Wikipedia Vandal Early Detection: From User Behavior to User Embedding

Shuhan Yuan[1], Panpan Zheng[2], Xintao Wu[2(✉)], and Yang Xiang[1]

[1] Tongji University, Shanghai 201804, China
{shxiangyang,4e66}@tongji.edu.cn
[2] University of Arkansas, Fayetteville, AR 72701, USA
{pzheng,xintaowu}@uark.edu

Abstract. Wikipedia is the largest online encyclopedia that allows anyone to edit articles. In this paper, we propose the use of deep learning to detect vandals based on their edit history. In particular, we develop a multi-source long-short term memory network (M-LSTM) to model user behaviors by using a variety of user edit aspects as inputs, including the history of edit reversion information, edit page titles and categories. With M-LSTM, we can encode each user into a low dimensional real vector, called user embedding. Meanwhile, as a sequential model, M-LSTM updates the user embedding each time after the user commits a new edit. Thus, we can predict whether a user is benign or vandal dynamically based on the up-to-date user embedding. Furthermore, those user embeddings are crucial to discover collaborative vandals. Code and data related to this chapter are available at: https://bitbucket.org/bookcold/vandal_detection.

1 Introduction

Wikipedia, as one of the world's largest knowledge bases on the web, heavily relies on thousands of volunteers to make contributions. This crowdsourcing mechanism based on the freedom-to-edit model (i.e., any user can edit any article) leads to a rapid growth of Wikipedia. However, Wikipedia is plagued by vandlism, namely "deliberate attempts to damage or compromise integrity"[1]. Those vandals who commit acts of vandalism damage the quality of articles and spread false information, misleading information, or nonsense to Wikipedia users as well as information systems such as search engines and question-answering systems.

Reviewing millions of edits every month incurs an extremely high workload. Wikipedia has deployed a number of tools for automatic vandalism detection, like ClueBot NG[2] and STiki[3]. These tools use heuristic rules to detect and revert apparently bad edits, thus helping administrators to identify and block vandals. However, those bots are mainly designed to score edits and revert the worst-scoring edits.

[1] https://en.wikipedia.org/wiki/Wikidata:Vandalism.
[2] https://en.wikipedia.org/wiki/User:ClueBot_NG.
[3] https://en.wikipedia.org/wiki/Wikipedia:STiki.

M. Ceci et al. (Eds.): ECML PKDD 2017, Part I, LNAI 10534, pp. 832–846, 2017.
https://doi.org/10.1007/978-3-319-71249-9_50

Detecting vandals and vandalized pages from crowdsourcing knowledge bases has attracted increasing attention in the research community [12,15,21]. For example, [12] focused on predicting whether an edit is vandalism. They developed a set of 47 features that exploit both content and context information of users and edits. The content features of an edit are defined at levels of character, word, sentence, and statement whereas the context features are used to quantify users, edited items, and their respective revision histories. [15] focused on predicting whether a user is a vandal based on user edits. The developed VEWS system adopted a set of behavior features based on edit-pairs and edit-patterns, such as vandals make faster edits than benign user, benign users spend more time editing a new page than vandals, or benign users more likely edit a meta-page than vandals. All the above features empirically capture the differences between good edits and vandalism to some extent and there is no doubt that classifiers (e.g., randomforest or SVM) with these features as inputs can achieve good accuracy of detecting vandalism.

Different from the existing approaches that heavily rely on hand-designed features, we tackle the problem of vandal detection by automatically learning user behavior representations from their edit sequences. Each edit in a user's edit sequence contains many attributes such as PageID, Title, Time, Categories, PageType, Revert Status, and Content. We transform each edit sequence into multiple aspect sequences and develop a multi-source long-short term memory network (M-LSTM) to detect vandals. Each LSTM processes one aspect sequence and learns the hidden representation of the corresponding aspect of user edits. The LSTM as a sequence model can represent the user edit sequence with variable-length as fixed-length real vectors, i.e., aspect representations. We then apply the attention model [4,33] to derive the user representation, called user embedding, by combining all aspect representations. The user embedding accurately captures all aspects of a user's edit information. Thus we can use user embeddings as classifier inputs to separate vandals from benign users. To the best of our knowledge, this is the first work to use the deep neural network to represent users as user embeddings which capture the information of user behavior for vandal detection.

