep learning perspective," in *ICAAI 2019: Proceedings of the 2019 3rd International Conference on Advances in Artificial Intelligence*, pp. 24–28, 2019.

[10] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," 2019.

[11] S. Girgis, E. Amer, and M. Gadallah, "Deep learning algorithms for detecting fake news in online text," in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pp. 93–97, 2018.

[12] S. Singhania, N. Fernandez, and S. Rao, "3han: A deep neural network for fake news detection," in *Neural Information Processing* (D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, eds.), (Cham), pp. 572–581, Springer International Publishing, 2017.

[13] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," 2019.

[14] Z. Zhou, H. Guan, M. M. Bhat, and J. Hsu, "Fake news detection via nlp is vulnerable to adversarial attacks," 2019.

[15] B. Guo, H. Wang, Y. Ding, W. Wu, S. Hao, Y. Sun, and Z. Yu, "Conditional text generation for harmonious human-machine interaction," 2019.