Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Context-Aware Deep Markov Random Fields for Fake News Detection

TIEN HUU DO[1,2], MARC BERNEMAN[1], JASABANTA PATRO[1,2], GIANNIS BEKOULIS[1,2], AND NIKOS DELIGIANNIS[1,2], (Member, IEEE)

[1]Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB), Pleinlaan 2, B-1050 Brussels, Belgium
[2]imec, Kapeldreef 75, B-3001 Leuven, Belgium

Corresponding author: Nikos Deligiannis (e-mail: ndeligia@etrovub.be).

This work was supported in part by the VUB through the Strategic Research Programme: Processing of large scale multi-dimensional, multi-spectral, multi-sensorial and distributed data (M³D²) and in part by the Fonds Voor Wetenschappelijk Onderzoek (FWO) under Grant G0A2617N.

ABSTRACT Fake news is a serious problem, which has received considerable attention from both industry and academic communities. Over the past years, many fake news detection approaches have been introduced, and most of the existing methods rely on either news content or the social context of the news dissemination process on social media platforms. In this work, we propose a generic model that is able to take into account both the news content and the social context for the identification of fake news. Specifically, we explore different aspects of the news content by using both shallow and deep representations. The shallow representations are produced with word2vec and doc2vec models while the deep representations are generated via transformer-based models. These representations are able to jointly or separately address four individual tasks, namely bias detection, clickbait detection, sentiment analysis, and toxicity detection. In addition, we make use of graph convolutional neural networks and mean-field layers in order to exploit the underlying structural information of the news articles. That way, we are able to take into account the inherent correlation between the articles by leveraging their social context information. Experiments on widely-used benchmark datasets indicate the effectiveness of the proposed method.

INDEX TERMS Fake news detection, deep learning, markov random field, representation learning, question answering, sentiment analysis, clickbait detection, toxicity detection, bias detection.

I. INTRODUCTION

Fake news, which refers to stories that are intentionally and verifiably false, is deliberately created to mislead people for financial or political gains and has existed for a long time, even before the appearance of traditional media such as the printing press [1]. Social media platforms such as Twitter or Facebook and their increasing popularity speed up the dissemination of fake news since news can quickly and freely circulate through a huge network of social media users, where everyone can view and share news without paying much attention to the veracity of each reported claim [2].

Early works in fake news detection are mainly based on fact-checking of external sources or the writing style of news content [1]. The fake news detection task is traditionally approached using linguistic features that are able to identify linguistic patterns of the text [3], [4]. The main limitation of such methods is that they are hand-crafted and involve manual labor for designing them. On the other hand, more recent deep neural networks have been proposed to alleviate the need for manually designing hand-crafted features since deep learning methods are able to automatically capture linguistic patterns. Note also that the articles (that are discussing particular events, e.g., the election of the government) are not unrelated the one with the other. This is because there are common users that are interacting with these articles. Thus, this is why in our previous research works, we have exploited the correlation among the aforementioned articles [5], [6].

In particular, in our previous work (see [5]), we adopted the strategy of leveraging the content of news articles and exploited their correlation from the articles' social context to improve the fake news detection performance. In [5], we formulated mean-field layers via Markov Random Fields (MRF), taking into account the structural information of the underlying graph of considered articles. We then used the mean-field layers to design a deep learning model, referred to as Deep MRF, for Fake News detection (DMFN). In this paper, we provide a new perspective of the mean-field layers as

proposed in [5] by illustrating the similarity of these layers to graph convolutional layers [7]. More precisely, we show that both graph convolutional layers and mean-field layers tend to smooth the characteristics of nodes within the same cluster of the underlying graph. Furthermore, we go beyond the DMFN model in [5] by extending its multiview component. Unlike our previous research work (see [5], [6]), in this work, we are able to take simultaneously into account hand-crafted features (e.g., the TF-IDF), deep neural network methods (i.e., BERT), and graph neural networks for considering the correlation among news articles.

This work extends our conference paper in [5]. Specifically, we (i) add a graph-based subcomponent to exploit the engagement of social media users toward news articles, and (ii) extend the multiview component by exploiting the use of transformer-based models and that way integrating the bidirectional deep representation of the articles' content. On top of that, we show with extensive experiments on popular benchmark datasets that the proposed method outperforms other existing state-of-the-art methods on the fake news detection task. In summary, our contribution is three-fold:

- We provide an alternative formulation for the mean-field layers proposed in our previous work (see [5]), and thus show the equivalence of the mean-field layers and graph convolutional layers in smoothing the characteristics of nodes within the same cluster.
- We extend the multiview component of our DMFN model in [5] by considering also the user engagements towards news articles and deep bidirectional representation of the articles' content. The deep bidirectional representation of the news content is generated via transformer-based models [8], [9], which have been jointly or independently trained on various tasks strongly related to the fake news detection task.
- We carry out comprehensive experiments on three benchmark datasets. We show that our method is able to achieve consistent improvements on top of our DMFN model [5] and outperforms existing state-of-the-art methods on the task of fake news detection.

The rest of our paper is structured as follows. In Section II, we briefly present the related work and indicate the difference between our method and existing studies. Section III presents the overall architecture that relies on the original DMFN and provides an alternative formulation of the mean-field layers. The extension to the DMFN model is described in Section IV and Section V. Section VI demonstrates the effectiveness of the proposed method via experimental studies and the conclusion and future work are given in Section VII.

II. RELATED WORK

In the last two decades, there is a substantial increase in the number of publications in the domain of media manipulation and fake news [10], [11]. A number of tasks have been introduced since then such as fact checking [12]–[14], rumor detection [15], stance detection [16], assessing credibility [17], and exaggeration [18], [19]. Moreover, several datasets have been introduced for these tasks (see in particular the datasets on claim verification [16], [20]–[22], entire article verification [23] and verification of social media posts [24]–[26]). For more details regarding tasks and datasets related to fake news detection, we refer to the survey in [13].

A. HAND-CRAFTED FEATURES

Early work on fake news detection has been focused on feature-based methods to separate fake from genuine news. Linguistic patterns, such as, special characters, specific keywords and expression types were exploited to spot fake news [3], [27], [28]. However, these methods are not very effective as fake news is intentionally created to mimic the true news [29]. Apart from textual features, user related features were also leveraged to detect fake news. In particular, features like the number of followers, the age and the gender of users [3], [4], and news' propagation patterns [3], [30] were shown to to improve performance when combined with textual patterns; however, the reported prediction accuracy of such models is still relatively low [10]. It is worthwhile mentioning that the majority of these works rely on combinations of the aforementioned features rather than only on a single feature. Similar to these works, we also extract hand-crafted features. However, we do not rely on manually engineered features such as special characters or keywords, but we rather extract TF-IDF representations and timeseries (e.g., number of tweets in different timeslots) for our multiview component due to their state-of-the-art performance in [5].

B. DEEP NEURAL NETWORKS

With the evolution of deep neural models, researchers have also investigated deep learning architectures for fake news detection, which led to reestablishment of state-of-the-art performance [10], [11]. Many of them have represented the claims and articles as latent embeddings and fed them to neural classifiers [31]–[35]. Different architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been used to encode the articles and the claims. Alternatively, deep neural networks that input multiple types of features have been studied to detect fake news [36]–[38]. Recently, researchers have started using transformers in the task of fake news detection [39] due to their state-of-the-art performance in a number of NLP tasks (e.g, text classification, named entity recognition) [9]. We follow a similar approach to prior work on similar problems such as fact verification [13] and fake news detection [10], and we rely on BERT for extracting feature-based representations. However, we pretrain the models on similar tasks (to our core task) and apply transfer learning instead of fine-tuning the model to the new dataset. This is because we aim at a general purpose model that is able to generalize well on several similar tasks.

C. CORRELATION AMONG NEWS ARTICLES

Most of the aforementioned fake news detection models ignore the correlation among the news articles when mak-

ing decisions (a.k.a., they treat each news article independently of others). Nevertheless, news articles' correlation has been found effective in analysing online news and social events [6], [29], [40]–[42]. The correlation between news articles has also been exploited in the works of Shu *et al.* [29] and Zhang *et al.* [40]. Unlike our work, where we consider directly the connection among the news articles, they indirectly capture the correlations among the articles via modeling the relationships of these articles among their publishers and the social media users interacting with the articles. Freire *et al.* [41] proposed to detect breaking news on Wikipedia by exploring the graph of related events, where the graph is created by connecting any pair of pages on Wikipedia edited by the same users during a small time frame. The breaking news is then detected using a traditional densest-subgraph extraction approach. Fairbanks *et al.* [42] constructed a graph of news by connecting the web pages referring to a specific event and estimated the credibility of news by employing a belief propagation algorithm on the constructed graph. In their experiments, they illustrated that the correlations among the news (encoded in the constructed graph) were more effective than the textual content of the news for predicting their credibility. Similarly, in [6], a graph of news articles was constructed, encoding their correlation. The graph is then used directly by a graph convolutional network for credibility inference. Our previous research [5] adopted the similar idea of exploiting the correlation between news articles. However, the correlation was exploited via mean-field layers derived from MRF. This work extends our previous research [5] by not only considering the correlation between news articles but also the correlation between users involved with the same article. In addition, we show the equivalence of the mean-field layers with popular graph convolutional layers [7] in smoothing the characteristics of nodes within the same cluster, which explains the effectiveness of the proposed mean-field layers.