Our approach has several advantages over past efforts. First, neither heuristic rules nor hand-designed features are needed. Second, while each user may have a different number of edits and each user may have different edit behavior, we map each user into the same low-dimensional embedding space. Thus user embeddings can be easily used for a variety of data mining tasks such as classification, clustering, outlier analysis, and visualization. Third, by using various aspect information (e.g., article title and categories), our M-LSTM is able to effectively capture hidden relationships between different users. The derived user embeddings can be used to analyze collaborative vandals who commit acts of vandalism together to impose big damages and/or evade detection. Fourth, our M-LSTM can naturally achieve early vandal detection and has great potential to be deployed for dynamically monitoring user edits and conducting real-time vandal detection.

The rest of the paper is organized as follows. Section 2 summarizes the related work about deep neural networks and the vandal detection. Section 3 introduces our M-LSTM model for vandal (early) detection. The experimental results and analysis are discussed in Sect. 4. Section 5 concludes the paper.

2 Related Work

Deep neural networks have achieved promising results in image [11], text [20], and speech recognition [9]. The key ingredient for the successful of deep neural network is because it learns meaningful representations of inputs [5]. For example, in text area, all the words are trained to represent as real-valued vectors called word embeddings which capture the semantic relations among words [20]. Then, a neural network model can further combine the word embeddings to represent the sentences or documents [33]. For image recognition, a deep neural network can learn different levels of image representations on different levels of the neural network [38]. In this work, we propose a M-LSTM model to train the representations of users and further use them to predict vandals.

Most work for vandalism detection extracts features, e.g., content-based features [1,14,21], context features to measure user reputation [2], spatial-temporal user information [31], and then uses those features as classifier inputs to predict vandalism. Moreover, [19] utilizes search engine to check the correctness of user edits. However, it is difficult to apply these approaches based on hand-design features to detect subtle and collaborative vandalism.

Wikipedia vandal detection is related to fake review detection. In [22], different types of behavior features were extracted and used to detect the fake reviews in Yelp. [17] have identified several representative behaviors of review spammers. [32] studied the co-anomaly patterns in multiple review-based time series. [8] proposed approaches to detect fake reviews by characterizing burstiness of review. There has been extensive research on detecting anomaly from graph data [3,28,34] and detecting Web ranking spams [27]. In [35], the authors developed a fraud detection framework that combines deep neural networks and spectral graph analysis. They developed two neural networks, deep autoencoder and convolutional neural network, and evaluated them in a signed graph extracted from Wikipedia data. [23] studied various aspects of content-based spam on the Web and presented several heuristic methods for detecting content based spam. Finding time points at which graph changes significantly given a sequence of graphs has also been studied [24]. Although some of above approaches can be used for vandal detection, they are not able to automatically extract and fuse multiple aspects of user edit behaviors.

Several network embedding methods including DeepWalk [26], LINE [30], Node2vec [10], SNE [36], and DDRW [16] have been proposed. These models are based on the neural language model. Some works learn the network embedding by considering the node attribute or other information from heterogeneous networks [6,29]. Unlike all the embedding works described above, in this paper, we explore to build user embedding from user edit sequence data.

3 Multi-source LSTM for Vandal Early Detection

Our key idea is to adopt multiple LSTMs to transform a variable-length edit sequence into multiple fixed-length aspect representations and further use the attention model to combine all aspect representations into the user embedding. As user embeddings capture all aspects of user edits as well as relationships among users, they can be used for detecting vandals and examining behaviors of vandalism.

