Exploiting Network Structure to Detect Fake News

Stanford University, CS229, Fall 2018

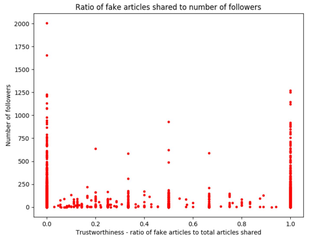
Neel Ramachandran (neelr@stanford.edu) , Anika Raghuvanshi (anikar@stanford.edu), Meghana Rao (mvrao@stanford.edu)

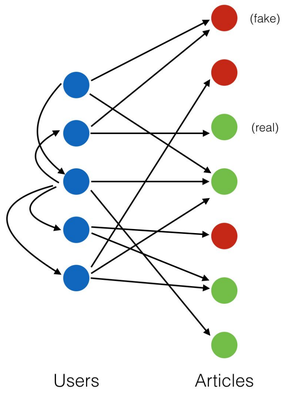
Motivation

The spread of fake news and false information through social media has become an important topic since the 2016 elections. Most work in fake news detection uses purely text-based models to classify articles as fake news. In this project, we aim to analyze the network structure of news propagation and leverage relationships between users and articles in order to improve on text-based models.

Dataset

Our dataset comes from *Shu et al.*'s FakeNewsNet repository [1], which contains 422 articles that were posted on Twitter in 2017, including 211 articles labeled as 'fake' and 211 labeled as not 'fake' by BuzzFeed and PolitiFact's fact-checking services. In addition to the article text and titles, the data contains information about user-user relationships (ie, who follows who) and user-article relationships (ie, who tweets which articles).





Methods and Models

Baseline

We implemented a Naive Bayes and a logistic regression model for our simplistic baseline approach.

Words Most Indicative of Fake vs. Real News:
Fake News, Naive Bayes: 1. Fake, 2. Marijuana, 3. Murder, 4. Sex, 5. Prison
Fake News, Logistic Regression: 1. President, 2. Your, 3. Were, 4. Comments, 5. White
Real News, Naive Bayes: 1. Debate, 2. Voting, 3. Candidates, 4. Keith, 5. September
Real News, Logistic Regression: 1. Story, 2. Black, 3. Debate, 4. Trump, 5. September

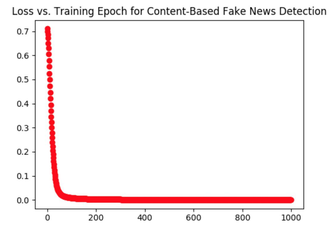
Observing the top words (especially those identified by Naive Bayes), there appears to be a stark difference in topics and content of the fake and real news stories in the dataset.

Iterative Classification

To improve on the baseline, we incorporate the network structure into our logistic regression classifier using iterative classification techniques suggested by Neville and Jensen [2]. In iterative classification, we utilize the baseline logistic regression classifier to compute 'soft assignments' for the articles in our test set. These soft assignments are used to compute new features for a second classifier that incorporates the user-user and user-article relationships. Then, we can compute the soft assignments and classify repeatedly until convergence.

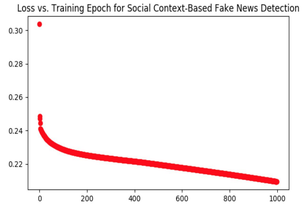
Article-Context Neural Network

We implemented a neural network in PyTorch, using the words from an article and its weighted title as input features. The network has a linear layer, a ReLU activation layer, a second linear layer, and a sigmoid output layer. The NN has 10 hidden units with binary cross entropy loss and stochastic gradient descent with momentum.



Social-Context Neural Network

For social context, we made use of certain information for each article including (1) how many times the article was shared, (2) users who shared the article, (2) users who viewed the article and (4) how many articles the sharer of the article has shared. Each article has one feature for each user, where the value at the index of user i is a certain weight if the user has shared the article and a certain lesser weight if the user has been exposed to the article (i.e. is a follower of someone who shared the article). The last 2 feature rows encompass (1) and (4) described above.



Hybrid: Article-Context and Social-Context

The hybrid method is simply a combination of the previous two methods.

Results

Model	Training Error	Test Error
Naive Bayes	1.6%	32.2%
Logistic Regression	0%	23.4%
LR with Iterative Classification	0%	**19.8%**
Article Context Neural Network	.06%	27.9%
Social Context Neural Network	45%	37.0%
Hybrid Social-Article Context Neural Network	15%	33.93%

Discussion

Iterative classification with logistic regression achieved the lowest test error. While it's performance was not the highest, a neural network with purely social context identified fake news articles with 63% accuracy. One of the main challenges of the project was understanding how to appropriately incorporate user-user interactions into neural networks. The training and testing were done on a mixed dataset of articles fact-checked by BuzzFeed and Politifact, creating a sparse matrix of user interactions between the sets of articles. The hybrid neural network was created by combining the input features of the social context and article context NN, but the expected lift in performance was not seen. This may be caused by redundancies in input features and the sparseness of the input matrix.

Future Work

- Improved social-context and article-context NN performance through feature engineering (more aggregated user information, more granular
- Try training exclusively on PolitiFact articles and testing on BuzzFeed, ie leverage entire network structure