IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 August 2023, accepted 22 October 2023, date of publication 8 November 2023, date of current version 14 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3331220

RESEARCH ARTICLE

MCWDST: A Minimum-Cost Weighted Directed Spanning Tree Algorithm for Real-Time Fake News Mitigation in Social Media

CIPRIAN-OCTAVIAN TRUICĂ[1], ELENA-SIMONA APOSTOL[1], RADU-CĂTĂLIN NICOLESCU[1], AND PANAGIOTIS KARRAS[2]

[1] Computer Science and Engineering Department, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, 060042 Bucharest, Romania
[2] Department of Computer Science, University of Copenhagen, 2100 Copenhagen, Denmark

Corresponding author: Elena-Simona Apostol (elena.apostol@upb.ro)

This work was supported by the National University of Science and Technology Politehnica Bucharest through the PubArt Program.

ABSTRACT The widespread availability of internet access and handheld devices confers to social media a power similar to the one newspapers used to have. People seek affordable information on social media and can reach it within seconds. Yet this convenience comes with dangers; any user may freely post whatever they please and the content can stay online for a long period, regardless of its truthfulness. A need arises to detect untruthful information, also known as *fake news*. In this paper, we present an end-to-end solution that accurately detects fake news and immunizes network nodes that spread them in real-time. To detect fake news, we propose two new stack deep learning architectures that utilize convolutional and bidirectional LSTM layers. To mitigate the spread of fake news, we propose a real-time network-aware strategy that (1) constructs a minimum-cost weighted directed spanning tree for a detected node, and (2) immunizes nodes in that tree by scoring their harmfulness using a novel ranking function. We demonstrate the effectiveness of our solution on five real-world datasets.

INDEX TERMS Fake news detection, fake news propagation, network-aware fake news mitigation, real-time network immunization.

I. INTRODUCTION

With the accelerated technology adoption by a growing number of users, social media have become the main medium for the dissemination of information on current news and events [1]. While these new media bring several benefits (e.g., a large number of consumers reached, instant and continuous updates on one's topics of interest), they also enable the spread of harmful information in the form of fake news, and may thus polarize public discourse regarding critical topics (e.g., elections [2], vaccination [3], health hazards [4]) and threaten democratic values [5]. Because of its detrimental effects on society at large [6], [7], [8], the *fake news* phenomenon has been studied by scientists and practitioners alike; fake news is defined as news articles that intentionally contain verifiably false misleading information inconsistent with factual reality [9], [10], [11], [12], [13], [14], [15]. To mitigate the threat of fake news, journalists have started to manually classify news and offer websites with fact-checking mechanisms that provide a verdict regarding its veracity, such as PolitiFact and Snopes. However, such solutions may fail in high-velocity information spreading social media, as news appears and spreads much faster than any manual verification; by the time it is checked, the news may have been already shared with many sources and its negative effect may have taken hold. In this paper, we propose new models and strategies for misinformation detection and mitigation to address the current real-world challenges posed by fake news.

In particular, we aim to:

(O_1) Propose new deep learning architectures to accurately detect fake news in social media; and

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Hao Chen.

(O_2) Propose new real-time strategies to mitigate the spread of detected fake news in a social network.

To reach these objectives, we answer the following questions:

(Q_1) Can we improve the accuracy of fake news detection using new deep learning architectures?

(Q_2) Can we immunize nodes that spread harmful content using network information in real-time?

To answer (Q_1) and achieve objective (O_1), we propose two novel stack deep learning architectures that utilize convolutional and bidirectional LSTM layers. We show the effectiveness of these architectures with ample benchmarking on four real-world datasets. To answer (Q_2) and reach objective (O_2), we propose a real-time algorithm for network immunization, which builds a minimum-cost weighted directed spanning tree for a detected source node r and chooses nodes in that tree to immunize by scoring their potential harmfulness using a ranking function. The proposed ranking functions consider the following network information for each node:

1. How long a chain of followers is (i.e., the length of the diffusion path),
2. How many nodes it reaches (i.e., the spread of information), and
3. How fast it spreads information (i.e., the information diffusion speed).

To show the effectiveness of network immunization, we evaluate our method on one real-world Twitter dataset. In summary, the main contributions of this paper are:

1. New deep learning architectures for fake news detection;
2. New real-time fake news mitigation strategy consisting of an algorithm for building the minimum-cost weighted directed spanning tree for a given node and a network-aware node harmfulness ranking function;
3. Benchmarking on multiple real-world datasets to evaluate the efficiency of our deep learning architectures for fake news detection;
4. Evaluation of our real-time mitigation algorithm on a real-world Twitter dataset.

The rest of this paper is structured as follows: In Section II, we present the state of the art. In Section III, we present the proposed architecture, deep learning models for fake news detection, minimum-cost weighted directed spanning tree algorithm for a given source node and harmful node ranking function. In Section IV, we detail the implementation of our architecture. In Section V, we describe the datasets and analyze in detail the experimental validation of our solution. Section VI provides an in-depth discussion of results, and Section VII concludes the paper and hints at future research.

II. RELATED WORK

In this section, we present the approaches of previous research on the two tasks of interest to our work: detecting fake news and mitigating its spread.

A. FAKE NEWS DETECTION

Nakamura et al. [16] introduced a dataset comprising Reddit posts along with an architecture designed to identify posts that contain fake pieces of information using image data, title data, or both combined. The advantages of working with such a dataset are its diversity, large size, and multidimensionality, which allows researchers to treat fake news detection as a 2-way, 3-way, or 6-way classification problem. Further, the authors combine text and image features to classify whether a post spreads fake pieces of information or not. Text is embedded using the BERT [17] and InferSent [18] models, while image data is extracted using the VGG16 [19], ResNet50 [20], and EfficientNet [21] models.

Kumar et al. [22] tackle the fake news detection challenge via sentiment analysis, implementing seven deep learning architectures, such as long short-term memory (LSTM), bidirectional LSTM, convolutional neural networks (CNN), and ensemble models that combine the aforementioned. After performing fake news classification using these models, the authors evaluate the models' performance against classic machine learning algorithms, such as logistic regression and support vector machines.

Khan et al. [23] address the fake news detection task via classical machine learning algorithms and more advanced deep-learning and neural network models, and compare their performance on two well-known datasets: Liar and Fake or Real News, along with a self-built dataset, which the authors claim to be more variate and denser than the other two.

Granik and Mesyura [24] tackle the fake news detection problem by using the Naive Bayes classification algorithm, which, despite its simplicity, yields good results on the task. Moreover, the authors suggest further adjustments that can be made to improve the results of the aforementioned method.

Pérez-Rosas et al. [25] collect two original fake news detection datasets and use a linear kernel SVM algorithm to perform classification. The first dataset is built by collecting real news from reliable US news websites and adding pieces of fake information to them to turn them into fake news. For the second dataset, the authors target celebrity news and search the web for pairs of fake and real articles regarding celebrity gossip. The classification model takes as input a set of predefined linguistic features or a mix of them, such as unigrams or bigrams, punctuation marks, psycholinguistic features, readability, and syntax. The authors evaluate the model's performances using these features on the two created datasets.

Nguyen et al. [26] suggest that implicit correspondences between articles can help improve the performance of a fake news detection algorithm. The authors convert the task of detecting whether an article contains untruthful pieces of information into an inference problem in a Markov random field and transform the algorithm that solves this kind of problem into a deep neural network.

Truică et al. [27] propose the use of transformer-based sentence embeddings and transfer learning for fake news detection for news articles in both English and German.

In their study, Truică and Apostol [28] show empirically that the embedding used for encoding the data is one of the key factors in accurately determining the veracity of news articles.

Mayank et al. [29] propose DEAP-FAKED, a new model that uses Natural Language Processing techniques, Graph Neural Networks, and Knowledge graphs to identify Fake News. The experimental results on the Kaggle dataset show this approach improves the performance of misinformation detection. We use the DEAP-FAKED model in our comparison.

Truică and Apostol [15] designed MisRoBÆRTa, a transformed-based ensemble model used for multi-class fake news detection. The ensemble uses RoBERTa [30] and BART [31] to encode the textual content of the news. The experimental results on the Kaggle and The Fake News Corpus (FNC) datasets show how this approach improves the overall performance on the task of misinformation detection. We use MisRoBÆRTa to compare the results obtained by our proposed models on Kaggles.