3.1 LSTM Revisited

Long short-term memory network [13], as one class of recurrent neural networks (RNNs), was proposed to model the long-range sequences and has achieved great success in natural language processing and speech recognition recently. Figure 1 shows the structure of the standard LSTM for classification. Given a sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ where $\mathbf{x}_t \in \mathbb{R}^d$ denotes the input at the t-th step, LSTM maintains a hidden state vector $\mathbf{h}_t \in \mathbb{R}^h$ to keep track the sequence information with the input from the current step $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. LSTM is composed by a special unit called memory block in the recurrent hidden layer to compute the hidden state vector $\mathbf{h}_t$. Each memory block contains self-connected internal memory cells and special multiplicative units called gates to control what kinds of information need to be encoded to the internal memory or discarded. Each memory block has an input gate to control the input information into the memory cell, a forget gate to forget or reset the current memory, and an output gate to control the output of cell into the hidden state. Formally, the hidden state $\mathbf{h}_t$ is computed by

$$\begin{aligned}
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where σ is the sigmoid activation function; $\odot$ represents element-wise product; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ indicate the input gate, forget gate, output gate, and cell activation vectors and $\tilde{\mathbf{c}}_t$ denotes the intermediate vector of cell state; $\mathbf{W}$ and $\mathbf{U}$ are the weight parameters; $\mathbf{b}$ is the bias term. We denote all LSTM parameters ($\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$) as $\boldsymbol{\Theta}_1$.

After the LSTM reaches the last step T, $\mathbf{h}_T$ encodes the information of the whole sequence and is considered as the representation of the sequence. It can then be used as an input of the softmax classifier,

$$P(\hat{y} = k|\mathbf{h}_T) = \frac{\exp\left(\mathbf{w}_k^T\mathbf{h}_T + b_k\right)}{\sum_{k'=1}^{K}\exp(\mathbf{w}_{k'}^T\mathbf{h}_T + b_{k'})}, \tag{2}$$

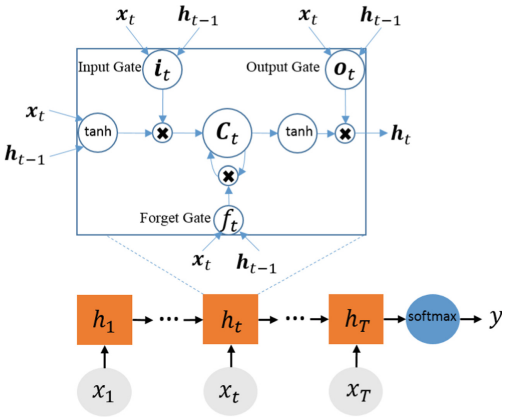

Fig. 1. Standard LSTM for classification

where K is number of classes, $\hat{y}$ is the predicted class of the sequence, $\mathbf{w}_k$ and b_k are the parameters of softmax function for the k-th class, and $\mathbf{w}_k^T$ indicates the transpose of $\mathbf{w}_k$. All softmax parameters $\mathbf{W}_k$ and $\mathbf{b}_k$ over K classes are denoted as $\mathbf{\Theta}_2$. The LSTM model aims to optimize $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ by minimizing the cross entropy loss function,

$$\boldsymbol{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(P(\hat{y}_i)), \tag{3}$$

where y_i is the true class of the i-th sequence, and N is the number of training sequences.

3.2 Multi-source LSTM

We develop a multi-source LSTM model to capture all useful aspects of edits. As different aspects carry different weights for vandal detection, we adopt the attention model [4,33] to dynamically learn the importance of each aspect. The user embeddings are then used as inputs of softmax classifier to separate vandals from benign users.

