Fake News Detection using Deep Markov Random Fields

Duc Minh Nguyen[⋆†], **Tien Huu Do**[⋆†], **Robert Calderbank**[‡], **Nikos Deligiannis**[⋆†],
[⋆]Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
[†]imec, Kapeldreef 75, B-3001 Leuven, Belgium
[‡]Duke University, Durham, North Carolina
[⋆†]{mdnguyen, thdo, ndeligia}@etrovub.be
[‡]robert.calderbank@duke.edu

Abstract

Deep-learning-based models have been successfully applied to the problem of detecting fake news on social media. While the correlations among news articles have been shown to be effective cues for online news analysis, existing deep-learning-based methods often ignore this information and only consider each news article individually. To overcome this limitation, we develop a graph-theoretic method that inherits the power of deep learning while at the same time utilizing the correlations among the articles. We formulate fake news detection as an inference problem in a Markov random field (MRF) which can be solved by the iterative mean-field algorithm. We then unfold the mean-field algorithm into hidden layers that are composed of common neural network operations. By integrating these hidden layers on top of a deep network, which produces the MRF potentials, we obtain our deep MRF model for fake news detection. Experimental results on well-known datasets show that the proposed model improves upon various state-of-the-art models.

1 Introduction

The term "fake news" refers to news articles that is intentionally and verifiably false (Shu et al., 2017). The problem of fake news has existed since the appearance of the printing press, but only gained a lot of momentum and visibility during the age of social media. This is due to the large audience, easy access and fast dissemination mechanism of social media, where more and more users are consuming news in a daily basis (Shu et al., 2017). Traditional methods for verifying the veracity of news that rely on human experts, despite being reliable, do not scale well to the massive volume of news nowadays. This renders the automatic detection of fake news on social media an important problem, drawing a lot of attention from both the academic and industrial communities.

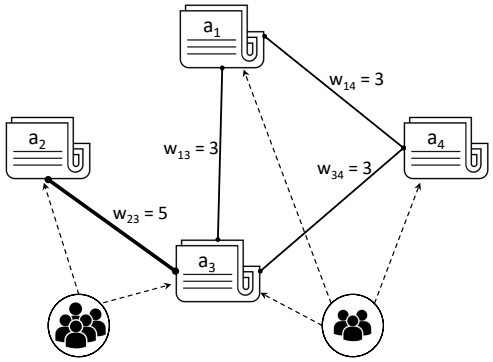


Figure 1: Modeling the relationship among news articles (or events): the dash lines represent engagements of users to articles, the solid lines represent the relationships between articles with weights determined by the number of common engaged users.

The recent literature has witnessed the success of deep learning models in detecting fake news on social media (Ma et al., 2016; Yu et al., 2017; Ruchansky et al., 2017; Rashkin et al., 2017; Ma et al., 2018a; Kochkina et al., 2018). By leveraging the capability of deep networks in learning high-level representations, these models have achieved state-of-the-art performance on various benchmark datasets. Nevertheless, one limitation of existing deep-learning-based methods is that they often ignore the correlations among news articles, which have been proved to be effective for analysing online events (Freire et al., 2016; Fairbanks et al., 2018).

To overcome this limitation, we aim at a model that leverages the capability of deep neural networks while effectively incorporating the correlations among articles when determining their credibility. To this end, we first model the relationship between two articles by the number of the com-

Proceedings of NAACL-HLT 2019, pages 1391–1400
Minneapolis, Minnesota, June 2 - June 7, 2019.

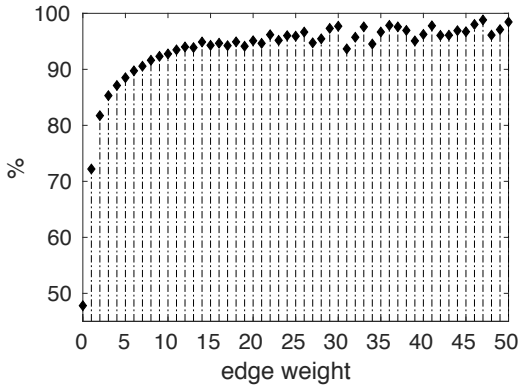


Figure 2: Percentage of edges having certain weights, which connects two articles with the same labels on the news graph constructed from the Weibo dataset (Ma et al., 2016).

mon users that engage to them, e.g., by means of tweeting, re-tweeting, commenting. An example is illustrated in Fig. 1: the articles a_2 and a_3 have a strong relationship as they are engaged to by 5 common users, whereas, there is no relationship between the articles a_1 and a_2 because there is no user engage to both of them. With this modeling, we can construct a *news graph*, where each node corresponds to an article and an edge encodes the relationship between two articles.

