Tracing Fake-News Footprints: Characterizing Social Media Messages by How They Propagate

Liang Wu
Computer Science and Engineering
Arizona State University
Tempe, AZ
wuliang@asu.edu

Huan Liu
Computer Science and Engineering
Arizona State University
Tempe, AZ
huanliu@asu.edu

ABSTRACT

When a message, such as a piece of news, spreads in social networks, how can we classify it into categories of interests, such as genuine or fake news? Classification of social media content is a fundamental task for social media mining, and most existing methods regard it as a text categorization problem and mainly focus on using content features, such as words and hashtags. However, for many emerging applications like fake news and rumor detection, it is very challenging, if not impossible, to identify useful features from content. For example, intentional spreaders of fake news may manipulate the content to make it look like real news. To address this problem, this paper concentrates on modeling the propagation of messages in a social network. Specifically, we propose a novel approach, TraceMiner, to (1) infer embeddings of social media users with social network structures; and (2) utilize an LSTM-RNN to represent and classify propagation pathways of a message. Since content information is sparse and noisy on social media, adopting TraceMiner allows to provide a high degree of classification accuracy even in the absence of content information. Experimental results on real-world datasets show the superiority over state-of-the-art approaches on the task of fake news detection and news categorization.

KEYWORDS

Misinformation, Fake News Detection, Graph Mining, Social Network Analysis, Social Media Mining, Classification

ACM Reference Format:
Liang Wu and Huan Liu. 2018. Tracing Fake-News Footprints: Characterizing Social Media Messages by How They Propagate. In *Proceedings of WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159677

1 INTRODUCTION

As online social networks continue to pervade our culture, social networking sites have become an attractive platform to facilitate the spread of information. A recent study from Pew Research claims that 62% of adults get their news from social media in United States, with 29% among them doing so very often[1]. Concomitant with the expansive and varied sources of data are the challenges for personalizing the massive amount of information andfiltering out unwanted messages such as fake news and spam. However, the sparse and noisy social media content makes it difficult for traditional approaches, which heavily rely on content features, to tackle these challenges.

By contrast, our study aims tofi nd additional data sources to solve the problem. In this work, we focus on the diffusion of information. A key driving force behind the diffusion of information is its spreaders. People tend to spread information that caters to their interests and/orfi ts their system of belief [8]. Hence, similar messages usually leads to similar traces of information diffusion: they are more likely to be spread from similar sources, by similar people and in similar sequences. Since the diffusion information is pervasively available on social networks, in this work, we aim to investigate how the traces of information diffusion in terms of spreaders can be exploited to categorize a message. The message can be a piece of news, a story or a meme that has been posted and forwarded in social networks, and those users who post or forward it are the spreaders. Traces of a message refer to by whom and when the message is spread, *i.e.*, posted or forwarded.

We propose TraceMiner, a novel approach for classifying social media messages with diffusion network information. TraceMiner takes traces of a message as input and outputs its category. Consider the huge number of social media users and all the possible combinations of spreaders, traces will be of high dimensionality and thus may result in sparsity in the feature space. To cope with the problem, TraceMiner utilizes the proximity of nodes [34] and social dimensions [35] manifested in the social network, which have been successfully applied to capture the intrinsic characteristics of social media users in a myriad of applications.

To demonstrate TraceMiner's potential on real-world applications, we evaluate it with traditional approaches on Twitter data. TraceMiner outperforms competitors on multi-label information classification problems in large graphs. Therefore, TraceMiner provides an alternative way for modeling social media messages through learning abundant diffusion data that has not be fully utilized. Existing graph mining research mainly focuses on learning representation of graphs and nodes, while little attention has been paid to classifying information circulating between nodes. TraceMiner distances from existing graph representation methods by directly modeling information and making predictions in an

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159677

[1] http://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/

end-to-end manner other than providing only an attribute vector or embedding vector.

TraceMiner is scalable and the optimization can be easily parallelized through open-source software libraries. Hence, our method can be useful for a variety of social media mining problems where information from content is insufficient. Our contributions can be summarized as follows:

- We propose a novel approach for classifying social media messages with diffusion network information.
- We derive efficient optimization methods for TraceMiner, and provide analysis to guarantee the correctness.
- We extensively evaluate the performance on real social network data, and the experimental results demonstrate the effectiveness on different tasks.

The rest of this paper is organized as follows. In Section 2, we provide the definition for the problem. In Section 3, we introduce the proposed approaches and the optimization methods. How TraceMiner can be utilized to classify information diffusion sequences is presented in Section 4. In Section 5, we show empirical evaluation with discussions. Related work is discussed in Section 6. Conclusion and future work are presented in Section 7.

2 PROBLEM DEFINITION

We consider the problem of classifying social media messages into one or more categories. We define a graph $G \in \langle V, E \rangle$, where $v_i \in V$ with $i \in [1, |V|]$ is a node (user) and $E \subseteq V \times V$ is the set of edges. If $e_{ij} \in E$, there is an edge between v_i and v_j, otherwise there is not. Let M be the set of messages where $m_i \in M$ with $i \in [1, |M|]$. Each message m_i has a corresponding set of spreaders $\{(v_1^{m_i}, t_1^{m_i}), \{(v_2^{m_i}, t_2^{m_i}), \cdots, \{(v_n^{m_i}, t_n^{m_i})\}$, where n is the number of spreaders for m_i and $v_j^{m_i}$ is a user who spreads m_i at the time of $t_j^{m_i}$. Messages are partially labeled and thus only some of them have an associated class label. We denote the set of labels as Y, where $y_i \in Y$ indicates that m_i is labeled. Our goal is to learn a model with the social network graph G and partially labeled message M with the corresponding diffusion traces and label information Y, to predict $\hat{y}$ for the unlabeled messages.

Problem definition for traditional approaches: In order to make predictions for messages, most existing methods take the problem as a text categorization task, hence, each message m_i has a set of spreaders $\{(v_1^{m_i}, t_1^{m_i}, c_1^{m_i}), \cdots, \{(v_n^{m_i}, t_n^{m_i}, c_n^{m_i})\}$, where $c_j^{m_i}$ is the content information.

3 PROPOSED METHOD

In this section, we introduce how a diffusion trace can be used to facilitate classification. Wefi rst utilize sequential modeling methods to enable sequences to be used as attribute vectors. To alleviate the sparsity of sequences, we present a novel embedding method.

