Transactions on Machine Intelligence 3(3) (2020) 149-158

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Model for Detecting Fake News on Twitter

M. Narangi Fard[1], Z. Heshmati[2,*]

[1] Department of Network Science and Technology, School of Advanced Science and Technology, University of Tehran, Iran
[2] Assistant Professor, Department of Network Science and Technology, School of Advanced Science and Technology, University of Tehran, Iran

ARTICLE INFO

Article History:
Received 4 May 2020
Received in revised form 23 June 2020
Accepted 5 September 2020
Available online 5 September 2020

Keywords:
Fake News, Social Networks, Machine Learning, Propagation Tree, News Dissemination

ABSTRACT

Due to the widespread use of social networks by people of all ages, distinguishing between fake and real news on these platforms has become a significant challenge in today's world. Individuals who disseminate fake news on social networks often aim to achieve various commercial, political, and economic goals. Therefore, identifying and distinguishing real news from fake news is crucial in addressing this issue. The objective of this research is to present an intelligent model for detecting fake news using a news propagation tree in the social network Twitter. The dataset used in this study is sourced from the political news section of the Fake News Net website. Initially, a news propagation tree was constructed for both real and fake news using this dataset, followed by the development of features from structural, temporal, and syntactic perspectives based on the news propagation tree. Finally, machine learning algorithms were employed to build a model for predicting fake and real news. The results indicated that among the algorithms used for modeling, the Random Forest algorithm, with an accuracy of 75.8%, was the best model for distinguishing fake news from real news.

1. INTRODUCTION

Given that a significant portion of human life is now devoted to communication through social networks, most individuals prefer to follow news via these platforms rather than traditional media outlets. Following news on social networks offers distinct advantages and disadvantages compared to traditional methods.

Advantages of using social networks for news compared to traditional methods include:

- Faster and more accessible news
- Lower cost of news dissemination
- Ease of news sharing
- Ease of sharing opinions and engaging in discussions

* Corresponding Author: zheshmati@ut.ac.ir
Department of Network Science and Technology, School of Advanced Science and Technology, University of Tehran, Iran

http://dx.doi.org/10.47176/TMI.2020.149

© 2020 by the authors. Licensee T.M.I, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).

- Ability to search for news

Disadvantages of using social networks compared to traditional methods include:

- Lower accuracy of news

- Rapid dissemination of fake news on these platforms

Fake news consists of information deliberately fabricated with the intention of achieving political, economic, or other benefits. The spread of such news on social networks often leads to unethical exploitation in various regions worldwide. Therefore, employing methods to detect fake news on these platforms can help prevent the spread of false information and aid communities in distinguishing between fake and real news.

Thus, the goals of this research are to implement a model using machine learning algorithms to detect and classify fake news from real news based on features derived from news propagation on the Twitter social network. To achieve these objectives, the study first constructs a propagation tree for tweets, retweets, and replies related to news items from the FakeNewsNet dataset, specifically focusing on the Politifact news category. Subsequently, features are extracted for each news item based on the propagation trees. Finally, these features are used to develop a model for predicting and identifying real and fake news.

2. LITERATURE REVIEW

In 2019 [1], efforts were made to identify the relationships between user characteristics and the dissemination of fake news on social networks. The primary objectives were to determine which user profiles are more prone to disseminating fake news and to identify the user characteristics that can be utilized to detect fake news.

User characteristics in this study were categorized into two types: implicit and explicit. Implicit characteristics included factors such as age, behavioral traits, geographical location, profile pictures, and political preferences, which were indirectly related. Explicit characteristics comprised information such as user authenticity and the age of the user account. The study's findings indicated that the age of the user account had the most significant impact on detecting fake news, as accounts with a shorter registration period are less credible. Following this, the factors of account authenticity, political preferences, behavioral traits, and finally, the average level of user activity on the social network were identified as having the most substantial influence on distinguishing fake news based on the characteristics of individuals involved in news creation and dissemination.

