Classifying Fake News

C. Fan

Introduction

Problem

Leading up to the 2016 election, fake news thrived on social media. Fake news stories generated millions of clicks, providing ad revenue to websites that hosted the news stories. One analysis found that, on Facebook, the top 20 fake news stories outperformed the top 20 real news stories by shares, comments, and reactions. The top election news story, "Pope Francis Shocks World, Endorses Donald Trump for President, Releases Statement," was fake (Silverman, 2016).

Following Donald Trump's victory in the 2016 election, some political pundits claimed that the proliferation of fake news influenced the election results. A study by two economics professors found that fake news stories supporting Trump were shared 30 million times on Facebook, while fake news stories supporting Clinton were shared 8 million times (Allcott & Gentzkow, 2017). The study concluded that though fake news may have contributed to the outcome, it likely did not change the outcome by swaying voters. Rather, fake news reinforced existing political biases. People were more likely to share stories that supported their own viewpoints.

Nonetheless, there were calls for tech companies to be more accountable, as their websites are what many internet users engage with and they are the arbiters of what content is displayed. In response, Google and Facebook stated they would make changes to starve fake news sites of

ad revenue (Love, 2016). In addition, they would update their algorithms so that fake news articles would be less likely to trend. In a blogpost, Facebook said it updated its Trending stories algorithm to use groups of articles shared on Facebook instead of only using mentions of a topic. This way, a highly-mentioned topic from one article does not trend (Cathcart, 2017). Facebook updated its UI to allow users to flag articles as fake, and fact-checked articles marked as fake display "Disputed by 3rd Parties" (Mosseri, 2016).

Even before the election, there have always been hoaxes and conspiracy theories shared on the internet. But the 2016 election brought this issue into prominence. In a reversal, following the election, an editor of fact-checking site Snopes said there has been a spike in fake news aimed at liberals (Meyer, 2017). Fake news is an ongoing problem.

In this paper, I attempt to build a classifier that categorizes news as fake or real.

Data

To make the classifier, I used news article copy. I used articles from *The New York Times* and the *Wall Street Journal*, which are known to be reputable news sources (even though they have a liberal and conservative bias, respectively). To push for subscriptions and maintain profitability, these newspapers limit the number of articles that can be viewed for free, and their APIs do not provide full article copy. I found a website that keeps copies of articles from major news sources (http://eventregistry.org/) and wrote a script to get 9,000 articles from the past month.

Finding a dataset for fake news proved to be trickier. For one thing, I did not want to nor could I have read and classified each article myself. I could not scrape and label articles from a single website as fake, because though some websites are known to have fake news, such as the alt-right *Breitbart News*, that does not mean that all the articles are fake. Also, for an individual fake news article, the article is not completely false; its content could run from the continuum of mostly true to mostly false statements. I did not want to include satirical articles, like the articles that could be found in *The Onion* or *The Borowitz Report*. These articles are known to be satire by the general public and are self-labelled as such, with no intention to mislead or profit from sensationalist headlines, and they are well-written like a real news article. I ended up using a dataset of ~13,000 fake news posts from this website: https://www.kaggle.com/mrisdal/fake-news. This dataset was generated from scraping websites marked as "bullshit" by the BS Detector Chrome extension (https://github.com/selfagency/bs-detector).

A classifier tailored to each domain would be better than a general model, so that there is less noise. Ideally, each real news article in the model had a fake news article counterpart. For example, a real news article that says Clinton won the popular vote has a counterpart in an article that says Trump won the popular vote (like the article from *70news* that was the top Google search result for "popular vote" following the election). But for many articles, they do not have a counterpart. For example, *New York Times* articles run the gamut from politics to lifestyles. While there are fake news articles about politics, I doubt there is much fake news about spring fashion trends or wedding announcements. On the other hand, some fake news

stories are so outlandish that one would be hard-pressed to find a corresponding real news story. The wide range of article topics increases the sparsity of the features.

