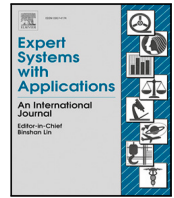




Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

GETAE: Graph Information Enhanced Deep Neural Network Ensemble Architecture for fake news detection

Ciprian-Octavian Truică ^{a,1}, Elena-Simona Apostol ^{a,b,*}, Marius Marogel ^{a,c}, Adrian Paschke ^{d,e}

^a Computer Science and Engineering Department, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Splaiul Independentei 313, Bucharest, 060042, Romania

^b Academy of Romanian Scientists, Ilfov 3, Bucharest, 050044, Romania

^c Department of Computer Science, Faculty of Mathematics and Computer Science, University of Bucharest, Academiei 14, Bucharest, 010014, Romania

^d Department of Mathematics and Computer Science, Freie Universität Berlin, Arnimallee 14, Berlin, 14195, Germany

^e Fraunhofer Institute for Open Communication Systems, Kaiserin-Augusta-Allee 31, Berlin, 10589, Germany

ARTICLE INFO

Keywords:

Fake news detection
Social network analysis
Deep Neural Network Ensemble Architectures
Propagation embeddings
Propagation-Enhanced Content Embedding

ABSTRACT

In today's digital age, fake news has become a major problem with serious consequences, ranging from social unrest to political upheaval. New methods for detecting and mitigating fake news are required to address this issue. In this work, we propose incorporating contextual and network-aware features into the detection process. This involves analyzing not only the content of a news article but also the context in which it was shared and the network of users who shared it, i.e., the information diffusion. Thus, we propose GETAE, Graph Information Enhanced Deep Neural Network Ensemble Architecture for Fake News Detection, a novel ensemble architecture that uses textual content together with the social interactions to improve fake news detection. GETAE contains two Branches: the Text Branch and the Propagation Branch. The Text Branch combines Word and Transformer embeddings with a Deep Neural Network architecture based on feed-forward and bidirectional Recurrent Neural Networks ([Bi]RNN) to capture contextual features and generate a Text Content Embedding. This integrated approach allows for a more comprehensive understanding of the textual information. The Propagation Branch considers the information propagation within the graph network and proposes a Deep Learning architecture that employs Node Embeddings to create novel Propagation Embedding. GETAE's Ensemble module combines the Text Content and Propagation Embeddings, to create a powerful and unique Propagation-Enhanced Content Embedding which is afterward used for classification. The experimental results obtained on two real-world publicly available datasets, i.e., Twitter15 and Twitter16, prove that this approach improves fake news detection and outperforms state-of-the-art models.

1. Introduction

As mass media becomes more digitalized, new journalistic models for information dissemination have emerged. These models have significantly transformed how society consumes news and information. In the race to stay ahead of competitors, journalists sometimes sacrifice the standards of traditional journalism, prioritizing speed and aiming to “go viral” by quickly generating views aiming for an increase in likes, comments, and shares. This new approach focuses on catering to users' needs, behaviors, and interests. While digital media offers many benefits, it also heightens the risk of misinformation (Mustafaraj

& Metaxas, 2017; Ruths, 2019), which can have harmful consequences for society by enabling the rapid spread of false information, such as fake news related to the Brexit referendum (Bastos & Mercea, 2017), the 2016 US presidential election (Bovet & Makse, 2019), COVID-19 vaccinations (Rzyski et al., 2021), and war in Ukraine (Marushchak, Petrov, & Khoperiya, 2025).

The current research explores models for detecting fake news, which are trained using either traditional machine learning or deep learning techniques on large datasets of news articles. New approaches also incorporate additional metadata (Zhang & Ghorbani, 2020), such as

* Corresponding author at: Computer Science and Engineering Department, Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica Bucharest, Splaiul Independentei 313, Bucharest, 060042, Romania.

E-mail addresses: ciprian.truica@upb.ro (C.-O. Truică), elena.apostol@upb.ro (E.-S. Apostol), marius.marogel@stud.acs.upb.ro (M. Marogel), paschke@inf.fu-berlin.de (A. Paschke).

¹ These authors contributed equally to this article.

<https://doi.org/10.1016/j.eswa.2025.126984>

Received 18 December 2024; Received in revised form 7 February 2025; Accepted 18 February 2025

Available online 28 February 2025

0957-4174/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

citation sources, verifiable authors, short paragraphs, third-person narration, and other factors that can enhance the accuracy of fake news detection models. However, training on such data can have drawbacks, especially when the model encounters new data that does not adhere to the same writing conventions. For example, social media content is typically more personal and informal (Aïmeur, Amri, & Brassard, 2023). This is why integrating extra information is crucial when developing fake news detection systems for social media. Various methods (Truică, Apostol and Karras, 2024) exist for representing features in this context, which can be divided into news-content features (e.g., headlines, text, images, or linguistic/visual elements) and social context features (e.g., user, post, or network-based data). A major shortcoming in the current literature is that few current methods for analyzing social media data incorporate information propagation for the detection task.

To address this major shortcoming in the current literature, we propose GETAE a novel Graph Information Enhanced Deep Neural Network Ensemble Architecture for Fake News Detection. Our approach utilizes social network information stored as a graph, creating node embeddings with advanced algorithms. These embeddings, combined with textual content, are then fed into feed-forward or bidirectional recurrent neural networks. This enables our model to leverage both contextual and network-aware data for fake news detection.

The main objective of this work is to design and implement the novel Graph Information Enhanced Deep Neural Network Ensemble Architecture for Fake News Detection, i.e., GETAE. GETAE uses textual content and social interactions to improve fake news detection. Our secondary objectives are fivefold:

- (1) Design a novel Text Content Embedding, a new vector representation for textual that combines multiple complex textual features such as lexical (e.g., character and word level features) and syntactic (e.g., sentence-level features) to improve prediction.
- (2) Create a new Propagation Embedding designed to encode information diffusion, i.e., the spread of information from a node to its followers.
- (3) Propose an innovative Propagation-Enhanced Content Embedding that encapsulates textual content, contextual data, and propagation information to improve detection performance.
- (4) Benchmark GETAE by training multiple models using cross-validation, ablation testing, and hyperparameter tuning on two real-world publicly available datasets, i.e., Twitter15 and Twitter16.
- (5) Compare the best GETAE models with current state-of-the-art models.

The main research questions we want to address with this work are:

- (Q1) How much dataset-reliant is the choice of models and algorithms for Fake News Detection?
- (Q2) What is the impact of word embeddings that combine multiple complex textual features such as lexical (e.g., character and word level features) and syntactic (e.g., sentence-level features), on the performance of Fake News Detection systems?
- (Q3) What is the impact of employing node embeddings that encode the information diffusion through social networks on the effectiveness of Fake News Detection methods?
- (Q4) What is the impact of embeddings that consider both the information diffusion and the textual content and context on the performance of Fake News Detection systems?
- (Q5) How are these models performing in comparison with state-of-the-art Fake News Detection models?

This article is structured as follows. In Section 2, we briefly present the current state-of-the-art. Section 3 presents the methodology for preprocessing the data and designing the GETAE architecture. In Section 4, we present ablation, cross-validation, and hyperparameter tuning for GETAE as well as a comparison with the state-of-the-art models. Section 5 discusses our findings and the limitations of our architecture. Finally, Section 6 concludes this work and hints at future research directions.

2. Related work

Textual data representation is a challenging task when it comes to generating linguistic features for the task of fake news detection (Broda & Strömbäck, 2024; Melchior & Oliveira, 2023; Singh, Ahmed, Alam, Raghuvanshi, & Kumar, 2023). Shu, Sliva, Wang, Tang, and Liu (2017) discuss the use of linguistic and semantic features such as n-grams, total word count, and punctuation that can provide meaningful representations for fake news detection. Ahmad, Yousaf, Yousaf, and Ahmad (2020) applied ensemble learning methods like Random Forests, Boosting, Bagging, and Voting Classifiers on the LIWC (Linguistic Inquiry and Word Count) dataset (Pennebaker, Boyd, Jordan, & Blackburn, 2015), achieving higher accuracy than traditional models across four datasets. N-gram models for feature extraction also yielded strong results (Khan, Khondaker, Afroz, Uddin, & Iqbal, 2021; Wynne & Wint, 2019). These approaches use classical machine learning algorithms with TF-IDF (Term Frequency-Inverse Document Frequency) for n-gram extraction to train models.

The current literature on detecting harmful content primarily focuses on techniques such as word embeddings (Ilie, Truică, Apostol, & Paschke, 2021), character-level embeddings (Khan et al., 2021), transformer embeddings (Petrescu, 2023; Truică & Apostol, 2022), sentence transformers (Truică, Apostol, & Paschke, 2022), and document embeddings (Truică & Apostol, 2023). Various deep learning architectures are used as classifiers for harmful content detection, with most state-of-the-art models relying on LSTMs (Long Short-Term Memory networks), GRUs (Gated Recurrent Units), and/or CNNs (Convolutional Neural Networks) as foundational components, as seen in studies such as (Ilie et al., 2021; Truică & Apostol, 2022).

Other effective linguistic approaches for extracting features for fake news detection include using contextual inputs such as the headline, first two sentences of a news article, and news content to generate representations (Borges, Martins, & Calado, 2019). These features used for training models that employ BiLSTM (Bidirectional Long Short-Term Memory) and BiGRU (Bidirectional Gated Recurrent Unit), manage to capture long-term dependencies in sequences. Other approaches use Hierarchical Recursive Neural Networks and transformer-based architecture to construct a linguistic tree of news content using an encoder-decoder model (Zhou, Li, Li, & Zafarani, 2023). This linguistic tree is then processed by a series of BiGRU layers within the network.

EvolveDetector (Ding et al., 2025) is a novel method that uses parameter-level knowledge transfer to detect fake news without storing historical data, leveraging a knowledge memory and event masks for efficient learning. This model retrieves neuron parameters of similar past events to guide new event learning and integrates feature outputs using multi-head self-attention to train a classifier. Ember (emulating the behaviors of readers) (Yin et al., 2025) is a fake news detection model that incorporates user behavior in news consumption and authenticity verification. To achieve this, Ember emulates readers' reading and verification behaviors by considering the news articles' components, i.e., headlines, images, comments, and content. QMFND (Qu, Meng, Muhammad, & Tiwari, 2024) is a quantum multimodal fusion-based model that integrates image and text features, utilizing a quantum convolutional neural network (QCNN) for enhanced detection. Tested on Gossip and Politifact datasets, QMFND matches or outperforms classical models, demonstrating strong expressibility, entangling capability, and robustness against quantum noise. NSEP Framework (Fang et al., 2024) is a novel approach that detects fake news by analyzing macro and micro semantic environments using a distinguisher module, graph convolutional networks, and multihead attention. Empirical experiments on Chinese and English datasets show that NSEP outperforms baseline models, proving its effectiveness of analyzing both news content and posts for early misinformation detection.

Graph Neural Networks (GNN) use node-oriented data to capture and encode information from networks, unlike classical deep learning models that operate on sequential data. Understanding local and global

features in graphs can help solve a great number of problems in areas like biological or social networks. Propagation-oriented methods use the networks to understand how features change when information is spread into a network.

Lu and Li (2020) propose the graph-aware co-attention network, GCAN, a model that takes the user features in a propagation system and uses a GCN (Graph Convolutional Network) to learn representations for rumor detection. Using a retweet order sorted based on the retweet time, they create GCN, GRU, and CNN embeddings for the tweet and social encodings, which are then passed through a dual co-attention mechanism. This approach uses features from patterns in response time, source tweet content, and user characteristics on Twitter15 and Twitter16 datasets.

Huang, Zhou, Wu, Wang, and Wang (2019) use a deep structure learning method for rumor detection on the same Twitter-based datasets. The authors create encoders for user data and propagation data with layers of GCN and RNN (Recursive Neural Networks), which are fed into an integrator module that uses both encoders to create representations of both user and propagation-tree data. On rumor detection using semantic and propagation information, Ke et al. (2020) propose KZWANG, a model that uses Multihead Attention mechanisms to generate a better representation for microblogs and GCN layers to fuse microblog features with propagation encoding. This approach was tested on three datasets, surpassing other state-of-the-art models in accuracy and F1-score for multi-class classification.