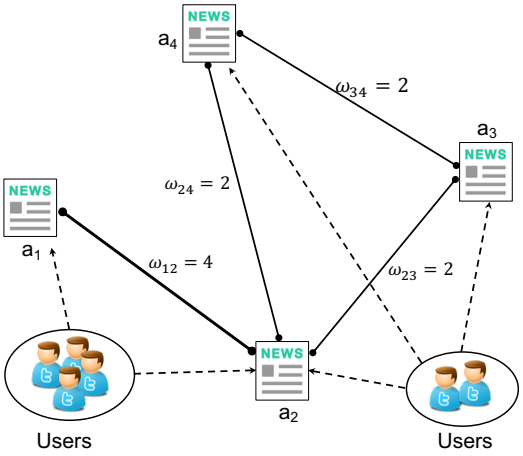

FIGURE 1: The construction of an article graph. A node represents an article and connections are based on common users. The weight of each connection indicates the number of common users.

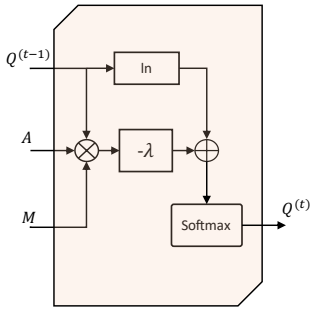

FIGURE 2: The structure of a mean-field layer that smooths out the label probabilities $Q^{(t-1)}$ of an arbitrary model. A and M denote the adjacency matrix of the graph of news articles and the compatibility matrix, λ is a constant, and t indicates the layer order.

III. MULTIVIEW DEEP MARKOV RANDOM FIELD MODEL

In this section, we first show how the correlation between news articles can be exploited using a deep MRF, which leads to the formulation of mean-field layers. We subsequently describe how these layers are used to create novel learning architectures for Fake News detection. Additionally, we describe the details of the considered features and their extraction procedure.

A. CORRELATION EXPLOITATION WITH DEEP MRF

As discussed in Section II, the correlation among news articles has been proven effective in many tasks including breaking news detection and fake news detection. To consider this correlation, a graph of articles is created. In this graph, nodes represent the articles and edges are formed based on the number of common associated social media users. Figure 1 illustrates how such a graph is created. Let $G = (V, E)$ denote the undirected article graph, where V ($|V| = n$) is the set of nodes and E is the set of edges. Let $\mathcal{L}$ denote the set of labels, $|\mathcal{L}| = s$. Let $A \in \mathbb{R}^{n\times n}$ be the symmetric adjacency matrix of graph G such that A_{ij} is equal to the weight of edge $(i, j) \in E$; the adjacency matrix captures the correlation of articles. Let $X = \{X_k\}_{k=1}^{n}$ define the set of random variable representing the labels of nodes of G. In our prior work [5], we introduced a Markov Random Field based model, where the distribution $P(X)$ can be estimated by:

$$P(X = x) = \frac{1}{Z}\exp(-E(x)) \tag{1}$$

$$= \frac{1}{Z}\exp\left\{\sum_{k\in V}\Phi(x_k^u) + \lambda\sum_{k,l\in\mathcal{N}}\Psi(x_k^u, x_l^v)\right\} \tag{2}$$

In Eq. (1), Z is the factor to ensure a valid distribution and $E(x)$ is the energy of the MRF, which can be decomposed into two components, the aggregated unary potential (i.e., the first term in Eq. (2)) and the aggregated pairwise potential (i.e., the second term in Eq. (2)). More precisely, $\Phi(x_k^u)$ is the cost of assigning label $\mathcal{L}_u$ to node k, and $\Psi(x_k^u, x_l^v)$ measures

the cost of assigning labels $\mathcal{L}_u$ and $\mathcal{L}_v$ to node k and node l, respectively. $P(X)$ is then approximated using a simplified factorization assumption $P(X) \approx \mathcal{Q}(X) = \prod_{k \in V} q_k$ such that:

$$q_k^u = \frac{1}{Z_k} \exp\left\{ -\Phi(x_k^u) - \lambda \sum_{l \in \mathcal{N}_k} \alpha(k,l) \sum_{v \in \mathcal{L}} q_l^v \mu(u,v) \right\}. \quad (3)$$

Equation (3) represents an iterative update rule, which we refer to as *mean-field update* rule. In Eq. (3), $Z_k = \sum_{u \in \mathcal{L}} q_k^u$, where q_k^u represents the probability that node k is assigned label u, $\Phi(x_k^u)$ is the *unary potential* and is given by $\Phi(x_k^u) = -\ln P(X_k = \mathcal{L}_u)$. The second term in Eq. (3) is the *pairwise potential*, representing the correlation between node k and node l. $\alpha(k,l) = A_{kl}$ is the weight of the edge (k,l), and $\mu(u,v)$ is *label compatibility* representing the discrepancy between the two labels, namely that $\mu(u,v) \in \{0,1\}, \forall u,v$, and $\mu(u,v) = 1$ if $u \neq v$.

Let $Q \in \mathbb{R}^{n \times s}$ be the matrix containing entries q_k^u, which are the output probabilities of a model such as a neural network. It follows that $\Phi = -\ln(Q)$. We denote by $M \in \mathbb{R}^{s \times s}$ the matrix with the label compatibility entries $\mu(u,v)$. As matrix M is symmetric, Eq. (3) can be re-written in a matrix form as:

$$Q^{(t)} = \text{softmax}\Big(\ln(Q^{(t-1)}) - \lambda A Q^{(t-1)} M\Big), \quad (4)$$

with t denoting the time step. Using Eq. (4), we design a mean-field layer, as illustrated in Fig. 2. Hence, t also indicates the t-th mean-field layer, which has $Q^{(t-1)}$ as an input and $Q^{(t)}$ as output. Multiple mean-field layers can be stacked together to obtain a higher level of smoothness of output probabilities. We observe that a mean-field layer acts similarly to a graph convolutional (GCONV) layer (see [7]) in that the two layers encourage the agreement of nodes in the same cluster. Specifically, the GCONV layer operates on node feature vectors and encourages the nodes in a cluster to obtain similar representations. This eventually helps in assigning similar labels for the nodes that belong in the same cluster (see [43]). Similar to a GCONV layer, a mean-field layer encourages the smoothing of output probabilities; however, this layer works directly on the output probabilities (i.e., $Q^{(t)}$). Specifically, the product $S = AQ^{(t-1)}M \in \mathbb{R}^{n \times s}$ represents the aggregated discrepancy of the nodes with regard to their neighboring nodes; hence, a small value of S_{ku} will increase the confidence of assigning label $\mathcal{L}_u$ to node k and vice versa. This means that node k is more likely to have label $\mathcal{L}_u$ if its neighboring nodes also have label $\mathcal{L}_u$. Eventually, nodes within the same cluster (i.e., having the same label) tend to have similar output probabilities. However, stacking too many layers leads to over-smoothing, which may reduce the performance of the model [43]. Thus, the number of the mean-field layers T is a hyper-parameter in the proposed method.

B. MULTIVIEW DEEP MRF FOR FAKE NEWS DETECTION

Fake news typically has special language patterns (e.g., exaggeration and rhetoric) [44] and is being shared by unreliable users (i.e., users with a history of sharing unreliable news) [6]. Moreover, the reaction of social media users towards fake news tends to be different compared to the reaction towards real news [45]. With that in mind, we design a Generic Deep MRF Neural Network architecture for detecting Fake News, referred to as GDMFN. The model exploits the aforementioned observations and also the correlation between news articles.

The architecture of the GDMFN model is presented in Fig. 3. It consists of three sequentially connected components, namely, feature learning, classifier and mean-field. The feature learning component has multiple branches. Each branch transforms a raw input feature to a high level feature (i.e., a vector embedding). The high-level features are then concatenated to obtain a shared representation of the inputs. The shared representation is then passed to the subsequent classifier component. This component consists of several fully connected layers followed by a softmax classifier to produce the class specific probabilities. Finally, these probabilities are passed through the last component that consists of several mean-field layers (see Fig. 2). The aim of this component is to smoothen the class probability values by leveraging the correlation between the news articles.