Our underlying assumption is that if there exists a strong relationship between two articles, they are likely to share the same labels. To verify this assumption, we calculate the percentages of the edges that connect two articles with the same labels, among those whose weights are equal to certain values, and plot the results for the news graph constructed from the Weibo dataset (Ma et al., 2016) in Fig. 2. It is clear from Fig. 2 that the higher the edge weight, the more likely it is that the corresponding articles share the same labels (*fake* or *true*). Similar patterns are observed on other datasets such as the Twitter (Ma et al., 2016) and the PHEME (Zubiaga et al., 2017) datasets. Evidently, these results support our assumption.

In order to incorporate the correlations among news articles, we formulate fake news detection as an inference problem in a Markov random field (MRF). Our motivation behind this formulation is to leverage the capability of MRF in capturing dependencies among random variables. We solve the resulting inference problem using the mean-field algorithm (Koller and Friedman, 2009). We then propose a method to unfold this algorithm into hidden layers that can be integrated on top of a deep network that computes the potentials of the MRF. By doing this, we obtain our deep MRF model for detecting fake news, referred to as the DMFN model. To the best of our knowledge, this is the first integration of deep networks and MRF for detecting fake news. Our main contributions are as follows:

- We formulate fake news detection as an inference problem in an MRF model. This allows us to incorporate the correlations among news articles when deciding their credibility.
- We propose a method to unfold the mean-field algorithm into specially-designed neural network layers, and build a deep MRF model for detecting fake news.
- We carry out comprehensive experiments on widely-used datasets collected from popular social networks. Experimental results show the effectiveness of the proposed model compared to various state-of-the-art models.

The remainder of the paper is organized as follows: we review the related work in Section 2, and describe our formulation of fake news detection as an inference problem MRF in Section 3. In Section 4, we describe our model in detail. We present our experimental studies in Section 5 and finally draw the conclusions in Section 6.

2 Related Work

Early work in fake news detection focused on finding a good set of features that are useful for separating fake news from genuine news. Linguistic patterns, such as special characters, specific keywords and expression types, have been explored to spot fake news (Castillo et al., 2011; Liu et al., 2015; Zhao et al., 2015). Different feature types have also been considered, such as the characteristics of users involved in spreading the news, e.g. the number of followers, the users' ages and genders (Castillo et al., 2011; Yang et al., 2012), and the news' propagation patterns (Castillo et al., 2011; Kwon et al., 2013). Instead of relying on a single feature type, existing works normally made use of multiple types of feature at the same time.

Recent years have witnessed the use of deep learning for fake news detection. The idea is to leverage deep neural networks to overcome the limitation of the *shallow* hand-craft features employed in the earlier methods (Ma et al., 2016).

Many works proposed to represent news articles as multivariate time series using the timestamp information and to formulate fake news detection as a sequence classification problem (Ma et al., 2016; Yu et al., 2017; Ma et al., 2018a; Liu and Brook Wu, 2018; Kochkina et al., 2018). As it is common in the literature to utilize multiple types of features in detecting fake news, deep networks with multiple branches to incorporate various feature types were also proposed (Ruchansky et al., 2017; Volkova et al., 2017; Yang et al., 2018). In general, deep learning based methods yield higher accuracy compared to shallow-feature-based approaches, leading to state-of-the-art performance.

The existing deep-learning models, however, often ignore the correlations among news articles which haven been shown to be effective cues for analysing online news and events (Freire et al., 2016; Fairbanks et al., 2018). Freire et al. proposed a method to detect breaking news on Wikipedia by exploring the graph of related events (Freire et al., 2016). This graph is built by connecting any pair of pages on Wikipedia which were edited by the same users in a small time window. In (Fairbanks et al., 2018), a graph of news was constructed by connecting the corresponding web pages using the links between them in form of `html` tags. The credibility of the news was assessed by applying the loopy belief propagation algorithm to perform semi-supervised learning on this graph. In their experiments, the authors showed 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.

In this work, we propose a deep MRF model for fake news detection. Apart from leveraging the power of deep networks, our model incorporates the correlations among news articles when determining their credibility. In this regard, our method shares a similar motivation with the graph-based methods for social event analysis as mentioned above. Nevertheless, while these graph-based methods do not utilize any information of the news articles beyond the labels, our model is more generic and allows incorporating an arbitrary number of features.