Raza and Ding [32] propose FND-NS, a framework that learns useful representations for predicting fake news. FND-NS uses both the textual content and the social context to determine the veracity of news articles and social media posts. For the Fakeedit corpus, we compare the results of our models with the ones obtained by FND-NS.

B. FAKE NEWS MITIGATION

Sharma et al. [33] start out with graphs constructed from two Twitter datasets and design two independent diffusion paths for real and fake posts, each with a separate set of learned parameters. Using these paths and the obtained parameters, they determine the user characteristics that matter most in news spread and propose two methods to mitigate the spread of fake news: blocking a node (i.e., the user sending the news) or blocking the edge (i.e., the news transmission path). Their experiments indicate that an account that spreads fake news has a low follower count and no account description available, while an account that posts real news has a large number of followers, and is associated with well-known sources.

Saxena et al. [34] consider a social network as a graph in which there are three types of users: positive — which spread true news, negative — which spread false news, and neutral — which are influenced by the other two types. The authors model these influences probabilistically, where neutral nodes influenced by one of the two types of news are less likely to change stance, the longer the diffusion process lasts. To mitigate false information circulating through the graph, the authors design an algorithm that assigns each true news broadcaster a metric called "truth score", and selects the top-k sources with the best scores.

Shu et al. [35] contribute to the field of false news detection and mitigation by building a comprehensive dataset. They collect news from PolitiFact, GossipCop, and E! Online. The first two are platforms where the level of truth of news is labeled following a fact-checking process, while the latter is considered a credible news source. The dataset includes user features, i.e., number of followers, number of followees, user location (if mentioned in the profile description), user comments or retweets, as well as post features, i.e., post content (written or visual). These features are obtained by searching the news collected from the 3 sources using the Twitter search tool. By virtue of its diverse features, the dataset can be used in both fake news detection and mitigation tasks. Their best-performing model on this dataset is CNN [35], which we will also use in our comparison.

Sayyadiharikandeh et al. [36] address the problem of fake news mitigation assuming that non-human entities in social networks are set up to spread false information in an automated way. The authors implement a mechanism to detect such entities, which they call Botometer. They build several models to detect people and different types of bots in a social network. The models are aggregated into an ensemble learning algorithm, which produces the final classification results.

Nevertheless, none of the above approaches assembles a combined fake news detection and mitigation pipeline, as we do, further discussed in the following section. Firstly, we identify the harmful nodes that spread fake news in the network using a novel deep learning architecture. Secondly, we detect and rank harmful nodes using a novel algorithm that uses the direct weighted graph structure. Lastly, we immunize the network using a blocking strategy based on the ranked list of harmful nodes.

III. METHODOLOGY

Figure 1 presents our pipeline for fake news detection and mitigation. We mine a Social Media platform in real-time to detect nodes that spread harmful content; for each such node, we immunize the network with a real-time mitigation strategy. To be labeled as true or fake, a post passes through a preprocessing stage, where its content is edited and converted to a real-number matrix, which is passed to the detection stage. The detection stage is represented by one of our two proposed architectures, which outputs a post label. In case the post is false, we start the mitigation process; we construct a propagation path, i.e., a minimum-cost weighted directed spanning tree starting from each node that spreads fake news, compute a score for and rank the harmfulness of each node, and eventually extract the top-k most harmful nodes.

A. PREPROCESSING

The preprocessing module cleans the text and minimizes the vocabulary while preserving meaning; it comprises three steps: (1) punctuation marks and stop words removal, (2) lemmatization, and (3) word encoding using word embeddings.

Due to their high frequency of occurrence and specificity, stop words and symbols do not add relevant information when fed into the detection model. Thus, they may better be removed

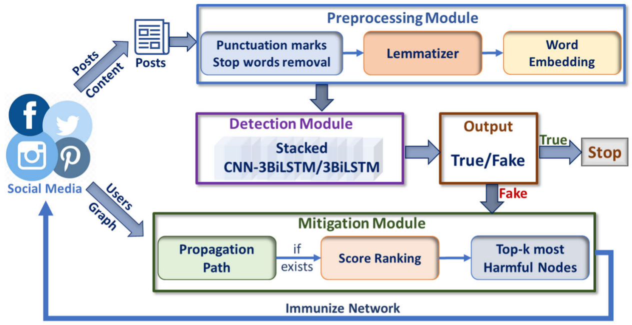


FIGURE 1. **Detection and mitigation pipeline.**

from posts. In the first stage of preprocessing, we remove these two types of elements. In the second stage, lemmatization, we transform words with similar forms into one unique form to homogenize the content of the posts. For the posts to be included in our proposed models, each word is transformed into a numerical representation using a word embedding vector. We use two pre-trained word embedding models and a model we trained on the chosen datasets.

B. FAKE NEWS DETECTION

The purpose of classification is to predict previously unseen items based on inferences derived by training on a set comprising news articles/posts and their labels. We first describe the basic elements of the two proposed architectures and motivate why we chose them.

We use an *Embedding* layer at the start of our models to store word embeddings that we obtain through the continuous bag-of-words model or pre-trained models. Thereby, our architectures have access to a word embedding dictionary. In general, an embedding layer would either generate embeddings or map every word of a post to a real-number vector representation; since we generate word embeddings in the preprocessing stage, we use the embedding layer with the latter function.

We employ a *Bidirectional Long Short-Term Memory (BiLSTM)* layer because the news we predict represents word sequences, and this type of layer is specialized in learning long-term dependencies in text. A further reason to employ such a layer is that it uses two simple LSTM architectures that look both forward and backward in the sequence, such that given a sequence element, both previous and future elements are available to the network.

We use a *Convolutional* layer with N filters and kernel size k to extract patterns from a k-size window in the data passed to the network, creating N features by means of a convolution operation between the text window and every distinct filter, and adding a bias term. Every filter has an associated channel where it stores features. The final result of the Convolutional layer with N filters and size k applied on a post of length L consists of N channels of length $L-k+1$, each containing the new features obtained by its corresponding filter. We equalize post lengths in the preprocessing stage by zero-padding or truncation, such that the post length is constant, leading to constant channel size for all posts. Following the Convolutional layer, we use a *max-pooling* layer with pooling size p to decrease the size of the feature channels by grouping elements into groups of length p and choosing only the feature with the maximum value. We use *dropout* layers to deactivate a percentage of the outputs coming from the previous layer; thus, we reduce overfitting by creating artificial noise and improve the generalization capacity of the network when new, unknown data is fed for prediction. We use *dense* layers with linear activation as a connection between the network layers, and a final dense layer with a softmax activation function to produce the classification result, i.e., the probability of the post to be true.

Concretely, we propose two deep learning models, namely CNN-3BiLSTM and 3BiLSTM. The main difference between them is that CNN-3BiLSTM creates new features through a convolutional layer, from which it extracts the best-generated features using a max-pooling layer, while 3BiLSTM feeds the initial features directly to the BiLSTM and dropout sequences.

In the stacked CNN-3BiLSTM architecture (Figure 2), we use an *embedding* layer where we pass word embeddings as weight parameters, followed by a *convolutional* layer with one group of 128 filters and kernel size 3 with ReLU as activation function and a *max-pooling* layer with pool size 2 and the number of strides set to 2. We add a *dense* layer with 256 units, followed by three BiLSTM layers having dropout layers in-between. We set the number of units for the BiLSTM layers to 64 and the dropout rate to 0.2. The architecture ends with two dense layers: one with 128 linear activation units and one with 1 sigmoid activation unit.

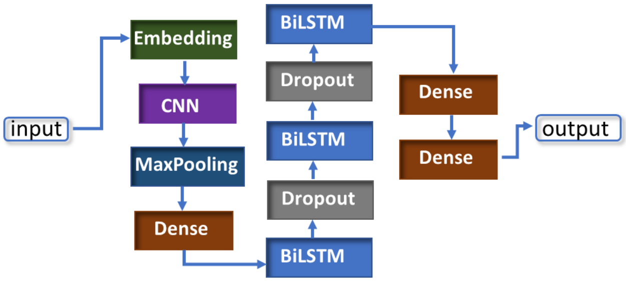


FIGURE 2. **Stacked CNN-3BiLSTM architecture.**

In the stacked 3BiLSTM architecture (Figure 3), we use an embedding layer with weights set as the word embeddings, then stack three 128 units BiLSTM layers with dropout layers in-between, with a dropout rate of 0.2. As in the previous model, the architecture ends with two Dense layers.