Topological information of a graph can be stored in a vector space, in the same way word embeddings are represented. High-dimensional vector representations for nodes, edges, and whole graphs are noted as node, edge, or graph embeddings. Goyal and Ferrara (2018) and Cai, Zheng, and Chang (2018) use graph embeddings that encode node, edge, and whole graph information for solving different problems. Xie, Huang, Xie, and Jiang (2020) use the GraphSage model for node embeddings, the Multihead Attention mechanism for creating user-based representations, and CNNs for content representations. These layers are then concatenated for a binary classifier which gets good results on BuzzFeed and PolitiFact datasets, compared to other embedding techniques. Imaduwage, Kumara, and Samaraweera (2022) propose a method for combining local and global graph-level embeddings. Each propagation tree contributes only partially with half priority with its local GraphSage embeddings to a global embedding mechanism. These are fed to an autoencoder-based network that generates whole propagation-tree embeddings used then in a classification model.

When dealing with information diffusion on networks, the current literature proposes multiple solutions that analyze the spread of online content (Petrescu, Truică, & Apostol, 2019; Petrescu, Truică, Apostol and Paschke, 2024; Truică, Apostol, Ștefu, & Karras, 2021) for both detection and network immunization tasks, such as preemptive (Petrescu, Truică, Apostol, & Karras, 2021), community-based (Apostol, Coban and Truică, 2024), or tree-based (Truică, Apostol, Nicolescu & Karras, 2023) strategies. Furthermore, the current literature also presents full solutions for real-time Fake News Detection. STOPHC (Truică, Constantinescu and Apostol, 2024) is a full solution that builds on top of the current literature and proposes a novel deep-learning architecture for harmful content detection and mitigation. CONTCOMMRTD (Apostol, Truică and Paschke, 2024) is an architecture that detects misinformation on social media posts during real-time disaster reporting. Petrescu, Truică and Apostol (2024) proposes a mixture of models to deal with harmful content detection.

3. Methodology

3.1. Feature representation

3.1.1. Text preprocessing and word embeddings

To preserve semantic relations while removing any elements that do not contribute to extracting the context (e.g., hyperlinks, utf8

characters, etc.), we process the textual data using a preprocessing pipeline. We apply this preprocessing pipeline to reduce the vocabulary size (Truică, Darmont, & Velcin, 2016) before training while increasing the model's performance during inference (Truica, Guille, & Gauthier, 2016). By standardizing the textual data, we allow the model to focus on the core content and generalize better when unseen data is used for inference (Truică, Apostol, & Leordeanu, 2017). The preprocessing pipeline involves applying the following steps on the textual data (Truică, Apostol, Șerban & Paschke, 2021; Truică et al., 2016; Truică & Leordeanu, 2017): (1) remove URL; (2) remove punctuation; (3) remove stopwords (only for the Word2Vec model); and (4) assign each token a unique integer identifier (id) starting with id 1.

We reconstruct the textual data using the tokens' unique identifiers and obtain a 2-dimensional document-to-token matrix $D \in \mathbb{N}^{n \times k}$, where n is the number of documents in the corpus and k is the maximum size of a document in the corpus. We use left zero padding for documents that are smaller than the maximum size k , i.e., we add 0 on the left side of the document to achieve the maximum length k .

Using the tokenized text, we create 3 word embeddings:

- (1) WORD2VEC SKIP-GRAM (Mikolov, Chen, Corrado, & Dean, 2013), a model that takes as input the word w and tries to detect the context (the words on the left and right sides of w);
- (2) BERT (Devlin, Chang, Lee, & Toutanova, 2019) is a deep bidirectional transformer architecture that employs transfer learning to generalize language understanding;
- (3) BERTTWEET (Nguyen, Vu, & Tuan Nguyen, 2020) is a pre-trained language model for Twitter data that uses BERT's architecture and the pre-training procedure of RoBERTa.

We train our own WORD2VEC embeddings and we use the pre-trained BERT and BERTTWEET transformer embeddings. Using these embeddings, we manage to better capture the relationships between tokens and encode local and global contexts (Truică, Apostol, Șerban et al., 2021).

We create a 2-dimensional matrix $W \in \mathbb{R}^{m \times s}$ that stores for each unique token its word embedding, where m is the size of the vocabulary (i.e., the number of unique tokens in the corpus plus the padding element) and s is the size of the word embedding (i.e., the number of the embedding components). The word embedding for the padding element is a vector that contains only the value 0.

3.1.2. Graph representation and node embeddings

To construct a graph from a social network dataset, we use the users as nodes and the posting and user interaction with a post (i.e., re-sharing, linking, commenting, etc.) as edges. Using this graph, we extract the information diffusion propagation graph $G = (V, E)$ of each node $u \in V$ to construct node embedding. The diffusion is given by the edges $(u, v) \in E$ that connect 2 nodes $u, v \in V$. For building node embeddings, we use NODE2VEC (Grover & Leskovec, 2016) and DEEPWALK (Perozzi, Al-Rfou, & Skiena, 2014). Both methods aim to capture the structural properties of the network and represent them in a low-dimensional space. NODE2VEC is an extension of DEEPWALK that uses a biased random walk to sample nodes in the network, which allows for the capture of both local and global network structures.

Specifically, NODE2VEC uses a bias factor $\alpha_{p,q}$, and two parameters p (return parameter) and q (in-out parameter). Eq. (1) presents the calculation of the bias coefficient $\alpha_{p,q}$ when traversing a graph with a second-order random walk for the current node $u \in V$ by traversing edge $(u, v) \in E$, where $d_{u,v} \in \{0, 1, 2\}$ is the length of path of the edge $(u, v) \in E$.

$$\alpha_{p,q} = \begin{cases} 1/p, & \text{if } d_{u,v} = 0 \\ 1, & \text{if } d_{u,v} = 1 \\ 1/q, & \text{if } d_{u,v} = 2 \end{cases} \quad (1)$$

Thus, the control of parameters (p, q) influences the sampling decision of nodes that are more likely to produce similar embeddings. The NODE2VEC model uses a Skip-Gram with Negative Sampling (SGNS)

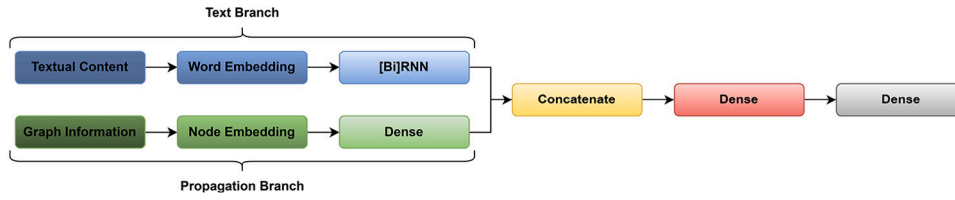


Fig. 1. GETAE architecture.

approach to generate embeddings that are trained to maximize the likelihood \mathcal{L} of predicting the neighborhood $p(N_s(u) | f(u))$ of a node given its embedding (Eq. (2)), where $N_s(u)$ is the set of nodes sampled with strategy s in the neighborhood of $u \in V$ and $f(u)$ is the mapping function from nodes to representations.

$$\mathcal{L} = \sum_{u \in V} \log p(N_s(u) | f(u)), \quad (2)$$

DEEPWALK uses a random walk to generate sequences of nodes and then applies the Skip-Gram model to learn node embeddings that capture the co-occurrence statistics of the nodes in the sequences. DEEPWALK is based on the idea that nodes that appear in similar contexts in the network are likely to have similar roles and functions, and therefore can be represented by similar embeddings. Eq. (3) presents the objective function \mathcal{L} that we try to maximize when creating embeddings with DEEPWALK, where $N(u)$ is the set of nodes within the k -length random walk of node $u \in V$, and $f(u)$ and $f(v)$ are the embeddings for nodes $u, v \in V$. In this case, the objective function \mathcal{L} uses the *softmax* function on the dot product of $f(u)$ and $f(v)$ to normalize the output in the interval $[0, 1]$.

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N(u)} \log \text{softmax}(f(u) \cdot f(v)), \quad (3)$$

Both objective functions are optimized using stochastic gradient descent (SGD) with negative sampling.

For each node embedding model, we create a 2-dimensional matrix $N \in \mathbb{R}^{|V| \times s}$ that stores the embedding vector of each node, where $|V|$ is the number of nodes in the graph and s is the size of the node embedding (i.e., the number of the embedding components).

3.2. GETAE architecture

3.2.1. Text branch

GETAE's Text Branch (see Fig. 1) contains 3 layers: (1) the input layer (i.e., Textual Content), (2) the Word Embedding layer, and (3) a hidden layer (i.e., [Bi]RNN). This particular combination allows us to effectively capture both semantic and contextual information from the text, which is essential for accurate fake news detection. The Textual Content layer requires 2 inputs: (1) the document-to-token matrix $D \in \mathbb{N}^{n \times k}$ and (2) the word embedding matrix $W \in \mathbb{R}^{m \times s}$. The Word Embedding layer is used to pair document tokens from D to their corresponding word embeddings from W . The hidden [Bi]RNN layer employs either Unidirectional Recurrent Neural Network (RNN) units or Bidirectional Recurrent Network (BiRNN) units. In GETAE's implementation, the RNN is replaced by either Recurrent Neural Networks (RNN) units, Long Short-Term Memory (LSTM) units, or Gated Recurrent Units (GRU) for ablation testing. We use the [Bi]RNN notation when discussing this layer, regardless if we use the RNN or the BiRNN layer. Thus, when using an RNN hidden layer, the actual layer units employed by GETAE are either standard RNN, LSTM, or GRU. When using a BiRNN hidden layer, the layer units are either standard BiRNN, BiLSTM, or BiGRU. During ablation testing, we interchange unidirectional with bidirectional RNNs. Thus, we aim to determine the importance of preserving the information from inputs when the information passes through the hidden state in a feed-forward manner (the unidirectional case) versus when the information is passed in both directions using a backward and forward approach (the bidirectional case). The output of the [Bi]RNN hidden layer is a new word

representation, i.e., Text Content Embedding. The novel Text Content Embeddings combines multiple complex textual features such as lexical (e.g., character and word level features) and syntactic (e.g., sentence level features) used to improve prediction.

As we employ multiple units for the [Bi]RNN hidden layer, the Text Content Embeddings manage to create different types of embeddings that also consider the input Word Embedding. This allows us to generate a diverse set of text representations, each capturing different nuances of the input word embeddings. This diversity is crucial for robust fake news detection, as different recurrent units may be better suited to capturing specific linguistic patterns or contextual relationships. Regardless of the word embedding used as input, we create 6 subtypes of Text Embeddings using [Bi]RNN: RNN-based, BiRNN-based, GRU-based, BiGRU-based, LSTM-based, and BiLSTM-based Embeddings. For each Text Embedding subtype, the word embedding passes through the corresponding recurrent layer to obtain the final text embedding.

3.2.2. Propagation branch

GETAE's Propagation Branch (see Fig. 1) contains 3 layers: (1) the input layer (i.e., Graph Information), (2) Node Embedding layer, and (3) a hidden layer (i.e., Dense). The Graph Information layer requires 2 inputs: (1) list of nodes V and (2) the node embedding matrix $N \in \mathbb{R}^{|V| \times s}$. The Node Embedding layer is used to pair nodes from V to their corresponding node embeddings from N . A hidden layer uses the pairs to obtain a novel Propagation Embedding. This hidden layer contains Dense units that employ the *ReLU* activation function to create a new Propagation Embedding. The novel Propagation Embedding is designed to encode the information diffusion in order to encode the spread of information from a node to its followers.

3.2.3. Ensemble

GETAE's Ensemble concatenates the output of the Text and Propagation Branches into one tensor, which is passed to a Dense layer used for classification. During the concatenation, we create a new *Propagation-Enhanced Content Embedding* that takes into account both the textual content and the information propagation within the social media graph, i.e., it combines the Text Content Embedding with the Propagation Embeddings. This novel *Propagation-Enhanced Content Embedding* is then used by a Dense layer that extracts hidden contextual features and creates a new vector representation by employing the *ReLU* activation function. The new vector representation is passed through the final Dense layer for classification which uses the *softmax* activation function to determine the veracity of a post.

