SI630 Final Project Report

Version 1.0

Matt Whitehead

Abstract

In this paper, I approach the task of Fake News detection using Convolutional Neural Networks (CNNs). I test and evaluate my approach against the FakeNewsNet dataset (Shu et al., 2018). My approach falls short of the best benchmark described by Oshikawa, but it provides a "second best" approach that requires minimal feature engineering, trains quickly, and is production ready. The architecture I propose in this paper could be used for other text classification problems that have social media context data.

1 Introduction

Fake News abounds throughout the internet, both in search results and social media sites. While many people would say, "you shouldn't believe everything you read on the internet", the 2016 US presidential election demonstrated the power of fake news and social media to influence the opinions of the public. Russian bots and Facebook echo chambers should be viewed as a threat to democratically fair elections, and free and open discourse on the internet more broadly.

In this project, I propose NLP and deep learning techniques that effectively identify and flag news articles as false. I show how features extracted from text by CNNs can be fed into a feed-forward network to produce a robust classifier. The method I propose is fast to train, easily reproducible, and production-ready via the TensorFlow library. The techniques I demonstrate show that text classification problems can be effectively resolved with purely neural approaches, reducing the need for time-consuming feature engineering or domain expert knowledge.

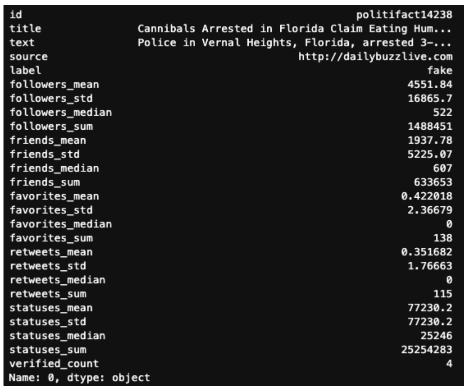
AI has often been criticized for creating complicated models that are beneficial in theory but offer no actionable information to organizations. I believe that the task of fake news detection is relevant to AI/NLP researchers and organizations alike. The CNN I propose here is a robust classifier of Fake News, and the model architecture may be deployed on news articles encountered in the wild instead of just neatly curated datasets. All the model requires to function is an article's title, body content, and a sample of data from the Twitter API for users that have shared the article.

The final model is highly accurate (over 90%) and clearly demonstrates the power of neural networks and artificial intelligence to help humans identify and help stop the proliferation of fake news. It also shows how augmenting raw text with context extracted from social media can improve the accuracy of a machine learning classifier.

2 Problem Definition and Data

Fake news can be hard for anyone to detect at times. Articles are often designed deceptively and mimic the look and style of legitimate news sources, sometimes rising to the level of blatantly infringing on a brand's trademarks. We cannot expect the average internet user to carefully example an article's content and metadata to determine if a news source is legitimate. Similarly, we cannot expect even expert human beings to manually flag articles as fake given the gigantic scale of the internet. Fortunately, this is exactly the type of problem that AI is well-positioned to solve.

In this project, I approached this task by training a neural classifier that takes a large corpus of labeled news articles and their social context from Twitter as input, and outputs a binary prediction as to whether the article is real or fake. Specifically, the output of the model is a probability measure of how confident the network is that the article is fake. This allows data scientists and developers to set a threshold for classification, depending on



```
id                                                   politifact14238
title             Cannibals Arrested in Florida Claim Eating Hum...
text              Police in Vernal Heights, Florida, arrested 3-...
source                                  http://dailybuzzlive.com
label                                                        fake
followers_mean                                            4551.84
followers_std                                             16865.7
followers_median                                              522
followers_sum                                             1488451
friends_mean                                              1937.78
friends_std                                               5225.07
friends_median                                                607
friends_sum                                                633653
favorites_mean                                           0.422018
favorites_std                                             2.36679
favorites_median                                                0
favorites_sum                                                 138
retweets_mean                                            0.351682
retweets_std                                              1.76663
retweets_median                                                 0
retweets_sum                                                  115
statuses_mean                                             77230.2
statuses_std                                              77230.2
statuses_median                                             25246
statuses_sum                                             25254283
verified_count                                                  4
Name: 0, dtype: object
```

Figure 1: An example entry in the dataset

whether they want a precision-oriented or a recall-oriented classifier. I simply used 0.5 as the decision threshold, but this is a parameter that could be explored in future work.