3 Fake News Detection on MRF

We employ a Markov random field (MRF) model (Freeman et al., 2000) to incorporate the correlations among the events due to its capability in capturing dependencies among random variables. To this end, we first construct an event graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ the set of vertices and $\mathcal{E}$ the set of edges as described in Section 1. We then define an MRF model over $\mathcal{G}$. In this model a node k is associated with a random variable X_k, which represents the label of the k-th event. The random variables $X_k, \forall k \in \{1, \dots, n\}$ have domain $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2 \dots \mathcal{L}_s\}$, which represent the s possible labels. For notation brevity, we refer to the nodes in $\mathcal{V}$ by their indices, namely, $k \in \{1, \dots, n\}$. We are interested in inferring the distribution $P(X)$ of the MRF, from which the labels for the events can be obtained. In the MRF, the probability $P(X = \mathrm{x})$, with x a set of values of the random variables, is given by (Koller and Friedman, 2009)

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

with Z the partition function ensuring a valid distribution and $E(\mathrm{x})$ the energy of the MRF, which has the following form

$$E(\mathrm{x}) = \sum_{k \in \mathcal{V}} \Phi\left(\mathrm{x}_k^u\right) + \lambda \sum_{k,l \in N} \Psi\left(\mathrm{x}_k^u, \mathrm{x}_l^v\right). \tag{2}$$

In (2), N is the set of nodes that are connected in the MRF . The *unary potential*, $\Phi\left(\mathrm{x}_k^u\right)$, measures the cost of assigning the label $\mathcal{L}_u$ to the node k, while the *pairwise potential*, $\Psi(\mathrm{x}_k^u, \mathrm{x}_l^v)$, measures the cost of assigning the nodes k and l respectively the labels $\mathcal{L}_u$ and $\mathcal{L}_v$. As such, the pairwise potentials capture the dependencies among the nodes of the MRF. λ is a hyperparameter.

As exactly computing $P(X)$ is intractable, we employ the mean-field algorithm (Koller and Friedman, 2009) to approximate $P(X)$ by a fully-factorized proposal distribution $Q(X) = \prod_{\forall k \in \mathcal{V}} Q_k(X_k)$. $Q_k(X_k = \mathcal{L}_u)$ is the probability that the node k have the label $\mathcal{L}_u$ according to the distribution Q_k. Denote $Q_k(X_k = \mathcal{L}_u)$ as q_k^u, the mean-field algorithm iteratively calculates $q_k^u, \forall k \in \{1, \dots, n\}$, $u \in \{1, \dots, s\}$ according to (Koller and Friedman, 2009)

$$q_k^u = \frac{1}{Z_k} \exp\left\{ -\left(\Phi\left(\mathrm{x}_k^u\right) + \lambda \sum_{l \in N_k} \sum_{v \in \mathcal{L}} q_l^v \Psi\left(\mathrm{x}_k^u, \mathrm{x}_l^v\right) \right) \right\}, \tag{3}$$

with N_k the set of nodes connected to k and $Z_k = \sum_{u \in \mathcal{L}} q_k^u$ the normalization factor ensuring q_k^u, $\forall u \in \{1, \ldots, s\}$, add up to 1. The generic mean-field update in (3) requires the unary potentials and pairwise potentials. In Section 4, we show how we realize these potentials.

4 Deep MRF Model for Fake News Detection

In this section, we present our realizations of the unary and pairwise potentials, with which we obtain the final mean-field update equation. After that, we present a method to unfold the mean-field update into specially-designed neural network layers, and describe our deep MRF model for fake news detection.

4.1 Computing the Unary and Pairwise Potentials

We compute the unary potential $\Phi(\mathrm{x}_k^u)$ of the MRF as the negative log likelihood:

$$\Phi(\mathrm{x}_k^u) = -\ln p(X_k = \mathcal{L}_u), \tag{4}$$

with $p(X_k = \mathcal{L}_u)$ the likelihood that X_k has label $\mathcal{L}_u$. As such, $\Phi(\mathrm{x}_k^u)$ will be high if this node is not likely to have the label $\mathcal{L}_u$, and vice versa. The likelihood $p(X_k = \mathcal{L}_u)$ is computed as $p(X_k = \mathcal{L}_u) = \mathcal{F}_\theta(\mathcal{O}_k)$, where $\mathcal{F}_\theta$ is a non-linear function represented by a deep neural network parameterized by θ and O_k is the observations, i.e, features associated to the k-th event. The design of this network is described later in Section 4.