After identifying these characteristics, the following results were obtained using various machine learning methods:

Table 1. Evaluation of Results

Dataset	Model	Acc	Prec	Recall	F1
PolitiFact	RF	**0.909**	**0.948**	**0.864**	**0.904**
	SVM	0.869	0.884	0.847	0.865
	DT	0.848	0.849	0.844	0.847
	LR	0.848	0.867	0.819	0.842
GossipCop	RF	**0.966**	**0.956**	**0.976**	**0.966**
	SVM	0.918	0.920	0.916	0.918
	DT	0.931	0.931	0.930	0.931
	LR	0.914	0.918	0.909	0.914

In [2], attempts were made to detect fake news by identifying features based on the dissemination of fake news on social networks. In this study, a propagation tree was created for each news item published on the network. After constructing the propagation tree for each news item, three categories of analysis—structural, temporal, and syntactic—were considered. Features were extracted based on these analyses, and subsequently, machine learning algorithms were employed to develop a model for predicting fake and real news.

Table 2. Evaluation of Results

Datasets	Type	Acc	Prec	Rec	F1
PolitiFact	Structural	0.681	0.681	0.672	0.676
	Temporal	0.793	0.716	**0.963**	0.821
	Linguistic	0.659	0.648	0.683	0.665
	All	**0.843**	**0.835**	0.851	**0.843**
GossipCop	Structural	0.826	0.828	0.823	0.826
	Temporal	0.826	0.827	0.825	0.826
	Linguistic	0.578	0.594	0.491	0.538
	All	**0.861**	**0.854**	**0.869**	**0.862**

3. DATASET DESCRIPTION

This study utilizes the FakeNewsNet dataset, which comprises two categories of news: Politifact and Gossipcop. Politifact news represents non-sensational news, while Gossipcop news pertains to sensational news on the Twitter network. Each category includes both fake and real news. For this research, the dataset specifically includes fake and real news from Politifact.

The features of the Politifact dataset are outlined in Table 3. In this study, we randomly select 362 real news items and 362 fake news items.

Table 3. Features of the Politifact Dataset

Attributes	Count
Real News	624
Fake News	432
Number of Users	384813
Number of Tweets	275058
Number of Retweets	293438
Number of Replies	125654

Each Politifact news item is associated with JSON files containing relevant information about the news. The details of these files are as follows:

- news article: Contains metadata about the news, including the text of the news, links to images, publication time, and other related details.

- Tweets: Includes a list of tweets related to the news published by users. Each tweet contains information such as the tweet text, the user ID who posted the tweet, the tweet ID, and other relevant details.

- Retweets: Consists of a list of retweets made on the tweets related to the news.

- Replies: Contains a list of replies to the tweets or retweets, made by users.

4. RESEARCH METHODOLOGY

The research methodology involves three main phases. In the first phase, we construct a propagation tree for each real and fake news item. In the second phase, we extract features from this propagation tree. In the third phase, we use these extracted features and apply machine learning algorithms to develop a model for predicting fake and real news.

4.1. Propagation Tree Construction Phase

As described in the dataset introduction, each real or fake news item includes information such as tweets related to the news, retweets on these tweets, and replies to the tweets. The structure of the propagation tree is as follows: it consists of a directed graph for each news item, starting with a root node with ID 1 and a creation time attribute corresponding to the news publication time. If the news publication time is missing in the dataset, we use the time of the first tweet related to the news as the publication time.

Next, all tweets related to the news are connected to the root node with ID 2 and their respective tweet publication times. It is important to note that the propagation tree is a directed graph, so edges are directed from the root node to the tweet nodes.

Subsequently, we address the retweets of each tweet. For each retweet, we create a node with ID 3 and the retweet's publication time, and establish a directed edge from the original tweet to its retweets.

Finally, we handle the replies to these news items. Each tweet may have a set of replies, and each reply can also have additional replies. Moreover, these replies may be retweeted and may have further replies. Therefore, we first create nodes with ID 4 and publication times for the replies to each tweet, and establish directed edges from the tweets to their replies. For replies to replies and retweets of replies, we recursively create nodes with ID 4 and publication times, and establish directed edges accordingly.