3.1 Sequence Modeling

Given the spreader information $\{(v_1^{m_i}, t_1^{m_i}), \cdots, \{(v_n^{m_i}, t_n^{m_i})\}$ and the graph G, the topology of information diffusion can be inferred by graph mining techniques [9]. The topology, which is usually a tree or forest (multiple trees) rooted with the initial spreader, contains informative patterns for characterizing a message. However, it is extremely difficult to directly deal with the tree structure. Consider two messages with similar diffusion networks, adding or removing one spreader, or changing any direction of the informationfl ow would lead to a different tree. Theoretically, there can be n^{n-2} different trees with n number of different nodes according to the Cayley's formula [7].

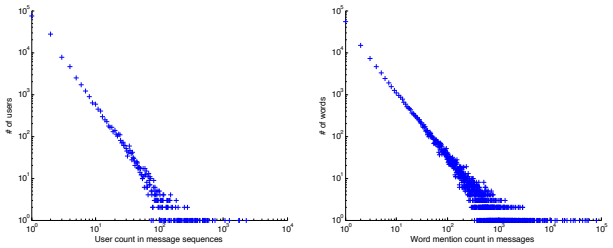
In order to solve this problem, we convert the tree structure into a temporal sequence. For example, given the spreaders of m_i $\{(v_1^{m_i}, t_1^{m_i}), \cdots, \{(v_n^{m_i}, t_n^{m_i})\}$, we generate a sequence $x_i = [(v_{q(1)}^{m_i}, t_{q(1)}^{m_i}), \cdots, (v_{q(n)}^{m_i}, t_{q(n)}^{m_i})]$ where for any two elements k and j in the sequence, if $k < j$, then $t_{q(k)}^{m_i} < t_{q(j)}^{m_i}$, meaning that $v_{q(k)}^{m_i}$ spread the information earlier than $v_{q(j)}^{m_i}$ did. Therefore, given n nodes, the number of all possible diffusion networks are reduced to $n!$. In order to further alleviate the sparsity, we incorporate social proximity and social dimensions in Section 3.2.

However, a possible problem of temporally sequencing spreaders is the loss of dependencies between users. Given v_i^m and v_j^m where $e_{ij} \in E$. If $t_i^m < t_j^m$, it is likely that user i spreads it to j or j is influenced by i [9]. Such direct dependency will be of vital importance in characterizing the information. For example, the informationfl ow from the controller account to the botnet followers is a key signal in detecting crowdturfing[11]. But if there is a spreader (u_k^m, t_k^m) where $< t_i^m < t_j^m$, in the sequence, i and j will be separated. Therefore, it would be appealing if the model can take advantage of dependencies between separated and distant items in a sequence. To this end, we propose to apply Recurrent Neural Networks (RNNs).

RNNs have been successfully applied in a myriad of domains for modeling sequential data [10], such as information retrieval [28], sentiment analysis [33] and machine translation [6]. We propose to use an RNN to sequentially accept each spreader of a message and recurrently project it into a latent space with the contextual information from previous spreaders in the sequence. As the RNN reaches the end of the sequence, a prediction can be made based on the embedding vector produced by the hidden activations. In order to better encode the distant and separated dependencies, we further incorporate the Long Short-Term Memory cells into the RNN model, *i.e.,* the LSTM-RNN.

In information diffusion, thefi rst spreader who initiates the diffusion process is more likely to be useful for classifying the message [1]. Hence, we feed the spread sequence in the **reverse** order, where thefi rst spreader in the sequence directly interacts with the prediction result, and thus it has more impact. Each spreader is represented by a local RNN. Parameters $\mathbf{W}$ of RNNs are shared across each replication in the sequence and h' is the previous recurrent output sent between RNNs to exploit the contextual information. In order to make the prediction, the last local RNNs are taking thefi rst spreader's attribute vector, prior recurrent output (and the label of the message) as input to predict the category of the message (or to train the RNNs model). In this work, we set the hidden node size (k) as 10. The way we obtain the attribute vector of nodes is introduced in Section 3.2.

Having chosen LSTM-RNNs as our method to classify messages, we now need a suitable way of learning attribute vectors $\mathbf{f}$, for social media users. An intuitive way is to utilize the social network



(a) Frequency of users in social media message traces.

(b) Frequency of words in social media message.

Figure 1: The frequency of users appearing in traces of social media messages follows a power-law distribution, which is similar to the distribution of word frequencies in messages.

graph G to generate embedding vectors [29, 34], and feed sequences of embedding vectors to the LSTM-RNNs [28]. Such embedding-based preprocessing for sequential data has been widely used for natural language processing. We follow the practice since 1) several social graph embedding approaches have been proven useful for classification tasks, such as LINE [34] and DeepWalk [29], and 2) users appear in spread traces follow similar distribution of how words appear in the social media posts.

Figure 1 illustrates the distribution of users and words. The distribution in Figure 1(a) comes from a real-world Twitter message trace dataset showing how users appear in message traces. The distribution in Figure 1(b) comes from the same dataset showing how words appear in message content. They both follow a power-law distribution, which motivates us to embed users into low dimensional vectors, as how embedding vectors of words are used in natural language processing [16, 28]. Several graph embedding algorithms are available, we will compare their performance and provide our solution and reasons behind our choice in the next subsection. For the rest of the subsection, we will introduce the optimization for the proposed LSTM-RNNs.

We show the training of the proposed LSTM-RNNs in Algorithm 1. We input the labeled spreader sequences X and the corresponding labels Y, which are randomly split into a training and a validation set in line *2*. In addition to the maximum number of iterations Max_{iter}, we also have a function *EarlyStop*() for controlling early termination of the training, which takes the loss on the validation set as the input. In line *1*, we initialize the model parameters randomly with Gaussian distribution. From line *3* to *7*, we update $\mathbf{W}$ with training data until the maximum epoch is reached or the early termination condition is met. The loss function used in line *4* is shown below:

$$\sum_{i=1}^{|X_{tr}|} |Y_{tr} = 0| y_i \log(\hat{y_i}) + |Y_{tr} = 1|(1 - y_i)(\log(1 - \hat{y_i})), \quad (1)$$

where y_i is the true label of i and $\hat{y_i}$ is the corresponding prediction. So Eq.(1) calculates the cross entropy between the true labels and the prediction. $|Y_{tr} = 0|$ ($|Y_{tr} = 1|$) is the number of negative (positive) instances in the training set. Since we aim to work on multi-label classification, the data is naturally imbalanced when we model one of them, introducing the weight helps the model balance the gradient of skewed data. In next subsection, we will introduce how we generate embeddings and the reason behind our choice.