The pairwise potential is computed by

$$\Psi(\mathrm{x}_k^u, \mathrm{x}_l^v) = a(k,l) \times \mu(u,v), \tag{5}$$

where $a(k,l)$ is the weight of the edge between the nodes k and l, namely, $a(k,l)$ represents how strong the relationship between the nodes k and l is; and $\mu(u,v)$ is the *label compatibility*, which is calculated using the Pott model (Boykov et al., 1998):

$$\mu(u,v) = \begin{cases} 1, & \text{if } u \neq v, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Substituting the pairwise potential in (5) into (3), we obtain the final mean-field update equation:

$$q_k^u = \frac{1}{Z_k} \exp\Bigg\{ -\Phi\left(x_k^u\right) - \lambda \sum_{l \in N_k} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu\left(u,v\right) \Bigg\}. \tag{7}$$

We refer to the term $\sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v)$ in (7) as the *compatibility transform* step, and to the term $\sum_{l \in N_k} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu\left(u,v\right)$ summing up information from the nodes connected to the node k as the *message passing* step.

4.2 Mean-Field Updates as Neural Network Layers

We now describe a method to implement the update equation in (7) using operations that are common in the deep learning literature. Abusing the notation, we denote also by $Q \in \mathbb{R}^{n \times s}$ a matrix, with an entry $Q_{k,u}$ corresponding to the value q_k^u, i.e., the probability that the node k has the label $\mathcal{L}_u$. Denote by $A \in \mathbb{R}^{n \times n}$ the adjacency matrix, containing all the edge weights in $\mathcal{G}$. We set $A_{k,k} = 0, \forall k \in \{1, \ldots, n\}$. We further denote by $M \in \mathbb{R}^{s \times s}$ a square matrix whose the entry $M_{u,v}$ corresponds the label compatibility $\mu(u,v)$ calculated using (6), and by $\Phi \in \mathbb{R}^{n \times s}$ the matrix containing the unary potentials $\Phi(\mathrm{x}_k^u)$, for all $k \in \{1, \ldots, n\}$ and $u \in \{1, \ldots, s\}$. Φ is calculated by taking the logarithm of Q element-wise.

4.2.1 The Compatibility Transform Step

The compatibility transform in (7) can be performed via a 1-D convolutional layer applied on the matrix Q. This convolutional layer has s filters of kernel size $1 \times s$. The weights of the u-th filter are set equal to the values along the u-th row of M. We do not employ any padding and set the stride to 1. When applying this operation, the u-th filter slides vertically across Q, and calculates the inner product between its weights and the rows of Q. The output is denoted as $\mathcal{Q}^{'} \in \mathbb{R}^{n \times s}$, with an entry $Q_{l,u}^{'}$ given by

$$\begin{aligned} Q_{l,u}^{'} &= \sum_{v \in \mathcal{L}} Q_{l,v} \times M_{u,v} \\ &= \sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v), \end{aligned} \tag{8}$$

for all $l \in \{1, \ldots, n\}$, $u \in \{1, \ldots, s\}$, which is equal to the result of the compatibility transform step in (7).

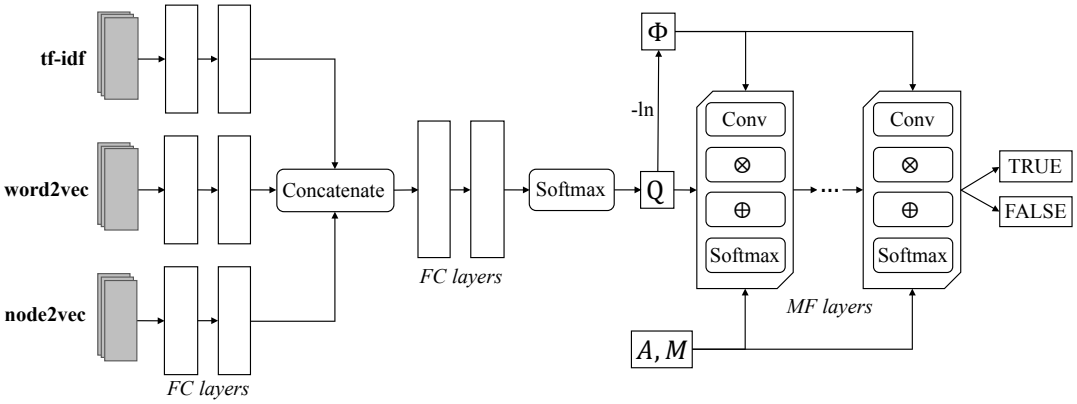


Figure 3: The architecture of the DMFN model: the first block consists of three feature branches, each with configurable numbers of fully-connected (FC) layers to process a type of feature, a concatenation layer and a number of FC layers with softmax activation function at the end to produce labels' probability matrix Q; Each *mean-field* (MF) layer consists of four main operations, namely convolution (conv), matrix multiplication ($\otimes$), point-wise addition ($\oplus$) and the softmax function. T MF layers are stacked one after another to implement the mean-field algorithm with T iterations. All MF layers share the same set of parameters, namely, the matrix of the unary potentials Φ, which is computed from Q, the adjacency matrix A encoding the relationships among the input events, and the compatibility matrix M.