Formally, for a user u with T edits, his edits can be modeled as a sequence $\mathbf{e}_u = (\mathbf{e}_{u_1}, \ldots, \mathbf{e}_{u_t}, \ldots, \mathbf{e}_{u_T})$ where $\mathbf{e}_{u_t}$ includes all related information about the t-th edit. Please note different users may have different numbers of edits. For each edit sequence $\mathbf{e}_u$, we transform it into M aspect sequences. Its m-th aspect sequence, denoted as $\mathbf{x}^{(m)} = (\mathbf{x}_1^{(m)}, \mathbf{x}_2^{(m)}, \ldots, \mathbf{x}_T^{(m)})$, captures the m-th aspect of edit information and is used as the input of the m-th LSTM in our multi-source LSTM. Figure 2 illustrates our M-LSTM model with M aspect sequences as inputs. The last hidden states $\mathbf{h}_T^{(m)}$ $(m = 1, \cdots, M)$ encode all the aspect

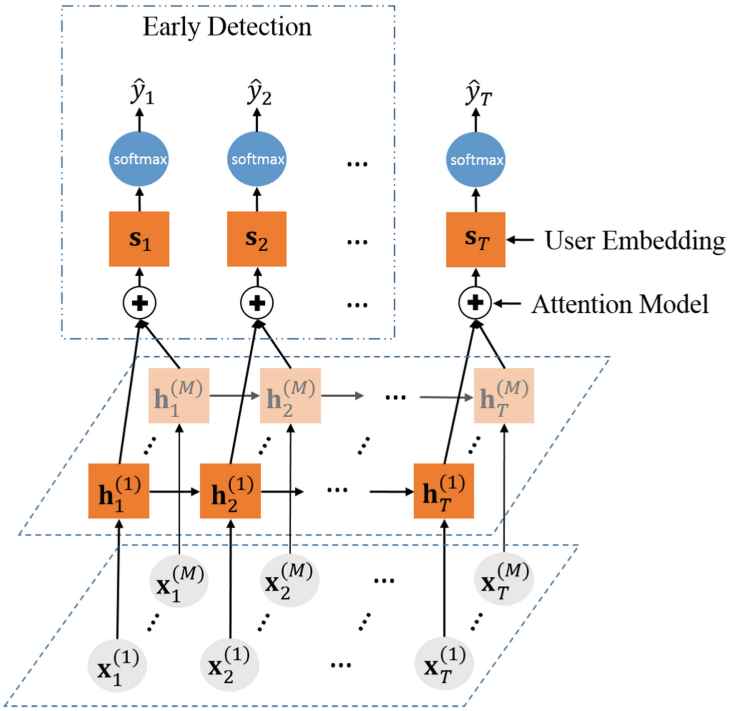

Fig. 2. Multi-source LSTM

information of the user's edit sequence. We apply the attention model as shown in Eqs. 4–6 to combine all aspect information into the user embedding.

$$\mathbf{z}_T^{(m)} = \tanh(\mathbf{W}_a \mathbf{h}_T^{(m)}), \tag{4}$$

$$\alpha_T^{(m)} = \frac{\exp(\mathbf{u}_a^T \mathbf{z}_T^{(m)})}{\sum_{m'=1}^{M} \exp(\mathbf{u}_a^T \mathbf{z}_T^{(m')})}, \tag{5}$$

$$\mathbf{s}_T = \sum_{m=1}^{M} \alpha_T^{(m)} \cdot \mathbf{h}_T^{(m)}, \tag{6}$$

where $\mathbf{W}_a \in \mathbb{R}^{h*h}$ is a trained projection matrix; $\mathbf{u}_a \in \mathbb{R}^h$ is a trained parameter vector. All the parameters, $\mathbf{W}_a$ and $\mathbf{u}_a$, in the attention model are denoted as $\mathbf{\Theta}_3$.

In the attention model, we first compute a hidden representation $\mathbf{z}_T^{(m)}$ of the last hidden state $\mathbf{h}_T^{(m)}$ based on a one-layer neural network by Eq. 4. After obtaining all the M hidden representations, $\mathbf{z}_T^{(1)}, \ldots, \mathbf{z}_T^{(M)}$, we apply the softmax function to calculate the weight of each hidden state $\alpha_T^{(m)}$ by Eq. 5. Finally, we compute the user embedding $\mathbf{s}_T$ as the weighted sum of the M hidden states by Eq. 6. The advantage of the attention model is that it can dynamically learn

a weight of each aspect according to its relatedness with the user class (e.g., vandal or benign).