The GDMFN model can be instantiated by using different sets of features. For instance, in our previous research work [5], our DMFN model leverages four types of features: term frequency-inverse term frequency (TF-IDF), word2vec embeddings, node2vec embeddings, and time series. The input features of the DMFN model are encompassed in the dashed box signified with *Base component*. The *Additional* component is not part of the DMFN architecture. The features in the Base component are described in what follows.

TF-IDF is a weighting scheme widely used in information retrieval and data mining. It has been recently used along with deep learning models leading to promising results in various tasks [5], [46], [47]. TF-IDF evaluates the level of importance of a particular term (e.g., token) for a document belonging to a corpus of documents. The importance increases proportionally to the frequency of the term in the document and it takes also into account the overall frequency of the term in the corpus. We extract TF-IDF features from tweets associated to the news articles. Specifically, tweets associated with an article are grouped to a pseudo tweet document. We then preprocess the documents by removing stop words, URLs, and converting the words into lower case. We then extract the TF-IDF features from the pre-processed tweet documents.

We also exploit the use of word embeddings, namely, the word2vec model [48] (see Fig. 3). Word2vec embeddings capture the semantics of individual terms, which has been proven beneficial in a number of NLP tasks such as entity recognition and relation extraction [49], text classification [50], fact verification [13], etc. We rely on the pre-trained

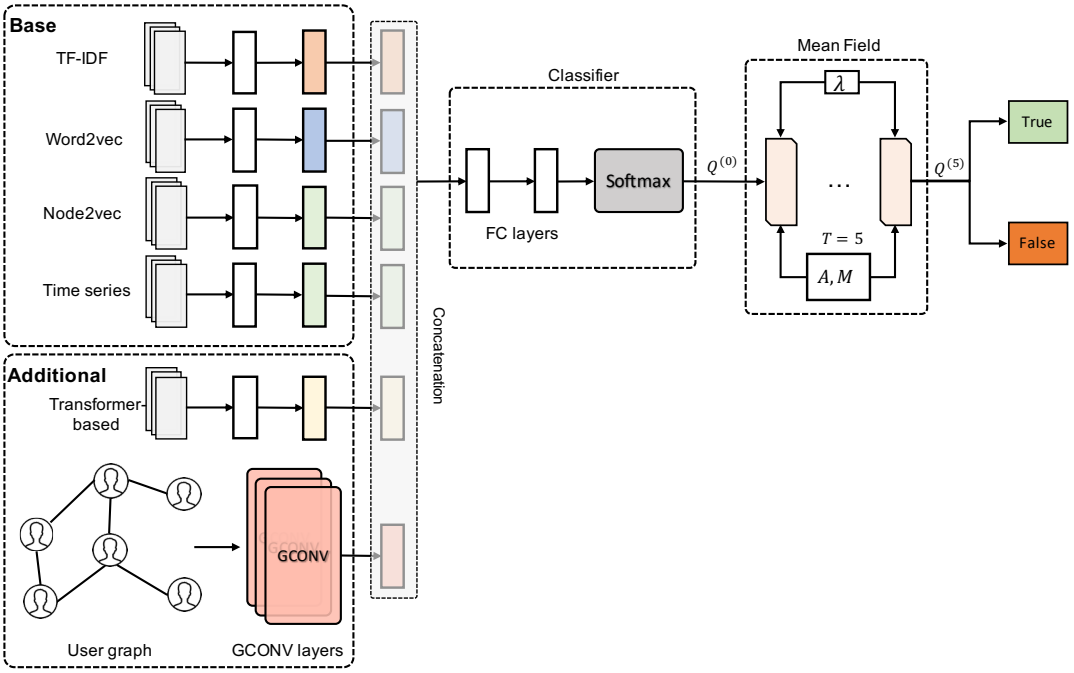

FIGURE 3: The GDMFN model for two-class fake news detection (i.e., True/False) has three main components: feature learning, classifier, and mean field. The feature learning component can has multiple inputs. Here, it is depicted with two subcomponents: Base and Additional. The Base subcomponent has four features (i.e., TF-IDF, word2vec, node2vec, and time series), which form the DMFN model, proposed in our previous work [5]. This work extends our previous research by adding the Additional subcomponent with features based on graph neural networks (GNN branch) and transformer models.

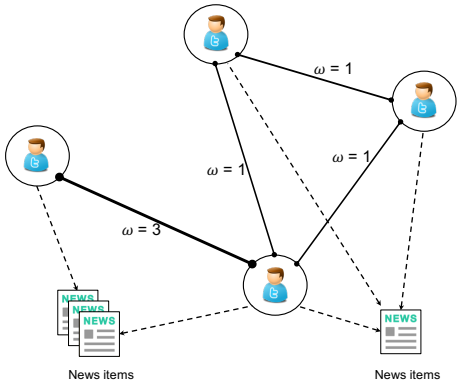

FIGURE 4: The construction of the user graph. A node represents a social media user (i.e., Twitter user) and a connection between two users is based on their common engagements with news articles. The weight of a connection is equal to the number of common engagements.

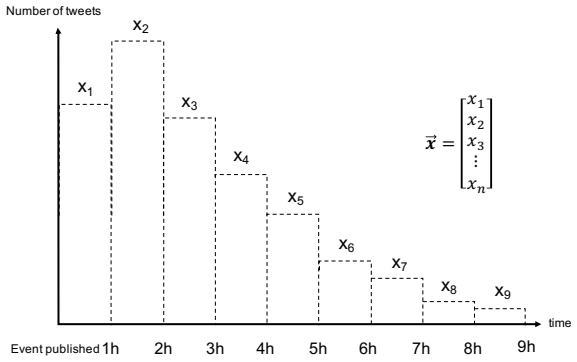

FIGURE 5: Time series feature extraction from tweets associated with a news article. Vector $\vec{x}$ contains elements indicating numbers of tweets associated with an article per hour after the article is shared on Twitter.

word2vec[1] model provided by Google for word2vec feature extraction.

Node2vec is a method proposed in [51] to learn continuous feature representations (embeddings) for nodes in a graph. The embeddings reflect the local connectivity pattern of the graph. We rely on node2vec embeddings to capture the peculiarities of the graph of social media users who are involved with events (a.k.a., articles or news items). We construct the user graph in the following way. First, users engaged with a set of all considered events are collected; these users are considered as nodes of the user graph. Connections between the nodes (a.k.a., users) are created based on common news articles these users interact with, where the weight of a connection between two users is the number of common news articles. Figure 4 illustrates the construction of the user

[1]https://code.google.com/archive/p/word2vec/

graph. The node2vec model [51] is then trained on the user graph, producing node2vec embeddings. The node2vec feature vector of an article is then computed by averaging over the node2vec embeddings of the users interacting with it.

The time series feature captures the number of reactions to news items on social media across time. As shown in [45], the per-hour number of social media posts associated to real news is different to that associated with fake news. Motivated by this, we extract the time when a news item appears on social media and measure the number of associated tweets during subsequent time instants (hours). This produces a time series vector representing the number of reactions per hour to the news item (see Fig. 5).

IV. GRAPH-BASED COMPONENT INTEGRATION

In the DMFN model, the structural information of the user graph is captured using node2vec embeddings [51]. This feature is learned in an unsupervised manner, and thus it is task-agnostic. Furthermore, the node2vec method leverages the shallow architecture of the skip-gram model [48], thus it may not express accurately the rich structural information of the user graph. Therefore, we extend the DMFN model by adding a graph-based model consisting of several graph convolutional layers so as to better express the underlying structure of the user graph. Specifically, we create an extra branch that contains graph convolutional (GCONV) layers [43] (see Fig. 3). The input of this branch is the graph of users interacting with a news article; hence, each article has one connected user sub-graph, which is a part of the entire user graph as described in Section III-B (see Fig. 4).

Let $\tilde{\mathbf{D}}$ be a diagonal matrix, where $\tilde{\mathbf{D}}_{ii} = \sum_{j=1}^{n} \tilde{\mathbf{A}}_{ij}$ and $\tilde{\mathbf{A}}$ is the adjacency matrix defined in Section III-A with self-connections added. We denote by $\mathbf{H}^{(k)}$ and $\mathbf{H}^{(k+1)}$ the input and output matrices of a GCONV layer, respectively. A row in these matrices is the feature vector of a node. The layer is parameterized by matrix $\mathbf{W}$. We employ the propagation rule in Kipf *et al.* [7], which can be written as[2]:

$$\mathbf{H}^{(k+1)} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k)}\mathbf{W} \tag{5}$$

We extract node feature vectors as follows. Since a social media user (e.g., Twitter or Weibo user) corresponds to a node, we employ user profile information for node feature vectors. Specifically, for Twitter users, we collected the *favourites_count*, *followers_count*, *friends_count*, *geo_enabled* status, *statuses_count*, *verified* status, *url* availability, and *screen_name* to form user feature vectors. In addition, the node degree is used as an extra feature. Finally, these vectors are normalized by removing the mean and dividing by the standard deviation (a.k.a., Z score). The Z-score vectors then become the input for the graph-based branch. For other social media platforms (e.g., Weibo), the feature extraction process is similar.