Algorithm 1 Training Algorithm of LSTM-RNNs

Input: Labeled sequences and labels X, Y
Maximum number of iterations: Max_{iter}
Early termination function : *EarlyStop()*
Output: Weights of LSTM-RNNs: $\mathbf{W}$
1: Initialize $\mathbf{W}$ randomly with Gaussian distribution, $VLoss[Max_{iter}]$, $i = 0$
2: Split X and Y into training and validation set, (X_{tr}, Y_{tr}) and (X_{val}, Y_{val})
3: **do**
4: Train RNNs with (X_{tr}, Y_{tr}) for 1 epoch with Eq.(1)
5: Test RNNs with (X_{val}, Y_{val}) to obtain loss $VLoss[i]$
6: $i = i + 1$
7: **while** $EarlyStop(VLoss, i) = FALSE$ AND $(i < Max_{iter})$

3.2 Embedding of Users

Given the framework of sequence modeling, the next problem is tofi nd the proper embedding method that captures the intrinsic features of social media users. As discussed previously, using embedding vectors can help alleviate the data sparsity through leveraging social proximity and social dimensions. In this work, among the existing embedding methods, we will mainly focus on two state-of-the-art approaches that have been proven effective on social graphs, LINE [34] and DeepWalk [29]. Both LINE and DeepWalk aim to provide a representation for data instances that captures the inherent properties, such as social proximity.

These methods mainly focus on the microscopic structure of networks. For example,fi rst-order proximity constrains users that are connected to be similar and second-order proximity constrains users that have common friends to be similar. LINE achieves this by sampling such nodes from the network and updating their representations jointly, while DeepWalk samples a sequence of data with a random walk algorithm. Nevertheless, for a large social graph, some mesoscopic structure such as social dimensions [35] and community structures [44] are more useful in characterizing information [17]. Therefore, the ideal embedding method should be able to capture both local proximity and community structures.

Table 1 illustrates our results of using different embedding methods. We test LINE, DeepWalk and SocDim [35] on Twitter data and show the distance between neighbors with the new representation. We also detect community structures in the network and calculate the average of distances between nodes that are in the same community. The community detection algorithm is an accelerated version of Louvain method [2]. As shown in the table, LINE captures thefi rst and second-degree proximity, while SocDim best captures the community-wise proximity. Based on the random walk, DeepWalk achieves better community-wise proximity, however, it is still outperformed by SocDim, which directly models the community structure.

In order to capture both the social proximity and community-wise similarity among users, we propose a principled framework that directly models both kinds of information. Given the social

Table 1: Average Euclidean distance between nodes with low dimensional representation.

Method	1^{st}-degree	2^{nd}-degree	Intra-group
LINE	**5.16**	**5.00**	10.76
DeepWalk	7.74	7.69	6.04
SocDim	6.87	6.12	**4.55**

graph G, we can derive an adjacency matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$, where n is the number of users. Our goal is to learn a transformation matrix $\mathbf{M} \in \mathbb{R}^{n\times k}$ which converts users to a latent space with the dimensionality of k. Note that we reuse k for brevity of presentation, and the number of features and hidden nodes in the LSTM-RNNs are not necessarily the same. In order to capture the community-wise similarity, we introduce two auxiliary matrices, a community indicator matrix $\mathbf{H} \in \mathbb{R}^{n\times g}$, where g is the number of communities and $tr(\mathbf{H}\mathbf{H}^T) = n$ (only one element is 1 in each row and all the others are 0), and a community representation matrix $\mathbf{C} \in \mathbb{R}^{g\times k}$, where each row $\mathbf{c}_i$ is an embedding vector describing the community. In order to capture the community structure, we embed the problem into an attributed community detection model [44]:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{H},\mathbf{C}} \sum_{i=1}^{n} ||\mathbf{s}_i\mathbf{M} - \mathbf{h}_i\mathbf{C}||_2^2 + \alpha||\mathbf{H} - \mathbf{M}\mathbf{C}^T||_F^2, \\ &s.t. \quad tr(\mathbf{H}\mathbf{H}^T) = n, \end{aligned} \tag{2}$$

where $\mathbf{s}_i\mathbf{M}$ is the embedding vector and we regularize it to be similar to the representation of its corresponding community $\mathbf{h}_i\mathbf{C}$. The second term aims to achieve the intra-group coherence by predicting the community assignment by group the embedding vectors of users and communities [44]. The objective function in Eq.(2) aims to cluster nodes with embedding vectors. In order to further regularize the clusters to be social communities, we adopt a modularity maximization-based method, which has been widely used to detect communities with network information [37]. Specifically, given the adjacency matrix $\mathbf{S}$ and the community membership indicator, the modularity is defined as follows [35]:

$$Q = \frac{1}{2|E|}\sum_{i,j}(S_{ij} - \frac{d_i d_j}{2|E|})(\mathbf{h}_i\mathbf{h}_j^T), \tag{3}$$

where $|E|$ is the number of edges and d_i is the degree of i. $\mathbf{h}_i$ is the community assignment vector for i, and $\mathbf{h}_i\mathbf{h}_j^T = 1$ if i and j belong to the same community, otherwise $\mathbf{h}_i\mathbf{h}_j^T = 0$. $\frac{d_i d_j}{2|E|}$ is the expected number of edges between i and j if edges are placed at random. Modularity Q measures the difference between the number of actual edges within a community and the expected number of edges placed at random. An optimal community structure $\mathbf{H}$ should maximize the modularity Q. By defining the modularity matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ where $B_{ij} = S_{ij} - \frac{d_i d_j}{2|E|}$ and suppressing the constant which has no effect on the modularity, we rewrite Eq.(3) as follows:

$$Q = tr(\mathbf{H}^T\mathbf{B}\mathbf{H}).$$

In order to guarantee that the embedding vectors preserve the community structure in the latent space, we propose to integrate modularity maximization into the embedding method. The objective function can be rewritten with the modularity maximization regularizer as follows:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{H},\mathbf{C}} \sum_{i=1}^{n} ||\mathbf{s}_i\mathbf{M} - \mathbf{h}_i\mathbf{C}||_2^2 + \alpha||\mathbf{H} - \mathbf{M}\mathbf{C}^T||_F^2 - \beta tr(\mathbf{H}^T\mathbf{B}\mathbf{H}) \\ &s.t. \quad tr(\mathbf{H}\mathbf{H}^T) = n, \end{aligned} \tag{4}$$

where β controls the influence of community structures. As discussed previously, the microscopic structure is also of vital importance for generating embedding vectors. In order to jointly consider both mesoscopic and microscopic structures, we decompose $\mathbf{M}$ into a conjunction of a global model parameter $\tilde{\mathbf{M}}$ and a localized variable $\mathbf{M_i}$ for each user i ($\mathbf{M} = \tilde{\mathbf{M}} + \mathbf{M_i}$ for each user i). Therefore, $\tilde{\mathbf{M}}$ captures the community structure and $\mathbf{M_i}$ can be used to directly apprehend the microscopic structure between nodes. Motivated by recent research on network regularization, we fortify the representation of nodes with proximity by the network lasso regularization term [12]:

$$\sum_{i,j} \mathbf{A}_{ij}||\mathbf{M}_i - \mathbf{M}_j||_F^2,$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the microscopic structure matrix, $A_{ij} = 1$ if we aim to preserve the proximity between i and j in the latent space. Following conventional graph embedding practices [34], we considerfi rst- and second-degree proximity, meaning that $A_{ij} = 1$ if i and j are connected or share a common friend. Note that A can be specified with particular applications. Imposing the Frobenius norm of the difference between $\mathbf{M}_i$ and $\mathbf{M}_j$ incentivizes them to be the same when $A_{ij} = 1$. By incorporating the network lasso regularizer, the objective function can be reformulated as follows:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{H},\mathbf{C}} \sum_{i=1}^{n} ||\mathbf{s}_i(\tilde{\mathbf{M}} + \mathbf{M_i}) - \mathbf{h}_i\mathbf{C}||_2^2 + \alpha||\mathbf{H} - \tilde{\mathbf{M}}\mathbf{C}^T||_F^2 \\ &- \beta tr(\mathbf{H}^\mathbf{T}\mathbf{B}\mathbf{H}) + \gamma \sum_{i,j} \mathbf{A}_{ij}||\mathbf{M}_i - \mathbf{M}_j||_F^2, \\ &s.t. \quad tr(\mathbf{H}\mathbf{H}^\mathbf{T}) = n, \end{aligned} \tag{5}$$

where γ controls the influence of the network lasso. As we can see, we establish the consensus relationship between mesoscopic and microscopic network structures by jointly considering the social communities and proximity. By introducing the global parameter $\tilde{\mathbf{M}}$ and the personal variable $\mathbf{M}_i$, we force both kinds of information to be preserved in the newly-learnt embedding vectors. However, Eq.(5) is not jointly convex to all the parameters $\mathbf{M}$,$\mathbf{H}$ and $\mathbf{C}$. In order to solve the problem, we separate the optimization into four subproblems and iteratively optimize them. We will introduce details of the optimization for the rest of the section.

Update $\tilde{\mathbf{M}}$ whilefixing $\mathbf{M_i}$, $\mathbf{H}$ and $\mathbf{C}$: By removing terms that are irrelevant to $\tilde{\mathbf{M}}$, we obtain the following optimization problem:

$$\min_{\tilde{\mathbf{M}}} \sum_{i=1}^{n} ||\mathbf{s}_i\tilde{\mathbf{M}} + \mathbf{s}_i\mathbf{M_i} - \mathbf{h}_i\mathbf{C}||_2^2 + \alpha||\mathbf{H} - \tilde{\mathbf{M}}\mathbf{C}^T||_F^2, \tag{6}$$

which is convex *w.r.t.* $\tilde{\mathbf{M}}$. In real applications, the number of users n may be huge. Hence, we adopt a gradient-based update rule as follows:

$$\tilde{\mathbf{M}} = \tilde{\mathbf{M}} - \tau\frac{\partial \epsilon_{\tilde{\mathbf{M}}}}{\partial \tilde{\mathbf{M}}}, \tag{7}$$

where τ is the step size that can be obtained through backtracking line search [27]. The derivative of $\tilde{\mathbf{M}}$ is shown as follows:

$$\frac{\partial \epsilon_{\tilde{\mathbf{M}}}}{\partial \tilde{\mathbf{M}}} = \mathbf{s}_i^T \sum_{i=1}^{n} (\mathbf{s}_i \tilde{\mathbf{M}} + \mathbf{s}_i \mathbf{M_i} - \mathbf{h}_i \mathbf{C}) + \alpha(\mathbf{H} - \tilde{\mathbf{M}}\mathbf{C}^T)\mathbf{C}. \quad (8)$$

Update $\mathbf{M_i}$ whilefixing $\tilde{\mathbf{M}}$, $\mathbf{H}$ and $\mathbf{C}$: By removing terms that are irrelevant to $\mathbf{M}_i$, we obtain the following optimization problem:

$$\min_{\mathbf{M}_i} \sum_{i=1}^{n} ||\mathbf{s}_i \tilde{\mathbf{M}} + \mathbf{s}_i \mathbf{M_i} - \mathbf{h}_i \mathbf{C}||_2^2 + \gamma \sum_{i,j} A_{ij} ||\mathbf{M}_i - \mathbf{M}_j||_F^2, \quad (9)$$

which is convex *w.r.t.* $\mathbf{M}_i$. Similarly, we derive the gradient:

$$\frac{\partial \epsilon_{\mathbf{M}_i}}{\partial \mathbf{M}_i} = \mathbf{s}_i^T \sum_{i=1}^{n} (\mathbf{s}_i \tilde{\mathbf{M}} + \mathbf{s}_i \mathbf{M_i} - \mathbf{h}_i \mathbf{C}) + \gamma \sum_{i,j} A_{ij} (\mathbf{M}_i - \mathbf{M}_j). \quad (10)$$

Update $\mathbf{C}$ whilefixing $\tilde{\mathbf{M}}$, $\mathbf{M_i}$, and $\mathbf{H}$: By removing terms that are irrelevant to $\mathbf{C}$, we obtain the following optimization problem:

$$\min_{\mathbf{C}} \sum_{i=1}^{n} ||\mathbf{s}_i (\tilde{\mathbf{M}} + \mathbf{M_i}) - \mathbf{h}_i \mathbf{C}||_2^2 + \alpha ||\mathbf{H} - \tilde{\mathbf{M}}\mathbf{C}^T||_F^2, \quad (11)$$

which is convex *w.r.t.* $\mathbf{C}$. Similarly, the gradient can be obtained as:

$$\frac{\partial \epsilon_{\mathbf{C}}}{\partial \mathbf{C}} = \sum_{i=1}^{n} \mathbf{h}_i^T (\mathbf{h}_i \mathbf{C} - \mathbf{s}_i \tilde{\mathbf{M}} - \mathbf{s}_i \mathbf{M_i}) + \alpha (\tilde{\mathbf{M}}\mathbf{C}^T - \mathbf{H})^T \tilde{\mathbf{M}}. \quad (12)$$