3.2.4. Algorithm

Algorithm 1 presents the pseudo-code for GETAE. As input, GETAE takes the dataset $X = \{\langle t_i, v_i, c_i \rangle | i = \overline{1, n}\}$, the graph $G = (V, E)$, the configuration for the Text Branch given as a pair $\langle bidirection, recurrent \rangle$, the word embedding model for the Text Branch we_model , and the node embedding model for the Propagation Branch ne_model . The dataset X contains for each observation $i = \overline{1, n}$ ($n = |X|$) a tuple $\langle t_i, v_i, c_i \rangle$, where t_i is the textual content, v_i is the author of the content, i.e., node, and c_i is the class. The graph G contains the list of nodes V and the list of edges E used to create the propagation graph of each node. The configuration for the Text Branch determines the types of units used by the [Bi]RNN and if it is unidirectional or bidirectional as follows: (1) *bidirection* is a boolean that indicates the use (True)

Algorithm 1: GETAE

```

Input : dataset  $X = \{ \langle t_i, v_i, c_i \rangle \mid i = \overline{1, n} \}$ 
         the graph  $G = \{V, E\}$ 
         configuration Text Branch  $\langle bidirection, recurrent \rangle$ 
         word embedding model for the Text Branch  $we\_model$ 
         node embedding model for the Propagation Branch  $ne\_model$ 

Output: GETAE model  $m$ 

1  $T \leftarrow \emptyset$  // text list
2  $V \leftarrow \emptyset$  // nodes list
3  $C \leftarrow \emptyset$  // label list
4 foreach  $\langle t, v, c \rangle \in X$  do
5    $T \leftarrow T \cup \{t\}$  // populate the text list
6    $V \leftarrow V \cup \{v\}$  // populate the node list
7    $C \leftarrow C \cup \{c\}$  // populate the label list
8  $D \leftarrow getDocument2Token(T)$  // get document to token matrix
9  $W \leftarrow getWordEmbedding(T, we\_model)$  // train the word embedding
10  $N \leftarrow getNodeEmbedding(G, V, ne\_model)$  // train the node embedding
    // ===== Text Branch =====
11  $i\_text \leftarrow InputLayer(D, W)$  // input for the Text Branch
12  $text\_branch.addLayer(i\_text)$  // Text Branch implementation
13  $text\_branch.addLayer(WordEmbedding)$  // Word Embedding layer
    // ===== Text Branch [Bi]RNN Layer =====
14 if  $bidirection = True$  then
15   if  $recurrent = RNN$  then
16      $text\_branch.addLayer(BiRNN())$ 
17   else if  $recurrent = LSTM$  then
18      $text\_branch.addLayer(BiLSTM())$ 
19   else
20      $text\_branch.addLayer(BiGRU())$ 
21 else
22   if  $recurrent = RNN$  then
23      $text\_branch.addLayer(RNN())$ 
24   else if  $recurrent = LSTM$  then
25      $text\_branch.addLayer(LSTM())$ 
26   else
27      $text\_branch.addLayer(GRU())$ 
    // ===== Propagation Branch =====
28  $i\_propagation \leftarrow InputLayer(V, N)$  // input for the Propagation Branch
29  $propagation\_branch.addLayer(i\_propagation)$  // Propagation Branch implementation
30  $propagation\_branch.addLayer(NodeEmbedding())$  // Node Embedding layer
31  $propagation\_branch.addLayer(Dense(ReLU))$  // Dense layer
    // ===== Ensemble model and training =====
32  $ensemble \leftarrow Concatenate(text\_branch, propagation\_branch)$  // concatenation layer
33  $ensemble.addLayer(Dense(ReLU))$  // Propagation-Enhanced Content Embedding
34  $o\_layer \leftarrow ensemble.addLayer(Dense(softmax))$  // model output
35  $m \leftarrow Model(inputs = [i\_text, i\_propagation], output = o\_layer)$  // model definition
36  $m.train([(D, W), (V, N)], c)$  // model training
37 return  $m$ 

```

or not (False) of a Bidirectional RNN, and (2) *recurrent* is a Layer object that indicates what units are used, i.e., standard RNN, LSTM, or GRU. The *we_model* determines which word embedding model is used, i.e., WORD2VEC, BERT, or BERTWEET. Finally, *ne_model* determines which node embedding model is used, i.e., NODE2VEC or DEEPWALK.

In Lines 1–7, we initialize and populate the T , V , and C lists where we store the text t_i , the nodes v_i , and the class c_i for each entry in the dataset X . After iterating through each record in X , the lists will look as follows: $T = \{t_i \mid i = \overline{1, n}\}$, $V = \{v_i \mid i = \overline{1, n}\}$, and $C = \{c_i \mid i = \overline{1, n}\}$. The list T is used to obtain the document-to-token matrix D (Line 8) and the word embedding matrix W (Line 9). The function *getWordEmbedding*(T, we_model) uses the *we_model* parameter to determine which word embedding model to use. The list V is used to obtain the node embedding matrix N (Line 10). The function *getNodeEmbedding*(G, V, ne_model) uses the Graph and the *ne_model* parameter to determine which node embedding model to use. We note that the architecture is not limited to using only the presented word and node embeddings, as this is a modular architecture, any word and node embedding can be used.

The Text Branch is configured using the configuration pair $\langle bidirection, recurrent \rangle$ (Lines 11–27), as explained in Section 3.2.1. The input layer (Line 11) receives the document-to-token matrix D and the word embedding matrix W . This layer together with the Word Embedding layer is added to the Text Branch (Lines 12–13). The [Bi]RNN layer is constructed in Lines 14 to 27. Depending on parameter *bidirection*, the architecture uses either a feed-forward or a bidirectional RNN layer. The type of units used by GETAE's RNN layer is selected based on the *recurrent* parameter from the following types: standard RNN, LSTM, or GRU.

The Propagation Branch is created similarly to the Text Branch. Lines 28–31 present the layers of the Propagation Branch, as explained in Section 3.2.2. The input layer (Line 29) receives the nodes list V and the node embedding matrix N . This layer together with the Node Embedding layer is added to the Propagation Branch (Lines 29–30). A Dense layer with the *ReLU* activation function is added to the Propagation Branch (Line 31) to obtain the Propagation Embedding.

The ensemble is created by concatenating the Text and the Propagation Branches (Line 32). The new Propagation-Enhanced Content

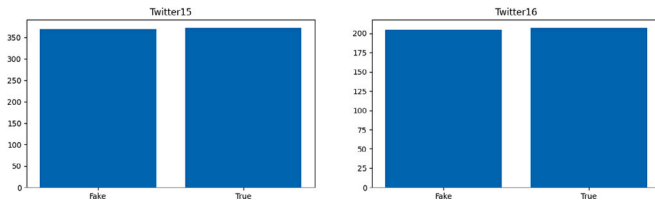


Fig. 2. Class distribution.

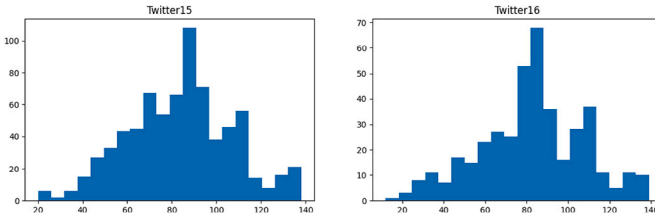


Fig. 3. Length distribution.

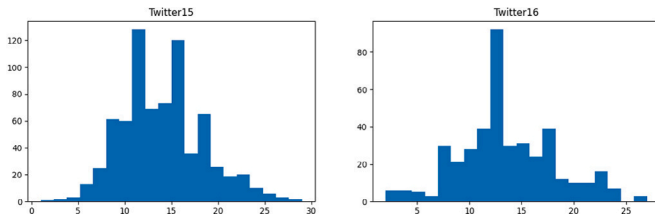


Fig. 4. Number of words distribution.

Embedding is created by adding a Dense layer with the *ReLU* activation function (Line 33). The output layer (Line 34) uses as input the Propagation-Enhanced Content Embedding and a Dense layer with the *softmax* activation function to obtain the final prediction. The model is defined using as input the input layers from the Text and Propagation Branches (Line 35) and as output the last Dense layer (Line 34). Finally, the model is trained (Line 36) and returned (Line 37). Section 3.2.3 explains in detail the GETAE's Ensemble.

4. Experimental results

4.1. Datasets

We use *Twitter15* and *Twitter16* datasets (Ma, Gao and Wong, 2017) for our experimental validation. Both datasets contain Twitter data separated into: (1) the *source tweets* with the tweet ID and its text content, (2) the *label* that assigns a class to each tweet ID, and (3) the *propagation tree* of each tweet.

4.1.1. Textual data analysis

Both datasets are labeled using 4 classes: fake, true, non-rumor, and unverified. Because we are building binary classification models, we only used the source tweets labeled as *true* and *fake* for training. Fig. 2 presents the class distribution for these two labels. We observe that both datasets are fairly balanced.

Fig. 3 represents the length distribution of tweets in both datasets. We observe that the most probable length is around 80–100 characters. Fig. 4 presents the histogram representing the frequency of the number of words. Both datasets have most tweets in 10–15 words range and a maximum word count under 30 words. This highlights the nature of tweets as short texts with several words and a low number of characters.

As both datasets are balanced, we use appropriate evaluation metrics (Truică & Leordeanu, 2017) to evaluate the trained models, i.e., Accuracy, Precision, Recall, and F1-Score.

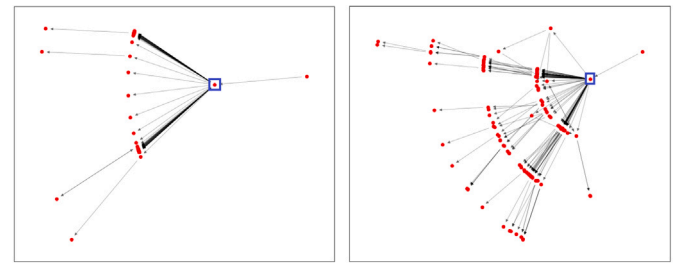


Fig. 5. Examples of propagation trees.

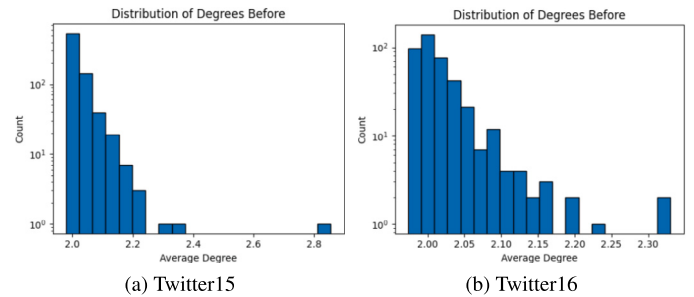


Fig. 6. Degree distribution.

4.1.2. Network data analysis

For both *Twitter15* and *Twitter16* datasets, each propagation tree is encoded using a list of edges from parent nodes to child nodes. The relationship between a child and a parent node represents retweets and mentions of the source tweet, only for the neighbor of the root node. Fig. 5 presents some examples of propagation trees to better visualize the actual information spread. These propagation trees are used to build the *Node2Vec* and *DeepWalk* embeddings employed by GETAE to create the *Propagation-Enhanced Content Embedding*.

The graphs are directed with the orientation specified in the dataset. Every row in the edge file is a connection from a parent node to a child node. Nodes are represented as tuples with three components: the $user_{id}$ of the Twitter account, the $tweet_{id}$ of the tweet post, and the $time$ in minutes passed since the first node posted (source node). Eq. (4) shows the edge information from the tree file:

$$(parent_{id}, tweet_{id}, t_{tweet}) \rightarrow (child_{id}, reply_{id}, t_{reply}) \quad (4)$$

Fig. 6 presents the graphs' Degree Distributions using histograms with a logarithmic scale. For the *Twitter15* dataset, the total number of graphs is 742 and we compute the average degree for each of these graphs. We observe that most graphs have a degree distribution around 2, with some of them even over 2.8. The same behavior is present in the *Twitter16* dataset as well, where we observe that most graphs have an average distribution degree over 2.