[2]Different propagation rules can be defined for the GCONV layers, such as the propagation rule, $\mathbf{H}^{(k+1)} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}$, proposed in [52], or the propagation rule, $\mathbf{H}^{(k+1)} = \tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1}\mathbf{H}^{(k)}\mathbf{W}$, proposed in [43]. However, we select the propagation rule in [7] as we found it the most effective in our experiments.

TABLE 1: Datasets used for fine-tuning BERT models. Note that the sentiment analysis task has 3 labels, while the other tasks have 2 labels (see Table 2 for details about the labels).

Task	Dataset	Weighted Sampling
Clickbait detection	Article headlines [53]	No
Sentiment analysis	Financial Phrase Bank [54]	No
Bias detection	BASIL (article sentences) [55]	Yes
Toxicity detection	Wikipedia discussions [56]	Yes

V. TRANSFORMER-BASED EXTENSION

In the DMFN model, we have used two types of textual representations, i.e., TF-IDF and word2vec. These representations focus on individual words, ignoring the overall semantics of the entire sequence (e.g., a sentence or a paragraph). To address this shortcoming, we leverage a transformer-based model, BERT [9], to represent the content of the news articles and their associated tweets via its deep bidirectional encoder representations. These representations, which are context-aware and known to perform well in a number of NLP tasks [9], are then used on top of the DMFN model (see Fig. 3). These transformer-based features capture different aspects of the content of news and tweets, which are derived from four individual tasks: (*i*) clickbait detection, (*ii*) sentiment analysis, (*iii*) bias detection, and (*iv*) toxicity detection. We present two approaches to extract the transformer-based features: we either train four individual single-task BERT models (i.e., one for each task) or a unified model for the four tasks; the unified model is called "Tetrathlon". In the following two sub-sections, we describe these models.

A. SINGLE-TASK MODELS

We follow the transfer learning paradigm to fine-tune pre-trained BERT models for the considered tasks. A pre-trained BERT model is taken from the Hugging Face repository[3]. This model was already fine-tuned from the original $\text{BERT}_{\text{base}}$ [9] to classify the sentiment of the IMDB reviews as either positive or negative.

We leverage four datasets to fine-tune the aforementioned BERT model for the considered tasks. For clickbait detection, we use the headlines of articles published by Chakraborty *et al.* [53]. For sentiment analysis, we employ a publicly available dataset from Kaggle[4], which was first introduced in [54]. For bias detection, the BASIL dataset is used [55]. Finally, we use the dataset published by Pavlopoulos *et al.* [56] to fine-tune the BERT model for the toxicity detection task. The details of the considered datasets are described on Table 1. The datasets for clickbait detection and sentiment analysis are balanced and thus a normal training procedure is used; namely, the training is performed with small batches.

[3]https://huggingface.co/textattack/bert-base-uncased-imdb
[4]https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news

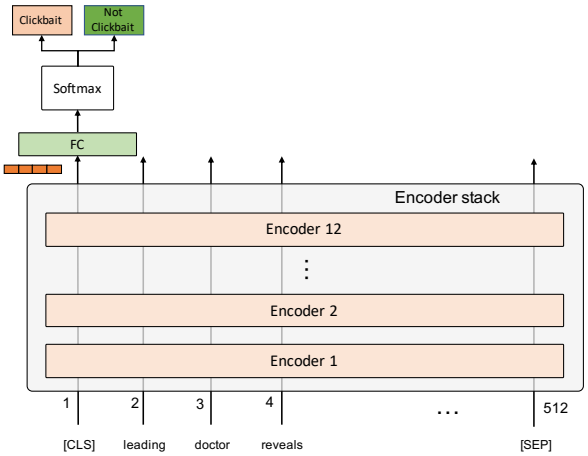

FIGURE 6: Architecture for the clickbait single-task model. The headline input is "Leading Doctor Reveals the No. 1 Worst Carb You Are Eating" and at the beginning of the sentence we add the [CLS] token similar to the work of [9]. The hidden state of the [CLS] token contains information for the entire input sequence and it is used for the classification task. In particular, on top of the [CLS] hidden state, we add a fully connected layer (i.e., denoted by FC) and a softmax classifier to produce the clickbait/not clickbait probabilities.

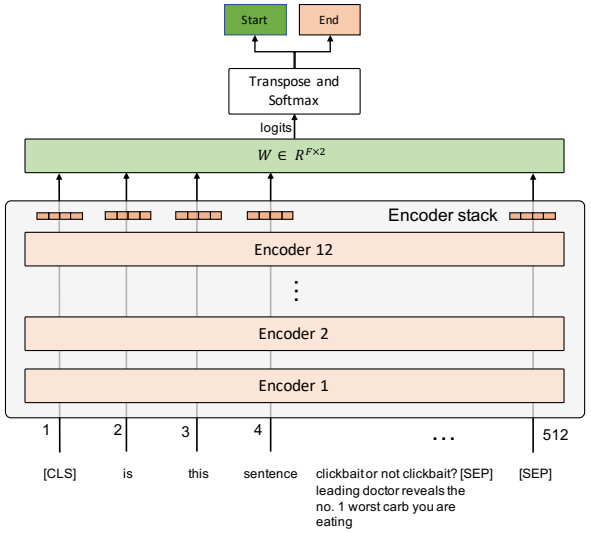

FIGURE 7: Architecture of the Tetrathlon model. Embeddings of all tokens are inputs to a linear layer to obtain *logits*. We then use the `softmax` function and `argmax` functions to obtain start and end indices of the extracted answer.

The bias and toxicity datasets are unbalanced; therefore, we rely on weighted sampling[5].

Figure 6 shows the architecture of a single-task model in the context of clickbait detection. The input is the headline of one clickbait article, which is "*Leading Doctor Reveals the No. 1 Worst Carb You Are Eating*". This input gets forwarded to the BERT model, which is followed by a binary classifier. The classifier determines if the input sentence is clickbait or not clickbait. Note that the classifier only acts on the output that corresponds to the [CLS] token. This token indicates the start of the sentence, and the output (i.e., hidden representation) corresponding to this token can be considered as the representation of the input sentence (a.k.a., sentence embedding).

B. THE TETRATHLON MODEL

In the previous section, we train one individual BERT model for a single task. However, it is known that training a model with multiple tasks (multi-task learning) can help improve performance over single-task models [57]. One noticeable example is the decaNLP model [58], where ten NLP tasks (e.g., question-answering, summarization, machine translation, sentiment analysis) are cast into a question-answering (QA) problem in order to train a unified model. As a result, the model can generalize to completely new tasks though different but related task descriptions [58]. We follow the concept of the decalNLP model to train a unified BERT model, called *Tetrathlon*, which can address the four tasks mentioned earlier (i.e., bias detection, sentiment analysis, clickbait detection, and toxicity detection). We formulate these tasks as a QA problem similar to the decalNLP model. However, different from decalNLP [58] that depends heavily on bidirectional long short-term memory networks (BiLSTMs), our method is based on the BERT model. BERT relies on the self-attention mechanism introduced in the work of transformers (see [8] for more details) and has been successfully used for many NLP tasks to produce state-of-the-art results.

In our formulation, two inputs are needed for the QA problem, including a question and its context. The model outputs the answer extracted from the context. Let us consider a simple example as follows.

Question: *Who do I owe money to?*
Context: *I owe Jack 10 euros.*
Answer: *Jack*

The BERT model is fine-tuned to return a part of the text from either the question or the context as the answer. Thus the considered model returns the start and end indices as output. These indices point to where the answer is in the input that is given to the BERT model. It is worth mentioning that this approach leads to a unique model that can be used for different tasks. This has several advantages. First, no changes are required if the task changes. For instance, the sentiment analysis task has three classes (i.e., negative, neural, positive) while the clickbait detection task has two classes (i.e., clickbait or not clickbait). Hence, a modification needs to be made to the single-task model if the task changes. On the other

[5]Formally, let us consider a dataset $\mathcal{D}$ with a set of labels $\mathcal{L} = \{L_1, L_2, \ldots, L_s\}$. To sample a batch with size B, examples are selected based on the proportion of each type of labels. Specifically, an example x_k with label u is selected with probability: $P(x_k^u) = \frac{1}{|\mathcal{L}|n_u}$, where n_u is number of examples with label u (i.e., $\sum_{u=1}^{s} n_u = |\mathcal{D}|$).

hand, no modifications are needed for the Tetrathlon model to handle both tasks. In addition, training the Tetrathlon model is extremely simple. Datasets of different tasks can be mixed together and used as a unique dataset since there are no distinctions at the output of the model for considered tasks. This could be helpful in case only small datasets are available for specific tasks and we want to leverage the availability of large datasets of other tasks.