C. FAKE NEWS MITIGATION

Having detected a source of fake news, we mitigate its spread by rating each user with a network-aware ranking score that represents how influential that user is. This ranking score weighs in the following network-based goals: firstly, we aim to immunize nodes that are followed extensively, by followers

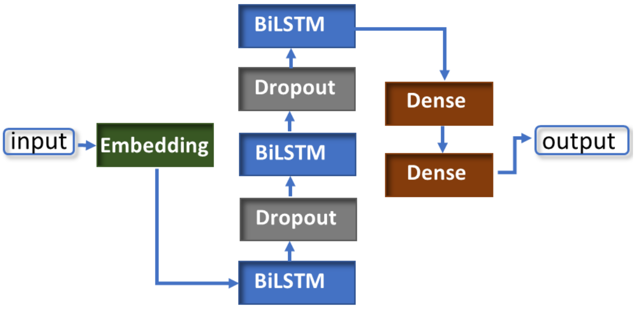


FIGURE 3. Stacked 3BiLSTM architecture.

themselves having many followers; thus, we take into account the height of the subtree spanning out from them, i.e., the length of diffusion paths. Secondly, we wish to immunize nodes that spread information to multiple nodes; therefore, we take into account the size of the spanning subtree, i.e., the number of nodes. Lastly, we intend to immunize nodes that spread information faster than others; ergo, we immunize nodes having high information diffusion speed. A high ranking score by these criteria implies that a user has a high impact on news spread. After calculating this score for each user, we select the top-k most harmful users to be proposed for removal or added to a blacklist.

Figure 4 presents an example that evaluates every node that may spread a harmful article for a given source node. Firstly, given a weighed directed graph $G = (V, E)$ with positive costs $t_{u,v} \in \tau$, where $\tau = \{t_{u,v} | (u, v, t_{u,v}) \in E\}$, with each $t_{u,v}$ measuring the time required to propagate the harmful content from u to v, we build a minimum-cost weighted directed spanning tree (MCWDST) T rooted at the detected source node r, using Algorithm 1.

Algorithm 1 works as follows. We initialize (1) the tree's set of vertices V_t to $\{r\}$ and E_t to the empty set, and (2) the sets of untested vertices V_c to $\{r\}$ and the set of untested edges E_c to all edges except the ones that point to r (Lines 1-4). While there are still untested vertices (Line 5), we initialize, in each iteration, a set for vertices V_n and one for edges E_n (Lines 6-7) which are needed to update V_c and E_c at the end of the iteration (Lines 20-21). We pass over all vertices $n \in V_c$ and find the edges in E_c where n is the parent of v (Lines 8-9). If the node v is not present in V_t then we add v to V_t and V_n and the edge (n, v) to E_t and E_n (Lines 10-14), otherwise, it means that there is an edge pointing to v in E_t (Lines 15-17). In this case, we verify if it is cheaper to pass through this edge than to use the existing one (Line 16). If it is cheaper, we update E_t by removing the existing edge (Line 17) and adding the new edge to E_n (Line 18). We update V_c with the newly added nodes in V_n, and E_c by removing the nodes in E_n (Lines 20-21). When the algorithm exits the while block, it returns the propagation tree T. The complexity is $O(|V||E|)$. Having obtained this tree, we evaluate the potential harmfulness of each node using a ranking function and sort the nodes in descending order of their scores to obtain a leaderboard of the most harmful nodes.

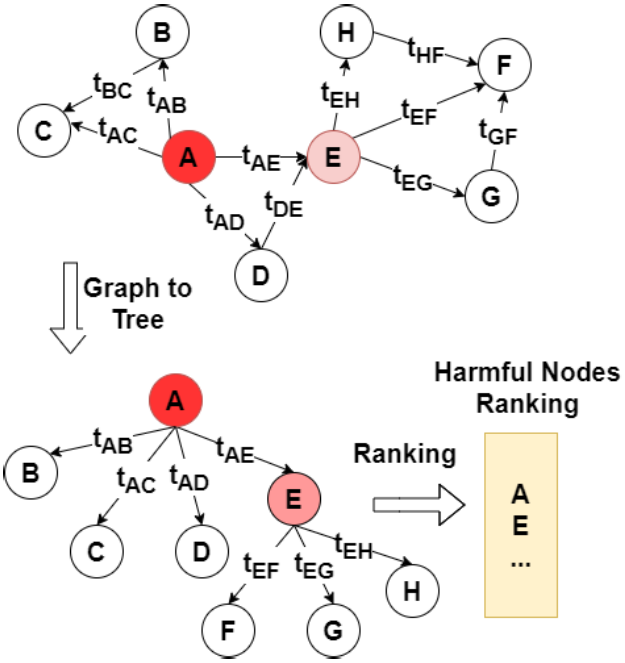


FIGURE 4. From graph to the harmful nodes ranking.

Algorithm 1 MCWDST for Starting Node r

```
Input  : the weighted directed graph $G = (V, E)$ the
         starting node $r$
Output : the MCWDST $T = (V_t, E_t)$
1  $V_t \leftarrow \{r\}$
2  $E_t \leftarrow \emptyset$
3  $V_c \leftarrow \{r\}$
4  $E_c \leftarrow E \setminus \{(u, r, t_{u,r}) | (u, r, t_{u,r}) \in E\}$
5  while $V_c \neq \emptyset$ do
6      $V_n \leftarrow \emptyset$
7      $E_n \leftarrow \emptyset$
8      foreach $n \in V_c$ do
9          foreach $(u, v, t_{u,v}) \in E_c$ do
10             if $n = u \wedge v \notin V_t$ then
11                 $V_t \leftarrow V_t \cup \{v\}$
12                 $E_t \leftarrow E_t \cup \{(n, v, t_{n,v})\}$
13                 $V_n \leftarrow V_n \cup \{v\}$
14                 $E_n \leftarrow E_n \cup \{(n, v, t_{n,v})\}$
15             if $n = u \wedge v \in V_t$ then
16                 if $t_{r,v} > t_{r,n} + t_{n,v}$ then
17                     $E_t \leftarrow$
                       $E_t \setminus \{(u', v, t'_{u',v}) | (u', v, t'_{u',v}) \in E_t\}$
18                     $E_t \leftarrow E_t \cup \{(u, v, t_{u,v})\}$
19                     $E_n \leftarrow E_n \cup \{(u, v, t_{u,v})\}$
20     $V_c \leftarrow V_n$
21     $E_c \leftarrow E_c \setminus E_n$
22 return $T = \{V_t, E_t\}$
```

Our ranking function assesses each node n (Equation (1)) in real-time in terms of three components that meet the criteria enumerated previously:

(1) $H(n)$: the normalized height of the subtree of n to promote nodes that have a long chain of perpetuating followers,
(2) $A(n)$: the normalized size of the subtree of n to promote nodes that reach many other nodes, and
(3) $f_t(n)$: a function applied to the timestamps of the descendants of n to promote nodes of high information diffusion speed.

$$rank(n) = H(n) + A(n) + (1 - f_t(n)) \tag{1}$$

The normalized weight $H(n) = \frac{h_{max\_n}}{h_{max\_\mathcal{R}}}$, where h_{max_n} is the maximum height of the subtree of n and $h_{max\_\mathcal{R}}$ the maximum height of the whole tree, takes into account the depth of the propagation path of harmful content, i.e., how many levels in the graph can be affected by harmful content spread by node n. This factor is in range $[0, 1]$ as $0 \leq h_{max\_n} \leq h_{max\_\mathcal{R}}$. Note, for leaf nodes, $H(n)$ is always 0 as a leaf does not have a subtree.

The normalized area $A(n) = \frac{A_n}{A_\mathcal{R}}$, where A_n is the area of the subtree of n and $A_\mathcal{R}$ is the area of the whole tree, encodes into the ranking function the hole propagation paths, i.e., how many nodes are affected by node n. This factor is in range $(0, 1]$ as $0 < A_n \leq A_\mathcal{R}$. Note, for leaf nodes, $A(n)$ is always 0 as a leaf does not have a subtree.