Fig. 7 presents plots that represent the average Degree Centrality, to measure the immediate risk for infection for a node by counting their neighbors. We observe that both datasets have similar centrality degrees, thus, the nodes have a similar number of neighbors.

4.2. Experimental setup and implementation

We conducted experiments on *Twitter15* and *Twitter16* datasets creating word embeddings, node embeddings, and deep learning models that combine this data. The text preprocessing for training word embeddings consists of removing (1) URLs, (2) double spaces, and (3) punctuations. To vectorize the text, we train a *Wor2Vec* Skip-Gram model. Table 1 specifies the hyperparameters used for obtaining node embeddings with *Node2Vec* and *DeepWalk*. For BERT and BERTweet, we use the pre-trained models from HuggingFace: 'ber-base-uncased' and

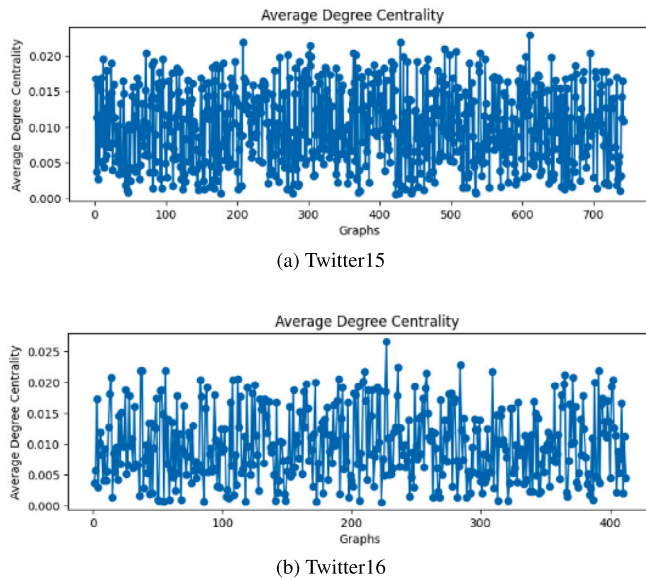


Fig. 7. Degree centrality.

Table 1
Embeddings hyperparameters.

Embeddings	Window	Min count	Walk length	Learning rate	Epochs
WORD2VEC	10	4	–	0.025	5
NODE2VEC	10	1	10	0.05	1
DEEPWALK	5	1	10	0.05	1

Table 2
BERT hyperparameters.

Tokenizer	add_special_tokens	Padding	Truncation	max_length
BERT	TRUE	TRUE	TRUE	128
BERTWEET	TRUE	TRUE	TRUE	128

Table 3
GETAE's layers hyperparameters.

Layer	Units	Dropout	Activation function
RNN	64	0.2	Tanh (Hyperbolic tangent)
BiRNN	128	0.2	Tanh (Hyperbolic tangent)
GRU	64	0.2	Tanh (Hyperbolic tangent)
BiGRU	128	0.2	Tanh (Hyperbolic tangent)
LSTM	64	0.2	Tanh (Hyperbolic tangent)
BiLSTM	128	0.2	Tanh (Hyperbolic tangent)
Text dense	32	–	ReLU (Rectified linear unit)
Graph dense	32	–	ReLU (Rectified linear unit)
Output dense	1	–	Sigmoid

'vinai/bertweet-base'. For Transformer embeddings, we applied the tokenizer with parameters presented in Table 2. We pass the input_ids and attention_mask tensors to the BERT models to obtain the transformer embeddings. For BERTweet, the only difference was loading a different tokenizer and model from huggingface. Table 3 presents GETAE's deep learning ensemble architecture hyperparameters.

We implemented the preprocessing elements of GETAE in Python version 3.10. For text preprocessing, we use the `nlTK` (Bird, 2009) package. For training the WORD2VEC word embeddings, we use the `gensim` (Řehůřek & Sojka, 2010) package. For training the node embeddings, we use the `node2vec` and `karateclub` (Rozemberczki, Kiss, & Sarkar, 2020) packages for training the embeddings with NODE2VEC and DEEPWALK, respectively. We implemented the GETAE architecture and trained the models using TensorFlow and Keras. The code is freely available on GitHub at <https://github.com/DS4AI-UPB/GETAE>.

4.3. Classification and ablation results

For this set of experiments, we use an 80–20 train–test ratio. We trained the multiple models using the GETAE architecture for 30 epochs, except for the models that employ the WORD2VEC word embeddings which were trained for only 8 epochs to avoid overfitting. For training the different GETAE models, we used a learning rate of 0.001. The training process for the models was carried out on Google Colab, with no hardware acceleration.

All experiments use k-fold cross-validation with $k = 10$ with the ratio 80-20 for the train and test sets. The tables present the mean and standard deviation for each metric, using the individual results obtained at each fold during the k-fold cross-validation.

Tables 4 and 5 present the results obtained on the test set after training the GETAE models on Twitter15 and Twitter16, respectively. We use three types of word embeddings (i.e., WORD2VEC Skip-Gram, BERT, and BERTWEET) and two types of node embeddings (i.e., NODE2VEC and DEEPWALK). These tables also present two types of ablation testing for the GETAE architecture: (1) adding or removing the Propagation Branch, (2) changing the RNN Layer for the Text Branch. The Text Branch uses six different recurrent neural network layers for ablation testing: RNN, BiRNN, GRU, BiGRU, LSTM, and BiLSTM.

The evaluation of the models obtained by training the GETAE architecture includes performance metrics such as Accuracy, Precision, Recall, and F1-Score, measured on both Twitter15 and Twitter16 datasets. The experiments are conducted with and without the Network Branch for ablation testing. The hyperparameters for NODE2VEC are: $p = 1$, $q = 1$, $d = 100$, where p is the return hyperparameter, q is in–out hyperparameter, and d is the embeddings' dimensions. For DEEPWALK node embeddings, the only hyperparameter tuned was the dimensionality of the embedding space, set to $d = 100$.

For the Twitter15 dataset, the best configuration for GETAE by metrics is a tie between configurations (BERT, NODE2VEC, BiLSTM) and (BERTWEET, DEEPWALK, BiRNN). When comparing these two models with other state-of-the-art ones, we choose the (BERT, NODE2VEC, BiLSTM) configuration due to a higher F1-Score. BERTWEET should also be mentioned as having promising results, especially when employing the DEEPWALK node embedding and BiRNN as the recurrent layer.

For the Twitter16, the best configuration for GETAE is (BERT, NODE2VEC, RNN) with 89.6% Accuracy, 90.1% Precision, 89.7% Recall, and 89.5% F1-Score. The second best configuration for GETAE is a tie between (BERT, NODE2VEC, BiRNN) and (BERT, NODE2VEC, BiLSTM) with a difference in metrics smaller than 1%. We observe that for the Twitter16 dataset, the model that employs BERTWEET for the word embeddings only performs better when using node embeddings and a BiRNN layer.

This set of experiments shows that the fake news detection process is enhanced by leveraging users' content and their interaction within a social network and encoding them within the novel Propagation-Enhanced Content Embedding.

4.4. Node embeddings hyperparameter tuning

For this set of tests, we use the same methodology when training the models using the GETAE as in Section 4.3.

4.4.1. NODE2VEC

Tables 6 and 7 present the detection results when varying Node2Vec parameters p (return hyperparameter), q (in–out hyperparameter), and d (dimensions) for the GETAE architecture. The return parameter p controls the likelihood of immediately revisiting a node in the walk, while the in–out hyperparameter q allows the search to differentiate between *inward* and *outward* nodes (Grover & Leskovec, 2016). For each of these configurations of parameters, we use the following configurations for GETAE: (1) (BERT, BiLSTM, NODE2VEC) for the Twitter15

Table 4
GETAE ablation testing on Twitter15 (Node2Vec $p = 1, q = 1$, dimensions = 100 for both Node2Vec and DeepWalk.).

Word embedding	Network embedding	Text branch	Propagation branch	RNN layer				BiRNN layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.749 ± 0.039 0.651 ± 0.028 0.705 ± 0.025	0.764 ± 0.036 0.683 ± 0.025 0.720 ± 0.023	0.750 ± 0.042 0.660 ± 0.027 0.711 ± 0.023	0.745 ± 0.043 0.642 ± 0.041 0.703 ± 0.025
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.790 ± 0.036 0.766 ± 0.070 0.736 ± 0.063	0.791 ± 0.034 0.800 ± 0.035 0.773 ± 0.044	0.788 ± 0.035 0.769 ± 0.066 0.738 ± 0.060	0.784 ± 0.037 0.755 ± 0.090 0.725 ± 0.071	0.797 ± 0.033 0.800 ± 0.028 0.795 ± 0.031	0.800 ± 0.034 0.810 ± 0.024 0.802 ± 0.032	0.797 ± 0.033 0.801 ± 0.029 0.796 ± 0.030	0.794 ± 0.034 0.797 ± 0.029 0.791 ± 0.031
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.735 ± 0.084 0.701 ± 0.097 0.736 ± 0.075	0.780 ± 0.024 0.773 ± 0.042 0.772 ± 0.039	0.736 ± 0.086 0.698 ± 0.095 0.725 ± 0.084	0.714 ± 0.128 0.664 ± 0.139 0.711 ± 0.113	0.792 ± 0.059 0.790 ± 0.050 0.827 ± 0.041	0.824 ± 0.026 0.812 ± 0.032 0.839 ± 0.031	0.797 ± 0.053 0.790 ± 0.044 0.827 ± 0.044	0.787 ± 0.064 0.783 ± 0.054 0.823 ± 0.046
Word embedding	Network embedding	Text branch	Propagation branch	GRU layer				BiGRU layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.481 ± 0.020 0.485 ± 0.027 0.485 ± 0.033	0.240 ± 0.010 0.351 ± 0.150 0.491 ± 0.034	0.500 ± 0.000 0.510 ± 0.017 0.492 ± 0.033	0.324 ± 0.008 0.371 ± 0.085 0.469 ± 0.044
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.764 ± 0.031 0.757 ± 0.044 0.751 ± 0.036	0.772 ± 0.027 0.765 ± 0.041 0.751 ± 0.036	0.771 ± 0.025 0.760 ± 0.046 0.750 ± 0.036	0.761 ± 0.029 0.755 ± 0.045 0.749 ± 0.035	0.780 ± 0.018 0.768 ± 0.030 0.776 ± 0.015	0.784 ± 0.018 0.769 ± 0.025 0.776 ± 0.017	0.783 ± 0.017 0.764 ± 0.027 0.775 ± 0.014	0.779 ± 0.018 0.764 ± 0.029 0.773 ± 0.016
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.686 ± 0.044 0.694 ± 0.072 0.712 ± 0.044	0.734 ± 0.038 0.757 ± 0.038 0.745 ± 0.034	0.688 ± 0.040 0.695 ± 0.058 0.713 ± 0.049	0.667 ± 0.060 0.670 ± 0.091 0.701 ± 0.057	0.742 ± 0.038 0.731 ± 0.023 0.748 ± 0.029	0.753 ± 0.030 0.752 ± 0.033 0.754 ± 0.029	0.746 ± 0.036 0.735 ± 0.019 0.750 ± 0.028	0.742 ± 0.038 0.727 ± 0.021 0.747 ± 0.028
Word embedding	Network embedding	Text branch	Propagation branch	LSTM				BiLSTM layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.667 ± 0.051 0.623 ± 0.040 0.642 ± 0.028	0.732 ± 0.046 0.649 ± 0.053 0.679 ± 0.034	0.668 ± 0.045 0.630 ± 0.040 0.653 ± 0.028	0.643 ± 0.066 0.617 ± 0.042 0.632 ± 0.030
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.756 ± 0.049 0.783 ± 0.045 0.764 ± 0.050	0.788 ± 0.019 0.805 ± 0.031 0.782 ± 0.035	0.758 ± 0.043 0.775 ± 0.049 0.760 ± 0.050	0.746 ± 0.054 0.773 ± 0.053 0.757 ± 0.059	0.799 ± 0.030 0.827 ± 0.042 0.814 ± 0.043	0.805 ± 0.025 0.831 ± 0.035 0.818 ± 0.041	0.796 ± 0.034 0.827 ± 0.040 0.815 ± 0.042	0.794 ± 0.036 0.825 ± 0.042 0.812 ± 0.044
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.651 ± 0.105 0.708 ± 0.078 0.732 ± 0.074	0.758 ± 0.040 0.783 ± 0.028 0.762 ± 0.037	0.666 ± 0.096 0.702 ± 0.078 0.735 ± 0.062	0.611 ± 0.139 0.675 ± 0.117 0.720 ± 0.093	0.748 ± 0.049 0.758 ± 0.047 0.770 ± 0.083	0.777 ± 0.029 0.787 ± 0.035 0.815 ± 0.029	0.752 ± 0.042 0.759 ± 0.049 0.774 ± 0.077	0.741 ± 0.053 0.749 ± 0.058 0.757 ± 0.103

Table 5
GETAE ablation testing on Twitter16 (Node2Vec $p = 1, q = 1$, dimensions = 100 for both Node2Vec and DeepWalk.).