A BERT model would slightly modify the question and context by adding special tokens, namely "[CLS]" and "[SEP]", as follows:

[CLS] Who do I owe money to? [SEP] I owe Jack 10 euros. [SEP]

Typically, the question goes before the context. The [CLS] token indicates the start of the input and the [SEP] token is used to separate sentences or to mark the end of the input sequence.

In order to leverage this QA approach, it is important to ask the right questions. For instance, we could ask "*Is this sentence clickbait?*" for the clickbait detection task. However, the BERT model would not know how to answer "*not clickbait*" since "*not clickbait*" is not present in the input sequence. Hence, a better question would be "*Is this sentence clickbait or not clickbait*", which allows the model to extract the right answer since both of the possible classes are present at the question passage. We apply this strategy for all considered tasks. The questions and possible answers for each task are given in Table 2.

The architecture of the model is illustrated in Fig. 7. The question and context are forwarded to the BERT model. Then the output embeddings for input tokens are passed to a linear layer with 2 outputs. The linear layer is the same for every output of the BERT model. These 2 outputs are the start index logits and the end index logits. These logits are determined by using a softmax function. Formally, let $Z \in \mathbb{R}^{N\times F}$ denote the output of the final encoder, where N is the length of the input sequence and F is the dimensionality of the embeddings. Adding a linear layer and a softmax function will result in:

$$Y = \text{softmax}(W^T \cdot Z^T), \tag{6}$$

where $W \in \mathbb{R}^{F\times 2}$. The largest outputs are then the start and end indices of the answer, namely that $y = \text{argmax}(Y, \text{axis} = 1)$. Thus, we have a range $[\text{start\_index}, \text{end\_index}]$ that locates the answer in the input sequence (i.e., the concatenation of the question and the context passage). One problem with this approach is that there is a possibility that the model outputs a nonsensical answer. Specifically, the model behaves correctly if it extracts either "*clickbait*" or "*not clickbait*" from the input. However, the model could extract the sub-sequence "*Reveals the No. 1*" from the input sequence, which does not properly answer the question. A possible approach to address this problem could be to assign the closest valid answer to the predicted sub-sequence as the answer. In addition, we can try to shorten the question to make the predicting of the answer less challenging. We plan to explore these possibilities in future work.

Similar to single-task models, we employ a pre-trained BERT model, which was previously trained by Hugging Face for the question-answering task on the SQuAD1.1 dataset[6]. The pre-trained model is available online[7]. We use the four datasets used for training the single-task models with the same train/test splitting settings (see Section V-A). In addition, we have randomly chosen 320 samples from the training set of each of the four datasets in order to form the validation set. We concatenate the sampled validation instances and we obtain a balanced validation set of 1280 samples. Since the four datasets have different sizes and some of the considered datasets are unbalanced (i.e., toxicity and bias datasets), we use again the weighted sampling strategy similar to the case of training single-task models (see section V-A)[8].

C. FEATURE EXTRACTION

The single-task and Tetrathlon models are able to learn the representation of textual content in order to classify sentences for different tasks. Hence, we can leverage these models to create extra features for the GDMFN model (see Section III). We can use the *logits*[9] as features, but the logits are only 2-dimensional (or 3-dimensional for the sentiment task), thus, they do not contain much information. The authors of [9] propose multiple ways for feature extraction using the $\text{BERT}_{\text{base}}$ model. Specifically, for the $\text{BERT}_{\text{base}}$ model consisting of 12 layers where the output of one layer is a 768-dimensional vector, their feature-based approach includes using the output of the last layer as a feature or to sum the outputs across all 12 layers. In the end, the authors of [9] found that concatenating the outputs of the last 4 layers gives the best results. This means that the feature vector has 3072 dimensions. We follow this approach to extract extra features for the GDMFN model.

Using the aforementioned approach, every token that is forwarded to the BERT model has an associated 3072-dimensional feature vector. For example, if we forward the following sentence:

[CLS] Hello there [SEP],

we will get a 3072-dimensional feature vector for each of the tokens "*[CLS]*", "*Hello*", "*there*" and "*[SEP]*". Following [9], we consider the feature vector corresponding to the [CLS] token as the representation for the entire input

[6]https://rajpurkar.github.io/SQuAD-explorer/

[7]https://huggingface.co/csarron/bert-base-uncased-squad-v1

[8]Formally, let $\mathcal{D} = \{\mathcal{D}_k\}_{k=1}^{n}$ denote the set of considered datasets where a dataset $\mathcal{D}_k$ has $\mathcal{L}(\mathcal{D}_k) = \{L_k^u\}_{u=1}^{m}$ labels. The probability of selecting an example from dataset $\mathcal{D}_k$ with label L_u is given by

$$P(x_k^u) = \frac{1}{|\mathcal{D}_k| \cdot |\mathcal{L}(\mathcal{D}_k)| \cdot n_k^u}, \tag{7}$$

where n_k^u represents the number of examples having label L_u in dataset $\mathcal{D}_k$.

[9]Logits are the output values before applying the softmax function to obtain probabilities

TABLE 2: Questions and possible answers for all the Tetrathlon tasks.

Task	Question	Answer
Clickbait detection	Is this sentence clickbait or not clickbait?	clickbait, not clickbait
Financial sentiment analysis	Is this sentence positive, neutral or negative?	positive, neutral, negative
Bias detection	Is this sentence biased or not biased?	biased, not biased
Toxicity detection	Is this sentence toxic or not toxic?	toxic, not toxic

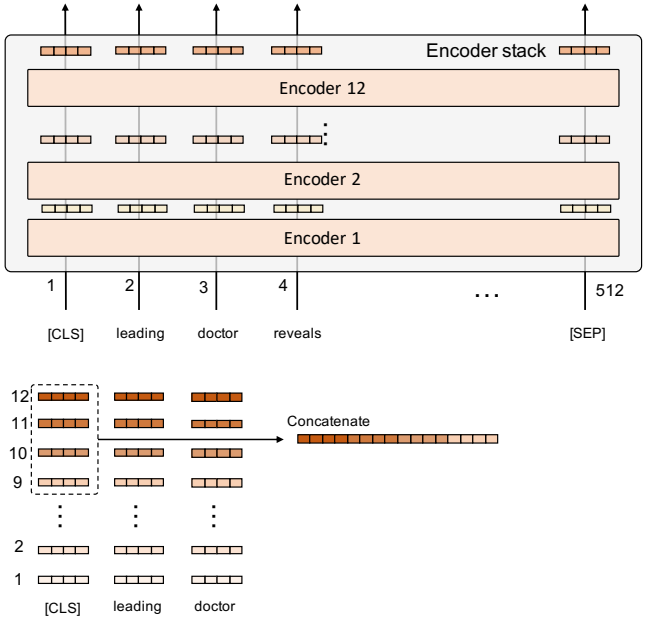

FIGURE 8: Feature extraction for an input sequence. Following the original work [9], the last four embeddings corresponding to token [CLS] are concatenated to obtain the final representation for the input sequence.

sequence. The entire procedure for feature extraction is described graphically in Figure 8.

1) Feature Extraction for Articles

We aim to extract features from articles to enrich the input information for the GDMFN model. This is done by forwarding the content of articles to the fine-tuned BERT models. The 3072-dimensional feature vector corresponding to the [CLS] token is then considered as the representation for that particular article. We consider both the single-task models and the Tetrathlon model. For the single-task models, the article is forwarded to every fine-tuned single-task BERT model. On the other hand, the article content and the corresponding questions are forwarded to the Tetrathlon model. As there are four questions, every article is forwarded to the Tetrathlon model four times, each time with a different question.

2) Feature Extraction for Tweets

Since there are many tweets related to a single article, we pass all the tweets to the BERT models one by one and extract a 3072-dimensional feature vector for each of the tweets[10].

[10]URLs were removed from the tweets before tokenizing them.

The final representation of the tweets associated to one event is found by averaging the feature vectors of the tweets that correspond to that specific event.