We include $f_t(n)$ to weigh the normalized timestamp of spreading fake news (the cost of each edge). For a node n with k edges and a sorted set of timestamps (costs of edges) $\tau_n = \{t_i | t_i = t_{n,v} \land (n, v, t_{n,v}) \in E \land i = \overline{1, k}\}$, we normalize the value of a timestamp t_i in $[0, 1]$ using Equation (2). Note: this function is applied only to nodes that have descendants in the tree. For leaf nodes, we do not apply the function as they have no descendants. Thus, in the ranking function $rank(n)$, the value of $f_t(n)$ for leaf nodes is always 0.

$$t_i' = \begin{cases} \dfrac{t - \min_{t_j \in \tau_n}(t_j)}{\max_{t_j \in \tau_n}(t_j) - \min_{t_j \in \tau_n}(t_j)} & \min_{t_j \in \tau_n}(t_j) \neq \max_{t_j \in \tau_n}(t_j) \\ 0.5 & \min_{t_j \in \tau_n}(t_j) = \max_{t_j \in \tau_n}(t_j) \end{cases} \tag{2}$$

We consider three ways of computing $f_t(n)$:
(1) average timestamp value, $average(t_n) = \frac{1}{k}\sum_{i=0}^{k} t_i$, which weighs in the average propagation time;
(2) median timestamp (Equation (3) using the sorted order), which may be more descriptive than the average when there are outliers in the sequence that might skew the average value; and
(3) the ratio of minimum to maximum timestamp value, $ratio(t_n) = \frac{min_{t_i \in \tau_n}(t_i)}{max_{t_i \in \tau_n}(t_i)}$, which emphasizes extremes.

If a node has no descendants, the value of the function $f_t(n)$ is 0. Regardless of the computational approach for $f_t(n)$, the function is in the range $[0, 1]$.

$$median(t_n) = \begin{cases} t_{\frac{k}{2}} & \text{if } k \text{ is even} \\ \dfrac{t_{\frac{k-1}{2}} + t_{\frac{k+1}{2}}}{2} & \text{if } k \text{ is odd} \end{cases} \tag{3}$$

In effect, $f_t(n)$ measures the latency by which a node n can disseminate fake news to its descendants if such exists; for a harmful node, this value is close to 0. To express speed, we use $1 - f_t(n)$ in the ranking function. Given that $rank(n) \in (1, 3]$, a harmful node will have a value that is close to 3. After scoring all nodes, we sort them in descending order of their scores and choose the top-k nodes, which consequently have a high potential to infect the network with fake news. Note, $rank(n) = 1$ for a leaf node as $A(n) = 0$, $H(n) = 0$, and $1 - f_t(n) = 1$

Returning to Figure 4, node A is marked by the detection module as a source node that spreads fake news. We build a propagation tree for A using Algorithm 1, score each node in that tree using the ranking function, and create a list of potentially harmful nodes. In effect, we find E, which may spread content from A to its followers, to be the second most harmful node after A.

After extracting node ranks, we start immunizing the network from the root node to the leaves. To do so, we suggest a minimally interventionist approach of lowering the rank of a harmful post to the end of its followers' feed. Thus, the harmful content will require more time to reach other users and followers if they are not expressly looking for it by browsing the feed. To mitigate the behavior where users search for content, we can monitor the next ranked nodes to stop the network infection and apply the same strategy as for the root. Such immunization is real-time, allowing for continuous monitoring nodes that may reshare the harmful content.

IV. IMPLEMENTATION

In this section, we present the details of our implementation in Python 3. The code is publicly available on GitHub at https://github.com/DS4AI-UPB/MCWDST.

A. PREPROCESSING

We load the data using *pandas* [37]. For processing, we use: (1) *SciKit-Learn* [38] *LabelEncoder* to transform labels from categorical to numerical, (2) regular expressions to clean the text, (3) *NLTK* [39] *WordNetLemmatizer* to extract the lemmas, and (4) *Keras* [40] *Tokenizer* to vectorize the clean text corpus.

To obtain word embeddings, we consider two approaches: (1) training a Word2vec CBOW model on each dataset using *Gensim* [41], and (2) using the pre-trained models *GloVe pre-trained* and *Word2Vec pre-trained*.

We train 100-dimensional *Word2Vec CBOW* [42] embeddings using a learning rate of 0.025 and a word window size of 5 for 5 epochs. We chose CBOW over Skip-Gram because it is faster to train and it uses distributed representations of context to predict words instead of using a distributed representation of the words to predict the context. For the pre-trained embeddings we chose: (1) *GloVe pre-trained*: a 100-dimensional GloVe [43] embedding trained on Wikipedia 2014 and Gigaword 5, and (2) *Word2Vec pre-trained*: 300-dimension Word2Vec embeddings trained on Wikipedia data.

B. FAKE NEWS DETECTION

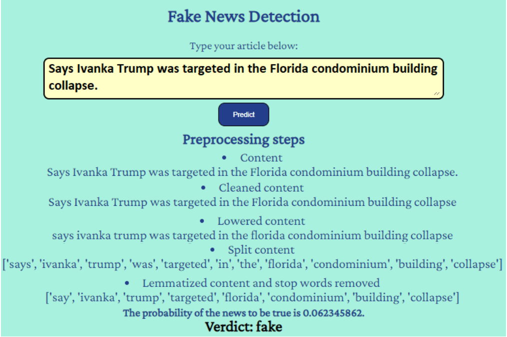
In the Detection module, we use the *Keras* interface of the *TensorFlow* [44] library to build our networks; we compile the model using the Adam optimizer with binary cross-entropy as the loss function, train the models for 200 epochs using a validation split of 0.2 and a batch size of 128, and add an early-stopping callback that monitors the validation loss.

C. FAKE NEWS MITIGATION

We construct propagation trees by reading the graph's adjacency matrix, along with the timestamps associated with each node. The propagation tree is the minimum-cost weighted directed spanning tree starting from a source node that spreads fake news. We iterate through all nodes in the tree and assign a ranking score to each. To calculate the function of descendant timestamps, we use the built-in Python library *statistics*. We sort the scores dictionary by descending values and take the first k nodes as the most harmful. We visualize the tree by *Graphviz* [45].

D. GRAPHICAL USER INTERFACE

We build a user-friendly graphical user interface consisting of three pages: a landing page, a fake news classification page, and a fake news mitigation page. For detection (Figure 5), the user inputs the content of the article or post in a text box, clicks the ''Predict'' button, and gets the classification result as a verdict; the results of preprocessing are thereby applied to the text, with stop words and punctuation removed, and lemmatization of the remaining words. Figure 6 presents the fake news mitigation page. The user can opt to visualize a node's minimum-cost weighted directed spanning tree in two formats: human-friendly format as source — (destination, timestamp), or Twitter15 [46] format. The format is toggled by a check box. Upon pressing ''Submit'', the tree image is shown along with a ranking of the most harmful nodes therein according to our evaluation metric.



Fake News Detection

Type your article below:

Says Ivanka Trump was targeted in the Florida condominium building collapse.

Preprocessing steps

- Content

Says Ivanka Trump was targeted in the Florida condominium building collapse.

- Cleaned content

Says Ivanka Trump was targeted in the Florida condominium building collapse

- Lowered content

says ivanka trump was targeted in the florida condominium building collapse

- Split content

['says', 'ivanka', 'trump', 'was', 'targeted', 'in', 'the', 'florida', 'condominium', 'building', 'collapse']

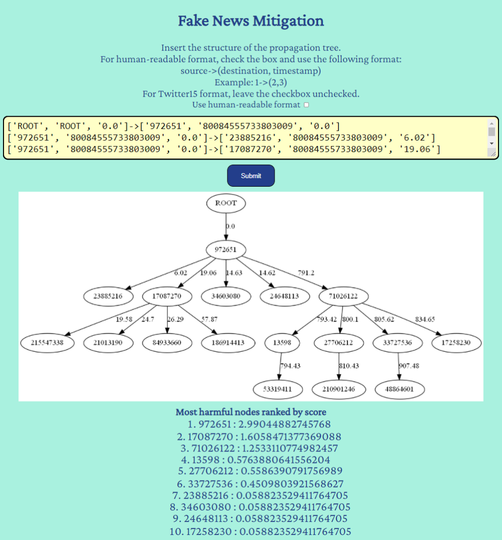
- Lemmatized content and stop words removed

['say', 'ivanka', 'trump', 'targeted', 'florida', 'condominium', 'building', 'collapse']

The probability of the news to be true is 0.062345862.

Verdict: fake

FIGURE 5. Fake news detection page.