Word embedding	Network embedding	Text branch	Propagation branch	RNN layer				BiRNN layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.705 ± 0.081 0.647 ± 0.048 0.702 ± 0.060	0.716 ± 0.075 0.685 ± 0.074 0.722 ± 0.067	0.705 ± 0.072 0.646 ± 0.048 0.707 ± 0.061	0.697 ± 0.080 0.627 ± 0.053 0.699 ± 0.062
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.848 ± 0.036 0.896 ± 0.025 0.801 ± 0.072	0.850 ± 0.032 0.901 ± 0.026 0.836 ± 0.037	0.849 ± 0.036 0.897 ± 0.028 0.800 ± 0.079	0.847 ± 0.037 0.895 ± 0.025 0.789 ± 0.089	0.856 ± 0.043 0.880 ± 0.031 0.851 ± 0.040	0.869 ± 0.029 0.881 ± 0.031 0.856 ± 0.036	0.863 ± 0.035 0.880 ± 0.031 0.832 ± 0.032	0.854 ± 0.045 0.879 ± 0.031 0.850 ± 0.038
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.714 ± 0.098 0.670 ± 0.101 0.612 ± 0.122	0.743 ± 0.162 0.666 ± 0.208 0.670 ± 0.233	0.695 ± 0.112 0.655 ± 0.113 0.621 ± 0.098	0.659 ± 0.161 0.601 ± 0.170 0.538 ± 0.162	0.789 ± 0.097 0.800 ± 0.099 0.829 ± 0.033	0.835 ± 0.055 0.847 ± 0.046 0.842 ± 0.030	0.799 ± 0.090 0.801 ± 0.096 0.831 ± 0.031	0.781 ± 0.107 0.784 ± 0.125 0.826 ± 0.034
Word embedding	Network embedding	Text branch	Propagation branch	GRU layer				BiGRU layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.508 ± 0.062 0.488 ± 0.070 0.463 ± 0.027	0.303 ± 0.158 0.438 ± 0.190 0.409 ± 0.095	0.501 ± 0.003 0.517 ± 0.028 0.475 ± 0.034	0.339 ± 0.029 0.382 ± 0.095 0.380 ± 0.040
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.858 ± 0.038 0.869 ± 0.049 0.845 ± 0.031	0.868 ± 0.035 0.876 ± 0.047 0.849 ± 0.025	0.858 ± 0.040 0.872 ± 0.051 0.847 ± 0.030	0.855 ± 0.039 0.866 ± 0.050 0.843 ± 0.032	0.837 ± 0.051 0.858 ± 0.051 0.857 ± 0.027	0.844 ± 0.045 0.860 ± 0.046 0.866 ± 0.024	0.838 ± 0.048 0.855 ± 0.053 0.862 ± 0.031	0.835 ± 0.052 0.855 ± 0.053 0.856 ± 0.027
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.693 ± 0.121 0.672 ± 0.107 0.684 ± 0.105	0.720 ± 0.168 0.755 ± 0.045 0.788 ± 0.045	0.697 ± 0.120 0.690 ± 0.094 0.683 ± 0.094	0.653 ± 0.170 0.644 ± 0.141 0.638 ± 0.132	0.781 ± 0.043 0.739 ± 0.063 0.767 ± 0.109	0.827 ± 0.024 0.784 ± 0.026 0.797 ± 0.067	0.788 ± 0.041 0.751 ± 0.046 0.774 ± 0.089	0.773 ± 0.049 0.728 ± 0.066 0.757 ± 0.125
Word embedding	Network embedding	Text branch	Propagation branch	LSTM				BiLSTM layer			
				Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
				WORD2VEC	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.471 ± 0.054 0.481 ± 0.056 0.467 ± 0.021	0.235 ± 0.025 0.374 ± 0.134 0.416 ± 0.098	0.500 ± 0.000 0.506 ± 0.022 0.478 ± 0.021	0.320 ± 0.024 0.366 ± 0.068 0.407 ± 0.057
BERT	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.844 ± 0.055 0.814 ± 0.113 0.825 ± 0.054	0.852 ± 0.044 0.853 ± 0.054 0.843 ± 0.046	0.846 ± 0.055 0.827 ± 0.099 0.832 ± 0.049	0.840 ± 0.059 0.806 ± 0.134 0.823 ± 0.054	0.776 ± 0.140 0.879 ± 0.036 0.874 ± 0.033	0.822 ± 0.060 0.883 ± 0.036 0.880 ± 0.027	0.789 ± 0.116 0.885 ± 0.036 0.877 ± 0.032	0.759 ± 0.170 0.878 ± 0.037 0.873 ± 0.032
BERTWEET	N/A NODE2VEC DEEPWALK	✓ ✓ ✓	N/A ✓ ✓	0.689 ± 0.106 0.698 ± 0.099 0.651 ± 0.120	0.682 ± 0.208 0.737 ± 0.089 0.709 ± 0.172	0.687 ± 0.120 0.680 ± 0.104 0.666 ± 0.094	0.637 ± 0.178 0.651 ± 0.147 0.609 ± 0.154	0.750 ± 0.080 0.796 ± 0.033 0.798 ± 0.047	0.806 ± 0.028 0.812 ± 0.039 0.820 ± 0.030	0.751 ± 0.084 0.794 ± 0.031 0.806 ± 0.041	0.729 ± 0.049 0.790 ± 0.031 0.794 ± 0.051

dataset, and (2) (BERT, RNN, Node2Vec) for the Twitter16 dataset. Beyond optimizing these hyperparameters, we also investigated their impact on our model performance. This sensitivity analysis revealed distinct behaviors. Tables 6 and 7 illustrate the effect of varying p, q, and d hyperparameters on the Accuracy, Recall, Precision, and F1-score metrics. Our sensitivity analysis revealed that the dimensions hyperparameter had the most significant impact on performance, followed by the return hyperparameter, with the in-out hyperparameter exhibiting the least influence.

For the Twitter15 dataset (Table 6), we observe that the best results are obtained for hyperparameters' values $p = 1, q = 1$, and $d = 100$ with 82.7% Accuracy, 83.1% Precision, 82.7% Recall, and 82.5% F1-Score. With this GETAE configuration, the node embedding model uses a balanced strategy for random walks, considering both breadth-first and depth-first traversals. The second best model trained using the GETAE architecture obtained the best results when using $p = 0.5, q = 0.5$, and $d = 100$ as hyperparameter values. The difference between the best and second best performing models is around 3%.

Table 6
NODE2VEC on Twitter15 (BERT + BiLSTM).

p	q	d	Accuracy	Recall	Precision	F1-score
1.0	1.0	32	0.686 ± 0.070	0.696 ± 0.056	0.691 ± 0.063	0.682 ± 0.075
1.0	0.5	32	0.715 ± 0.022	0.724 ± 0.029	0.717 ± 0.022	0.714 ± 0.022
0.5	1.0	32	0.701 ± 0.033	0.710 ± 0.030	0.707 ± 0.032	0.700 ± 0.035
0.5	0.5	32	0.709 ± 0.036	0.716 ± 0.039	0.711 ± 0.040	0.707 ± 0.038
2.0	1.0	32	0.685 ± 0.056	0.704 ± 0.036	0.689 ± 0.046	0.678 ± 0.066
1.0	2.0	32	0.713 ± 0.032	0.722 ± 0.029	0.714 ± 0.030	0.712 ± 0.033
1.0	1.0	100	0.827 ± 0.042	0.831 ± 0.035	0.827 ± 0.040	0.825 ± 0.042
1.0	0.5	100	0.806 ± 0.023	0.807 ± 0.024	0.804 ± 0.024	0.804 ± 0.024
0.5	1.0	100	0.780 ± 0.042	0.790 ± 0.034	0.785 ± 0.037	0.778 ± 0.044
0.5	0.5	100	0.823 ± 0.032	0.827 ± 0.034	0.824 ± 0.034	0.821 ± 0.035
2.0	1.0	100	0.806 ± 0.038	0.814 ± 0.037	0.809 ± 0.035	0.805 ± 0.038
1.0	2.0	100	0.791 ± 0.051	0.813 ± 0.030	0.794 ± 0.048	0.784 ± 0.058

Table 7
NODE2VEC on Twitter16 (BERT + RNN).

p	q	d	Accuracy	Recall	Precision	F1-score
1.0	1.0	32	0.826 ± 0.059	0.849 ± 0.039	0.831 ± 0.053	0.822 ± 0.066
1.0	0.5	32	0.836 ± 0.128	0.815 ± 0.198	0.842 ± 0.119	0.821 ± 0.171
0.5	1.0	32	0.824 ± 0.091	0.796 ± 0.173	0.815 ± 0.113	0.799 ± 0.151
0.5	0.5	32	0.861 ± 0.042	0.867 ± 0.041	0.860 ± 0.040	0.856 ± 0.044
2.0	1.0	32	0.878 ± 0.040	0.884 ± 0.034	0.873 ± 0.045	0.872 ± 0.044
1.0	2.0	32	0.868 ± 0.042	0.874 ± 0.039	0.862 ± 0.044	0.864 ± 0.044
1.0	1.0	100	0.896 ± 0.025	0.901 ± 0.026	0.897 ± 0.028	0.895 ± 0.025
1.0	0.5	100	0.841 ± 0.089	0.871 ± 0.040	0.841 ± 0.087	0.830 ± 0.110
0.5	1.0	100	0.853 ± 0.071	0.868 ± 0.043	0.858 ± 0.062	0.851 ± 0.074
0.5	0.5	100	0.822 ± 0.093	0.856 ± 0.050	0.828 ± 0.078	0.813 ± 0.099
2.0	1.0	100	0.845 ± 0.054	0.858 ± 0.044	0.846 ± 0.048	0.842 ± 0.055
1.0	2.0	100	0.858 ± 0.049	0.864 ± 0.045	0.863 ± 0.041	0.855 ± 0.049

This hyperparameters' configuration is not leaning towards breadth-first or depth-first strategies. Thus, in the case of Twitter15, we notice that having a bias for the random walk does not matter that much, while $d = 100$ tells us that a fine-grained representation is needed for understanding more complex patterns.

For the Twitter16 dataset (Table 7), the best model trained using the GETAE architecture uses the following hyperparameters' values $p = 1$, $q = 1$, and $d = 100$. This model obtains 89.6% Accuracy, 90.1% Precision, 89.7% Recall, and 89.5% F1-Score. These hyperparameters offer a balanced random walk strategy. The second best model is trained using the following hyperparameters' values $p = 2$, $q = 1$, and $d = 32$. The differences between the best and second best-performing models are between 2%–7% for each metric. The second best performing model is leaning towards a depth-first sampling strategy due to a higher likeliness of the returning hyperparameter p . Furthermore, the second-best results are obtained when the dimension is small, i.e., $d = 32$. Thus, we can conclude that a 32-dimensional representation is also not only enough, but better at capturing node information than many other strategies and dimensions.