We use the user embedding $\mathbf{s}_T$ to predict $P(\hat{y}|\mathbf{s}_T)$, i.e., the probability of user u belonging to each class k based on the softmax function shown in Eq. 2. We adopt the standard cross-entropy loss function (Eq. 3) to train our M-LSTM model.

Algorithm 1 shows the pseudo-code of M-LSTM training. Given a training dataset D that contains edit sequences and class labels of N users, we first construct the M aspects of edit sequences for each user. After initializing the parameters, $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, and $\mathbf{\Theta}_3$, in M-LSTM, in each running, we compute the M last hidden states by LSTM networks (Line 8). Then, we adopt the attention model to combine the M hidden states to the user embedding (Line 9). Finally, we update the parameters of M-LSTM by using the user embedding to predict the user label (Line 10). The parameters of M-LSTM are optimized by Adadelta [37] with the back-propagation.

Algorithm 1. Multi-source LSTM Training

Inputs : $D = \{(\mathbf{e}_u, y_u); u = 1, \cdots, N\}$
Maximum training epoch *Epoch*
Outputs: Well-trained parameters $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, $\mathbf{\Theta}_3$

1 **foreach** *user u in D* **do**
2 construct M aspect sequences $\mathbf{x}^{(m)}$ $(m = 1, \ldots, M)$ from the edit sequence $\mathbf{e}_u$;
3 **end**
4 initialize parameters $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, $\mathbf{\Theta}_3$ in M-LSTM;
5 $j \leftarrow 0$;
6 **while** $j <$ *Epoch* **do**
7 **foreach** *user u in D* **do**
8 compute $\mathbf{h}_T^{(m)}$ $(m = 1, \ldots, M)$ on M sequences of aspect vectors;
9 compute the user embedding $\mathbf{s}_T$ by the attention model (Eqs. 4, 5, 6) on M last hidden states;
10 optimize the parameter $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, $\mathbf{\Theta}_3$ in M-LSTM based on the loss function shown in Eq. 3 with Adadelta.
11 **end**
12 $j \leftarrow j + 1$;
13 **end**

M-LSTM for Vandal Early Detection. Our trained M-LSTM model can then be used to predict whether a new user v is vandal or benign given his edit sequence $\mathbf{e}_v = (\mathbf{e}_{v_1}, \cdots, \mathbf{e}_{v_t}, \cdots)$. The upper-left region of Fig. 2 shows our M-LSTM based vandal early detection. At each step t, we first derive its M aspect sequences from the user's edit sequence till the step t. The hidden states are updated with the new input $\mathbf{e}_{v_t}$. Thus, they are able to capture all user's edit aspects until the t-th step. We then adopt the attention model shown in

Eqs. 4, 5, and 6 (replacing all subscript T with t) to calculate the user embedding $\mathbf{s}_t$. The user embedding $\mathbf{s}_t$ captures all the user's edit information till the step t. Then, we can use the classifier to predict the probability $P(\hat{y}|\mathbf{s}_t)$ of the user to be a vandal based on $\mathbf{s}_t$. We set a threshold τ to evaluate whether the user is vandal. When $P(\hat{y}|\mathbf{s}_t) > \tau$, the user is labeled as vandal.

4 Experiments

We conduct our evaluation on UMDWikipedia dataset [15]. This dataset contains information of around 770K edits from Jan 2013 to July 2014 (19 months) of 17105 vandals and 17105 benign users. We focus on identifying the user behaviors on the Wikipedia articles. We remove those edits on meta pages (i.e., with titles containing "User:", "Talk:", "User talk:", "Wikipedia:") because they do not cause damages.

For each edit, we extract three aspects, article title, article categories, and revert status. We choose these three aspects because both the title and categories capture the topic information of the edited article and revert status (reported by bots) indicates whether the edit is good or bad. It is imperative to use them to derive user embeddings and then predict whether users are vandal or benign.