Fake News Mitigation

Insert the structure of the propagation tree.
For human-readable format, check the box and use the following format:
source->(destination, timestamp)
Example: 1->(2,3)
For Twitter15 format, leave the checkbox unchecked.
Use human-readable format □

```
['ROOT', 'ROOT', '0.0']->['972651', '80084555733803009', '0.0']
['972651', '80084555733803009', '0.0']->['23885216', '80084555733803009', '6.02']
['972651', '80084555733803009', '0.0']->['17087270', '80084555733803009', '19.06']
```

Most harmful nodes ranked by score

1. 972651 : 2.99044882745768
2. 17087270 : 1.6058471377369088
3. 71026122 : 1.2533110774982457
4. 13598 : 0.5763880641556204
5. 27706212 : 0.5586390791756989
6. 33727536 : 0.4509803921568627
7. 23885216 : 0.058823529411764705
8. 34603080 : 0.058823529411764705
9. 24648113 : 0.058823529411764705
10. 17258230 : 0.058823529411764705

FIGURE 6. Fake news mitigation page.

V. EXPERIMENTAL RESULTS

In this section, we present the datasets we use for evaluation, compare the performance of the two proposed architectures, and analyze how each metric in our ranking function performs, individually and collectively, in mitigating harmful content propagation.

A. DATASETS AND PREPROCESSING ANALYSIS

We use four datasets to evaluate the detection models and one dataset to evaluate the mitigation metrics. We split each dataset for classification in training and testing sets using an 80%-20% ratio. From the training set, we extract 20% for validation.

The *Fake News Corpus (FNC)* dataset contains over 9,4 million articles labeled using 11 predefined labels. We chose a balanced subset of 20 000 articles tagged as either reliable or fake. As the original dataset is automatically filtered, we manually verified each article's content to ensure it is correctly labeled.

The *Kaggle* dataset, downloaded from the Kaggle Fake News Detection challenge, consists of 20 800 articles tagged as reliable or unreliable. After eliminating null rows, the resulting dataset consists of 10 387 reliable and 10 374 unreliable tagged news.

The *GossipCop* [35] dataset consists of 22 155 news titles, article' URLs, tweet IDs of users that retweeted the article, and tags that label articles as true or fake. As the dataset is initially imbalanced, we select 5 323 articles representing both tags.

The *Fakeddit* [16] is a multimodal dataset that allows 2-way, 3-way, or 6-way classification using images, text, or both. We use the title of the post and the 2-way label to perform our task. We edit the dataset to exclude entries that contain images and select 107 742 posts with a 1:1 true-to-fake ratio.

The *Twitter15* [46] contains news articles and their labels, along with information on the directed network structure among users, followers, and followees. We use this dataset to analyze the performances of our mitigation module. It consists of 1 490 news articles labeled as true, unverified, non-rumor, or false. We select 361 articles identified as false, construct their propagation traces, and compute scores for the nodes.

Table 1 presents the features of the datasets with cleaned content, where every article contains words that are lemmatized. In terms of average and maximum text length, we can group the datasets into two categories: large ones (i.e., FNC, Kaggle) and small ones (i.e., Fakeddit, GossipCop). The articles in FNC and Kaggle consist of sequences of sentences or phrases, while the posts in Fakeddit and GossipCop consist of a summary sentence. In terms of size, the largest dataset is Fakeddit and the smallest is GossipCop.

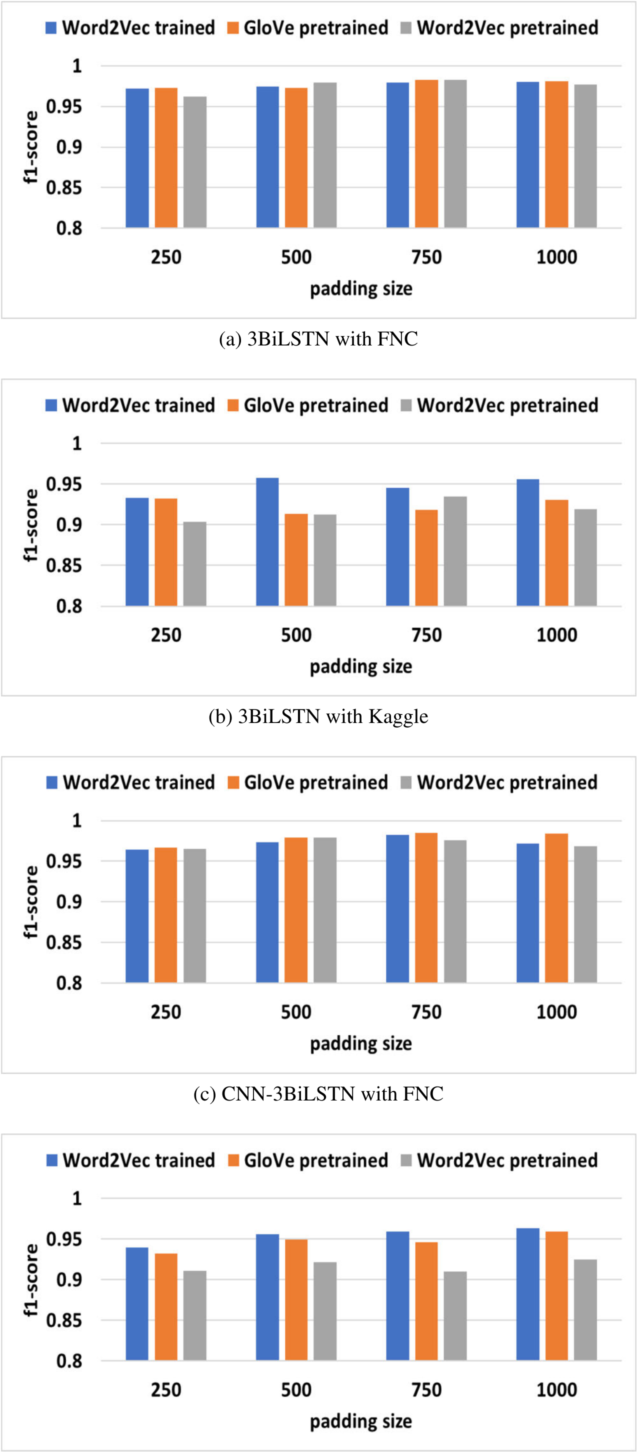
TABLE 1. Post-processing stage datasets details.

Dataset	Number of Article	Average Tokens	Maximum Tokens	True Articles	Fake Articles
FNC	20 000	412	10 736	10 000	10 000
Kaggle	20 761	419	12 059	10 387	10 374
GossipCop	10 646	8	26	5 323	5 323
Fakeddit	107 742	6	41	53 975	53 767

B. FAKE NEWS DETECTION RESULTS

Table 2 provides the classification results of the proposed architectures using the 3 embedding methods, i.e., Word2Vec trained, GloVe pre-trained, and Word2Vec pre-trained. The results are the average for each score after 10 distinct executions using random seeding. We use a default padding size of 10 for small-length data (Fakeddit and GossipCop) and 1 000 for large-length (i.e., Kaggle and FNC) data. CNN-3BiLSTM obtains the overall best results on Fakeddit, Kaggle, and FNC, although the difference between CNN-3BiLSTM and 3BiLSTM is relatively small. On FNC, the best accuracy is obtained when using CNN-3BiLSTM with GloVe pre-trained, i.e., 98.37. The difference between CNN-3BiLSTM and BiLSTM with the same embedding is insignificant, i.e., 0.22. CNN-3BiLSTM with Word2Vec trained registers an accuracy of 96.38 on Kaggle, with only a small increase of 0.74 over BiLSTM with the same embedding. On GossipCop, CNN-3BiLSTM with Word2Vec pre-trained obtains the best accuracy, i.e., 74.97. On Fakeddit, 3BiLSTM with Word2Vec pre-trained embeddings obtains the best accuracy. We emphasize that, in fake news detection, recall is an important metric as it tells how many fake articles are correctly classified. On datasets having a large average text length (i.e., FNC and Kaggle), recall scores are between 90 and 100. On small-length data, recall is lower than 77.

Table 3 presents a comparison between the results obtained by our models and the current state-of-the-art models. We can observe that the proposed architecture outperforms or obtains similar results as the current models. MisRoBÆRTa [15] and the proposed CNN-3BiLSTM with Word2Vec-trained embeddings obtain very similar results. The small difference between the models is a direct result of the embedding: MisRoBÆRTa uses two context-aware stat-of-the-art transformers, i.e., BART [31] and RoBERTa [30], as embeddings, while we use a static embedding.



(a) 3BiLSTN with FNC

(b) 3BiLSTN with Kaggle

(c) CNN-3BiLSTN with FNC

(d) CNN-3BiLSTN with Kaggle

FIGURE 7. The influence of padding on the f1-scores by architecture.