4.4.2. DEEPWALK

Tables 8 and 9 present the results obtained by the models trained using the GETAE architecture when using DEEPWALK node embeddings (Perozzi et al., 2014). For this set of experiments, we use BERT for embedding the textual data and we vary both the node embedding dimension d as well as the recurrent neural network layer of GETAE's Text Branch. Using sensitivity analysis, we observe the distinct behavior of the hyperparameter d on the Accuracy, Recall, Precision, and F1-score metrics.

For the Twitter15 dataset (Table 8), the best results are obtained when using $d = 100$ for the node embedding and the BiLSTM layer for the Text Branch. The model trained using this configuration obtains 81.4% Accuracy, 81.8% Precision, 81.5% Recall, and 81.2% F1-Score on the test set. The second best model obtains the best results when the hyperparameters configuration is $d = 32$ and a BiLSTM. The differences

between the best and second best-performing models are between 1%–2% for each metric. Using sensitivity analysis, we observe that the models trained using the GETAE architecture that employs $d = 100$ obtain better results than when using $d = 32$. We conclude that having a higher number of dimensions yields better results when training models with the GETAE architecture for this dataset. However, we observe that choosing to use a BiLSTM layer for GETAE's Text Branch has a greater influence than an increase in dimensions of the node embedding for GETAE's Propagation Branch.

For the Twitter16 dataset (Table 9), we observe that smaller node embedding dimensions are better for training models with the GETAE architecture. The best model uses the BiLSTM for GETAE's Text Branch and a 32-dimensional node embedding for GETAE's Propagation Branch. This model obtains 88.1% Accuracy, 88% Precision, 87.9% Recall, and 88% F1-Score on the test set. In comparison, the sensitivity analysis reveals that the results obtained by the model with $d = 32$ are around 1% better than when using $d = 100$ for the node embedding. We can conclude that for this dataset, the difference between using a lower or higher dimension representation of the node embedding changes very little the models' overall performance.

4.5. Comparison with state-of-the-art models

To better determine the performance of GETAE, we compare the results obtained by our model with state-of-the-art models trained on the Twitter15 and Twitter16 datasets. In our comparison, we employ the following models also used by Lu and Li (2020) in their comparison:

- **DTC** (Castillo et al., 2011) uses Decision Trees from user and tweet data.
- **SVM-TS** (Ma et al., 2016) uses a Support Vector Machine with data from the source tweet and the retweets sequence.
- **mgUR** (Ma et al., 2016) is a modified GRU neural network using the source tweet and the patterns in retweets.

Table 8
DeepWalk on Twitter15 + BERT.

d	RNN layer	Accuracy	Recall	Precision	F1-score
32	RNN	0.775 ± 0.045	0.786 ± 0.036	0.771 ± 0.042	0.768 ± 0.045
32	BiRNN	0.789 ± 0.040	0.797 ± 0.032	0.789 ± 0.038	0.786 ± 0.042
32	GRU	0.744 ± 0.026	0.749 ± 0.025	0.744 ± 0.027	0.743 ± 0.026
32	BiGRU	0.765 ± 0.017	0.770 ± 0.014	0.767 ± 0.019	0.764 ± 0.018
32	LSTM	0.758 ± 0.043	0.787 ± 0.033	0.758 ± 0.034	0.752 ± 0.043
32	BiLSTM	0.805 ± 0.037	0.819 ± 0.034	0.808 ± 0.038	0.801 ± 0.039
100	RNN	0.736 ± 0.063	0.773 ± 0.044	0.738 ± 0.060	0.725 ± 0.071
100	BiRNN	0.795 ± 0.031	0.802 ± 0.032	0.796 ± 0.030	0.791 ± 0.031
100	GRU	0.751 ± 0.036	0.751 ± 0.036	0.750 ± 0.036	0.749 ± 0.035
100	BiGRU	0.776 ± 0.015	0.776 ± 0.017	0.775 ± 0.014	0.773 ± 0.016
100	LSTM	0.764 ± 0.050	0.782 ± 0.035	0.760 ± 0.050	0.757 ± 0.059
100	BiLSTM	0.814 ± 0.043	0.818 ± 0.041	0.815 ± 0.042	0.812 ± 0.044

Table 9
DeepWalk on Twitter16 + BERT.

d	RNN layer	Accuracy	Recall	Precision	F1-score
32	RNN	0.844 ± 0.098	0.870 ± 0.068	0.847 ± 0.090	0.838 ± 0.108
32	BiRNN	0.843 ± 0.095	0.860 ± 0.064	0.845 ± 0.083	0.838 ± 0.102
32	GRU	0.853 ± 0.039	0.861 ± 0.036	0.852 ± 0.039	0.852 ± 0.040
32	BiGRU	0.846 ± 0.026	0.853 ± 0.021	0.849 ± 0.024	0.846 ± 0.026
32	LSTM	0.860 ± 0.053	0.870 ± 0.044	0.859 ± 0.050	0.856 ± 0.056
32	BiLSTM	0.881 ± 0.025	0.880 ± 0.023	0.879 ± 0.025	0.880 ± 0.026
100	RNN	0.801 ± 0.072	0.836 ± 0.037	0.800 ± 0.079	0.789 ± 0.089
100	BiRNN	0.851 ± 0.040	0.856 ± 0.036	0.856 ± 0.032	0.850 ± 0.038
100	GRU	0.845 ± 0.031	0.849 ± 0.025	0.847 ± 0.030	0.843 ± 0.032
100	BiGRU	0.857 ± 0.027	0.866 ± 0.024	0.862 ± 0.031	0.856 ± 0.027
100	LSTM	0.825 ± 0.054	0.843 ± 0.046	0.832 ± 0.049	0.823 ± 0.054
100	BiLSTM	0.874 ± 0.033	0.880 ± 0.027	0.877 ± 0.032	0.873 ± 0.032

Table 10
Fake news detection comparison with state-of-the-art models.

Model	Twitter15				Twitter16			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
DTC (Castillo, Mendoza, & Poblete, 2011)	0.4949	0.4963	0.4806	0.4948	0.5612	0.5753	0.5369	0.5616
SVM-TS (Ma et al., 2016)	0.5195	0.5195	0.5186	0.5190	0.6932	0.6928	0.6910	0.6915
mGRU (Ma et al., 2016)	0.5547	0.5145	0.5148	0.5104	0.6612	0.5603	0.5618	0.5563
RFC (Kwon, Cha, & Jung, 2017)	0.5385	0.5718	0.5302	0.4642	0.6620	0.7315	0.6587	0.6275
tCNN (Yang et al., 2018)	0.5881	0.5199	0.5206	0.5140	0.7374	0.6248	0.6262	0.6200
CRNN (Liu & Wu, 2018)	0.5919	0.5296	0.5305	0.5249	0.7576	0.6419	0.6433	0.6367
CSI (Ruchansky, Seo, & Liu, 2017)	0.6987	0.6991	0.6867	0.7174	0.6612	0.6321	0.6309	0.6304
dEFEND (Shu, Cui, Wang, Lee, & Liu, 2019)	0.7383	0.6584	0.6611	0.6541	0.7016	0.6365	0.6384	0.6311
DANES (Truică, Apostol et al., 2024)	0.7790	0.7856	0.7800	0.7827	0.7795	0.7854	0.7795	0.7824
GCAN-G (Lu & Li, 2020)	0.8636	0.7959	0.7990	0.7938	0.7939	0.6785	0.6802	0.6754
GCAN (Lu & Li, 2020)	0.8767	0.8257	0.8295	0.8250	0.9084	0.7594	0.7632	0.7593
GETAE	0.825	0.827	0.831	0.827	0.895	0.897	0.901	0.896

- **RFC** (Kwon et al., 2017) is a Random Forest-based classifier that learns on the source tweet and user retweets.
- **tCNN** (Yang et al., 2018) is a modified convolutional neural network that learns local patterns of user profile sequence and tweet data.
- **CRNN** (Liu & Wu, 2018) is an RNN and CNN approach using the source tweet data.
- **CSI** (Ruchansky et al., 2017) uses the tweets and a scoring mechanism of the users that retweet them.
- **dEFEND** (Shu et al., 2019) is a co-attention-based model that correlates the source tweet and the user profile.
- **DANES** (Truică, Apostol et al., 2024) is a model that uses a new node embedding from social and textual context data.
- **GCAN** and **GCAN-G** (Lu & Li, 2020). GCAN uses user features and combines them with recurrent and convolutional networks for retweet propagation. GCAN-G learns the same features but without the convolutional network.

Table 10 presents the results of our comparison. On Twitter15, GETAE outperforms the other model obtaining 82.7% F1-Score, 83.1% Recall, 82.7% Precision, while on Twitter16 obtaining 89.6% F1-Score, 90.1% Recall and 89.7% Precision. The best accuracies are obtained by GCAN, while the second best results come from GCAN, DANES, and GCAN-G. Except for DANES, both GCAN and GCAN-G are large deep neural networks.

For testing our best node embedding techniques, we compared GETAE with two state-of-the-art models:

- **HiMaP** (Mishra, 2020) learns indirect relationships between users with graph embeddings.
- **SSGE** (Vu & Jung, 2021) uses graph embeddings to learn propagation representations.

Table 11 presents the comparison between the two state-of-the-art models and GETAE. On the Twitter15 dataset, GETAE obtained the best results when using `Node2Vec`. We observe that our model is slightly

Table 11
Comparison with state-of-the-art w.r.t. node embeddings.

Model	Node embedding	Twitter15 accuracy	Twitter16 accuracy
HiMaP (Mishra, 2020)	Node2Vec	0.846	0.867
	DeepWalk	0.825	0.850
SSGE (Vu & Jung, 2021)	Node2Vec	0.600	0.600
	DeepWalk	0.500	0.500
GETAE	Node2Vec	0.827	0.896
	DeepWalk	0.814	0.881

surpassed by HiMaP with a small difference in Accuracy of only 1.9%. On the Twitter16 dataset, GETAE obtains the overall best performance with an Accuracy of 89.6%, outperforming the other state-of-the-art models used in this comparison.

5. Discussion and limitations

GETAE is a promising ensemble architecture for training fake news detection models that consider the information propagation of content. By considering how harmful nodes infect the network, the models trained outperform existing state-of-the-art models. Furthermore, GETAE manages to better detect fake news by combining the textual content and its propagation within social media by leveraging user interactions and encoding them within the novel Propagation-Enhanced Content Embedding. Thus, by incorporating the social dynamics among users and how users interact with content, the models trained using the GETAE architecture also underline the way malicious actors manage to misuse information and how they can take advantage of users' biases to manipulate them.

On Twitter15, in terms of Accuracy and F1-Score, the model trained using BiLSTM for the [Bi]RNN layer, BERT as the word embedding model, and NODE2VEC as the node embedding outperforms the other models. Using this configuration for the GETAE architecture, the trained model obtains an Accuracy score of 82.7% and an F1-Score of 82.5%. When configuring the GETAE architecture to use during training BiRNN for the [Bi]RNN layer, BERT_{TWEET} as the word embedding model, and DEEPWALK as the node embedding, we obtain a Precision score of 83.9% and an F1-Score of 82.7%. During hyperparameter tuning, when using the BiLSTM for the [Bi]RNN layer and BERT as the word embedding model, we observe that the best hyperparameters for NODE2VEC are $p = 1, q = 1, dim = 100$ followed by $p = 0.5, q = 0.5, dim = 100$ with a lower mean accuracy by $\sim 4\%$, but a higher standard deviation by $\sim 0.1\%$. DEEPWALK also provided the best overall results when using the BERT as the word embeddings and BiLSTM for the [Bi]RNN layer. The best GETAE configuration that uses WORD2VEC as the word embedding mode employs BiRNN for the [Bi]RNN layer and no node embedding. For this dataset, we can conclude that the *Propagation-Enhanced Content Embedding* performs better when using contextual word embeddings, such as BERT and BERT_{TWEET}, as well as a bidirectional approach for the [Bi]RNN layer.