Possible model features

When I was thinking about possible features for detecting fake news, I noticed that they could be separated into two major categories: indicators of writing quality and indicators of strong sentiment.

Writing quality

Real news sources have higher quality writing. Newspapers employ trained journalists, fact-checkers, and editors. Fake news articles are less likely to be written by professional writers—the articles may very well be written by teenagers in Macedonia (Kirby, 2016)! Presumably, being a salaried journalist allows one to spend more time writing each article, and to report with less bias, instead of resorting to sensationalist tactics to make a quick buck.

Misspelled words

Newspapers editors check spelling and grammar, so I would expect real news sources to have fewer misspellings. I used the PyEnchant library to check for misspelled words, and set the feature to be the number of misspelled words divided by the total number of words for each article.

Punctuation (,;)

Colons and semi-colons suggest complex sentence structure and thought, an indicator of quality writing.

Average sentence length and average word length

Intuitively, a long article is a good indicator of quality investigative journalism, but I did not want the model to penalize short articles, so average sentence length and average word length was used as a feature to suggest complex thought.

Strong sentiment

Real news sources are more likely to use neutral words, as they try to appear unbiased, describing events with facts. In opinion pieces, writers try to make rational, evidence-based arguments (op-eds with unsubstantiated claims are less likely to be published).

Fake news is written to elicit strong emotions. In a study of hotel reviews, fake reviews had more exaggerated sentiment (Ott, 2013). Emotional language makes a reader more likely to click and share an article.

Capitalized words and punctuation (?!)

Capitalized words, exclamation marks, and question marks could represent strong sentiment, especially multiple punctuation marks at the end of a sentence (e.g. “YOU WON’T BELIEVE THIS!!!!”), something fake news would be more likely to have. This was confirmed in the sample (see Table 2).

Word count/n-grams

Short phrases have more context than individual words, so short phrases capture sentiment better. I tested 1-grams, 2-grams, and 3-grams, but with higher n-grams, frequency would be too sparse.

Longer documents have higher word count values than shorter documents. If raw word counts were used as features, then longer documents would have an outsize influence. So, I converted the tokens to term frequencies (tf).

Stop words such as “the” and “a” are common, so they do not provide meaningful information to the classifier. These stop words would overshadow less common, more interesting terms (Pedregosa et al., 2011). I used a term frequency times inverse document frequency transform (tf-idf) to downscale the frequencies of common words.

For a term t, news article/document d, and total number of documents n_d, we have:

$$tf\text{-}idf(t,d) = \text{term frequency}(t,d) \times \text{inverse document-frequency}(t)$$

$$\text{inverse document-frequency}(t) = log\frac{1+n_d}{1+df(d,t)} + 1$$

where $df(d,t)$ is the number of documents that contain t.

Then each $tf\text{-}idf(t,d)$ was normalized with the L2 norm.

First person, second person, third person pronouns

An article that is reporting events is written in third person, so there would be a higher frequency of third person pronouns (he, she, it, him, her). First person and second person

pronouns (I, we, you) are likely used to indicate someone's opinion or experience, so will convey stronger sentiment.

Building the model

Comparison of some classification methods

The following is a high-level overview of methods used for classification.

Naïve Bayes

Naïve Bayes follows from the Baye's theorem, and applies the "naïve" assumption that features are independent, that is, the presence of a word in an article does not have any bearing on the presence of another word in the same article. For news articles, this independence assumption is obviously false, because if an article has "Hillary" in it, it will likely have "Clinton." If two features are dependent, then Naïve Bayes will multiply the probabilities as if they were independent, overestimating the probability that an article belongs to a certain class. For that reason, Naïve Bayes ought not to perform well for text classification. However, it turns out that Naïve Bayes performs well even with strong feature dependencies, because the dependencies tend to cancel each other out (Zhang, 2014).