VI. EXPERIMENTS

A. EXPERIMENTAL SETTINGS

In order to evaluate the proposed models, we employ three benchmark datasets: Twitter, Weibo, and PHEME [31], [59]. In the Twitter dataset, there are 992 events, 233K Twitter users and 592K tweets. The Weibo dataset is a larger dataset with 4664 events, 2.8M users, and 3.8M posts. An event, described by a news article, is associated with a set of tweets (or posts for the Weibo dataset). For both datasets, an event is associated to a True or a False label. Specifically, a True label means an event has actually happened while a False label suggests that the event has been fabricated. The PHEME dataset contains 5802 discussion threads on Twitter related to 5 main events. Note that an event has multiple threads, and a thread contains a source tweet and many reaction tweets. The total number of tweets for the PHEME dataset is approximately 103K. Similar to the Weibo and Twitter datasets, the PHEME dataset has also binary labels (i.e., *rumor* and *non-rumor*). Following existing works [5], [31], we use a 4-fold cross-validation setting to evaluate the performance of the proposed model on the Twitter and Weibo datasets. For the PHEME dataset, we employ the 5-fold *leave-one* setting presented in [59]. In this setting, for each fold, an event, associated with a number of discussion threads, is kept for testing and the four remaining events are used for training. It is worth noting that cross-validation is typically used for hyper-parameter optimization and model selection. However, to ensure a fair comparison with existing methods, we opt for using this procedure. We evaluate the performance of our models in terms of the accuracy, precision, recall, and F1 score. Since the parameters of the proposed models are initialized randomly, the results may vary at different runs. In order to obtain robust results, we measure the performance of the proposed model over 10 runs, where each run produces an intermediate result of the 5-fold *leave-one* cross validation. We report the average results over the 10 runs along with the standard deviation. The set of benchmark models includes DTC [3], SVM-RBF [4], RFC [30], SVM-TS [31], GRU-2 [31], CAMI [32], and TD-RvNN [60] for the Twitter and the Weibo datasets. Similarly, for the PHEME dataset, we employ Naive Bayes, CRF, and TD-RvNN as benchmark models following [33], [59]. We also compare the proposed method (i.e., the GDMFN model with both Base component and Additional component) against the original DMFN

model from our previous work [5].

B. PARAMETER SETTINGS

Similar to [5], we employ one hidden layer for each feature branch; a hidden layer has 100 hidden units. Likewise, one hidden layer with a dimensionality of 100 is used after the concatenation. The number of mean-field layers is set to $T = 5$, and the pairwise potential coefficient γ (see Eq. (4)) is set to 0.05. Regarding the GNN module, we chose the propagation rule proposed in the work of Kipf *et al* [7] compared to the rest of the approaches indicated in Section IV as we found that this propagation rule performs the best. Two hidden GCONV layers are used, and each of them has 100 dimensions. In order to address over-fitting, we deploy dropout with a drop rate of 0.9. In addition, early stopping is used and we set the maximum number of training epochs to 100. The learning rate is set to 0.001.

C. RESULTS

1) Classification Result

The results for the Twitter and Weibo datasets are shown in Table 3. Unlike the rest of the benchmark models, for our models, we present results that are averaged over 10 runs along with the corresponding standard deviations. Among the benchmark models for these datasets, CAMI [32] is the most effective one, achieving F1 scores of 0.776 and 0.933 for the Twitter and Weibo datasets, respectively. The DMFN model, which our model relies on, outperforms the CAMI model by a noticeable margin (i.e., 0.1 and 0.2 in terms of F1 score) on both datasets. The proposed model, namely the generic DMFN (GDMFN), achieves the best performance on both datasets.

The results on the PHEME dataset are presented in Table 4. The Naive Bayes models are not able to perform well due to the imbalance between the rumor/non-rumor labels in the PHEME dataset, which results in a noticeable difference between the Precision and Recall scores. This leads to overall low F1 scores (i.e., approximately 0.43). Although, the CRF model suffers from the same problem, it is able to perform better in terms of the F1 score evaluation metric. The TD-RvNN method (see Table 4) performs the best among the benchmark methods, and achieves an F1 score of 0.609. Compared to these methods, the original DMFN model performs much better by a large margin for all the performance evaluation metrics. Again, the generic DMFN (GDMFN) achieves the best performance thanks to its capability of exploiting many aspects of the data.

2) Ablation Study

In this section, we evaluate the effectiveness of the extra features and the additional module (see Section IV and Section V for more details) added on top of the original DMFN model. Specifically, in our ablation study, we are able to identify the contribution of the features extracted from *(i)* single-task models (see Section V-A), *(ii)* the Tetrathlon model (see Section V-B), and *(iii)* the GNN module (see Section IV) when they are added to the original set of features used in the DMFN model (see Section III-B). Our naming convention is as follows. Features extracted from the Tetrathlon model are referred to as "Multi" (e.g., "Tweet-Multi") as the Tetrathlon is a multi-task model. It should be noted that "Tweet-Multi" is not a single feature. Instead, it refers to a feature set that consists of four types of features, including bias, sentiment, clickbait, and toxicity. The features extracted from single-task BERT models are denoted with "Single" (e.g., "Tweet-Single"). Similarly, the "Tweet-Single" term refers to the four features mentioned earlier.

Table 5 shows the ablation study results for the Twitter dataset. For ease of comparison, the result for the original DMFN model is also included (i.e., the row "Original"). It can be seen that adding more features generally improves the performance of the proposed model. The GNN module helps the GDMFN model achieve the best performance with an F1 score of 0.792. However, using all features (i.e., row "All") does not guarantee a better performance compared to the original set of features. Instead, the model will be more prone to overfitting as more parameters are included in the model.

The ablation study results for the Weibo dataset are summarized in Table 6. Similar to Table 5, the result for the DMFN model with the original feature set is included (i.e., row "Original"). As we were not able to find Chinese datasets for bias detection, toxicity detection, and clickbait detection, only one single-task BERT model for sentiment detection was fine-tuned. Thus for the Weibo dataset, the only new feature extracted from BERT is the sentiment feature (i.e., row "Sentiment"). Similar to the Twitter dataset, adding a new feature or the GNN module produces slightly better results. The best performance on this dataset is achieved when all features are used. In particular, the proposed model achieves the best numbers in terms of accuracy (96.3%), precision (0.963), recall (0.963), and F1 score (0.963).

Tables 7 and 8 show the ablation study results for the PHEME dataset. We study two settings for this dataset, Specifically, *(i)* the normal 4-fold cross validation and *(ii)* 5-fold leave-one settings are considered. For setting *(i)*, adding one feature type (e.g., Tweet-Multi, Tweet-Single) or the GNN module increases the performance of the GDMFN model in terms of all the performance evaluation metrics. The improvement increases when all features are exploited. Similar results could be observed with setting *(ii)*. However, the best performance in that setting is obtained only when the GNN module is added.

VII. CONCLUSION

In this paper, we showed the analogy between mean-field layers and GCONV layers in terms of smoothing the characteristics of nodes in a graph, which explains the effectiveness of the proposed GDMFN model. In addition, we extended the original DMFN model by adding an extra GNN module to exploit the correlation of social media users involved with news articles. Furthermore, we formulated four different

TABLE 3: Fake news detection performance of the proposed model (i.e., GDMFN) in comparison with baseline models. We report the best result for the GDMFN model with the original feature set (i.e., TF-IDF, word2vec, node2vec, time series) and the transformer-based features as well as the GNN module. Our results are calculated by averaging over 10 runs, hence they are more robust than the results of baseline methods.

Model	Twitter				Weibo			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
SVM-RBF [4]	0.715	0.720	0.710	0.709	0.818	0.819	0.818	0.818
DTC [3]	0.718	0.718	0.718	0.718	0.831	0.831	0.831	0.831
RFC [30]	0.728	0.728	0.728	0.728	0.849	0.866	0.849	0.847
SVM-TS [31]	0.745	0.741	0.741	0.740	0.857	0.859	0.858	0.859
GRU-2 [31]	0.757	0.760	0.757	0.771	0.910	0.914	0.910	0.910
CAMI [32]	0.777	0.782	0.777	0.776	0.933	0.933	0.933	0.933
DMFN [5]	0.789 ± 0.006	0.793 ± 0.006	0.789 ± 0.008	0.788 ± 0.007	0.958 ± 0.002	0.959 ± 0.001	0.959 ± 0.002	0.958 ± 0.001
GDMFN (our)	$\mathbf{0.793 \pm 0.007}$	$\mathbf{0.798 \pm 0.008}$	$\mathbf{0.793 \pm 0.010}$	$\mathbf{0.792 \pm 0.007}$	$\mathbf{0.964 \pm 0.002}$	$\mathbf{0.964 \pm 0.002}$	$\mathbf{0.964 \pm 0.002}$	$\mathbf{0.964 \pm 0.002}$

TABLE 4: Results for PHEME dataset with 5-fold leave-one setting in comparison with existing methods.