Update $\mathbf{H}$ whilefixing $\tilde{\mathbf{M}}$, $\mathbf{M_i}$, and $\mathbf{C}$: By removing terms that are irrelevant to $\mathbf{H}$, we obtain the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{H}} ||\mathbf{SM} - \mathbf{HC}||_F^2 + \alpha ||\mathbf{H} - \tilde{\mathbf{M}}\mathbf{C}^T||_F^2 - \beta tr(\mathbf{H^T}(\mathbf{S} - \hat{\mathbf{B}})\mathbf{H}), \\ &s.t. \quad tr(\mathbf{H}\mathbf{H}^T) = n, \end{aligned} \quad (13)$$

where $\hat{\mathbf{B}}_{ij} = \frac{d_i d_j}{2|E|}$. Consider that $\mathbf{H}$ is an indicator matrix, the constraint makes the problem in Eq.(13) NP-complete, which is extremely difficult to solve. In order to cope with the problem, we relax the constraint to orthogonality $\mathbf{H}^T\mathbf{H} = \mathbf{I}$ and nonnegativity $\mathbf{H} \geq 0$ and reformulate the objective function as follows:

$$\begin{aligned} \epsilon_{\mathbf{H}} = &- \beta tr(\mathbf{H}^T \mathbf{S}\mathbf{H}) + \beta tr(\mathbf{H}^T \hat{\mathbf{B}}\mathbf{H}) \\ &+ ||\mathbf{SM} - \mathbf{HC}||_F^2 + \alpha ||\mathbf{H} - \hat{\mathbf{M}}\mathbf{C}^T||_F^2 \\ &+ \lambda ||\mathbf{H}^T\mathbf{H} - \mathbf{I}||_F^2, \end{aligned} \quad (14)$$

where $\lambda > 0$ should be a large number to guarantee the orthogonal constraint to be satisfied, and we set it as 10^8 in this work. We then utilize the property that $||\mathbf{X}||_F^2 = tr(\mathbf{X}^T\mathbf{X})$ to reformulate the loss function as follows:

$$\begin{aligned} \epsilon_{\mathbf{H}} = &- \beta tr(\mathbf{H}^T \mathbf{S}\mathbf{H}) + \beta tr(\mathbf{H}^T \hat{\mathbf{B}}\mathbf{H}) \\ &+ tr(\mathbf{S}\mathbf{M}\mathbf{M}^T\mathbf{S}^T + \mathbf{H}\mathbf{C}\mathbf{C}^T\mathbf{H}^T - 2\mathbf{S}\mathbf{M}\mathbf{C}^T\mathbf{H}^T) \\ &+ \alpha tr(\mathbf{H}\mathbf{H}^T + \hat{\mathbf{M}}\mathbf{C}^T\mathbf{C}\tilde{\mathbf{M}}^T - 2\mathbf{H}\mathbf{C}\tilde{\mathbf{M}}^T) \\ &+ \lambda tr(\mathbf{H}^T\mathbf{H}\mathbf{H}^T\mathbf{H} - 2\mathbf{H}^T\mathbf{H} + \mathbf{I}) + tr(\mathbf{\Theta}\mathbf{H}^T), \end{aligned} \quad (15)$$

where $\mathbf{\Theta} = [\Theta_{ij}]$ is a Lagrange multiplier matrix to impose the nonnegative constraint. Set the derivative of $\frac{\partial \epsilon_{\mathbf{H}}}{\partial \mathbf{H}}$ to 0, we have:

$$\begin{aligned} \mathbf{\Theta} = &\ 2\mathbf{S}\mathbf{H} - 2\beta\tilde{\mathbf{B}}\mathbf{H} - 2\mathbf{C}\mathbf{C}^T\mathbf{H}^T + 2\mathbf{S}\mathbf{M}\mathbf{C}^T \\ &- 2\alpha\mathbf{H}^T + 2\alpha\mathbf{C}\tilde{\mathbf{M}}^T - 4\lambda\mathbf{H}\mathbf{H}^T\mathbf{H} + 4\lambda\mathbf{H}. \end{aligned} \quad (16)$$

Following the Karush-Kuhn-Tucker (KKT) condition for the nonnegativity, we have the equation as follows:

$$\begin{aligned} &(2\mathbf{S}\mathbf{H} - 2\beta\tilde{\mathbf{B}}\mathbf{H} - 2\mathbf{C}\mathbf{C}^T\mathbf{H}^T + 2\mathbf{S}\mathbf{M}\mathbf{C}^T - 2\alpha\mathbf{H}^T \\ &+2\alpha\mathbf{C}\tilde{\mathbf{M}}^T - 4\lambda\mathbf{H}\mathbf{H}^T\mathbf{H} + 4\lambda\mathbf{H})_{ij} H_{ij} = \theta_{ij} H_{ij} = 0, \end{aligned} \quad (17)$$

which is thefi xed point equation that the solution must satisfy at convergence. The update rule for $\mathbf{H}$ can be written as follows:

$$\mathbf{H} = \mathbf{H} \odot \sqrt{\frac{-2\beta\tilde{\mathbf{B}}\mathbf{H} + \sqrt{\Delta}}{8\lambda\mathbf{H}\mathbf{H}^T\mathbf{H}}}, \quad (18)$$

where Δ is defined as:

$$\begin{aligned} \Delta = &\ 2\beta(\tilde{\mathbf{B}}\mathbf{H}) \odot (\tilde{\mathbf{B}}\mathbf{H}) + 16\lambda(\mathbf{H}\mathbf{H}^T\mathbf{H}) \\ &\odot (2\mathbf{S}\mathbf{H} - 2\mathbf{C}\mathbf{C}^T\mathbf{H}^T + 2\mathbf{S}\mathbf{M}\mathbf{C}^T \\ &- 2\alpha\mathbf{H}^T + 2\alpha\mathbf{C}\tilde{\mathbf{M}}^T + 4\lambda\mathbf{H}). \end{aligned} \quad (19)$$

The convergence of Eq.(19) can be proven as an instance of nonnegative matrix factorization (NMF) problem [18].

3.3 Time complexity

TraceMiner consists of two components, LSTM-RNNs and the embedding method. Though LSTM-RNNs take $O(|E| + |V|)$-time for backpropagations, the scalability can be easily increased with deep learning software library like Theano[2], especially when GPU is available.

Since the number of users is usually far larger than the number of features and number of communities, the embedding method takes $O(n^2)$-time. Only matrix multiplication is used in all update rules, so the optimization can be accelerated by utilizing matrix optimization library like OpenBLAS[3].

4 ALGORITHM-TRACEMINER

In this section, we introduce the detailed procedure of TraceMiner method for network diffusion classification. The overall process consists of two steps:

- *Learning embeddings based on network connectivity.* In this work, we aim to utilize the social identity of a user to infer the information she spreads. Hence, we learn embeddings from friendships and social community memberships.
- *Construct a sequence classifier with LSTM-RNNs.* After we obtain the embeddings of social media users, we consider a social media message as a sequence of its spreaders. We employ LSTM-RNNs to model the sequence, and thefinal hidden output are aggregated using softmax to produce a predicted class label.