We represent article titles and categories to their title embeddings and category embeddings based on word embeddings. Specifically, we first map each word in the titles and categories to its word embedding and then adopt average operation over the word embeddings to get the title embeddings and category embeddings, respectively. The title embeddings and category embeddings reflect the hidden features about the pages. We use the off-the-shelf pre-trained word embeddings[4] provided by [25]. These word embeddings are widely used and have been shown to achieve good performances on many NLP tasks. We randomly initialize the words which don't have pre-trained word embeddings. The dimension of the word embeddings is 50. The dimension of the hidden layer of the M-LSTM network is 32. The training epoch is 25.

4.1 Vandal Detection

To evaluate the performance of vandal detection, we split the training and testing dataset chronologically. We use the first 9 months of users as the training dataset and the last 10 months of users as the testing dataset. The training dataset has 8620 users and the testing dataset has 10418 users. We report the mean values of 10 different runs.

Table 1 shows the precision, recall, F1 and accuracy for vandal detection with different thresholds. Precision indicates the ratio of vandals who are correctly detected. Recall indicates the ratios of vandals who are correctly detected from the test dataset. The default threshold for binary classification used in vandal detection is 0.5, where our model achieves the best performance. We can also

[4] http://nlp.stanford.edu/projects/glove/.

Table 1. Experimental results on precision, recall, F1, and accuracy of vandal detection with different thresholds

τ	Precision	Recall	F1	Accuracy
0.5	88.35%	96.67%	92.32%	91.33%
0.6	88.69%	96.01%	92.20%	91.24%
0.7	89.31%	94.85%	92.00%	91.10%
0.8	90.36%	92.27%	91.31%	90.52%
0.9	93.13%	74.10%	82.53%	83.09%

observe that the model achieves good performances of vandal detection with different thresholds τ from 0.5 to 0.8. Meanwhile, with increasing τ, the precision increases accordingly while the recall decreases, which indicates with a higher threshold, the model can detect vandal more accurately but can mis-classify the vandals as benign users. Overall, the F1 and accuracy decease significantly at the 10^{-4} level with t-tests while the threshold $\tau > 0.6$.

We further compare our results with the VEWS approach [15]. The VEWS approach uses a set of hand-crafted features to detect vandals. When incorporating the revision status information, the VEWS can achieve around 90% classification accuracy. Our M-LSTM achieves better accuracy on vandal detection. More importantly, our M-LSTM does not need to design dozens of features to predict vandals. Hence our model can be easily extended to identify vandals from other crowdsourcing knowledge bases like Wikidata.

4.2 User Embeddings

As each user has different edits and each edit has many different aspects, it is challenging to derive users' edit patterns. Our M-LSTM derives user embeddings based on user edits. As user embeddings capture user edit behaviors, they can then be used to differentiate between benign users and vandals and detect potential collaborative vandals.

Visualization. We randomly select user embeddings of 3000 users and map them to a two-dimensional space based on t-SNE approach [18]. Figure 3 shows the visualization plot of user embeddings from M-LSTM. We observe that the benign users and vandals are clearly separated in the projected space. This indicates the user embeddings successfully capture the information whether a user is benign or vandal. Hence, they can be used for vandal detection.

Clustering of User Embedding. In this experiment, we adopt the classic DBSCAN algorithm [7] to cluster user embeddings. We set the maximum radius of the neighborhood $\epsilon = 0.05$ and the minimum number of points $minPts = 3$. DBSCAN produces 211 clusters. Among them, 139 clusters contain only vandals and the total number of vandals is 502 whereas 46 clusters contain only benign users and the total number of benign users is 495. It indicates the benign users