The dataset for this project is a subset of the FakeNewsNet (`https://github.com/KaiDMML/FakeNewsNet`) dataset (Shu et al., 2018). Due to problems with the code used to distribute the dataset, I was not able to download the whole dataset in a timely manner. However, I was able to gather 20,759 articles, 5,100 of which were examples of fake news and 15,659 were legitimate stories. From this data, I chose to look at the title of the articles and their body text. The text provided in the dataset appears to be fairly clean already, but some preprocessing is required for robust analysis. Additionally, the dataset contains Twitter API data for various users who have Tweeted out the articles. For my purposes I decided to collect the mean, median, standard deviation, and sum of: follower count, friend count, number of statuses posted, favorites count, and retweet count for each article. I also counted the number of verified users who Tweeted each article. All of this information was compiled into a simple CSV file for quick analysis. (See Figure 1.)

Several approaches to this task have been described in the literature review conducted by Oshikawa et al. Most of those approaches use a high degree of feature engineering and rather complicated models (Oshikawa et al., 2018). The state of the art model proposed by Della Vedova et al. makes use of extensive feature engineering and hard coded rules to create logistic regression and harmonic boolean label crowdsourcing models. They achieve an accuracy score of 0.938 on the Politifact data(Della Vedova et al., 2018). The state of the art score on the Buzzfeed data is achieved by Deligiannis et al. using Graph Convolutional Networks (GCNs). They also perform extensive feature engineering to represent the dataset as a network suitable for analysis and acheive an accuracy score of 0.944(Deligiannis et al., 2018).

The algorithm I propose does not produce state of the art results, however it comes close with an accuracy score of 0.9084 and avoids a lot of complications with the state of the art models. My goal was to create a practical algorithm that could be easily deployed into the real world with minimal feature engineering and a fast training time. Della Vedova's model requires extensive feature engineering and domain expertise, while Delingiannis's model requires a whole dataset to assemble a graph representation of the articles and an extensive knowledge of network analysis. My model is straightforward, quick, and easy to deploy in TensorFlow. Even without extensive machine learning or NLP knowledge, any developer should be able to deploy my algorithm.

3 Related Work

In ”Exploiting Tri-Relationship for Fake News Detection” (Shu et al., 2017), the authors compared RST, LIWC, Castillo, and TriFN models. They found the TriFN approach to be the most effective. The RST and LIWC were fairly ineffective approaches and are outperformed by even a very simple CNN, however the TriFN does seem to still be a very effective approach.

In ”Automatic Online Fake News Detection Combining Content and Social Signals” (Della Vedova et al., 2018), the authors combine article content with social media interactions to produce a classification. They used ML algorithms including logistic regression and harmonic boolean label crowdsourcing. I found their approach to be interesting, but the high degree of feature engineering used was very specific to the features of this particular dataset. Ideally, I wanted to create a model architecture that could potentially generalize a bit better. Thus, I didn't decide to reproduce this approach despite it being one of the state of the art methods.

Finally in ”Deep Learning for Geolocating Social Media Users and Detecting fake news” (Deligiannis et al.), the authors propose using a graph convolutional network (GCN) approach to the problem. In my initial research of the problem, I thought recurrent networks (RNNs) would be more useful due to their sensitivity to sequence data, however, ultimately I found a convolutional approach to be superior. While I didn't implement the type of graph the authors constructed in this paper (a directed social network graph), I was inspired to create a "graph" of my own using the Keras functional API rather than a simple sequential model. This ultimately led to my final solution for the problem.

Finally in "Are We Safe Yet? The Limitations of Distributional Features for Fake News Detection", the authors caution against relying heavily on source identification as an approach to identifying fake news (Schuster et al., 2019). While exploring the data, I initially wanted to include some source information despite the cautions given in this paper. However, the more I explored the data, the more I realized that the number of unique sources was so high that encoding source in any meaningful way would quickly result in exploding dimensionality. The authors of this paper also point out the potential of source information to bias the model toward a particular dataset. Therefore, I decided to disregard source information when crafting my algorithm.