Thefi rst step utilizes network structures to embed social media users into space of low dimensionality, which alleviates the data sparsity of utilizing social media users as features. The second step represents user sequences of information diffusion, which allows for the classification of propagation pathways.

[2] http://deeplearning.net/software/theano/
[3] http://www.openblas.net/

Table 2: Statistics of the datasest used in this study.

	Messages	Posts	Unique Users	Class Ratio
Real News	68,892	288,591	121,211	0.27(b):0.25(t):0.37(e):0.11(m)
Fake News	3,600	17,613	9,153	0.5:0.5

5 EXPERIMENT

In this section, we introduce experiment details to validate the effectiveness of the proposed framework. Through the experiments, we aim to answer two questions:

- How well can network information be used to classify social messages compared with content information?
- How effective are the LSTM-RNNs by integrating with the proposed embedding method?

Therefore, we test the methods on two different classification tasks with real-world datasets and include both content-based and network-based baselines for comparison.

5.1 Datasets

Over 200 million posts are posted per day on Twitter[4] and the popularity has made Twitter a testbed for informationfiltering research. In this work, we aim to collect a large dataset that includes tweets about specific messages. Following [30], we leverage Twitter Search API[5] to retrieve tweets of interests by compiling queries with certain topics.

We deal with two tasks in this work, standard news classification and fake news detection. News classification is a classical multi-label text categorization problem and existing efforts have mainly focused on the content. We obtain a news dataset which was originally used for content-based classification[6] by selecting news that has at least two posts on Twitter. Queries for Twitter Search API are compiled by words in the title of the corresponding news. Based on the spreaders of news, we try to use TraceMiner to classify the news into four categories: business (b), science and technology (t), entertainment (e), medical (m). Statistics about the dataset are shown in Table 2. We sample 68, 892 pieces of news, which relate to 288, 591 posts with 121, 211 unique users. The ratio of different categories is also presented.

The other task is fake news detection. The openness of social media platforms enables timely information to be spread at a high rate. Meanwhile, it also allows for the rapid creation and dissemination of fake news. Following [30], we retrieve tweets related to fake news by compiling queries with a fact-checking website. In this work, we choose Snopes[7] to obtain ground truth, where we collect articles tagged with fake news[8]. In order to obtain non-fake news posts pertaining to the same topic, we extract keywords in regular expressions as queries to retrieve posts. Statistics of the dataset is shown in Table 2. We collect 3, 600 messages with 50% are fake news.

[4]https://blog.twitter.com/2011/200-million-tweets-per-day
[5]https://dev.twitter.com/rest/public/search
[6]https://archive.ics.uci.edu/ml/datasets/News+Aggregator
[7]http://www.snopes.com/
[8]https://www.snopes.com/tag/fake-news/

5.2 Experimental Settings

A core contribution of our work is the idea that spreaders of information can be used to predict message categories. Therefore, we try to test the effectiveness of the proposed method comparing with the state-of-the-art content-based approaches. We experiment a variety of approaches, and report the following two which achieve better results.

- **SVM** [13] trains on content information, which isfirst preprocessed with Stanford CoreNLP toolkit [24]. We adopt bigram and trigram features based on results on the validation set.
- **XGBoost** [5] is an optimized distributed gradient boosting library that implements machine learning algorithms under the Gradient Boosting framework. It has been successfully applied to various problems and competitions. We feed it with the preprocessed content produced by Stanford CoreNLP. XGBoost presents the best results among all content-based algorithms we tested.

We propose a novel embedding method to cater to TraceMiner. In order to evaluate its effectiveness, we introduce two variants of TraceMiner and present their results for comparison:

- **TM(DeepWalk)** is a variant of TraceMiner by adopting the embedding vectors from DeepWalk as input. As discussed earlier, DeepWalk captures proximity between nodes with random walk: nodes that are sampled together with one random walk are forced to preserve the similarity in the latent space. Therefore, DeepWalk does not directly model thefi rst and second-degree proximity or the community structure.
- **TM(LINE)** is a variant of TraceMiner by adopting the embedding vectors from LINE. LINE modelsfi rst and second-degree proximity while does not consider the community structure between users.

To test the prediction accuracy in terms of both precision and recall, we adopted the F_1-measure to evaluate the performance. Since there are multiple labels to be predicted, for each task t, F_1^t can be computed. In order to get the overall performance, wefirst adopt the Macro-averaged F_1-measure as:

$$Macro - F_1 = \frac{1}{|\boldsymbol{T}|} \sum_{t \in \boldsymbol{T}} F_1^t, \tag{20}$$

where $\boldsymbol{T}$ is the set of all identity labels and F_1^t is the F_1-measure of task t.

A possible problem of Macro-F_1 is, since the sizes of different categories are different, the task with fewer instances may be overemphasized. In order to cope with this problem, we adopted Micro-averaged F_1-measure. First, we calculate the micro averaged

Table 3: The F_1-measure of different methods on the task of social media news categorization.

	Training Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F_1(%)	SVM	0.6967	0.7138	0.7447	0.7577	0.7988	0.8096	0.8499	0.8787	0.8996
	XGBoost	0.7121	0.7349	0.7512	0.7794	0.8248	0.8250	0.8638	0.8951	0.9047
	TM(DeepWalk)	0.7895	0.8081	0.8149	0.8374	0.8569	0.8627	0.8852	0.8917	0.9184
	TM(LINE)	0.7691	0.7926	0.8163	0.8379	0.8467	0.8744	0.8980	0.9106	0.9253
	TraceMiner	**0.8275**	**0.8460**	**0.8658**	**0.8835**	**0.8885**	**0.9141**	**0.9218**	**0.9357**	**0.9380**
Macro-F_1(%)	SVM	0.6988	0.7260	0.7425	0.7754	0.7665	0.7872	0.8118	0.8314	0.8722
	XGBoost	0.7305	0.7438	0.7857	0.7887	0.8144	0.8344	0.8726	**0.8941**	0.9044
	TM(DeepWalk)	0.7746	0.8010	0.8156	0.8313	0.8377	0.8611	0.8646	0.8734	0.8839
	TM(LINE)	0.7561	0.7895	0.8019	0.8138	0.8235	0.8568	0.8775	0.8896	**0.9153**
	TraceMiner	**0.8181**	**0.8347**	**0.8359**	**0.8549**	**0.8635**	**0.8788**	**0.8779**	0.8882	0.9064

precision and recall:

$$Micro-precision = \frac{\#TP}{\#TP + \#FP} \quad (21)$$
$$Micro-recall = \frac{\#TP}{\#TP + \#FN},$$

where #TP is the number of true positives, #FP is the number of false positives and #FN is the number of false negatives. Micro-F_1 is the harmonic average of Micro-precision and Micro-recall.