For the Twitter16 dataset, the model trained using RNN for the [Bi]RNN layer, BERT as the word embedding model, and NODE2VEC as the node embedding outperforms the other models. With this configuration, the model obtains an Accuracy score of 89.6%, a Precision score of 90.1%, a Recall score of 89.7%, and an F1-Score of 89.5%. When configuring the GETAE architecture to use during training BiRNN for the [Bi]RNN layer, BERT_{TWEET} as the word embedding model, and DEEPWALK as the node embedding, we obtain an Accuracy score of 82.9%, Precision score of 84.2%, Recall score of 83.1% and an F1-Score of 82.6%. During hyperparameter tuning, when using the RNN for the [Bi]RNN layer and BERT as the word embedding model, we observe that the best hyperparameters for NODE2VEC are $p = 1, q = 1, dim = 100$ followed by $p = 2, q = 1.0, dim = 32$ with a lower mean accuracy by $\sim 3\%$. DEEPWALK also provided the best overall results when using the BERT as the word embeddings and BiLSTM for the [Bi]RNN layer for

both small and large vector dimensions. The best GETAE configuration that uses WORD2VEC as the word embedding mode employs BiRNN for the [Bi]RNN layer and NODE2VEC as node embedding.

In comparison with other state-of-the-art models on Fake News Detection or Rumor Detection, our best models outperform GCAN (Lu & Li, 2020) in terms of Precision, Recall, and F1-Score, but it is outperformed in terms of Accuracy. As we used Stratified K-Fold validation and conducted 10 experiments for each measure while randomizing the train–test-validation split each time, we observed that the standard deviation was small enough to conclude that our models generalize well on the two datasets. During hyperparameter tuning, we obtain better results on high-dimensional node embeddings. Thus, we can conclude that more detailed representations are needed for capturing the topological information of the graph as well as better representing the information diffusion in order to create better *Propagation-Enhanced Content Embeddings*. Furthermore, for the NODE2VEC, the best return parameters and in–out parameters show that a balanced random walk strategy is preferred, with depth-first search in second place. Based on these observations, we can conclude that:

- The choice of models for word embeddings, node embeddings, and recurrent layers needs to be done after proper testing, in order to get a better fit to the data at hand — answering Q1.
- The Text Content Embedding manages to better encode multiple complex textual features such as lexical (e.g., character and word level features) and syntactic (e.g., sentence-level features) to improve prediction — answering Q2.
- The Propagation Embedding is specifically designed to encode the information diffusion to determine the spread of information from a node to its followers — answering Q3.
- The Propagation-Enhanced Content Embedding manages to better encapsulate textual content and context and the propagation information to improve fake news detection — answering Q4.
- Our data-driven best models trained with GETAE manage to outperform existing state-of-the-art models trained on the same datasets — answering Q5.

The rapid spread of harmful content (i.e., fake news, misinformation, disinformation, offensive language, hate speech, etc.) on social platforms is another thread that needs to be addressed. To deal with this thread, in the current literature there have been proposed two main approaches: network immunization, (i.e., stopping the spread of harmful content in real-time) and architectures that detects harmful content in real-time. The GETAE architecture can be used in both of these approaches by training models to deal with specific or general harmful content detection tasks. The models trained with GETAE can be integrated into current solutions as follows. For network immunization strategies (i.e, SparseShield (Petrescu et al., 2021), CONTAIN (Apostol, Coban et al., 2024), or MCWDST (Truică, Apostol et al., 2023)), GETAE models can be used to detect the seeds before applying network immunization. For real-time detection systems (i.e., STOPHC (Truică, Constantinescu et al., 2024), CONTCOMMRTD (Apostol, Truică et al., 2024)), GETAE models can be used within their harmful content detection modules.

We conclude the discussion by addressing one final question: *Can a propagation-enhanced deep neural network ensemble architecture provide a*

definitive solution to the challenge of Fake News Detection? The answer is a qualified no. While the proposed architecture offers potential improvements, no single solution can fully address the multifaceted nature of fake news. A mixture of models and techniques, e.g., Mixture of Experts (MoE), is necessary to improve the accuracy of predicting fake news. In such ecosystems, the proposed GETAE architecture demonstrates promising accuracy when incorporating information propagation and textual context. Numerous challenges remain and need to be addressed to help users make informed decisions while browsing news feeds on social networks.

The primary challenge we identified through our review of the current literature includes:

- (1) Stable datasets that preserve all metadata information, rather than relying on unstable URLs and unique IDs.
- (2) Multi-modal datasets on which we can train richer feature representations.
- (3) Human verified dataset that can enhance feature selection (Parikh & Atrey, 2018).

As a final thought, a significant limitation in this research area is the absence of large, annotated datasets that capture social media content and underlying network characteristics, such as post reactions, diffusion, etc. Furthermore, another shortcoming with current datasets is the veracity of data (Trajanov et al., 2024) and how uncertainties and inaccuracies that appear within the data are inherently learned by the trained models (Zarei, Yazdi, Moradi, & BahooToroodi, 2024). Such datasets would enable us to train and fine-tune our models for enhanced performance and validate and generalize our findings.

6. Conclusions

In this paper, we propose GETAE, Graph Information Enhanced Deep Neural Network Ensemble Architecture for Fake News Detection, a novel ensemble architecture that uses textual content together with the social interactions to improve fake news detection. We propose 3 types of embeddings, (1) a Text Content Embedding that manages to better encode lexical and syntactic textual features, (2) a Propagation Embedding specifically designed to encode the information diffusion, and (3) a Propagation-Enhanced Content Embedding that encapsulates both textual and propagation information. Using cross-validation, ablation testing, and hyperparameter tuning, we train multiple models on two real-world datasets, i.e., Twitter15 and Twitter16. We observe that the models trained using GETAE manage to outperform the current state-of-the-art, showing the importance of integrating network and diffusion information into the training step to improve inference.

BERT and BERT_{TWEET} Transformer embeddings paired with NODE2VEC or DEEPWALK node embeddings and BiLSTM or RNN recurrent layers gained a meaningful and accurate representation of both tweet content and network topology of the source tweets. We conclude that for NODE2VEC there is no simple strategy for biased random walks, our results despite having good accuracy, have no consistency leaning towards breadth-first or depth-first strategies. Our best recurrent layers in terms of accuracy were BiLSTM for Twitter15 and RNN for Twitter16 paired with BERT Transformer embeddings for text vectorization.

In the future, we plan to incorporate additional context-based features, such as the number of likes, retweets, and comments, into this model. We aim to add new convolutional layers to the GETAE architecture in order to better understand the topological patterns of the source nodes that spread fake news. Moreover, we plan to experiment on larger datasets and other social networks where we can create domain-specific lexicon techniques (Truică, Istrate and Apostol, 2023) for misinformation and propaganda detection and mitigation. Based on our previous approaches, we aim to create a Mixture of Experts (MoE) based solution for fake news detection. Finally, we plan to investigate the role of multimodal features in fake news detection enhanced with network information, building upon the findings presented here by incorporating network, textual, visual, and potentially audio cues.

CRediT authorship contribution statement

Ciprian-Octavian Truică: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Elena-Simona Apostol:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Marius Marogel:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization. **Adrian Paschke:** Conceptualization, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research presented in this paper was supported in part by (1) a grant from the National Program for Research of the National Association of Technical Universities (GNAC ARUT 2023) through the project “DEPLATFORM: Intelligent interactive system for detecting the veracity of news published on social platforms” (Contract no. 63/10.10.2023), (2) the German Academic Exchange Service (DAAD) through the project “iTracing: Automatic Misinformation Fact-Checking” (DAAD grant no. 91809005), (3) The Academy of Romanian Scientists through the funding of project “SCAN-NEWS: Smart system for detecting and mitigating misinformation and fake news in social media” (AOȘR-TEAMS-III).

Data availability

The data is publicly available online. We have reference the original articles describing the data accordingly.

References

- Ahmad, I., Yousaf, M., Yousaf, S., & Ahmad, M. O. (2020). Fake news detection using machine learning ensemble methods. *Complexity*, 2020, 1–11(8885861). <http://dx.doi.org/10.1155/2020/8885861>.
- Aïmeur, E., Amri, S., & Brassard, G. (2023). Fake news, disinformation and misinformation in social media: a review. *Social Network Analysis and Mining*, 13(1), 1–36(30). <http://dx.doi.org/10.1007/s13278-023-01028-5>.
- Apostol, E.-S., Coban, Ö., & Truică, C.-O. (2024). CONTAIN: A community-based algorithm for network immunization. *Engineering Science and Technology, an International Journal*, 55, 1–10(101728). <http://dx.doi.org/10.1016/j.jestech.2024.101728>.
- Apostol, E.-S., Truică, C.-O., & Paschke, A. (2024). ContCommRTD: A distributed content-based misinformation-aware community detection system for real-time disaster reporting. *IEEE Transactions on Knowledge and Data Engineering*, 1–12. <http://dx.doi.org/10.1109/tkde.2024.3417232>.
- Bastos, M. T., & Mercea, D. (2017). The Brexit botnet and user-generated hyperpartisan news. *Social Science Computer Review*, 37(1), 38–54. <http://dx.doi.org/10.1177/0894439317734157>.
- Bird, S. (2009). *Natural language processing with Python*. O'Reilly.
- Borges, L., Martins, B., & Calado, P. (2019). Combining similarity features and deep representation learning for stance detection in the context of checking fake news. *Journal of Data and Information Quality*, 11(3), 1–26. <http://dx.doi.org/10.1145/3287763>.
- Bovet, A., & Makse, H. A. (2019). Influence of fake news in Twitter during the 2016 US presidential election. *Nature Communications*, 10(1), 1–14. <http://dx.doi.org/10.1038/s41467-018-07761-2>.
- Broda, E., & Strömbäck, J. (2024). Misinformation, disinformation, and fake news: lessons from an interdisciplinary, systematic literature review. *Annals of the International Communication Association*, 48(2), 139–166. <http://dx.doi.org/10.1080/23808985.2024.2323736>.