4.2.2 The Message Passing Step

Given the adjacency matrix $A \in \mathbb{R}^{n \times n}$ and the output $Q^{'} \in \mathbb{R}^{n \times s}$ from the compatibility transform step, the message passing step can be performed simply by multiplying A with $Q^{'}$. This multiplication results in $Q^{''} \in \mathbb{R}^{n \times s}$, in which:

$$
\begin{aligned}
Q^{''}_{k,u} &= \sum_{l=1}^{n} A_{k,l}.Q^{'}_{l,u} \\
&= \sum_{l=1}^{n} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v). \qquad (9)
\end{aligned}
$$

As $A_{k,l} = 0$ if $l \notin N_k$, the operation in (9) is equivalent to the message passing step in (7).

4.2.3 The Mean-field Layer

To finish the update in (7), one needs to element-wise multiply $Q^{''}$ with λ, add Φ and negate the result. The exponential and normalization can be performed jointly using the *softmax* function, resulting in the new matrix $Q \in \mathbb{R}^{n \times s}$ whose entries correspond to the values of q_k^u after one mean-field iteration, for all $k \in \{1, \ldots, n\}$, $u \in \{1, \ldots, s\}$. We can consider these operations together with those implementing the compatibility transform and the message passing steps, as the operations of specially-designed a neural network layer, which we call the mean-field (MF) layer. As all the component operations are differentiable, the operation implemented by the MF layer is also differentiable. Each MF layer, hence, implements one iteration of the mean-field algorithm. Clearly, by stacking T MF layers, we can implement the mean-field algorithm with T iterations.

4.3 The DMFN Model

4.3.1 Model Architecture

The architecture of the DMFN model is illustrated in Figure 3, with two blocks: the first block corresponds to the deep network $\mathcal{F}_\theta$ that produces the unary potentials, and the second block is composed of T MF layers, which implements the T-iteration mean-field algorithm. The deep network $\mathcal{F}_\theta$ takes as inputs several feature types extracted to represent the given set of events. Each feature type is processed by a feature branch with a number of fully-connected (FC) layers. The outputs of the last layers in all feature branches are concatenated to produce high-level representations for all the events. These representations are then fed to another set of FC layers and a softmax function to produce the label probabilities. These label probabilities form the matrix Q, as described in Section 4.2. All the FC layers in the model are followed by a batch normalization layer (Ioffe and Szegedy, 2015), a ReLU activation function (Glorot et al., 2011), and the dropout regularization (Srivastava et al., 2014).

In the second block, all the MF layers share the same parameters, namely, the matrix Φ, the adjacency matrix A that encodes the relationships

among the given events, and the label compatibility matrix M. The first MF layer takes as input the matrix Q, whereas, the later MF layers operate on the outputs from their previous MF layers. The output after the last MF layer is the final label probabilities predicted for the given events. It should be noted that the parameters of the MF layers are pre-computed and shared. As such, adding MF layers after the layers in the first block does not increase the the risk of overfitting of the model.

4.3.2 Feature Extraction

We extract multiple types of features to capture the characteristics of the events, namely, the textual contents and social engagements. For the textual features, we first group all the tweets related to an event into a document. We pre-process the documents by removing stop words, converting the words into lower case and tokenizing them. From the pre-processed documents, we extract the *term frequency - inverse document frequency* (tf-idf) (Wu et al., 2008), and *word2vec* (Mikolov et al., 2013) feature vectors. We use the word2vec model pre-trained on the Google News dataset to map each word in a document to a $300-$dimensional embedding vector, and then obtain the embedding for the document by taking the average of the word embeddings.

We use a graph embedding technique to capture the social engagements associated to the events. Concretely, we first build a graph of users from the given dataset. Two users are connected by an edge if they engage to at least one event in common, with the edge's weight determined by the total number of common engaged events. We then employ the *node2vec* algorithm (Grover and Leskovec, 2016) to learn an embedding for each user. The social engagement feature of an event is then calculated by taking the average of the embeddings of all users engaged to it. It is worth noting that the graph of users used in this step is different from the graph of events, constructed as described in Section 1.

4.3.3 Training and Testing the DMFN Model

We employ the weighted cross entropy loss function to train the DMFN model. When calculating the loss, the weight given to the training samples of a particular label is inversely proportional to the number of samples in the current batch which have this label. This technique is highly beneficial when dealing with imbalanced dataset, e.g., the PHEME dataset (Zubiaga et al., 2016). The model's parameters are learned by using the SGD algorithm with the Adam parameter update (Kingma and Ba, 2015).