5.3 Experimental Results

Social Media News Categorization: The performance of different methods on Twitter News data with varying training ratio, from 10% to 90%, is illustrated in Table 3. For each experiment, samples are randomly split into training and testing set. We repeat this process 10 times and report the average results. The highest performance under each setting is highlighted in bold face.

In terms of Micro-F_1, our proposed model TraceMiner outperforms all the baselines and its variations, TM(DeepWalk), TM(LINE). Diffusion-based methods perform better than content-based methods. XGBoost performs slightly better than SVM. TM(DeepWalk) is the runner-up method for 10%, 20% and 50%, and TM(LINE) is the runner-up for the rest cases. The result shows that when less network data is available, the random walk-based approach produces better embeddings of users; And a more deterministic method constraining on social proximity better apprehends user behaviors when the network information is more complete. TraceMiner achieves the best result for all tasks. By jointly modeling the microscopic and mesoscopic structures, TraceMiner is more robust to data sparsity.

In terms of Macro-F_1, XGBoost outperforms SVM for all cases. Similar pattern has again been observed: TM(DeepWalk) outperforms TM(LINE) with less training information, while TM(LINE) outperforms TM(DeepWalk) when the information is more complete. TraceMiner still performs the best among most cases until we increase the training ratio up to 80%. XGBoost and TM(LINE) achieves the best result for 80% and 90%, respectively. Two observations can be made here: with more training information becoming available, 1) the margin between proposed methods and the content-based methods becomes smaller; and 2) the margin between TraceMiner and its variants TM(LINE) and TM(DeepWalk) becomes smaller. Based on the observations we can draw conclusions that TraceMiner is more useful when less training information is available, and the proposed TraceMiner can well handle scarce data in the early phase of learning when less training information is known. XGBoost gets the best when 80% of information is available. Since text-based categorization is a well-studied problem, and it is easy to solve when rich information is available, TraceMiner will be able to complement those cases that are difficult for content-based approaches to deal with, and such cases are pervasively present in social media mining tasks where content information is insufficient and noisy.

Another observation that again validates ourfi ndings is that TraceMiner performs better in terms of Micro-F_1. As shown in Eq.(20) and (21), in a multi-label classification task, the category with fewer instances is more advantageous for Macro-F_1. The results show that TraceMiner actually ends up with correctly classifying more instances.

Fake News Detection: The performance of different methods on Twitter fake news data with varying training ratio, from 10% to 90%, is illustrated in Table 4. Since the dataset is balanced, Micro- and Macro-F_1 are the same, so only one set of results are presented. For the content-based approaches, XGBoost consistently outperforms SVM for all cases. For the two variants of TraceMiner, similar patterns are observed: TM(DeepWalk) outperforms TM(LINE) when less training information is available. TM(LINE) outperforms TM(DeepWalk) when more information is available for training. It again proves that random walk-based sampling is more effective for scarce data, and proximity-based regularization better captures data structures with more training information.

An interesting difference between the results for fake news and the previous experiment is the larger margin between proposed methods and content-based methods. Unlike posts related to news where the content information is more self-explanatory, content of posts about fake news is less descriptive. Intentional spreaders of fake news may manipulate the content to make it look more similar to non-rumor information. Hence, TraceMiner can be useful for many emerging tasks in social media where adversarial attacks are present, such as detecting rumors and crowdturfing. The margin between content-based approaches and TraceMiner becomes smaller when more information is available for training, however, in these emerging tasks, training information is usually time-consuming and labor-intensive to obtain.

Table 4: The F_1-measure of different methods on the task of fake news detection.

Training Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
SVM	0.5825	0.5779	0.6122	0.6194	0.6658	0.7114	0.7224	0.7252	0.7581
XGBoost	0.6558	0.7004	0.7002	0.7153	0.7288	0.7703	0.7984	0.8115	0.8226
TM(DeepWalk)	0.7804	0.7810	0.8078	0.8264	0.8194	0.8491	0.8542	0.8738	0.8894
TM(LINE)	0.7542	0.7547	0.7913	0.8015	0.8083	0.8485	0.8733	0.8936	0.8971
TraceMiner	**0.7867**	**0.7935**	**0.8344**	**0.8459**	**0.8547**	**0.8751**	**0.8988**	**0.9089**	**0.9124**

Another point we would like to discuss is the performance when the training information is very insufficient. When 10% of information is available, SVM has an F_1 score of 58% which is slightly better than a random guess, while TraceMiner has an F_1 score of 78%. Although such margin is reduced when more information is available, the optimal performance with very few training information is of crucial significance for tasks which emphasize on the earliness. For example, detecting fake news at an early stage is way more meaningful than detecting it when 90% percent of its information is known [30, 31, 40]. In conclusion, TraceMiner provides an effective method for modeling messages diffused in social media with only network information, which provides a complementary tool for emerging tasks that require earliness and/or suffers from the scarcity of content information.

6 RELATED WORK

This work mainly focuses on classifying social media messages, which is a fundamental problem in social media mining. It can be useful for many classical tasks including social recommendation, personalization and targeted advertising. Accurate categorization of social media content allows for precisefi ltering of information, which helps alleviate the information overloading. A recent surge for social media platforms is the attacks of disinformation launched by malicious users. Both content and network information has been studied to detect malicious users, such as spammers [38, 39] and crowdturfers [41]. In terms of network information, traditional approaches usually derive features from the social networks and spreaders of a message. For example, Hu et al. assume that the information spread by similar users tend to share similar properties [38], and the network information mainly centers around the user instead of information itself. Our work distances from the existing work by directly studying the network information.

Our work is also related to network structure mining methods. Neural network models have been applied on network data for tasks such as classification [43] and clustering [36]. These existing methods focus on the nodes in the graph, while our work focuses on the network structure itself, which is manifested by the diffusion of messages. In addition, unlike existing graph representation methods, our goal is to provide an end-to-end system with prediction results, instead of offering only the embedding vectors. Recent research has been proposed to utilize RNNs for classification in an semi-supervised manner [25], which is also related to our work.

We present a novel graph embedding model, which is related to existing embedding methods and feature selection on networked data [20]. For example, DeepWalk [29] links a network embedding problem into a word embedding problem by showing the similar distribution of nodes appearing in random walks and words appearing in sentences. They employ a Skip-Gram model, which was originally proposed for modeling natural languages, to learn embedding of graphs. LINE [34] aims to preserve thefi rst- and second-order proximity between nodes, and provides an embedding vector by concatenating results on both levels. Our work focuses on encoding both social proximity and social community information to alleviate the data sparsity, instead of investigating only one of them [22, 23]. Recent studies also study and utilize network dynamics by observing the change of social networks over time [19, 21]. We focus on a snapshot since the newly established/withdrawn links during the study are very few.