TABLE 2. Classification results (Notes: 1. results are the average of 10 runs and 2. bold marks the overall best result).

Dataset	Embedding	Model	Accuracy	Precision	Recall	F1-Score
FNC	Word2Vec trained	CNN-3BiLSTM	97.15	97.17	97.13	97.14
		3BiLSTM	98.02	98.02	98.02	98.02
	GloVe pre-trained	CNN-3BiLSTM	**98.37**	98.44	98.34	98.37
		3BiLSTM	98.15	98.16	98.13	98.14
	Word2Vec pre-trained	CNN-3BiLSTM	96.85	96.87	96.84	96.84
		3BiLSTM	97.67	97.69	97.66	97.67
Kaggle	Word2Vec trained	CNN-3BiLSTM	**96.38**	96.39	96.38	96.38
		3BiLSTM	95.64	95.70	95.65	95.64
	GloVe pre-trained	CNN-3BiLSTM	95.95	95.95	95.95	95.95
		3BiLSTM	93.06	93.09	93.07	93.06
	Word2Vec pre-trained	CNN-3BiLSTM	92.46	92.46	92.46	92.46
		3BiLSTM	91.93	91.93	91.93	91.93
GossipCop	Word2Vec trained	CNN-3BiLSTM	68.82	71.28	69.41	68.28
		3BiLSTM	69.24	70.45	69.55	68.98
	GloVe pre-trained	CNN-3BiLSTM	73.66	73.69	73.60	73.60
		3BiLSTM	74.08	74.08	74.03	74.04
	Word2Vec pre-trained	CNN-3BiLSTM	**74.97**	75.26	75.03	74.92
		3BiLSTM	73.61	74.32	73.72	73.47
Fakeddit	Word2Vec trained	CNN-3BiLSTM	71.38	71.59	71.40	71.32
		3BiLSTM	71.91	71.97	71.88	71.87
	GloVe pre-trained	CNN-3BiLSTM	74.26	74.27	74.25	74.25
		3BiLSTM	75.66	75.89	75.70	75.62
	Word2Vec pre-trained	CNN-3BiLSTM	73.80	73.90	73.78	73.76
		3BiLSTM	**76.90**	77.27	76.89	76.82

TABLE 3. Comparison with state-of-the-art models (Note: bold text marks the overall best result).

Dataset	Model	Accuracy	Precision	Recall	F1-Score
Kaggle	CNN-3BiLSTM + Word2Vec trained	**96.38**	96.39	96.38	96.38
	DEAP-FAKED [29]	88.66	N/A	N/A	89.55
	MisRoBÆRTa [15]	**97.85**	97.85	97.85	97.85
GossipCop	CNN-3BiLSTM + Word2Vec pre-trained	**74.97**	75.26	75.03	74.92
	CNN [35]	72.30	N/A	N/A	72.50
Fakeddit	3BiLSTM + Word2Vec pre-trained	**76.90**	77.27	76.89	76.82
	FND-NS [32]	74.80	72.40	77.60	74.90

We also study the impact of padding size on performance with large data, i.e., Kaggle and FNC. We do not include small-length data in our analysis, since their average text length is 6-8 and their content consists of one sentence; thus, their padding size range is small. We evaluate the two datasets in terms of f1-score for 4 padding sizes, namely 250, 500, 750, and 1 000. Figure 7 shows the f1-scores we obtain after 10 distinct executions using random seeding. On FNC with 3BiLSTM, the best-performing embedding strategy is Word2Vec pre-trained with a padding size of 750. On Kaggle with 3BiLSTM, the Word2Vec-trained embeddings obtain the best results when the padding size is 500. On FNC with CNN-3BiLSTM, GloVe pre-trained with a padding size of 750 achieves the best f1-score. The Word2Vec-trained embeddings with a 1 000 padding size have the best score on Kaggle with CNN-3BiLSTM. We note that a larger text size does not necessarily translate into better results. We conclude that the padding size has a small impact on model performance.

C. FAKE NEWS MITIGATION RESULTS

To analyze the performance of our proposed algorithm in identifying harmful nodes by measuring the distribution of news by users, we use 361 trees with an average of 335 nodes per tree, a minimum number of nodes of 97, a maximum number of nodes of 2 971, an average tree height of 4, a minimum height of 2, and a maximum height of 10.

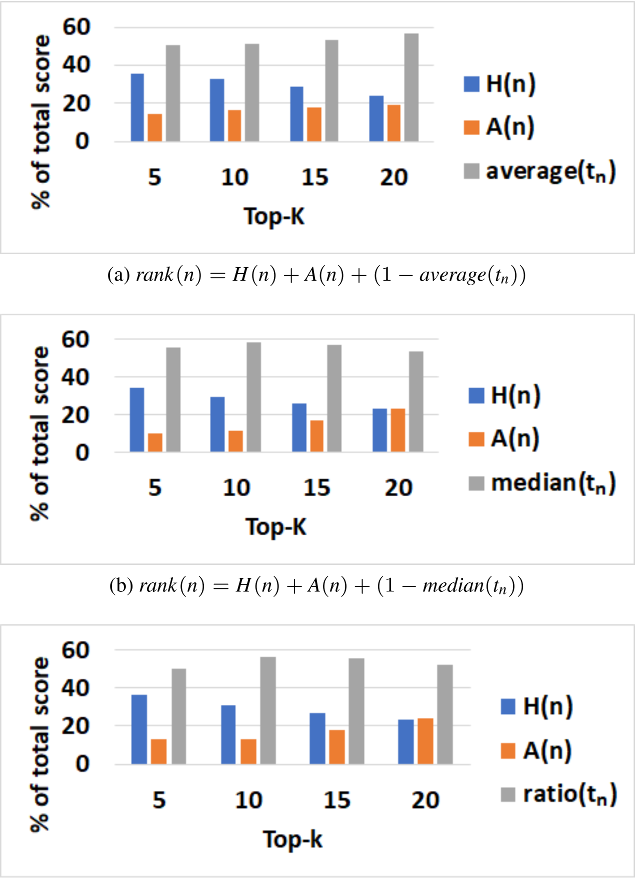
We first analyze the importance of $f_t(n)$ in ranking nodes. We divide the interval $(1, 3]$ into two intervals and study the influence of the function $f_t(n)$ on the obtained subintervals. Table 4 shows the percentages of the scores obtained for the evaluation metric in each range, for the three strategies applied in the $f_t(n)$ function, when collecting the top-k most harmful nodes with $k \in \{1\,000, 2\,000, 3\,000\}$. A higher score represents a greater capability to spread untruthful information faster. Thus, a score in $(2, 3]$ indicates a very harmful node to be immunized first, a score in $[1, 2]$ indicates a mildly harmful node to be immunized in second priority, where nodes with a score close to 1 suggest that a node is weakly to not harmful and should be immunized in last priority. A node with a score of 1 is a lead node. We observe that when using $median(t_n)$, our ranking function scores with higher values more harmful nodes (i.e., nodes $\in (2, 3]$) than when using $average(t_n)$ or $ratio(t_n)$ (i.e., harmful nodes are ranked in $\in (1, 2]$). Furthermore, as we know which nodes are harmful and which are not, $rank(n)$ identifies correctly all the harmful nodes regardless of the method used to compute $f_t(n)$.

Figure 8 plots the contribution that each component of the proposed ranking function has in a node's final score. We analyze the top-k most harmful nodes, where $k = \{5, 10, 15, 20\}$. The $f_t(n)$ function influences the score the

TABLE 4. The distribution of the top-k nodes.

k	$f_t(n)$	% in (1, 2]	% in (2, 3]
1 000	$average(t_n)$	68.51	31.49
	$median(t_n)$	60.10	39.90
	$ratio(t_n)$	89.10	10.90
2 000	$average(t_n)$	83.31	16.69
	$median(t_n)$	80.05	19.95
	$ratio(t_n)$	94.55	5.45
3 000	$average(t_n)$	90.20	9.80
	$median(t_n)$	86.70	13.30
	$ratio(t_n)$	96.36	3.64

most, followed by the $H(n)$ and $A(n)$ functions. Notably, the weight of $f_t(n)$ is more than 50% in all evaluation situations. In all three subgraphs, as we increase k, the contribution of $H(n)$ is transferred to $A(n)$. With $k = 20$, $A(n)$ surpasses $H(n)$ (Figure 8(b) and 8(c)). In conclusion, the information diffusion speed has the most contribution to our ranking function. That is a reasonable design, as, for a small k, nodes that have long chains of followers (i.e., a longer maximum diffusion path) play a more important role than the nodes that can reach many nodes. As k increases, the impact of the nodes with many followers decreases, and the impact of nodes that reach many nodes increases.