The structure of the propagation tree for each real news item is illustrated in Figure 1, and for each fake news item, it is depicted in Figure 2. As shown, the yellow node in the figures represents the root node or the original news item, while the red nodes indicate tweets published at various times about the news. The green nodes represent retweets associated with the news, and the blue nodes denote replies generated in response to the news.

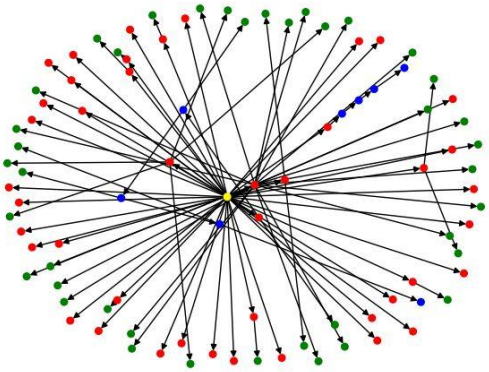

Fig.1. Propagation Tree for a Real Politifact News Item

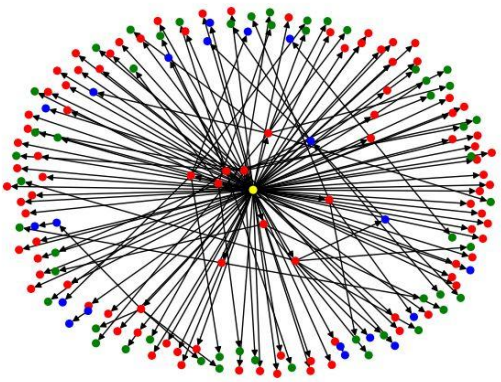

Fig.2. Propagation Tree for a Fake Politifact News Item

4.2. Feature Extraction Phase for News Propagation Trees

The news propagation tree network can be analyzed hierarchically at two levels: the micro-level and the macro-level. At the micro-level, we assess the reply network, which contains local information. At the macro-level, we analyze information disseminated through retweets, providing a global perspective.

By examining the propagation tree from both micro-level and macro-level perspectives, we can extract features that ultimately aid in constructing a model to predict whether news is real or fake.

At the micro-level, features are evaluated from structural and temporal aspects, while at the macro-level, features are assessed from structural, temporal, and syntactic perspectives.

Table 4 lists the features extracted from each news item based on its propagation tree. Features A1, A2, A3, and A4 pertain to structural aspects; features A5, A6, A7, A8, and A9 relate to temporal aspects; and features A10 and A11 are associated with the syntactic aspects of the propagation tree.

The extraction of the features listed in Table 4 is performed using Python programming. Initially, we extract the desired features by invoking the functions detailed in Table 5. Table 6 provides a description of the implementation of the feature extraction functions or methods used, including an overview of their algorithmic structures.

Table 4. Description of Features Constructed Based on the News Propagation Tree Network

Features	Feature Descriptions
A1	The number of tweets in the distribution tree shows any news that has no retweets and no replies
A2	Maximum Yal shows the output of tweets of each news
A3	It represents the number of retweets and tweet replies with the maximum number of output edges of each news item.
A4	Calculating the depth of the propagation tree
A5	It shows the time interval between the first tweet and the last retweet of each news item
A6	Indicates the time interval between two consecutive tweets
A7	Expressing the time interval between two consecutive retweets
A8	Expressing the time interval between the first tweet and the tweet with the maximum output edge
A9	It shows the average time interval between the first tweet of each subtree and the first retweet of that tweet.
A10	It represents the average sentiment of the replies that are directly connected to the tweets
A11	It expresses the feeling of replays in the deepest chain of replays
A12	The class label for fake or real news, which is equal to one if the news is real and zero if the news is not real

Table 5. Functions Implemented for Constructing Features of News Propagation Trees