Model	Precision	Recall	F1
Naive Bayes Content	0.309	0.723	0.433
CRF Content	0.683	0.545	0.606
Naive Bayes Content + Social	0.310	0.723	0.434
CRF Content + Social	0.667	0.556	0.607
TD-RvNN + Social	0.616	0.612	0.609
DMFN	0.668 ± 0.006	0.673 ± 0.008	0.657 ± 010
GDMFN	$\mathbf{0.670 \pm 0.003}$	$\mathbf{0.676 \pm 0.004}$	$\mathbf{0.661 \pm 0.004}$

TABLE 5: Ablation results for Twitter dataset. A row in this table shows the performance of the GDMFN model when a new feature set is added to the original feature set.

Added Feature	Accuracy	Precision	Recall	F1
Original	0.789 ± 0.006	0.793 ± 0.006	0.789 ± 0.008	0.788 ± 0.007
Tweet-Multi	0.792 ± 0.007	0.797 ± 0.008	0.792 ± 0.008	0.790 ± 0.008
Tweet-single	0.791 ± 0.009	0.796 ± 0.009	0.790 ± 0.010	0.789 ± 0.009
Article multi	0.791 ± 0.009	0.796 ± 0.009	0.791 ± 0.009	0.789 ± 0.009
Article single	0.791 ± 0.012	0.795 ± 0.012	0.791 ± 0.013	0.790 ± 0.013
GNN module	$\mathbf{0.793 \pm 0.007}$	$\mathbf{0.798 \pm 0.008}$	$\mathbf{0.793 \pm 0.010}$	$\mathbf{0.792 \pm 0.007}$
All	0.789 ± 0.008	0.791 ± 0.008	0.787 ± 0.009	0.785 ± 0.008

NLP tasks as a QA problem, which we addressed by using either the Tetrathlon or the single-task transformer-based models, enabling unified deep bidirection representations of news articles, which contribute to the ultimate task of fake news detection. Experiments on popular benchmark datasets show that the proposed method consistently improves over the state of the art in fake news detection approaches. While promising results have been obtained, the proposed model is not fully end-to-end, namely that the transformer-based models are fine-tuned separately from the training of the GDMFN model. Hence, our future work will focus on designing a truly end-to-end model, which will simplify the training process.

TABLE 6: Ablation results for Weibo dataset. As Chinese datasets for clickbait detection, bias detection, and toxicity detection are not publicly available, only feature for sentiment analysis is extracted from the corresponding BERT model.

Added Feature	Accuracy	Precision	Recall	F1
Original	0.958 ± 0.002	0.959 ± 0.001	0.959 ± 0.002	0.958 ± 0.001
Sentiment	0.960 ± 0.002	0.960 ± 0.002	0.960 ± 0.002	0.960 ± 0.002
GNN module	0.959 ± 0.002	0.959 ± 0.002	0.959 ± 0.002	0.959 ± 0.002
All	$\mathbf{0.963 \pm 0.002}$	$\mathbf{0.963 \pm 0.002}$	$\mathbf{0.963 \pm 0.002}$	$\mathbf{0.963 \pm 0.002}$

TABLE 7: Ablation results for PHEME dataset with 4-fold cross validation setting, which is identical to the setting used for the Twitter and Weibo datasets.

Added Feature	Accuracy	Precision	Recall	F1
Original	0.844 ± 0.004	0.825 ± 0.005	0.844 ± 0.004	0.832 ± 0.004
Tweet-Multi	0.847 ± 0.003	0.828 ± 0.003	0.847 ± 0.003	0.835 ± 0.003
Tweet-single	0.856 ± 0.003	0.837 ± 0.003	0.856 ± 0.003	0.844 ± 0.003
GNN module	0.853 ± 0.004	0.834 ± 0.004	0.844 ± 0.005	0.839 ± 0.004
All	$\mathbf{0.860 \pm 0.003}$	$\mathbf{0.841 \pm 0.003}$	$\mathbf{0.853 \pm 0.003}$	$\mathbf{0.846 \pm 0.003}$

TABLE 8: Ablation results for PHEME dataset with 5-fold leave-one setting. A row in this table shows the performance of the GDMFN model when the corresponding feature is added to the original feature set.

Added Feature	Accuracy	Precision	Recall	F1
Original	0.699 ± 0.007	0.668 ± 0.006	0.673 ± 0.008	0.657 ± 0.010
Tweet-Multi	0.704 ± 0.004	0.670 ± 0.003	$\mathbf{0.676 \pm 0.004}$	$\mathbf{0.661 \pm 0.004}$
Tweet-single	0.695 ± 0.006	0.664 ± 0.003	0.671 ± 0.003	0.657 ± 0.005
GNN module	$\mathbf{0.713 \pm 0.006}$	$\mathbf{0.687 \pm 0.004}$	0.671 ± 0.006	0.658 ± 0.008
All	0.705 ± 0.006	0.681 ± 0.006	0.667 ± 0.005	0.653 ± 0.006

REFERENCES

[1] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1):22–36, 2017.

[2] B. Doerr, M. Fouz, and T. Friedrich. Why rumors spread so quickly in social networks. Communications of the ACM, 55(6):70–75, 2012.

[3] C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In International conference on World wide web (WWW), pages 675–684, 2011.

[4] F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In ACM SIGKDD workshop on mining data semantics, pages 1–7, 2012.

[5] D. M. Nguyen, T. H. Do, R. Calderbank, and N. Deligiannis. Fake news detection using deep markov random fields. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 1391–1400, 2019.

[6] N. Deligiannis, T. H. Do, D. M. Nguyen, and X. Luo. Deep learning for geolocating social media users and detecting fake news. In NATO IST-160 Specialist's Meeting Big Data and AI. NATO, 2018.

[7] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2016.

[8] A. Waswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Conference on Neural Information Processing Systems (NeurIPS), 2017.

[9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 4171–4186, June 2019.

[10] X. Zhou and R. Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. ACM Computing Surveys (CSUR), 53(5):1–40, 2020.

[11] P. Meel and D. K. Vishwakarma. Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, chal-

lenges and opportunities. Expert Systems with Applications, 153:112986, 2020.

[12] J. Thorne and A. Vlachos. Automated fact checking: Task formulations, methods and future directions. In International Conference on Computational Linguistics, pages 3346–3359, 2018.

[13] G. Bekoulis, C. Papagiannopoulou, and N. Deligiannis. A Review on Fact Extraction and Verification. arXiv:2010.03001, 2020.

[14] G. Bekoulis, C. Papagiannopoulou, and N. Deligiannis. Understanding the impact of evidence-aware sentence selection for fact checking. In Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda, pages 23–28, 2021.

[15] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter. Detection and resolution of rumours in social media: A survey. ACM Computing Surveys (CSUR), 51(2):1–36, 2018.

[16] W. Ferreira and A. Vlachos. Emergent: a novel data-set for stance classification. In Conference of the North American chapter of the association for computational linguistics: Human language technologies (NAACL), pages 1163–1168, 2016.

[17] J. Patro and P. S. Rathore. A sociolinguistic route to the characterization and detection of the credibility of events on twitter. In ACM Conference on Hypertext and Social Media, pages 241–250, 2020.

[18] J. Patro, S. Baruah, V. Gupta, M. Choudhury, P. Goyal, and A. Mukherjee. Characterizing the spread of exaggerated health news content over social media. In ACM Conference on Hypertext and Social Media, pages 279–280, 2019.

[19] J. Patro and S. Baruah. A simple three-step approach for the automatic detection of exaggerated statements in health science news. In Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (EACL), pages 3293–3305, 2021.

[20] A. Vlachos and S. Riedel. Fact checking: Task definition and dataset construction. In ACL 2014 workshop on language technologies and computational social science, pages 18–22, 2014.

[21] W. Y. Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In Annual Meeting of the Association for Computational Linguistics, pages 422–426, 2017.

[22] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. Fever: a large-scale dataset for fact extraction and verification. arXiv:1803.05355, 2018.

[23] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu. FakeNewsNet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media. arXiv e-prints, pages arXiv–1809, 2018.

[24] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein. A stylometric inquiry into hyperpartisan and fake news. arXiv:1702.05638, 2017.

[25] E. Tacchini, G. Ballarin, M. L. D. Vedova, S. Moret, and L. Alfaro. Some like it hoax: Automated fake news detection in social networks. arXiv:1704.07506, 2017.

[26] A. Zubiaga, M. Liakata, R. Procter, G. W. S. Hoi, and P. Tolmie. Analysing how people orient to and spread rumours in social media by looking at conversational threads. PloS one, 11(3):e0150989, 2016.