(a) $rank(n) = H(n) + A(n) + (1 - average(t_n))$

(b) $rank(n) = H(n) + A(n) + (1 - median(t_n))$

(c) $rank(n) = H(n) + A(n) + (1 - ratio(t_n))$

FIGURE 8. The influences of the 3 components of the metric in evaluating the top-k most harmful nodes.

D. MITIGATION SCALABILITY RESULTS

We want to test the scalability of our solution and determine the mitigation strategy performance in a real-world setting. We collect from Twitter a graph structure containing over 10 000 nodes and 89 000 edges and split it into 3 subgraphs for scalability testing. Table 5 presents the MCWDST construction and ranking list creation time in seconds for these subgraphs. We can observe that for a small graph, the mitigation is done almost instantly. This is useful when we detect and target the harmful content spreaders within a disinformation network. For a larger network, the entire process is done in near real-time, as it takes less than 5 minutes. Thus, the best strategy is to immunize the network by targeting the topmost harmful nodes. Finally, we compare our method with two preventive network immunization strategies, i.e., NetShield [47] and SparseShield [48] an improved version of NetShield, and one counteractive network immunization in large networks, i.e, DAVA [49].

Although both preventive network immunization algorithms are faster, the main problems with NetShield and SparseShield that we solved with MCWDST are: *(1)* nodes are blocked in no particular order if they converge, *(2)* nodes are not ranked, *(3)* the edge weights are not used in the immunization process, *(4)* the directed graph structure is not considered, and *(5)* the number of immunized nodes is dependent on the budget.

Whereas our proposed solution gives an immunization strategy for a weighted directed graph by returning a blocking order by ranking the harmfulness of nodes.

When comparing MCWDST with DAVA, another counteractive network immunization algorithm, we observe that:

(1) DAVA blocks all the nodes in no particular order while our MCWDST manages to block nodes in the order given by the information diffusion paths,
(2) DAVA does not rank nodes, while MCWDST ranks them from most to least harmful,
(3) DAVA is dependent on the budget and immunizes a given number of nodes, while MCWDST immunizes the entire network, and
(4) DAVA does not consider the directed graph structure, while MCWDST takes this graph property into account.

VI. DISCUSSION

This section highlights the results and lessons learned from the implementation and presents an in-depth analysis of the two modules.

A. FAKE NEWS DETECTION

We observe that the pre-trained models (i.e., Word2Vec pre-trained and GloVe pre-trained) achieve better results on smaller datasets, i.e., GossipCop and Fakeddit, while Word2Vec trained and GloVe pre-trained obtain better results on larger datasets, i.e., FNC and Kaggle. These results of the classification models directly reflect the typical text length in each dataset. For small-length texts, it is better to use pre-trained embeddings as they better encode

TABLE 5. Mitigation scalability tests.

Nodes	Edges	MCWDST (s)	Ranking (s)	DAVA	NetShield (s)	SparseShield
978	10 217	2.01	0.66	1.78	0.29	0.07
5 210	49 124	53.68	20.69	60.75	12.02	0.42
10 210	89 124	411.96	138.44	386.99	102.13	1.33

semantic, syntactic, and contextual information that cannot be found in short statements. This is mitigated in large datasets by the texts' length, which offers sufficient semantic, syntactic, and contextual information to train accurate and data-specific word embeddings. The proposed novel deep learning architectures offer state-of-the-art results on all tested datasets, i.e., an accuracy of 88.66 on Kaggle [29], 74.80 on Fakeddit [32], and 72.30 on GossipCop [35]. We note that the longer texts enable the bidirectional LSTM layers with and without a CNN layer on top to detect meaningful patterns for fake news detection. Even though the experimental results show that 3BiLSTM architecture obtains good results, the CNN-3BiLSTM architecture achieves the best scores on FNC, Kaggle, and GossipCop. This result arises from the combination of a CNN layer with a MaxPooling layer, which generates new features and selects the best features among those generated. Moreover, on the Kaggle data, CNN-3BiLSTM outperforms 3BiLSTM with all embedding strategies we use. Furthermore, we observe that, on large datasets, a large-size padding strategy does not necessarily improve performance. It follows that an ample analysis w.r.t. size and word embedding should be performed on each dataset to determine the best padding approach.

B. FAKE NEWS MITIGATION

The proposed ranking function takes into account 3 network-specific components for mitigating harmful nodes: (1) $H(n)$, the length of the diffusion path, (2) $A(n)$, the spread of information, and (3) $f_t(n)$, the speed of information diffusion.

In our experiments on a real-world dataset, we observe that: (1) regardless of graph size, $f_t(n)$ has the largest contribution in the function, (2) for a small number of nodes to mitigate, the length of the diffusion path $H(n)$ adds more weight to the ranking function than the number of nodes that can be infected directly $A(n)$, and (3) for a large number of nodes to mitigate, both $H(n)$ and $A(n)$ have a similar impact on the ranking.

As $f_t(n)$ has the largest contribution of the three components to the score value of a node, it is, in effect, the most important factor in mitigating fake news in a network, according to our empirical findings. The comparison of different calculation strategies of $f_t(n)$ indicates the impact of the potential to spread fake news quickly, while the individual evaluation of the three components of the ranking function indicates that network structure is also important for the immunization task.

VII. CONCLUSION

In this paper, we proposed (1) two novel deep learning architectures for fake news detection, (2) a real-time algorithm for building the minimum-cost weighted directed spanning tree (i.e., MCWDST), and (3) a function that ranks harmful nodes in real-time, to combat the spread of fake news on social media.

The proposed deep learning architectures use convolutional and bidirectional LSTM layers that capture semantic, syntactic, and context information. These models offer state-of-the-art performance on the tested datasets. Our mitigation algorithm greedily constructs a minimum-cost weighted directed spanning tree for a given source node. The novel raking function takes into account network information (i.e., the length of the diffusion path, the spread of information, and the speed of information diffusion) as well as its harmful content to mitigate the spread of fake news in a network-aware, real-time manner. Ample benchmarking on five real-world datasets shows that the proposed models for fake news detection and the mitigation strategy are effective in stopping the spread of harmful content online.

For future research directions, we will consider using: (1) transformers as a way of generating word embeddings, and (2) graph embedding techniques to develop a more complex network-aware model that identifies harmful sources in real-time.

ACKNOWLEDGMENT

(Ciprian-Octavian Truică, Elena-Simona Apostol, and Radu-Cătălin Nicolescu contributed equally to this work.)

REFERENCES

[1] C. Truică, E. S. Apostol, T. Stefu, and P. Karras, "A deep learning architecture for audience interest prediction of news topic on social media," in *Proc. Int. Conf. Extending Database Technol.*, 2021, pp. 588–599.

[2] N. Pyrhönen and G. Bauvois, "Conspiracies beyond fake news. Produsing reinformation on presidential elections in the transnational hybrid media system," *Sociol. Inquiry*, vol. 90, no. 4, pp. 705–731, Nov. 2020.

[3] J. Petit, C. Li, B. Millet, K. Ali, and R. Sun, "Can we stop the spread of false information on vaccination? How online comments on vaccination news affect readers' credibility assessments and sharing behaviors," *Sci. Commun.*, vol. 43, no. 4, pp. 407–434, Aug. 2021.

[4] S. B. Naeem, R. Bhatti, and A. Khan, "An exploration of how fake news is taking over social media and putting public health at risk," *Health Inf. Libraries J.*, vol. 38, no. 2, pp. 143–149, Jun. 2021.

[5] J. Rose, "To believe or not to believe: An epistemic exploration of fake news, truth, and the limits of knowing," *Postdigital Sci. Educ.*, vol. 2, no. 1, pp. 202–216, Jan. 2020.

[6] C.-O. Truică, E.-S. Apostol, and P. Karras, "DANES: Deep neural network ensemble architecture for social and textual context-aware fake news detection," 2023, *arXiv:2302.01756*.

[7] E.-S. Apostol, C.-O. Truica, and A. Paschke, "ContCommRTD: A distributed content-based misinformation-aware community detection system for real-time disaster reporting," 2023, *arXiv:2301.12984*.