Features	Feature Functions
A1	findTws(G)
A2	MaxOutDegree(dict)
A3	NumEngageTwMOut(dict,maxOut, G)
A4	totalHeight(G)
A5	ComputeFtlr(G)
A6	ComputeRFrq(G, 2)
A7	ComputeRFrq(G, 3)
A8	ComputeDifFtMt(dict, maxOut, G, baseTime)
A9	ComputeDifTfrCas(G)
A10	f_reply(G)
A11	d_replies(G)
A12	Class label for fake or real news

Table 6. Description of Methods Used for Implementing Feature Extraction Functions|

Description of the implementation of functions related to the construction of features
calBaseTime(times): The input of this function is all the creation times of the tweets, and the output of this function is such that it sorts the list of times and returns the creation time of the first tweet.
findTws(G) : To implement this method, we go to all the nodes that are tweets, and if they don't have a reply or retweet, we increase our counter by one and at the end we return our counter.
ComputeFtlr(G): To implement this method, we first enter the creation time of all tweet nodes in one array and the creation time of all retweet nodes in another array. In the first step, we sort the tweet array and we sort the retweet array in reverse so that the longest retweet release time is placed first, and in the next step, the time interval between the first tweet and the last retweet is calculated with the help of the difference of the first values of these two arrays.
NumOutDegree(G): The output of this method is a dictionary that specifies the number of output edges for all tweets of each news item. The key of this dictionary is the unique identifier of the nodes of the propagation tree and its value is the maximum number of output edges. To implement this method for each node in the broadcast tree that is a tweet. We enter the number of output nodes of this edge into a list, and by calculating the length of this array, the number of output edges of each tweet is determined.
MaxOutDegree(dict): The output of this method is the maximal edge value of the output of each tweet. The implementation of this method is that the values of the dictionary that checks all the output degrees of the tweets of each news and Maximism returns the value as the output of the function.
NumEngageTwMOut(dict,max,G): The first variable of this method is a dictionary of the number of edges output from each tweet. The second variable is the maximum number of output edges from all news tweets. To implement this method, we first calculate the unique ID of the tweet node that has the maximum output edge. Next, we calculate the subtree of that tweet node, and by calculating the length of this subtree, we calculate the number of replies and retweets of this tweet node with the highest output edge.
totalHeight(G): For each tweet node in the propagation tree, it calculates the shortest path from this node to another node, and in the next step, it stores the shortest path for each tweet maximum. In the next step, we calculate the maximum shortest path of all tweets and add one to calculate the root node and return this value as the output of the method.
ComputeRFrq(G,id): If this method is called with an ID value of 2, it calculates the time interval between two consecutive tweets, and if it is called with an ID value of 3, it calculates the time interval between two consecutive retweets. To implement this method, based on the input ID, which is either two or three, we store all the tweets or retweets of the distribution tree in a dictionary and count the number of these tweets or retweets. In the next step, we sort this dictionary once from descending to ascending and once from ascending to descending in order to obtain the earliest and latest publication time of two tweets or retweets. In the next step, these two times are subtracted from each other and divided by the total number of tweets or retweets, thus the time between two consecutive tweets or retweets is calculated.
ComputeDifFtMt(dic,max,G,baseTime): The first variable of this method is a dictionary of the number of output edges from each tweet node. The second variable is the maximum number of output edges from all news tweets. The fourth variable is the time of the first tweet published on the news. In this method, we calculate the unique ID of the tweet node that has the largest number of output edges and obtain the node during the construction time. In the next step, we calculate the time interval between basetime and this time and return it as the output of the function.
ComputeDifTfrCas(G): For all the tweet nodes, we calculate the creation time of that node. In the next step, we get the subtree of that node, and we get the time of the first retweet of each tweet, and we store the time difference between the two in an array, and at the end, the average All these time intervals are the output of this method.
sentiment_analyzer_scores(sentence): This method gives the input text a score between -1 and 1 based on the feelings it takes from it.
f_reply(G): To implement this method, we first add all replays to an array. In the next step, for each tweet node, we increase the value of the counter variable by one and check all the nodes connected