At the testing stage, we select for the testing event k the label $\mathcal{L}_{\hat{u}} \in \mathcal{L}$ with $\hat{u}$ determined by $\hat{u} = \arg\max_{u \in \{1,\dots,s\}} q_k^u$. As we want to utilize the correlations among events in the prediction, we provide an adjacency matrix encoding the relationships among a set of events and run a forward pass with all their features as input. Without the adjacency matrix or when setting the adjacency matrix to the identity matrix, the model predicts the labels for each events without considering their correlations.

5 Experiments

5.1 Experimental Settings

We employ three well-known benchmark datasets, namely the Twitter, Weibo (Ma et al., 2016) and PHEME datasets (Zubiaga et al., 2017) for our experiments. The Twitter dataset consists of 992 events, involving 233.7 thousand users and 592.4 thousand tweets. The Weibo dataset is larger, with $4,664$ events, 2.8 million users and 3.8 million posts. Events in these datasets are labeled as either *fake* or *true*, and the two labels are relatively balanced. The PHEME dataset consists of $5,802$ comment threads collected from Twitter, with approximately 103 thousand tweets in total. This dataset is imbalanced, with $1,972$ threads labeled as *rumour* and $3,830$ threads labeled as *non-rumour*.

5.2 Hyperparameter Selection

For the DMFN model, we employ one hidden layer in each feature branch, and one hidden layer after the concatenation layer, all with 100 hidden units. We train the model for maximum 100 epochs with learning rate 0.001 and stop training early if the validation loss does not improve over the average of those of the previous 25 epochs. To control overfitting, we employ dropout with high dropping rate of 0.9.

We determine the values of λ which balance the weight between the unary and pairwise potentials in the MRF, and of the number of MF layers, T by cross validation on a separate split on the Weibo dataset. First, we fix T to 30, with which the mean-field algorithm is highly likely to converge (Krähenbühl and Koltun, 2011), and experi-

λ	0.005	0.050	0.100	0.500
Weibo	0.958	**0.960**	0.959	0.949

Table 1: Accuracy of the DMFN model when varying λ on the Weibo datasets (Ma et al., 2016).

T	1	5	10	15
Weibo	0.949	0.960	0.960	0.960

Table 2: Accuracy of the DMFN model when varying T on the Weibo datasets (Ma et al., 2016).

ment with different values of λ. The result of this is presented in Table 1.

Fixing λ to 0.05, we experiment with different number of MF layers by varying T. The results are summarized in Table 2. As can be seen from the table, employing multiple MF layers improves the results over just one MF layer. Even though we still observe improvements in the performance with more than 5 MF layers, the difference is small. As a result, we select $T = 5$ as it produces the best trade-off between accuracy and computational complexity.

5.3 Experimental Results

We compare the results of different models in two experimental settings, namely *late detection* and *early detection*. The former setting allows the models to use all the available users posts in the entire time span of the given datasets, whereas in the latter setting, the models are only allowed to use posts that have appeared within a specific deadline (in hours) since the appearances of the events.

5.3.1 Late Detection Setting

We compare the results of the proposed models in the late detection setting with those of reference models, including the decision tree classifier (DTC) (Castillo et al., 2011), the SVM classifier (SVM-RBF) (Yang et al., 2012), the random forest classifier (RFC) (Kwon et al., 2013), the SVM classifier with timeseries features (SVM-TS) (Ma et al., 2015), the 2-layer GRU model (GRU-2) (Ma et al., 2016) and the convolutional neural network (CAMI) (Yu et al., 2017), the Tree-structured Recursive Neural Networks (TD-RvNN) (Ma et al., 2018b), and the CRF and Naive Bayes classifiers in (Zubiaga et al., 2017). The performance of the models is assessed using four metrics, namely, accuracy, average precision, average recall and macro F1 score. Similar to (Yu et al., 2017), on the Twitter and Weibo datasets, we randomly reserve 10% of the samples for parameter tuning and perform four-fold cross validation on the remaining. On the PHEME dataset, we follow the leave-one-event-out approach as in (Zubiaga et al., 2017). We report the average results for the models.

The results for different models on the Twitter and the Weibo datasets (Ma et al., 2016) are shown in Table 3. On these two datasets, we do not include the results of the TD-RvNN model (Ma et al., 2018b) as this model requires a tree-like connections among the tweets to represent an event. As can be seen from the table, the DMFN model consistently outperforms the reference models on both datasets. The results of all the models are better on the Weibo dataset than on the Twitter dataset. This is possibly because number of posts available on the Weibo dataset is much larger than that available on the Twitter dataset.