4 Methodology

Before delving into the details of the model, it is necessary to describe the libraries used and the preprocessing steps that I took when cleaning and preparing the data. I began by downloading the dataset with the code provided by the authors on GitHub (`https://github.com/KaiDMML/FakeNewsNet`). Due to privacy concerns with the Twitter data and copyright concerns with the news data, the author's cannot distribute this dataset directly. Unfortunately, the code to download it manually is not very well optimized and after several days of downloading, I had a sample of 20,759 articles but not the entire dataset. I decided to proceed under the assumption that the subset of data that I was able to collect, closely approximates the distribution of the entire dataset. Unfortunately, having a limited subset of the data makes my results harder to verify. In future work, I would explore optimizations to the data collection code that could make the full dataset easier to download in a timely manner.

After the data collection was complete, I wrote a Jupyter notebook to extract the relevant information needed for the model and compile the dataset into an easily digestible CSV that can be quickly split into training and testing sets during the modeling process. Figure 1 shows what a single entry in the CSV contains. Since the dataset was fractured across many files, I decided to process it on my local machine rather than trying to upload over 20,000 files to the cloud. Unfortunately, due to the size of the data, special considerations had to be taken to avoid running out of RAM. I was able to complete the data processing on a 2012 Macbook Pro with 8GB of RAM after a little bit of optimization, but if one wanted to process the data with even less resources, the data would probably have to be streamed which would require re-engineering the program completely.

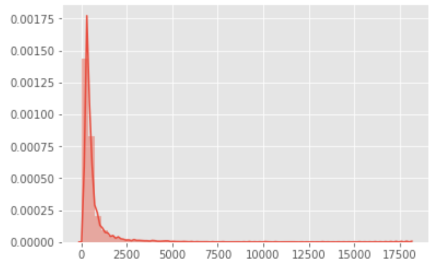


Figure 2: The distribution of article length.

After compiling the dataset, I conducted some basic analysis to allow the majority of the article texts to be encoded without creating an excessive number of dimensions. I plotted the distribution of the length of each article in the dataset and quickly determined that the data approximated to a power law distribution. (see Figure 2) I decided to compute the mean and standard deviation for the number of words in each article and round to the nearest integer. I then added two standard deviations to the mean and used this as a cutoff point to set the max article length. The cutoff point used was 2,986 words. I then calculated the proportion of articles with 2,986 words or less and determined that using this cutoff point would have no impact on 96.999% of articles. I then used the preprocessing tools provided by TensorFlow (`https://www.tensorflow.org/`) and Keras (`https://keras.io/`) to tokenize the corpus and split each article into sequences. The same procedure was performed for the article titles and a cutoff point of 19 words was set. The encoded data was then split into training and testing sets.

I then began trying different machine learning methods using Google Colab GPU-accelerated notebooks. I began by working with a subset of that data in an attempt to find the ideal model architecture before moving on to the full dataset. Eventually I settled on a ”graph-like” CNN using the Keras functional API. Rather than accepting a single input and piping it throught the model in a linear and sequential fashion, I decided to make the model accept 3 inputs at the same time. The first input was the encoded article title, the second was the encoded article body text, and the final input was the social media context data from Twitter.

The article title input takes in a 19 dimensional vector, where each element represents a word in a corpus of 22,405 unique words. The input is passed into a trainable embedding layer, such that each word becomes represented by a 50-dimensional vector. The resulting matrix is then convolved into a 256 dimensional vector using a kernel size of 3. The article body input takes in a 2,986 dimensional vector, corresponding to a corpus of 161,431 unique words. Each word in the article body is also converted into a 50 dimensional embedding. The embedded text is then convolved into a 256 dimensional vector using kernel size 10. Both of the convolved layers were passed through global max pooling layers. Finally, the output of both layers were concatenated with the corresponding Twitter data to form a single vector with 533 dimensions.

The final vector of the concatenated inputs was then passed into a 100 node fully-connected layer with a relu activation function, a 50 node fully-connected layer with a relu activation function, and finally a single node output layer with a sigmoid activation. The decision threshold was set at .5 and any articles producing an output in this layer greater than or equal to .5 were classified as fake. The final model was compiled and trained using the Adam optimizer with a learning rate of 0.0005. Binary cross entropy was used as the loss function. The final model was trained using early stopping with a patience value of 5. After 21 epochs, the patience threshold was reached and the model training was stopped. The final classification accuracy score was 90.837%, with a precision of 82.590%, a recall of 78.74%, and an F1 score of .80619. A visual representation of the model can be seen in Figure 3.