to that tweet. In this way, let's calculate all the replies that are directly connected to the tweet node. In the next step, for all the replies connected to the tweet node, we calculate the total points related to the sentiments of the reply texts, and at the end, we divide the total sentiment points of the replies by the number of tweets and consider it as the output of the method. we take.
d_replies(G): To implement this method, first, for all tweets, we calculate the length of the shortest path from the tweet node to the rest of its nodes and store these values in an array. In the next step, we calculate the longest and shortest path for each tweet and value it in a presentation. In the next step, we calculate the maxheight, which actually means the longest shortest path among all tweets in the distribution tree. In the next step, we calculate the ID of the tweet that has the longest and shortest path among all tweet nodes and store the subtree of that node in a temporary graph. At the end, for all the replays of that subtree, we calculate the total score related to the sentiments of the replays' texts and consider it as the output of the function.

To obtain all features of the constructed propagation trees in phase 4-1, the methods listed in Table 5 are called to store the features of each propagation tree representing a news item's dissemination on Twitter. Finally, after invoking all these methods, the final data for each tree is saved as a row in a CSV file.

The features for all propagation trees are compiled, whether the information pertains to real or fake news. This prepares the dataset for the subsequent modeling phase. As shown in Figure 3, the correlation of the constructed features with the class label indicates that feature A5 has the highest correlation.

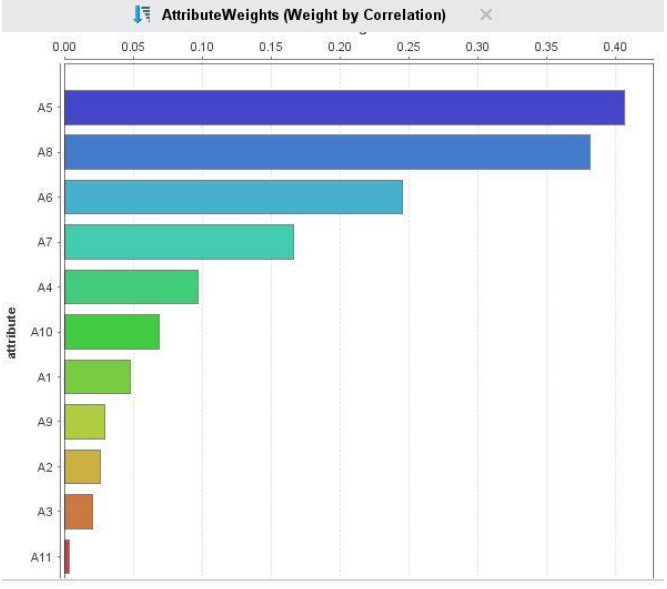

Fig.3. Correlation of Constructed Features with Class Label

4.3. Modelling Phase

In this phase, we use the features extracted from both real and fake news propagation trees to develop models employing machine learning algorithms for the identification and prediction of fake versus real news on Twitter.

4.3.1. Data Processing Steps

The features constructed for fake news are stored in one file, and those for real news are stored in another file. These two datasets were merged, and a column titled "Real or Fake" was added, with a value of one indicating the class label for the `feature_real` file and a value of zero indicating the class label for the `feature_fake` file. The final merged data was then shuffled and normalized before proceeding to the modeling phase.

4.3.2. Algorithms Used in the Modelling Phase

- Decision Tree

The decision tree is a conventional and popular method for classification. It is widely used across various fields. In machine learning, a decision tree serves as a predictive model and is one of the most commonly used machine learning methods. In a decision tree, the leaves represent classifications, and the branches represent the different features used to reach a specific class of data. A decision tree can also be represented using a set of conditions or rules [3, 4].

- Random Forest Algorithm

The Random Forest algorithm is a classification method and is regarded as one of the most powerful and popular techniques for high-dimensional and complex problems. This algorithm consists of a group of decision trees where each tree casts a vote for each sample. The class with the majority vote is selected as the classification of the data [5].

- XGBoost Algorithm

The XGBoost algorithm is a type of gradient boosting algorithm known for its high performance in classification, regression, and ranking tasks. It is favored for its precise predictions, high speed, and support for parallel and distributed execution [6].