The results on the PHEME dataset is illustrated in Table. 4. As this dataset is imbalanced, the prediction accuracy is not a good metric for comparison. Similar to (Zubiaga et al., 2017), we focus on the other metrics, namely, precision, recall and macro F1 score in the comparison. The CRF model with content features (Zubiaga et al., 2017) yields the highest precision, whereas the Naive Bayes classifiers yield the highest recalls.. While the reference models are either biased toward high precision or high recall scores, the DMFN model produces more balanced precision and recall, and the best Macro F1 score. The DF-RvNN model (Ma et al., 2018a) also achieves balanced precision and recall, nevertheless its performance is lower than that of the DMFN model in all metrics (with p-values of 0.004, 0.04, 0.03 respectively for the precision, recall and macro f1 scores in a pairwise t-test).

On the PHEME dataset, the average number of tweets per event is 17.8, which is much smaller than those on the Twitter and Weibo datasets (805 and 815 posts per event respectively). The lower number of tweets, and the class imbalance render this dataset highly challenging. As such, the performance of all the considered models on this dataset is lower than that on the Twitter and Weibo datasets. Overall on all the considered datasets, we observe consistent performance of the DMFN model: The variance of the Macro F1 score over 10 repetitions are relatively low, which are equal to

Model	Twitter				Weibo			
	Acc.	Prec.	Rec.	Macro F1	Acc.	Prec.	Rec.	Macro F1
SVM-RBF	0.715	0.720	0.710	0.709	0.818	0.819	0.818	0.818
DTC	0.718	0.718	0.718	0.718	0.831	0.831	0.831	0.831
RFC	0.728	0.728	0.728	0.728	0.849	0.866	0.849	0.847
SVM-TS	0.745	0.741	0.741	0.740	0.857	0.859	0.858	0.859
GRU-2	0.757	0.760	0.757	0.771	0.910	0.914	0.910	0.910
CAMI	0.777	0.782	0.777	0.776	0.933	0.933	0.933	0.933
DMFN	**0.800**	**0.803**	**0.803**	**0.799**	**0.962**	**0.963**	**0.962**	**0.970**

Table 3: Results for different models on the Twitter and Weibo datasets (Ma et al., 2016).

Model	Prec.	Rec.	Macro 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	0.616	0.612	0.609
DMFN	0.667	0.670	**0.657**

Table 4: Results for different models on the PHEME datasets (Zubiaga et al., 2017).

Model	Twitter		Weibo		PHEME	
	Acc.	Macro F1	Acc.	Macro F1	Acc.	Macro F1
DMFN-base	0.763	0.762	0.944	0.944	0.682	0.607
DMFN-separate	0.789	0.788	0.958	0.959	0.690	0.641
DMFN	**0.800**	**0.799**	**0.962**	**0.970**	**0.703**	**0.657**

Table 5: Results of different variants of the DMFN model on the three datasets.

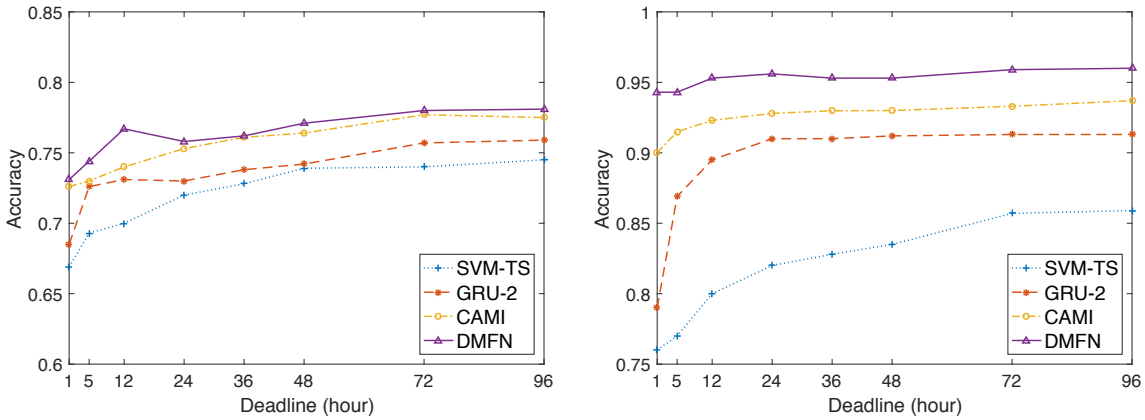
$1e{-}4$, $2e{-}4$ and $6e{-}5$ respectively on the Twitter, Weibo and PHEME datasets.