[8] O. Coban, C.-O. Truică, and E.-S. Apostol, "CONTAIN: A community-based algorithm for network immunization," *arXiv:2303.01934*, 2023.

[9] H. Allcott and M. Gentzkow, ''Social media and fake news in the 2016 election,'' *J. Econ. Perspect.*, vol. 31, no. 2, pp. 211–236, 2017.

[10] M. D. Molina, S. S. Sundar, T. Le, and D. Lee, '''Fake news' is not simply false information: A concept explication and taxonomy of online content,'' *Amer. Behav. Scientist*, vol. 65, no. 2, pp. 180–212, 2021.

[11] C.-C. Wang, ''Fake news and related concepts: Definitions and recent research development,'' *Contemp. Manag. Res.*, vol. 16, no. 3, pp. 145–174, Sep. 2020.

[12] A. Gelfert, ''Fake news: A definition,'' *Informal Log.*, vol. 38, no. 1, pp. 84–117, Mar. 2018.

[13] D. C. Brody and D. M. Meier, ''Mathematical models for fake news,'' 2018, *arXiv:1809.00964*.

[14] V.-I. Ilie, C.-O. Truică, E.-S. Apostol, and A. Paschke, ''Context-aware misinformation detection: A benchmark of deep learning architectures using word embeddings,'' *IEEE Access*, vol. 9, pp. 162122–162146, 2021.

[15] C.-O. Truică and E.-S. Apostol, ''MisRoBERTa: Transformers versus misinformation,'' *Mathematics*, vol. 10, no. 4, p. 569, 2022.

[16] K. Nakamura, S. Levy, and W. Y. Wang, ''Fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection,'' in *Proc. 12th Lang. Resour. Eval. Conf.*, 2020, pp. 6149–6157.

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, ''BERT: Pre-training of deep bidirectional transformers for language understanding,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.

[18] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, ''Supervised learning of universal sentence representations from natural language inference data,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 670–680.

[19] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[20] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[21] M. Tan and Q. V. Le, ''EfficientNet: Rethinking model scaling for convolutional neural networks,'' in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[22] S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, and M. Akbar, ''Fake news detection using deep learning models: A novel approach,'' *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3767, Feb. 2020.

[23] J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, and A. Iqbal, ''A benchmark study of machine learning models for online fake news detection,'' *Mach. Learn. Appl.*, vol. 4, Jun. 2021, Art. no. 100032.

[24] M. Granik and V. Mesyura, ''Fake news detection using naive Bayes classifier,'' in *Proc. IEEE 1st Ukraine Conf. Electr. Comput. Eng. (UKRCON)*, May 2017, pp. 900–903.

[25] V. Perez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, ''Automatic detection of fake news,'' in *Proc. 27th Int. Conf. Comput. Linguistics*, 2018, pp. 3391–3401.

[26] D. M. Nguyen, T. H. Do, R. Calderbank, and N. Deligiannis, ''Fake news detection using deep Markov random fields,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 1391–1400.

[27] C.-O. Truică, E.-S. Apostol, and A. Paschke, ''Awakened at CheckThat! 2022: Fake news detection using BiLSTM and sentence transformer,'' in *Proc. Work. Notes Conf. Labs Eval. Forum*, 2022, pp. 749–757.

[28] C.-O. Truică and E.-S. Apostol, ''It's all in the embedding! Fake news detection using document embeddings,'' *Mathematics*, vol. 11, no. 3, p. 508, 2023.

[29] M. Mayank, S. Sharma, and R. Sharma, ''DEAP-FAKED: Knowledge graph based approach for fake news detection,'' 2021, *arXiv:2107.10648*.

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, ''RoBERTa: A robustly optimized BERT pretraining approach,'' 2019, *arXiv:1907.11692*.

[31] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, ''BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,'' in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.

[32] S. Raza and C. Ding, ''Fake news detection based on news content and social contexts: A transformer-based approach,'' *Int. J. Data Sci. Anal.*, vol. 13, no. 4, pp. 335–362, May 2022.

[33] K. Sharma, X. He, S. Seo, and Y. Liu, ''Network inference from a mixture of diffusion models for fake news mitigation,'' in *Proc. 15th Int. AAAI Conf. Web Social Media*, 2021, pp. 668–679.

[34] A. Saxena, H. Saxena, and R. Gera, ''Competitive influence propagation and fake news mitigation in the presence of strong user bias,'' 2020, *arXiv:2011.04857*.

[35] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, ''FakeNewsNet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media,'' *Big Data*, vol. 8, no. 3, pp. 171–188, Jun. 2020.

[36] M. Sayyadiharikandeh, O. Varol, K.-C. Yang, A. Flammini, and F. Menczer, ''Detection of novel social bots by ensembles of specialized classifiers,'' in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2020, pp. 2725–2732.

[37] W. McKinney, ''Data structures for statistical computing in Python,'' in *Proc. Python Sci. Conf.*, 2010, pp. 56–61.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, ''Scikit-learn: Machine learning in Python,'' *J. Mach. Learn. Res.*, vol. 12, no. 10, pp. 2825–2830, Jul. 2017.

[39] S. Bird, E. Loper, and E. Klein, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*. Sebastopol, CA, USA: O'Reilly Media, 2009.

[40] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[41] R. Řehůřek and P. Sojka, ''Software framework for topic modelling with large corpora,'' in *Proc. LREC Workshop New Challenges NLP Frameworks*, 2010, pp. 45–50.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ''Efficient estimation of word representations in vector space,'' in *Proc. 1st Int. Conf. Learn. Represent.*, 2013, pp. 1–12.

[43] J. Pennington, R. Socher, and C. Manning, ''GloVe: Global vectors for word representation,'' in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[44] M. Abadi. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

[45] E. R. Gansner and S. C. North, ''An open graph visualization system and its applications to software engineering,'' *Softw., Pract. Exper.*, vol. 30, no. 11, pp. 1203–1233, 2000.

[46] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah, ''Real-time rumor debunking on Twitter,'' in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2015, pp. 1867–1870.

[47] C. Chen, H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, ''Node immunization on large graphs: Theory and algorithms,'' *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 113–126, Jan. 2016.

[48] A. Petrescu, C.-O. Truică, E.-S. Apostol, and P. Karras, ''SparseShield: Social network immunization vs. harmful speech,'' in *Proc. ACM Conf. Inf. Knowl. Manag.*, 2021, pp. 1426–1436.

[49] Y. Zhang and B. A. Prakash, ''Data-aware vaccine allocation over large networks,'' *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 2, pp. 1–32, Oct. 2015.

CIPRIAN-OCTAVIAN TRUICĂ received the Ph.D. degree in data management and text mining from the University Politehnica of Bucharest, Romania, in 2018. He is currently an Assistant Professor of computer science with the Computer Science and Engineering Department, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania. He held research positions with Uppsala University, Sweden; Aarhus University, Denmark; and Lyon University, France. His current research interests include social media analysis, deep learning, machine learning, graph mining, network immunization, natural language processing, big data analytics, and data management.

ELENA-SIMONA APOSTOL was a Postdoctoral Researcher with Microsoft Research Center in collaboration with the INRIA Institute, where she worked on state-of-the-art big data analysis, multisite cloud computing, and bioinformatics. She also held research positions with the Department of Information Technology, Uppsala University, and the Fraunhofer FOKUS Institute. She is currently an Associate Professor of computer science with the Computer Science and Engineering Department, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Romania. Her current research interests include big data analytics, data science, text and data mining, and cloud computing.

RADU-CĂTĂLIN NICOLESCU received the bachelor's degree in computer engineering from the University Politehnica of Bucharest, Romania, in 2021. He is currently pursuing the M.Sc. degree in advanced software services with the National University of Science and Technology Politehnica Bucharest, Romania. The bachelor's thesis was on the subject of fake news detection using deep learning methods. His current research interests include natural language processing, computer vision, and data mining. He aims to find new approaches and improvements to current challenges in artificial intelligence through his research.

PANAGIOTIS KARRAS received the M.Sc. degree in electrical and computer engineering from the National Technical University of Athens and the Ph.D. degree in computer science from The University of Hong Kong. He is currently a Professor of computer science with the University of Copenhagen. His current research interests include the design of robust, scalable, and versatile methods for data access, mining, analysis, exploration, and representation. He has been awarded the Hong Kong Young Scientist Award, the Singapore Lee Kuan Yew Postdoctoral Fellowship, the Rutgers Business School Teaching Excellence Fellowship, and the Skoltech Best Faculty Performance Award.