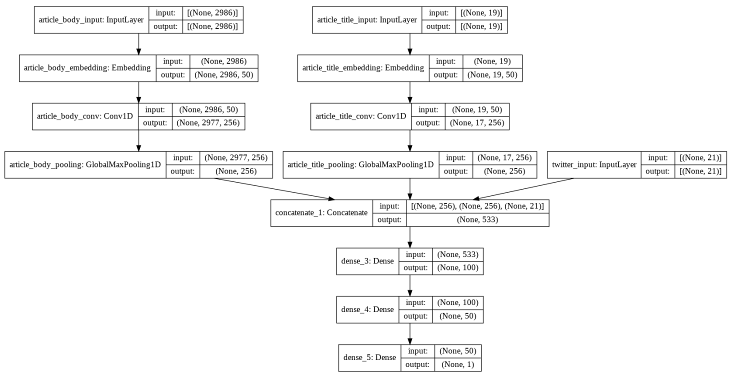
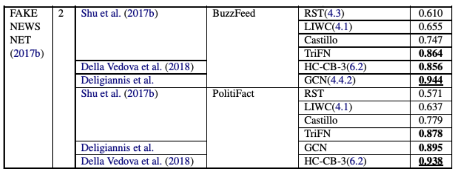

article_body_input: InputLayer | input: [(None, 2986)] | output: [(None, 2986)]
article_title_input: InputLayer | input: [(None, 19)] | output: [(None, 19)]
article_body_embedding: Embedding | input: (None, 2986) | output: (None, 2986, 50)
article_title_embedding: Embedding | input: (None, 19) | output: (None, 19, 50)
article_body_conv: Conv1D | input: (None, 2986, 50) | output: (None, 2977, 256)
article_title_conv: Conv1D | input: (None, 19, 50) | output: (None, 17, 256)
article_body_pooling: GlobalMaxPooling1D | input: (None, 2977, 256) | output: (None, 256)
article_title_pooling: GlobalMaxPooling1D | input: (None, 17, 256) | output: (None, 256)
twitter_input: InputLayer | input: [(None, 21)] | output: [(None, 21)]
concatenate_1: Concatenate | input: [(None, 256), (None, 256), (None, 21)] | output: (None, 533)
dense_3: Dense | input: (None, 533) | output: (None, 100)
dense_4: Dense | input: (None, 100) | output: (None, 50)
dense_5: Dense | input: (None, 50) | output: (None, 1)

Figure 3: The architecture of the final model.



FAKE NEWS NET (2017b)	2	Shu et al. (2017b)	BuzzFeed	RST(4.3)	0.610
				LIWC(4.1)	0.655
				Castillo	0.747
				TriFN	**0.864**
		Della Vedova et al. (2018)		HC-CB-3(6.2)	**0.856**
		Deligiannis et al.		GCN(4.4.2)	**<u>0.944</u>**
		Shu et al. (2017b)	PolitiFact	RST	0.571
				LIWC(4.1)	0.637
				Castillo	0.779
				TriFN	**0.878**
		Deligiannis et al.		GCN	**0.895**
		Della Vedova et al. (2018)		HC-CB-3(6.2)	**<u>0.938</u>**

Figure 4: Various benchmarks from the Oshikawa paper.

5 Evaluation and Results

The first and most obvious baseline that I tested against was simply picking the majority class. Since there are many more legitimate articles than fake articles in the dataset, a dummy classifier that always tries to classify articles as real provides a solid baseline. Using this classifier yielded an accuracy of 75.432%, which is significantly worse than the accuracy score of my final model (90.837%). Since the model is meant to be used to detect fake news, the precision and recall scores of this dummy classifier would both be 0.00%, indicating horrible performance on the task.

I also compared my performance with the reported accuracy scores in Oshikawa et Al’s literature review on the topic of fake news detection (See Figure 4). While my performance is not the best of the best (94.4% and 93.8%), it still out performs the next best scores of 89.5% (Politifact) and 85.6% (Buzzfeed) on this dataset. An accuracy score of 90.837% seems to fall right in line with methods that other researchers have tried. However, it is important to note that the Oshikawa paper only reports accuracy scores and does not provide other metrics such as precision, recall, or F1, which would give more insight into the performance of the baseline models.