Our work is related to information diffusion. There are various models which are designed to abstract the pattern of information diffusion, such as SIR Model [15], Tipping Model [4], Independent Cascade Model [14] and Linear Threshold Model [14]. However, traditional information diffusion models abstract the diffusion process to estimate the virality of information and ignore the interaction between multiple campaigns, which cannot be directly applied here.

Our work can be particularly helpful for identifying messages that cannot be easily detected with the content. An emerging problem that has the feature is rumor and fake news detection. For example, supervised learning approaches have been used to detect rumors [40] and the spreaders [26]. The proposed process usually consists of two stages: employing a feature engineering approach to distinguish misinformation [42] from Twitter's normal content stream and utilize a supervised learning approach to train a detector. However, supervised approaches depend on content information, which can be easily manipulated by malicious spreaders. Previous studies have explored how malicious information can be detected from node to node [3], however, the proposed systems can only help visualize and track known events and require experts to observe it and make decisions. The process requires certain domain knowledge and expertise, while TraceMiner is an end-to-end method taht directly studies the information diffusion.

7 CONCLUSION

In this work, we aim to classify messages spread in social networks, which is a fundamental problem for social media mining. We observe that for many emerging tasks, content information is usually insufficient or less descriptive, while pervasively available network information is left unused. Therefore, we propose a novel method TraceMiner that classifies social media messages with diffusion traces in social networks. To address the problem, we propose an end-to-end classification model based on LSTM-RNNs. In order to alleviate the data sparsity, we propose an embedding method that captures both social proximity and community structures.

Experimental results with real-world datasets show that TraceMiner effectively classifies social media messages and is especially useful when content information is insufficient.

Future work can be focused on two aspects. First, we would like to investigate if TraceMiner can be used to facilitate other network mining tasks, like recommendation and link prediction. Second, since content information is also readily available, we will study how content and network information can be jointly utilized.

ACKNOWLEDGMENTS

This work was supported, in part, by the Office of Naval Research grants N000141410095 and N000141310835.

REFERENCES

[1] Geoffrey Barbier, Zhuo Feng, Pritam Gundecha, and Huan Liu. 2013. Provenance data in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery* 4, 1 (2013), 1–84.
[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008.
[3] Nan Cao, Conglei Shi, Sabrina Lin, Jie Lu, Yu-Ru Lin, and Ching-Yung Lin. 2016. TargetVue: Visual analysis of anomalous user behaviors in online communication systems. *IEEE transactions on visualization and computer graphics* 22, 1 (2016), 280–289.
[4] Damon Centola. 2010. The spread of behavior in an online social network experiment. *science* 329, 5996 (2010), 1194–1197.
[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.
[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[7] LE Clarke. 1958. On Cayley's formula for counting trees. *Journal of the London Mathematical Society* 1, 4 (1958), 471–474.
[8] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences* 113, 3 (2016), 554–559.
[9] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. 2010. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1019–1028.
[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT Press.
[11] Guofei Gu, Junjie Zhang, and Wenke Lee. 2008. BotSniffer: Detecting botnet command and control channels in network traffic. (2008).
[12] David Hallac, Jure Leskovec, and Stephen Boyd. 2015. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 387–396.
[13] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.
[14] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.
[15] William O Kermack and Anderson G McKendrick. 1927. A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Vol. 115. 700–721.
[16] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[17] Edward O Laumann and Franz U Pappi. 2013. *Networks of collective action: A perspective on community influence systems*. Elsevier.
[18] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.
[19] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. *arXiv preprint arXiv:1706.01860* (2017).
[20] Jundong Li, Xia Hu, Liang Wu, and Huan Liu. 2016. Robust unsupervised feature selection on networked data. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 387–395.
[21] Jundong Li, Cheng Kewei, Liang Wu, and Huan Liu. 2018. Streaming Link Prediction on Dynamic Attributed Networks. In *Proceedings of 11th ACM International Conference on Web Search and Data Mining (WSDM 2018)*. ACM.
[22] Jundong Li, Liang Wu, Dani Harsh, and Huan Liu. 2018. Unsupervised Personalized Feature Selection. In *Proceedings of The 32nd AAAI Conference on Artificial Intelligence*. AAAI.
[23] Jundong Li, Liang Wu, Osmar R Zaïane, and Huan Liu. 2017. Toward personalized relational learning. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 444–452.
[24] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010
[25] John Moore and Jennifer Neville. 2017. Deep collective inference. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.
[26] Fred Morstatter, Liang Wu, Tahora H Nazer, Kathleen M Carley, and Huan Liu. 2016. A new approach to bot detection: Striking the balance between precision and recall. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE, 533–540.
[27] Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.
[28] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing* 24, 4 (2016), 694–707.
[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 701–710.
[30] Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1589–1599.
[31] Justin Sampson, Fred Morstatter, Liang Wu, and Huan Liu. 2016. Leveraging the implicit structure within social media for emergent rumor detection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2377–2382.
[32] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. *ACM SIGKDD Explorations Newsletter* 19, 1 (2017), 22–36.
[33] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, and others. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Vol. 1631. Citeseer, 1642.
[34] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1067–1077.
[35] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 817–826.
[36] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning Deep Representations for Graph Clustering.. In *AAAI*. 1293–1299.
[37] Liang Wu, Xia Hu, and Huan Liu. 2016. Relational learning with social status analysis. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 513–522.
[38] Liang Wu, Xia Hu, Fred Morstatter, and Huan Liu. 2017. Adaptive Spammer Detection with Sparse Group Modeling.. In *ICWSM*. 319–326.
[39] Liang Wu, Xia Hu, Fred Morstatter, and Huan Liu. 2017. Detecting Camouflaged Content Polluters.. In *ICWSM*. 696–699.
[40] Liang Wu, Jundong Li, Xia Hu, and Huan Liu. 2017. Gleaning wisdom from the past: Early detection of emerging rumors in social media. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 99–107.
[41] Liang Wu and Huan Liu. 2017. Detecting Crowdturfing in Social Media. (2017).
[42] Liang Wu, Fred Morstatter, Xia Hu, and Huan Liu. 2016. Mining misinformation in social media. *Big Data in Complex and Social Networks* (2016), 123–152.
[43] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1365–1374.
[44] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th international conference on*. IEEE, 1151–1156.