For class c (real or fake) and features $x_1, x_2, \dots x_n$ (the models described later will use tf or tf-idf for features):

$$P(c|x_1, x_2, \dots x_n) = \frac{P(c)P(x_1, x_2, \dots x_n|c)}{P(x_1, x_2, \dots x_n)}$$

Since we assume that features/words are independent:

$$P(x_i|c, x_1, \dots, x_{i-1}, x_{i+1}, \dots x_n) = P(x_i|c)$$

$$P(c|x_1, x_2, \dots x_n) = \frac{P(c)\prod_{i=1}^{n} P(x_i|c)}{P(x_1, x_2, \dots x_n)}$$

$P(x_1, x_2, \dots x_n)$ is constant, so for a given article, the classification of real or fake is:

$$\arg\max_c P(c) \prod_{i=1}^{n} P(x_i|c)$$

I used the scikit learn library's multinomial Naïve Bayes model with Laplace smoothing. The multinomial distribution with Laplace smoothing is so that if a feature/word is not present in the training data, it will not result in a zero posterior probability; otherwise, the model will perform poorly, erroneously suggesting certain events are highly likely or impossible.

Naïve Bayes has the benefit of training quickly, but it does not learn how features interact.

Logistic regression

With logistic regression, each word feature x_i is given weight w_i. Then the probability for each classification (real or fake) can be represented by logistic functions:

$$P(c = \text{real}|x_1, x_2, \dots x_n) = \frac{1}{1 + e^{w_0 + \sum_{i=1}^{n} w_i x_i}}$$

$$P(c = \text{fake}|x_1, x_2, \dots x_n) = 1 - P(c = \text{real}|x_1, x_2, \dots x_n) = \frac{e^{w_0 + \sum_{i=1}^{n} w_i x_i}}{1 + e^{w_0 + \sum_{i=1}^{n} w_i x_i}}$$

where w_0 is the bias.

Logistic regression has many pros: it allows for probabilities to be modeled (unlike decision trees and SVMs), features can be dependent (unlike Naïve Bayes), and it is easy to update the model with new data.