[27] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah. Real-time rumor debunking on twitter. In ACM International on conference on information and knowledge management (CIKM), pages 1867–1870, 2015.

[28] Z. Zhao, P. Resnick, and Q. Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In International conference on world wide web (WWW), pages 1395–1405, 2015.

[29] K. Shu, S. Wang, and H. Liu. Beyond news contents: The role of social context for fake news detection. In ACM international conference on web search and data mining, pages 312–320, 2019.

[30] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang. Prominent features of rumor propagation in online social media. In IEEE International conference on data mining (ICDM), pages 1103–1108, 2013.

[31] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K. F. Wong, and M. Cha. Detecting rumors from microblogs with recurrent neural networks. In International Joint Conference on Artificial Intelligence (IJCAI), page 3818–3824. AAAI Press, 2016.

[32] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan. A convolutional approach for misinformation identification. In International Joint Conference on Artificial Intelligence (IJCAI), pages 3901–3907, 2017.

[33] J. Ma, W. Gao, and K. F. Wong. Detect rumor and stance jointly by neural multi-task learning. In Companion proceedings of the the web conference 2018, pages 585–593, 2018.

[34] W. Liu and Y. F. Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In AAAI Conference on Artificial Intelligence, 2018.

[35] E. Kochkina, M. Liakata, and A. Zubiaga. All-in-one: Multi-task learning for rumour verification. In International Conference on Computational Linguistics, pages 3402–3413, 2018.

[36] N. Ruchansky, S. Seo, and Y. Liu. CSI: A hybrid deep model for fake news detection. In ACM on Conference on Information and Knowledge Management (CIKM), pages 797–806, 2017.

[37] S. Volkova, K. Shaffer, J. Y. Jang, and N. Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In Annual Meeting of the Association for Computational Linguistics, pages 647–653, 2017.

[38] Y. Yang, L. Zheng, J. Zhang, Q. Cui, Z. Li, and P. S. Yu. TI-CNN: Convolutional neural networks for fake news detection. arXiv:1806.00749, 2018.

[39] T. Zhang, D. Wang, H. Chen, Z. Zeng, W. Guo, C. Miao, and L. Cui. BDANN: BERT-Based Domain Adaptation Neural Network for Multi-Modal Fake News Detection. In International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2020.

[40] J. Zhang, B. Dong, and P. S. Yu. Fakedetector: Effective fake news detection with deep diffusive neural network. In IEEE International Conference on Data Engineering (ICDE), pages 1826–1829, 2020.

[41] A. Freire, M. Manca, D. Saez-Trumper, D. Laniado, I. Bordino, F. Gullo, and A. Kaltenbrunner. Graph-based breaking news detection on wikipedia. In International AAAI Conference on Web and Social Media (ICWSM), 2016.

[42] J. Fairbanks, N. Fitch, N. Knauf, and E. Briscoe. Credibility assessment in the news: do we need to read. In MIS2 Workshop held in conjuction with 11th Int'l Conf. on Web Search and Data Mining, pages 799–800, 2018.

[43] T. H. Do, D. M. Nguyen, G. Bekoulis, A. Munteanu, and N. Deligiannis. Graph convolutional neural networks with node transition probability-based message passing and dropnode regularization. arXiv:2008.12578, 2020.

[44] N. O'Brien, S. Latessa, G. Evangelopoulos, and X. Boix. The language of fake news: Opening the black-box of deep learning based detectors. 2018.

[45] T. H. Do, X. Luo, D. M. Nguyen, and N. Deligiannis. Rumour detection via news propagation dynamics and user representation learning. In IEEE Data Science Workshop (DSW), pages 196–200, 2019.

[46] T. H. Do, D. M. Nguyen, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Multiview deep learning for predicting twitter users' location. arXiv:1712.08091, 2017.

[47] L. Li, L. Xiao, N. Wang, G. Yang, and J. Zhang. Text classification method based on convolution neural network. In IEEE International Conference on Computer and Communications (ICCC), pages 1985–1989, 2017.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (NeurIPS), pages 3111–3119, 2013.

[49] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. Joint entity recognition and relation extraction as a multi-head selection problem. Expert Systems with Applications, 114:34–45, 2018.

[50] Y. Kim. Convolutional neural networks for sentence classification. In Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, October 2014.

[51] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In ACM SIGKDD international conference on Knowledge discovery and data mining, pages 855–864, 2016.

[52] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In AAAI Conference on Artificial Intelligence, 2018.

[53] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pages 9–16, 2016.

[54] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the Association for Information Science and Technology, 65(4):782–796, 2014.

[55] L. Fan, M. White, E. Sharma, R. Su, P. K. Choubey, R. Huang, and L. Wang. In plain sight: Media bias through the lens of factual reporting. In Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6343–6349, November 2019.

[56] J. Pavlopoulos, J. Sorensen, L. Dixon, N. Thain, and I. Androutsopoulos. Toxicity detection: Does context really matter? In Annual Meeting of the Association for Computational Linguistics, pages 4296–4305, Online, July 2020.
[57] S. Ruder. An overview of multi-task learning in deep neural networks. arXiv:1706.05098, 2017.
[58] B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: Multitask learning as question answering. arXiv:1806.08730, 2018.
[59] A. Zubiaga, M. Liakata, and R. Procter. Exploiting context for rumour detection in social media. In International Conference on Social Informatics, pages 109–123. Springer, 2017.
[60] J. Ma, W. Gao, and K. F. Wong. Rumor detection on Twitter with tree-structured recursive neural networks. In Annual Meeting of the Association for Computational Linguistics, pages 1980–1989, Melbourne, Australia, July 2018.

TIEN HUU DO received the BSc in Electrical Engineering from Hanoi University of Science and Technology in 2009. He received the MSc degree in Applied Computer Science in 2016 and the Ph.D. degree in engineering sciences in 2021 from Vrije Universiteit Brussel (VUB), Brussels. He is currently working as a postdoctoral researcher at the Department of Electronics and Informatics (ETRO), VUB, Belgium. His research interests include machine learning, deep learning, graph learning, natural language processing, and data mining.

MARC BERNEMAN received his BSc and MSc in Electrical Engineering from the Vrije Universiteit Brussel (VUB). He also received the BRussel Engineering Alumni (BREA) best master thesis prize in 2020. He is currently completing an Advanced MSc in Nuclear Engineering as part of the Belgian Nuclear higher Education Network (BNEN) program.

JASABANTA PATRO is a postdoctoral researcher at Department of Electronics and Informatics, Vrije Universiteit Brussel. He obtained his Ph.D. in NLP from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. Before that, he received his MTech from Maulana Azad National Institute of Technology, Bhopal, India in 2013 and B.Tech from Parala Maharaja Engineering College, Berhampur, Odisha. His research interests include computational linguistics, social computing and machine learning. Dr. Patro has published papers in the top venues in these fields such as EMNLP, ACL, EACL and ACM HT. Further, he serves as a reviewer or PC member for the venues like EMNLP, ACL and ICWSM.

GIANNIS BEKOULIS obtained his joint bachelor and master degree in Computer Science at the University of Patras in 2012. Then, he worked for two years as a research assistant at Information Technologies Institute (ITI), Center of Research and Technology – Hellas. In 2014 he moved to Paris where he conducted his master studies on Applied Mathematics for Data Science at École Polytechique. On January 2016, he joined the IDLab research group at the Information Technology (INTEC) department of Ghent University, as a Ph.D. student. Since January 2020, he joined ETRO-VUB as a postdoctoral researcher. His current research interests focus on the Natural Language Processing field and in particular the tasks of entity recognition, relation extraction, fact extraction and verification, and identification of fake news.

NIKOS DELIGIANNIS (Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, Patras, Greece, in 2006, and the Ph.D. degree (Hons.) in engineering sciences from Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 2012.

From 2013 to 2015, he was a Senior Researcher with the Department of Electronic and Electrical Engineering, University College London, London, U.K. He is currently an Associate Professor with the Department of Electronics and Informatics (ETRO), VUB, and a Senior Scientist with IMEC, Leuven, Belgium. Since 2021, he serves as the Chair for the EURASIP Technical Area Committee on Signal and Data Analytics for Machine Learning. He has authored or coauthored more than 130 journal and conference publications, five book chapters, and five international patent applications. His current research interests focus on interpretable and explainable machine learning, signal processing, and distributed learning for computer vision, data mining, and natural language processing.

Dr. Deligiannis is a member of the EURASIP. He serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and he was the Lead Guest Editor of the special issue on "Understanding and Designing Deep Neural Networks" at the EURASIP Journal on Advances in Signal Processing