- SVM Algorithm

Support Vector Machines (SVM) are a class of supervised machine learning algorithms used for classification and regression based on statistical learning theory [7,8]. An SVM maps each data point to a new space according to its class so that the data becomes linearly separable (hyperplane). It then finds the hyperplane that maximizes the margin between classes, thus providing the best separation [9,10].

5. RESULTS

Initially, the feature data were partitioned, with 30% allocated for testing and 70% for training and validation. Subsequently, the methods corresponding to the used algorithms were implemented, and the model was created using the training and validation data. Parameters were set accordingly, and the model was evaluated using the test data to measure its accuracy, as detailed in Table 7.

Table 7. Results for Modeling Real and Fake News

Algorithm	Accuracy	Precision	Recall	F-score
Random Forest	**75.8**	**75.8**	**75.7**	**75.7**
XGBoost	74.4	74.4	74.3	74.3
Decision Tree	73.5	73.5	73.6	73.5
SVM	67.1	72.2	68.0	65.8

As observed, the machine learning algorithms provided a model for predicting and identifying fake versus real news. The results indicate that among the machine learning algorithms used for modeling, the Random Forest algorithm achieved the highest accuracy of 75.8% in distinguishing between fake and real news.

6. CONCLUSION

This study explores the critical challenge of distinguishing fake news from real news on social media platforms, particularly Twitter, where misinformation can spread rapidly and influence public opinion, decision-making, and even societal stability. The ability to accurately classify fake news is essential in preventing its misuse for political, economic, or social manipulation.

The proposed approach consists of three key stages. First, propagation trees are constructed for various news items to represent the way information spreads across the platform. Second, temporal, structural, and semantic features are extracted from these propagation trees to analyze differences between real and fake news dissemination patterns. Finally, machine learning models are developed and evaluated to classify news items as either fake or real.

Extensive experiments were conducted using multiple machine learning algorithms to determine the most effective model for fake news detection. Among the tested classifiers, the Random Forest algorithm achieved the highest accuracy, with a performance rate of 75.8%, demonstrating its effectiveness in distinguishing fake news based on propagation characteristics. These findings highlight the potential of propagation-based features in enhancing the reliability of automated fake news detection systems, contributing to the broader effort of combating misinformation on social media.

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Shu, K., Zhou, X., Wang, S., Zafarani, R., & Liu, H. (2019). The role of user profiles for fake news detection. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 436–439). https://doi.org/10.1145/3341161.3342927

[2] Shu, K., Mahudeswaran, D., Wang, S., & Liu, H. (2020). Hierarchical propagation networks for fake news detection: Investigation and exploitation. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 14, pp. 626–637). https://doi.org/10.1609/icwsm.v14i1.7329

[3] Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

[4] Mahjoubi, J., & Etemad-Shahidi, A. (2007). Estimation of wind-induced wave heights in Neka using regression decision trees. *Ocean Engineering*, 35(5-6), 539-548.

[5] Kumar, S., & Sahoo, G. (2017). A random forest classifier based on genetic algorithm for cardiovascular diseases diagnosis. *International Journal of Engineering Transactions B: Applications*, 30(11), 1723–1729. https://doi.org/10.5829/ije.2017.30.11b.13

[6] Mitchell, R., & Frank, E. (2017). Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, 3, e127. https://doi.org/10.7717/peerj-cs.127

[7] Belaid, S., & Mellit, A. (2016). Prediction of daily and mean monthly global solar radiation using support vector machine in an arid climate. *Energy Conversion and Management*, 118, 105–118. https://doi.org/10.1016/j.enconman.2016.03.082

[8] Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 1–27. https://doi.org/10.1145/1961189.1961199

[9] Wang, H., Wang, Y., Zhou, Z., Ji, X., Li, Z., Gong, D., ... & Liu, W. (2018). CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern* Recognition (pp. 5265–5274). https://doi.org/10.1109/CVPR.2018.00552

[10] Yadgari, V., & Matinfar, A. R. (2019). Detection of web denial-of-service attacks using entropy and support vector machine algorithm. *Electronic and Cyber Defense*, 4(4), 79–89.