While not state of the art, I believe that the performance of my model is acceptable because of its relative simplicity and ease of replication. The approaches proposed by many of the researchers in the Oshikawa paper are highly domain specific and require a lot of precise tuning to the dataset of interest. Additionally, many of the methods used are unusual and not likely to be found in any popular machine learning (or deep learning) libraries. My methodology is straightforward, easy to replicate on nearly any dataset, and doesn't require a lot of custom coding to implement. My model should be able to be implemented using nearly any machine learning library that supports the use of convolutional neural networks with multiple input channels.

I also compared the model performance against a similar neural network that used only the processed text data and did not augment it with any additional social context from the Twitter data. This model achieved an accuracy score of 86.25%. Due to time constraints I was not able to investigate this too deeply, but I believe that this result demonstrates that using social media context in addition to raw text data makes for a more robust classifier than a classifier built using the text data only. The performance increase is modest, indicating that most of the predictive power comes from the text itself, but not insignificant.

Due to the high dimensionality of the data it's hard to show an example of misclassification that fits neatly into an image, but it does seem that there are a few trends in the articles that the model does not classify correctly. Namely, the model appears to lack the nuance to understand ideas like sarcasm or parody. An article that might be obviously satirical to a human reader may be classified as true by the model. Likewise, an article that is true but written in a sarcastic or critical tone may be classified as false by the model.

6 Discussion

I believe that the results of this project demonstrate a promising opportunity for AI to be deployed as a tool to help humans better detect fake news, however I do not believe that the tool is robust enough at this stage to operate completely autonomously. As with many AI tools, I tend to believe that a fake news detector should be used to augment human abilities rather than supplant them.

I think that my results also demonstrate the power of deep learning to solve NLP tasks. In some ways using deep learning almost feels like "cheating". Traditional ML and NLP methods have required a lot of careful thought and clever approaches to obtain acceptable performance on a particular classification. With deep learning approaches, even someone relatively new to NLP (like me) can create an algorithm using an existing machine learning library, fiddle with the various knobs and switches (hyperparameters), and come up with a decent result. However, it is my opinion that we should see such developments as an asset, rather than a hindrance to or cheapening of the NLP field. Rather than banging our heads against the wall, consulting domain experts, and desperately trying to think of novel mathematical formulations to a given task, we can use deep learning to achieve practical results right away for many tasks.

I don't mean to suggest that deep learning is some type of "magic bullet" or should always be considered desirable. There are plenty of situations where a deep learning approach may be difficult or even impossible. For instance, if one is working with a limited dataset or data that cannot be easily converted into a suitable format for neural networks. There may be other situations where a deep learning approach is practical but undesirable. Consider the case of an AI lie detector that is used to analyze the testimony of crime suspects. If an AI program labels a person as a liar, "this person is a liar because a linear combination of weights and biases discovered through a long and complex training process" is probably not an acceptable explanation. (As an aside, I am not advocating for the use of an AI lie detector in such circumstances nor do I wish to dive into the inherent ethical issues raised by such a proposal.) However, broadly speaking, I think the proven utility of deep learning methods on NLP tasks should be seen as a positive development in the field overall.

I think it is also important to consider the use case of the typical NLP practitioner versus an academic researcher. Most individuals implementing NLP at a commercial scale are interested in solving a particular business problem, such as document classification or customer segmentation. For such tasks, the explanation of why a particular method works well is much less important than the actual performance of the method on the task. In other words, business users tend to be results oriented and do not care for the gory details of why a particular method works, so long as they are able to deliver an acceptable level of performance. Researchers, on the other hand, have a different goal entirely. A paper stating ”model x performed well on task y” without any further explanation or justification would be considered completely useless. The primary goal of an academic researcher is to make discoveries that advance their field of interest. If the researcher isn't able to explain their results, they are unlikely to convince their peers of the merit of their achievement. Given the juxtaposition of these goals, it's easy to see why deep learning may be appropriate in many business and practical use cases, while it would not be appropriate in similar academic and/or research use cases.

7 Conclusion

In this project, my goal was to use neural network methods to create a tool to detect and classify fake news articles using social media context data from Twitter. While my method did not produce the highest accuracy ever recorded on this dataset, I believe that the results are still significant. My method shows that convolutional neural networks can be leveraged to accurately classify fake news, without requiring extensive preprocessing and feature engineering. I also demonstrated that social media context data, specifically from Twitter, can be utilized to make a neural network classifier more robust on the task of fake news detection by comparing a baseline network that used only the raw text data, with the final network I proposed in this project.