Decision trees

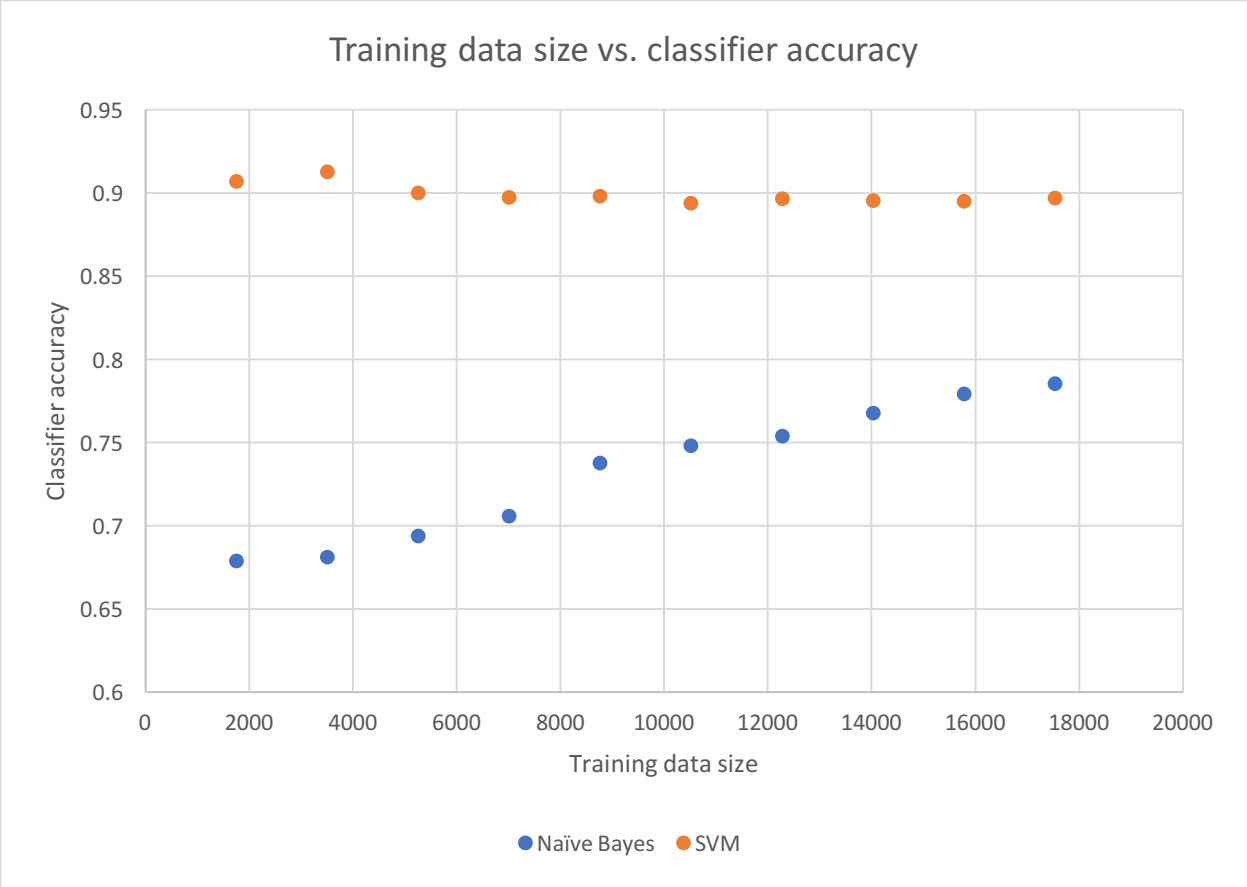
A decision tree is a set of decision nodes with a tree graph structure, like a flowchart. To make a decision, we start at the root. At each decision node, the input (information about the news article) is tested, and based on the outcome, we go along a branch to the next node. If the node is another decision node, we are again passed to a branch depending on the test outcome. If the node is a leaf node, that is the classification that the decision tree has arrived at (real news or fake news). Pros of decision trees are, features can be dependent, classes do not need to be linearly separable, outliers are handled well, and the decision tree itself is easy to interpret. For the fake news classifier, I generated 161,870 features. Given the large number of sparse features, a decision tree would overfit, so perform poorly on test data.

Support vector machines

A support vector machine model generates a hyperplane that separates the training data as far as possible, and in this way is a binary classifier. SVMs generally require tuning and are memory-intensive. Since there is an extremely high number of features in a text classification problem, the support vector machine performs well.

Naïve Bayes vs. SVM classifier performance

The real news and fake news datasets were randomly split so that 80% of the data was used for training and 20% for testing. From that training dataset, I tested randomly generated subsets of different sizes to see how the size of the training data impacted the classifier.



The accuracy of the Naïve Bayes classifier improved with the training data size, whereas the accuracy of the SVM classifier remained roughly the same (see Table 2).

Tuning the SVM model

Since SVM significantly outperformed the Naïve Bayes classifier, I decided to focus on optimizing the SVM. I performed a grid search to find optimal values for tuning parameters: alpha of 0.01 or 0.001; n-grams of size 1, 2, or 3; and to use tf or tf-idf. The grid search used those 12 combinations of parameters to find a best value, which was alpha 0.001, 1-grams, and to use tf-idf. The 1-grams finding the best value was contrary to my intuition. I thought 2-grams

would perform better, to retain some contextual information, but perhaps the 2-grams were too sparse and noisy.

Conclusion

The SVM fake news classifier performed well in-sample. But I wonder how it would perform using news articles written a year from now. In the future, there will be new hoaxes and fake stories, and this model will become outdated. Unless the classifier discovered some underlying information about fake news in general, the classifier would need to be retrained with new articles to stay relevant.

If I had more time, I would try other classifiers, such as boosted trees (with careful feature selection and pruning to prevent overfitting) or logistic regression. Alternatively, different types of classifiers and heuristics could be combined into an ensemble model, which has good success in industry (Conroy, 2015).

