CIS 410 Final Project

Cameron Cunningham

March 2021

1 Introduction

Fake news has been an important and much-discussed problem in recent years, and every year it seems to get worse. With the increase in usage of social media websites such as Facebook and Twitter, users are being exposed to articles that are inaccurate and misleading, often with politically- or profit-driven motives, and this is what we call fake news. After the influence of fake news in the 2016 presidential election, websites began initiatives to fight its spread and influence. Despite this, it's still a very real problem.

With this in mind, my goal was to approach this issue as a classification problem and create a system that could identify a given article as fake or real news using NLP methods and techniques. My original idea was to do this using a Naïve Bayes Classifier, and using a neural network, and comparing the results; unfortunately I didn't have enough time to implement the neural network, so the final result was simply a Naïve Bayes Classifier for detecting fake news.

2 Related work

Fake news has been in the public sphere since around 2016, with interest increasing in 2018. The term grew quickly in popularity and usage, and it wasn't long until it became the subject of academic studies and articles [Lazer et al., 2018], with interest in systems that could automatically detect it showing up as early as 2017 [Pérez-Rosas et al., 2017]. As detailed in *A Survey on Natural Language Processing for Fake News Detection* [Oshikawa et al., 2018], fake news studies usually approach fake news as either a classification problem, which puts an article into the class of "fake news" or "real news," or a regression problem, which would assign a numeric score representing how "fake" or "real" an article is. It's not just articles either; other media that can be fake news includes

general claims, which may not be confined to one publication, and posts on social media. Models used to evaluate fake or real news include Support Vector Machines, Naïve Bayes Classifiers, and neural networks such as RNNs and CNNs, as well as less conventional models such as Rhetorical Structure Theory and Recognizing Textual Entailment-based models.

There is additional research on the vulnerabilities of fake news detecting models. For example, while models can quite accurately predict obvious fake news articles, they are less accurate when it comes to articles that mimic real news articles [Zhou et al., 2019].

3 Methods

I framed this project as a classification problem; that is, I wanted to create a model that could classify news as fake or real news. I wanted to make a Naïve Bayes Classifier and a neural network (probably a recurrent neural network) and compare the performance of each. Unfortunately I ran out of time, so I just ended up creating a Naïve Bayes Classifier from scratch. I followed the common approach as described in the textbook, using a bag of words approach. I used two classes, though I found datasets that could have allowed me to have more classes. Thus, my model essentially returns $\max\{P(\text{fake news}|d), P(\text{real news}|d)\}$ for a given document d. Using the naïve Bayes assumption, for a given class c and assuming that d is represented as a set of features $\{f_1, f_2, \ldots, f_n\}$, we can write

$$P(c|d) = P(f_1|c)P(f_2|c)\ldots P(f_n|c)P(c)$$

and thus, the likelihood $P(c|d)$ can be calculated.

In practice, the hardest part turned out to be preprocessing. I needed to create a corpus of training documents for training and testing from the corpus document I downloaded at `www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset`, and then go through each one and create a vocabulary. I took this opportunity to also count the number of occurrences of each word in each class. This took a while, so to save time during testing the program saves the structures in pickle files so that training can be run without compiling these structures each time.

The model is then trained using this data, calculating the logprior (i.e. $P(c)$) for each class, and the probability of each token occurring with that class ($P(f_i|c)$), and stores those in files that

can be used for testing.

4 Experiments

To begin with, I just wanted to get my classifier working. To my surprise, simply implementing the basic classifier on data processed with whitespace tokenizing performed with an accuracy of 97.51%. Interestingly, it always seemed to do better with fake news documents than with real news documents. I suspect the reason has to do with the training data I used. Fake news detection models often do better with obvious fake news (e.g. articles that don't go along with headlines and articles including inflammatory or biased words) than with more subtly crafted articles that mimic real news [Zhou et al., 2019]. So, if a corpus has more obviously fake news articles than subtly fake, then it would make sense that it would perform well. If I'd had more time, I would have experimented with more corpora, but unfortunately this will remain an untested hypothesis for now.

I also experimented with using treebank tokenizing instead, rather than simple whitespace tokenizing. This actually brought the overall score up to 98.76% at first. Unfortunately I changed something in the process of experimentation, and the score dropped even when I thought I'd changed it back, so it may have been a mistake. The performance after this was just 96.77% accuracy, so I can't say exactly what happened.

There was more experimenting I wanted to do, but unfortunately this is all I could do in the time I had. If I'd had more time I would have wanted to figure out what happened to bring down the score when parsing documents using whitespace tokenizing. I also would have wanted to experiment with adding more features, such as website, publisher, title, author, and so on, as well as trying out more datasets, perhaps datasets that include more subtly fake news. Sadly, I will have to end my experimentation here and save these other questions for another time.

5 Conclusion

An NLP model as simple as even a Naïve Bayes Classifier can label fake and real news with surprising accuracy, at least given certain restrictions. This could have plenty of good applications in combating fake news, such as apps and programs, websites, or browser plugins, being a relatively

light program that isn't too hard to retrain as articles are encountered. Preprocessing the text is very important in getting the model to work, and for now it seems simple whitespace tokenizing may be the best approach, though this might not be the case. NLP is a powerful tool that promises to be very effective in combating the influence of fake news, now and in coming years.

References

[Lazer et al., 2018] Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J., and Zittrain, J. L. (2018). The science of fake news. *Science*, 359(6380):1094–1096.

[Oshikawa et al., 2018] Oshikawa, R., Qian, J., and Wang, W. Y. (2018). A survey on natural language processing for fake news detection. *CoRR*, abs/1811.00770.

[Pérez-Rosas et al., 2017] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., and Mihalcea, R. (2017). Automatic detection of fake news. *CoRR*, abs/1708.07104.

[Zhou et al., 2019] Zhou, Z., Guan, H., Bhat, M. M., and Hsu, J. (2019). Fake news detection via NLP is vulnerable to adversarial attacks. *CoRR*, abs/1901.09657.