8 Other Things I Tried

In my early experiments, I tried using recurrent neural networks (RNNs) to approach this task. My intuition was that identifying a news article as real or fake was highly order dependent and that due to their high levels of performance on sequence data, RNNs were the logical choice. However, I quickly found that this was not the case and that a CNN model was the superior choice. This may suggest that the choice of words and the particular combinations of words used in an article are more predictive of the authenticity of the article than the ordering of the words. Additionally, most of the deep learning literature on text classification makes use of CNNs.

I also attempted to implement the ”Pre-training of Deep Bidirectional Transformers for Language Understanding” or BERT approach described by Devlin et al, as it is said to be the state of the art in text classification as of the time of writing this report (Devlin et al., 2018). However, due to limitations on the dimensionality of the data, I was unable to implement this method without severely truncating the text of the articles in my dataset. In my naive understanding, it seems that BERT is intended for the classification of shorter snippits of text rather than the typical length of articles found in my dataset. It should also be noted that BERT takes significantly longer to train than the model proposed in this paper. This may or may not be important depending on the time and computational constraints of the end user.

9 What I Would Have Done Differently or Next

In an ideal world with no time constraints, there are a lot of things I would have done differently. First and foremost, I would have made sure to download the entire FakeNewsNet as described by Shu et al. and distributed on their GitHub repository. While I don't think this situation occurred, there is the possibility that my results are due to me getting a ”lucky” subset of the data that responded particularly well to the approaches I tried. I would have also liked to try a little bit more experimentation with the feature selection. In particular, I would have liked to explore different combinations of the social media features to try to identify which are actually predictive on this task, and which are simply noise. I would have also liked to experiment with different methods of generating word embeddings, including trying different numbers of dimensions, and using pretraied word2vec, GLOVE, or FastText vectors. Finally, I would have liked to train a BERT model (Devlin et al., 2018) on this dataset, but the complexity of implementing the model proved to be overly difficult given the time and computational constraints I was working with.

10 Work Plan

In my original proposal, I focused on using Recurrent Neural Networks to approach the task. I wanted to find a novel algorithm that would outperform (or at least come close to) the benchmark accuracy scores on this dataset from the Oshikawa paper. However, as I progressed further with the project, I realized that RNNs were not the ideal way to approach this task (or at least I was unable to find a way to create a useful RNN model) and

I pivoted toward the more conventional CNN approaches described in the literature for text classification. I also realized pretty quickly that I would not be able to outperform the benchmarks on this dataset. Thus, my goal shifted from outperforming the current standards toward producing a model that is easy to understand, quickly reproducible with modern machine learning libraries, and still provided a robust classification without the need for excessive feature engineering.

Acknowledgments

I would like to acknowledge Professor David Jurgens for his help and support throughout this process and in SI 630. I certainly would not have had the skills to complete this project without his guidence. I would also like to acknowledge the authors and open source contributors to TensorFlow (`https://www.tensorflow.org/`) and Keras (`https://keras.io/`) who's software enabled the rapid prototyping and development required for this project.

References

Nikos Deligiannis, Tien Do Huu, Duc Minh Nguyen, and Xiao Luo. 2018. Deep learning for geolocating social media users and detecting fake news.

Nioks Deligiannis, Tien Huu Do, Duc Minh Nguyen, and Xiao Luo. ???? Deep learning for geolocating socialmedia users and detecting fake news .

Marco L Della Vedova, Eugenio Tacchini, Stefano Moret, Gabriele Ballarin, Massimo DiPierro, and Luca de Alfaro. 2018. Automatic online fake news detection combining content and social signals. In *2018 22nd Conference of Open Innovations Association (FRUCT)*. IEEE, pages 272–279.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A survey on natural language processing for fake news detection. *CoRR* abs/1811.00770. `http://arxiv.org/abs/1811.00770`.

Tal Schuster, Roei Schuster, Darsh J Shah, and Regina Barzilay. 2019. Are we safe yet? the limitations of distributional features for fake news detection. *arXiv preprint arXiv:1908.09805* .

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2018. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *CoRR* abs/1809.01286. `http://arxiv.org/abs/1809.01286`.

Kai Shu, Suhang Wang, and Huan Liu. 2017. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709* .