5.3.2 Early Detection Setting

We perform the early detection experiments on the Twitter and Weibo datasets with different deadlines, in $\{1, 5, 12, 24, 36, 48, 72, 96\}$ (hours). Fig. 4a and Fig. 4b illustrate the results on the Twitter and the Weibo datasets, respectively. As can be seen, on both datasets, the DMFN model performs the best among the selected models, followed by the CAMI model. On the Weibo dataset, the average number of posts per event is approximately 168, 353, 497 within the 1-hour, 5-hour and 12-hour deadlines respectively.These figures suggest that Weibo users are responsive and quickly react to a newly broadcasted event. The large number of posts per event, even at 1-hour deadline, gives enough information for the DMFN, as well as the CAMI models to produce good results, even within short deadlines.

5.3.3 The Effects of Jointly Training the Deep Network with Mean-field Inference

An advantage of the proposed model is that it allows training the deep network that produces the unary potentials with feedback from the mean-field inference. To verify this argument, we compare the performance when using different variants of the proposed model: (*i*) training and testing without the MF inference, (*ii*) training without the MF inference and testing with the MF inference, and (*iii*) training and testing with the MF inference (the full DMFN model). We denote the three variants as *DMFN-base*, *DMFN-separate* and *DMFN*. As can be seen from the table, applying the MF inference improves the results of the base model even if it has been trained without the MF inference. This proves the benefits of enforcing the dependencies between the events in the MRF when making predictions. Nevertheless, training and testing with MF inference consistently yields the best results among the three variants. This proves



(a) Twitter dataset

(b) Weibo dataset

Figure 4: Accuracy of different models when considering only posts within specific deadlines on the Twitter and Weibo datasets.

the benefits of unfolding the MF inference and incorporate it on top of the base network.

6 Conclusion

We formulated the fake news detection on social media as an inference problem in an MRF model that can be solved using the mean-field algorithm. By translating each update step in this algorithm into common operations in the deep learning literature, we can unfold it into hidden layers that can be integrated on top of another deep neural network. This results in our deep MRF model (DMFN) for detecting fake news. As such, the DMFN carries the advantages of both deep neural networks in learning high-level representations, and of MRF in incorporating correlations among the news articles. Experiments on well-known benchmark datasets show that the proposed model consistently improves over the state of the art in fake news detection in both the late and early detection settings.

Acknowledgments

The authors acknowledge the financial support from the Vrije Universiteit Brussel (PhD bursary - Duc Minh Nguyen), the Fonds Wetenschappelijk Onderzoek Vlaanderen (FWO) and the Francqui Foundation (2016-2017 International Francqui Chair - Robert Calderbank).

References

Yuri Boykov, Olga Veksler, and Ramin Zabih. 1998. Markov random fields with efficient approximations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–655.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *ACM International Conference on World Wide Web (WWW)*, pages 675–684.

James Fairbanks, Natalie Fitch, Nathan Knauf, and Erica Briscoe. 2018. Credibility assessment in the news : Do we need to read ? In *Misinformation and Misbehavior Mining on the Web Workshop (MIS2)*, pages 1–8.

William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. 2000. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47.

Ana Freire, Matteo Manca, Diego Saez-Trumper, David Laniado, Ilaria Bordino, Francesco Gullo, and Andreas Kaltenbrunner. 2016. Graph-based breaking news detection on wikipedia. In *AAAI Conference on Web and Social Media (ICWSM) Workshops*, pages 1–2.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *International Conference on Computational Linguistics (COLING)*.

Daphne Koller and Nir Friedman. 2009. In *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Philipp Krähenbühl and Vladlen Koltun. 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *IEEE International Conference on Data Mining (ICDM)*, pages 1103–1108.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1867–1870.

Yang Liu and Yi-Fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI Conference on Artificial Intelligence*, pages 1–8.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Joint Conference on Artificial Intelligence (IJCAI)*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1751–1754.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018a. Detect rumor and stance jointly by neural multi-task learning. In *The Web Conference (WWW)*, pages 585–593.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018b. Rumor detection on twitter with tree-structured recursive neural networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1980–1989.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2931–2937.

Natali Ruchansky, Sungyong Seo, and Yan Lin. 2017. CSI : A hybrid deep model for fake news detection. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 797–806.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explorations Newsletter*, 19(1):22–36.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15:1929–1958.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 647–653.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting TF-IDF term weights as making relevance decisions. *ACM Transactions on Information Systems*, 26(3):13:1–13:37.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *ACM SIGKDD Workshop on Mining Data Semantics*, pages 13:1–13:7.

Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Xiaoming Zhang, Zhoujun Li, and Philip S. Yu. 2018. TI-CNN : Convolutional neural networks for fake news detection. *ArXiv e-prints 1806.00749*.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2017. A convolutional approach for misinformation identification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3901–3907.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *ACM International Conference on World Wide Web (WWW)*, pages 1395–1405.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *Social Informatics*, pages 109–123.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE*, 11(3):1–29.