- Cai, H., Zheng, V. W., & Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9), 1616–1637. <http://dx.doi.org/10.1109/tkde.2018.2807452>.
- Castillo, C., Mendoza, M., & Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on world wide web* (pp. 675–684). ACM, <http://dx.doi.org/10.1145/1963405.1963500>.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American chapter of the association for computational linguistics* (pp. 4171–4186). ACL, <http://dx.doi.org/10.18653/v1/N19-1423>.
- Ding, Y., Guo, B., Liu, Y., Jing, Y., Yin, M., Li, N., et al. (2025). EvolveDetector: Towards an evolving fake news detector for emerging events with continual knowledge accumulation and transfer. *Information Processing & Management*, 62(1), Article 103878. <http://dx.doi.org/10.1016/j.ipm.2024.103878>.
- Fang, X., Wu, H., Jing, J., Meng, Y., Yu, B., Yu, H., et al. (2024). NSEP: Early fake news detection via news semantic environment perception. *Information Processing & Management*, 61(2), Article 103594. <http://dx.doi.org/10.1016/j.ipm.2023.103594>.
- Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151, 78–94. <http://dx.doi.org/10.1016/j.knsys.2018.03.022>.
- Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864). <http://dx.doi.org/10.1145/2939672.2939754>.
- Huang, Q., Zhou, C., Wu, J., Wang, M., & Wang, B. (2019). Deep structure learning for rumor detection on Twitter. In *2019 international joint conference on neural networks* (pp. 1–8). IEEE, <http://dx.doi.org/10.1109/ijcnn.2019.8852468>.
- Ilie, V.-I., Truică, C.-O., Apostol, E.-S., & Paschke, A. (2021). Context-aware misinformation detection: A benchmark of deep learning architectures using word embeddings. *IEEE Access*, 9, 162122–162146. <http://dx.doi.org/10.1109/ACCESS.2021.3132502>.
- Imaduwa, S., Kumara, P., & Samaraweera, W. (2022). Capturing credibility of users for an efficient propagation network based fake news detection. In *2022 2nd international conference on computer, control and robotics* (pp. 212–217). IEEE, <http://dx.doi.org/10.1109/iccrc54399.2022.9790186>.
- Ke, Z., Li, Z., Zhou, C., Sheng, J., Silamu, W., & Guo, Q. (2020). Rumor detection on social media via fused semantic information and a propagation heterogeneous graph. *Symmetry*, 12(11), 1–14(1806). <http://dx.doi.org/10.3390/sym12111806>.
- Khan, J. Y., Khondaker, M. T. I., Afroz, S., Uddin, G., & Iqbal, A. (2021). A benchmark study of machine learning models for online fake news detection. *Machine Learning with Applications*, 4, 1–12(100032). <http://dx.doi.org/10.1016/j.mlwa.2021.100032>.
- Kwon, S., Cha, M., & Jung, K. (2017). Rumor detection over varying time windows. *PLoS ONE*, 12(1), 1–19(e0168344). <http://dx.doi.org/10.1371/journal.pone.0168344>.
- Liu, Y., & Wu, Y.-F. (2018). Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 355–361). AAAI, <http://dx.doi.org/10.1609/aaai.v32i1.11268>.
- Lu, Y.-J., & Li, C.-T. (2020). GCAN: Graph-aware co-attention networks for explainable fake news detection on social media. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 505–514). ACL, <http://dx.doi.org/10.18653/v1/2020.acl-main.48>.
- Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K.-F., et al. (2016). Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 3818–3824). AAAI Press.
- Ma, J., Gao, W., & Wong, K.-F. (2017). Detect rumors in microblog posts using propagation structure via kernel learning. In *Annual meeting of the association for computational linguistics* (pp. 708–717). <http://dx.doi.org/10.18653/v1/P17-1066>.
- Marushchak, A., Petrov, S., & Khoperiya, A. (2025). Countering AI-powered disinformation through national regulation: learning from the case of Ukraine. *Frontiers in Artificial Intelligence*, 7, <http://dx.doi.org/10.3389/frai.2024.1474034>.
- Melchior, C., & Oliveira, M. (2023). A systematic literature review of the motivations to share fake news on social media platforms and how to fight them. *New Media & Society*, 26(2), 1127–1150. <http://dx.doi.org/10.1177/14614448231174224>.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *International conference on learning representations* (pp. 1–12).
- Mishra, R. (2020). Fake news detection using higher-order user to user mutual-attention progression in propagation paths. In *2020 IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 2775–2783). IEEE, <http://dx.doi.org/10.1109/cvprw50498.2020.00334>.
- Mustafaraj, E., & Metaxas, P. T. (2017). The fake news spreading plague. In *ACM on web science conference* (pp. 235–239). <http://dx.doi.org/10.1145/3091478.3091523>.
- Nguyen, D. Q., Vu, T., & Tuan Nguyen, A. (2020). BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 9–14). ACL, <http://dx.doi.org/10.18653/v1/2020.emnlp-demos.2>.
- Parikh, S. B., & Atrey, P. K. (2018). Media-rich fake news detection: A survey. In *IEEE conference on multimedia information processing and retrieval* (pp. 436–441). <http://dx.doi.org/10.1109/MIPR.2018.00093>.
- Pennebaker, J. W., Boyd, R. L., Jordan, K., & Blackburn, K. (2015). The development and psychometric properties of LIWC2015.
- Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710). <http://dx.doi.org/10.1145/2623330.2623732>.
- Petrescu, A. (2023). Leveraging MiniLMv2 pipelines for EXIST2023. In *CEUR workshop proceedings: vol. 3497, Working notes of the conference and labs of the evaluation forum* (pp. 1037–1043).
- Petrescu, A., Truică, C.-O., & Apostol, E.-S. (2019). Sentiment analysis of events in social media. In *2019 IEEE 15th international conference on intelligent computer communication and processing* (pp. 143–149). IEEE, <http://dx.doi.org/10.1109/icccp48234.2019.8959677>.
- Petrescu, A., Truică, C.-O., & Apostol, E.-S. (2024). Language-based mixture of transformers for EXIST2024. In *Conference and labs of the evaluation forum* (pp. 1157–1164).
- Petrescu, A., Truică, C.-O., Apostol, E.-S., & Karras, P. (2021). SparseShield: Social network immunization vs. harmful speech. In *ACM conference on information and knowledge management* (pp. 1426–1436). <http://dx.doi.org/10.1145/3459637.3482481>.
- Petrescu, A., Truică, C.-O., Apostol, E.-S., & Paschke, A. (2024). EDSA-ensemble: an event detection sentiment analysis ensemble architecture. *IEEE Transactions on Affective Computing*, 1–18. <http://dx.doi.org/10.1109/TAFFC.2024.3434355>.
- Qu, Z., Meng, Y., Muhammad, G., & Tiwari, P. (2024). QMFND: A quantum multimodal fusion-based fake news detection model for social media. *Information Fusion*, 104, Article 102172. <http://dx.doi.org/10.1016/j.inffus.2023.102172>.
- Rozemberczki, B., Kiss, O., & Sarkar, R. (2020). Karate club: An API oriented open-source python framework for unsupervised learning on graphs. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 3125–3132). ACM, <http://dx.doi.org/10.1145/3340531.3412757>.
- Ruchansky, N., Seo, S., & Liu, Y. (2017). CSI: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM conference on information and knowledge management* (pp. 797–806). ACM, <http://dx.doi.org/10.1145/3132847.3132877>.
- Ruths, D. (2019). The misinformation machine. *Science*, 363(6425), 348. <http://dx.doi.org/10.1126/science.aaw1315>.
- Rzyski, P., Borkowski, L., Drag, M., Flisiak, R., Jemielity, J., Krajewski, J., et al. (2021). The strategies to support the COVID-19 vaccination with evidence-based communication and tackling misinformation. *Vaccines*, 9(2), 109. <http://dx.doi.org/10.3390/vaccines9020109>.
- Shu, K., Cui, L., Wang, S., Lee, D., & Liu, H. (2019). dEFEND: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 395–405). ACM, <http://dx.doi.org/10.1145/3292500.3330935>.
- Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1), 22–36. <http://dx.doi.org/10.1145/3137597.3137600>.
- Singh, M. K., Ahmed, J., Alam, M. A., Raghuvanshi, K. K., & Kumar, S. (2023). A comprehensive review on automatic detection of fake news on social media. *Multimedia Tools and Applications*, 83(16), 47319–47352. <http://dx.doi.org/10.1007/s11042-023-17377-4>.
- Trajanov, D., Apostol, E., Garabik, R., Gkirtzou, K., Gromann, D., Liebeskind, C., et al. (2024). From linguistic linked data to big data. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation* (pp. 7489–7502). ELRA and ICCL, URL: <https://aclanthology.org/2024.lrec-main.661/>.
- Truică, C.-O., & Apostol, E.-S. (2022). MisRoBERTa: Transformers versus misinformation. *Mathematics*, 10(4), 1–25(569). <http://dx.doi.org/10.3390/math10040569>.
- Truică, C.-O., & Apostol, E.-S. (2023). It's all in the embedding! Fake news detection using document embeddings. *Mathematics*, 11(3), 1–29(508). <http://dx.doi.org/10.3390/math11030508>.
- Truică, C.-O., Apostol, E.-S., & Karras, P. (2024). DANES: Deep neural network ensemble architecture for social and textual context-aware fake news detection. *Knowledge-Based Systems*, 294, 1–13(111715). <http://dx.doi.org/10.1016/j.knsys.2024.111715>.
- Truică, C.-O., Apostol, E. S., & Leordeanu, C. A. (2017). Topic modeling using contextual cues. In *IEEE international symposium on symbolic and numeric algorithms for scientific computing* (pp. 203–210). <http://dx.doi.org/10.1109/synasc.2017.00041>.
- Truică, C.-O., Apostol, E.-S., Nicolescu, R.-C., & Karras, P. (2023). MCWdST: A minimum-cost weighted directed spanning tree algorithm for real-time fake news mitigation in social media. *IEEE Access*, 11, 125861–125873. <http://dx.doi.org/10.1109/ACCESS.2023.3331220>.
- Truică, C.-O., Apostol, E.-S., & Paschke, A. (2022). Awakened at CheckThat! 2022: Fake news detection using BiLSTM and sentence transformer. In *Working notes of the conference and labs of the evaluation forum* (pp. 749–757).
- Truică, C.-O., Apostol, E.-S., Şerban, M.-L., & Paschke, A. (2021). Topic-based document-level sentiment analysis using contextual cues. *Mathematics*, 9(21), 2722. <http://dx.doi.org/10.3390/math9212722>.
- Truică, C.-O., Apostol, E.-S., Ştefu, T., & Karras, P. (2021). A deep learning architecture for audience interest prediction of news topic on social media. In *International conference on extending database technology* (pp. 588–599). <http://dx.doi.org/10.5441/002/EDBT.2021.69>.

- Truică, C.-O., Constantinescu, A.-T., & Apostol, E.-S. (2024). StopHC: A harmful content detection and mitigation architecture for social media platforms. In *IEEE international conference on intelligent computer communication and processing* (pp. 1–5). <http://dx.doi.org/10.1109/ICCP63557.2024.10793051>.
- Truică, C.-O., Darmont, J., & Velcin, J. (2016). A scalable document-based architecture for text analysis. In *International conference on advanced data mining and applications* (pp. 481–494). Springer, http://dx.doi.org/10.1007/978-3-319-49586-6_33.
- Truica, C.-O., Guille, A., & Gauthier, M. (2016). CATS: Collection and analysis of tweets made simple. In *ACM conference on computer supported cooperative work and social computing companion* (pp. 41–44). ACM, <http://dx.doi.org/10.1145/2818052.2874320>.
- Truică, C.-O., Istrate, N.-O., & Apostol, E.-S. (2023). A distributed automatic domain-specific multi-word term recognition architecture using spark ecosystem. In *2023 22nd international symposium on parallel and distributed computing* (pp. 31–38). IEEE, <http://dx.doi.org/10.1109/ispd59212.2023.00018>.
- Truică, C.-O., & Leordeanu, C. A. (2017). Classification of an imbalanced data set using decision tree algorithms. *University Politehnica of Bucharest Scientific Bulletin - Series C Electrical Engineering and Computer Science*, 79(4), 69–84.
- Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *workshop on new challenges for NLP frameworks* (pp. 45–50).
- Vu, D. T., & Jung, J. J. (2021). Rumor detection by propagation embedding based on graph convolutional network. *International Journal of Computational Intelligence Systems*, 14(1), 1053–1065. <http://dx.doi.org/10.2991/ijcis.d.210304.002>.
- Wynne, H. E., & Wint, Z. Z. (2019). Content based fake news detection using N-gram models. In *International conference on information integration and web-based applications and services* (pp. 669–673). ACM, <http://dx.doi.org/10.1145/3366030.3366116>.
- Xie, Y., Huang, X., Xie, X., & Jiang, S. (2020). A fake news detection framework using social user graph. In *Proceedings of the 2020 2nd international conference on big data engineering* (pp. 55–61). ACM, <http://dx.doi.org/10.1145/3404512.3404515>.
- Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., & Yu, P. S. (2018). TI-CNN: Convolutional neural networks for fake news detection. <http://dx.doi.org/10.48550/ARXIV.1806.00749>, arXiv preprint [arXiv:1806.00749](https://arxiv.org/abs/1806.00749).
- Yin, J., Gao, M., Shu, K., Zhao, Z., Huang, Y., & Wang, J. (2025). Emulating reader behaviors for fake news detection. *IEEE Transactions on Big Data*, 1–12. <http://dx.doi.org/10.1109/tbdata.2025.3527230>.
- Zarei, E., Yazdi, M., Moradi, R., & BahooToroody, A. (2024). Expert judgment and uncertainty in sociotechnical systems analysis. In *Safety causation analysis in sociotechnical systems: advanced models and techniques* (pp. 487–530). Springer, http://dx.doi.org/10.1007/978-3-031-62470-4_18.
- Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management*, 57(2), 1–26(102025). <http://dx.doi.org/10.1016/j.ipm.2019.03.004>.
- Zhou, X., Li, J., Li, Q., & Zafarani, R. (2023). Linguistic-style-aware neural networks for fake news detection. <http://dx.doi.org/10.48550/ARXIV.2301.02792>, [arXiv:2301.02792](https://arxiv.org/abs/2301.02792).