There are many ways to improve this model. As is the case with most models, additional data will improve the model, such as articles that provide some explanatory power, or additional significant features. The word frequency features could be improved by upweighting important words. For example, title words, the first sentence of each paragraph, and sentences that contain title words could be counted twice (Manning, Raghavan, & Schütze, 2008).
The news could be tagged with additional metadata, such as science or politics. Then the classifier could pick up idiosyncrasies particular to that domain. I wanted to see how the

classifier would perform solely based on article text. But in practice, other metadata would be useful. For example, the URL of an article would be a strong indicator. For news agencies that uphold journalistic standards, news stories are likely to be real, so an article from wsj.com (the *Wall Street Journal*) strongly implies a real news story.

Bibliography

Allcott, H., & Gentzkow, M. (2017, January). *Social Media and Fake News in the 2016 Election.* Retrieved from https://web.stanford.edu/~gentzkow/research/fakenews.pdf

Cathcart, W. (2017, January 25). *http://newsroom.fb.com/news/2017/01/continuing-our-updates-to-trending/.* Retrieved from Facebook newsroom: http://newsroom.fb.com/news/2017/01/continuing-our-updates-to-trending/

Conroy, N. J., Rubin, V. L. and Chen, Y. (2015), Automatic deception detection: Methods for finding fake news. Proc. Assoc. Info. Sci. Tech., 52: 1–4. doi:10.1002/pra2.2015.145052010082

Kirby, E. (2016, December 5). *The city getting rich from fake news*. Retrieved from BBC News: http://www.bbc.com/news/magazine-38168281

Love, J. (2016, November 15). *Google, Facebook move to restrict ads on fake news sites.* Retrieved from Reuters: http://www.reuters.com/article/us-alphabet-advertising-idUSKBN1392MM

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press.

Meyer, R. (2017, February 3). *The Rise of Progressive 'Fake News'.* Retrieved from The Atlantic: https://www.theatlantic.com/technology/archive/2017/02/viva-la-resistance-content/515532/

Mosseri, A. (2016, December 15). *News Feed FYI: Addressing Hoaxes and Fake News.* Retrieved from Facebook newsroom: http://newsroom.fb.com/news/2016/12/news-feed-fyi-addressing-hoaxes-and-fake-news/

Ott, M., Cardie, C. & Hancock, J. (2013). Negative Deceptive Opinion Spam. Proceedings of NAACLHLT. pp. 497–501

Pedregosa et al. (2011, October 12). Feature extraction. Retrieved from scikit learn: http://scikit-learn.org/stable/modules/feature_extraction.html

Pedregosa et al. (2011, October 12). Working With Text Data. Retrieved from scikit learn: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Silverman, C. (2016, November 16). *This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook.* Retrieved from Buzzfeed: https://www.buzzfeed.com/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

Spencer, S. (2017, January 25). *How we fought bad ads, sites and scammers in 2016*. Retrieved from The Keyword: https://blog.google/topics/ads/how-we-fought-bad-ads-sites-and-scammers-2016/

Zhang, Harry. (2004). *The Optimality of Naive Bayes*. http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

Zhang, H., Fan, Z., Zeng, J. & Liu, Q. (2012). An Improving Deception Detection Method in Computer-Mediated Communication. Journal of Networks, 7 (11).

Appendix

Table 1

Article counts for sentence-ending punctuation

	# articles				
	!!	??	!	?	total
real news	16	2	700	2575	8922
fake news	309	184	3725	5739	12999

Table 2

Training data size vs. classifier accuracy data

Training data size	Naïve Bayes	SVM
17536	0.785404789	0.896921323
15782	0.779019384	0.894868871
14028	0.767616876	0.895096921
12275	0.753705815	0.896465222
10521	0.748004561	0.89372862
8768	0.737742303	0.898061574
7014	0.705815279	0.897149373
5260	0.69395667	0.899885975
3507	0.681185861	0.912656784
1753	0.678905359	0.90672748