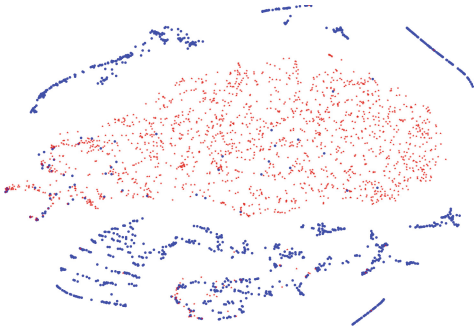

Fig. 3. Visualization of 3000 users in the dataset. Color of a node indicates the type of the user. Red: "Vandals", blue:"Benign Users". (Color figure online)

often form large-size clusters. On the contrary, the vandals usually cluster to small groups to damage articles. For the rest 26 clusters that contain mixed vandals and benign users, there are 17 clusters in which the vandals constitute the majority and 9 clusters in which the benign users constitute the majority. Similarly, the 17 (vandal-majority) clusters are small with only 52 vandals whereas there are 3663 benign users in the 9 (benign-majority) clusters. Embeddings of benign users are closer to each other than that of vandals, which can also be observed in Fig. 3. We conclude that "Benign users are much more alike; every vandal vandalizes in its own way."

When setting the maximum radius of the neighborhood $\epsilon = 0$ and the minimum number of points $minPts = 2$, DBSCAN produces 701 groups containing 1687 user embeddings. Note that under this setting all embeddings within the same group are exactly the same. Among them, 575 groups only contain vandals and the total number of vandals is 1396 whereas 68 groups only contain benign users and the total number of benign users is 144. The largest vandal group contains 13 vandals and the largest benign group contains 17 benign users.

Table 2 shows three examples of potential collaborative vandal groups. In Row 1, the group has 8 vandals who attacked the same page consecutively within a short time window. In Row 2, the group has three vandals who attacked one same page on different days. Because all these vandals were blocked after revising the page, these vandals have high chance to be controlled by a malicious user or group and aim to vandalize the specific page. In Row 3, we show a vandal group containing five vandals. All the five vandals edited the same two pages, "Shakugan no Shana" and "The Familiar of Zero", which are both Japanese light novels, consecutively within a short time window. These three examples demonstrate one advantage of our M-LSTM, i.e., detecting potential collaborative vandals with different behavior patterns.

In Table 3, we further show article titles edited by three pairs of vandals. Each pair of vandals are closed to each other in the embedding space. The first row shows that two vandals damage almost the same pages, which indicates vandals who edit the same pages are close to each other. The second row shows pages edited by two vandals have common words in titles although the title

Table 2. Three example of potential collaborative vandal groups. The vandals of each group damage the same page(s). Group 1 damages the page "List of the X factor finalists (U.S. season 2)" in 2013-01-05 within a short time window. Group 2 damages the page "Niels Bohr" on different days. Group 3 damages the two pages, "Shakugan no Shana" and "The Familiar of Zero", consecutively in 2014-04-18 within a short time window.

Group ID	User ID	Page ID	Revision time
Group 1 2013-01-05	4203021	37310371	02:36:32
	4203016		02:42:02
	4203009		02:42:55
	4203006		02:44:58
	4202998		02:45:32
	4203002		02:47:12
	4202988		02:52:12
	4202986		02:56:21
Group 2	4584127	21210	2013-10-04
	4597541		2013-10-23
	4939865		2014-01-08
Group 3 2014-04-18	5063994	2548832	21:33:51
		5982921	21:34:07
	5063996	2548832	21:35:53
		5982921	21:35:53
	5063998	2548832	21:45:06
		5982921	21:45:28
	5064002	2548832	21:47:21
		5982921	21:47:29
	5064006	2548832	21:48:56
		5982921	21:49:01

names are different. This indicates our M-LSTM can discover the semantic collaborative behaviors on pages based on user embeddings. The last row shows that our M-LSTM can further identify vandals who damage the pages with similar subject areas although there are no any common words in the titles. This example shows the usefulness of incorporating page category information in our M-LSTM. All above examples demonstrate that users who are closed in the low-dimensional user embedding space have similar edit patterns. Therefore, analyzing user embeddings can help capture and understand collaborative vandal behaviors.

Table 3. Three pairs of vandals and their edited page titles. Each pair has similar embeddings based on the cosine similarity.

Vandal IDs	Page title	Page title
4266603 and 4498466	Live While We're Young, What Makes You Beautiful, Up All Night (One Direction album), Take Me Home (One Direction album)	Live While We're Young, Best Song Ever (song), What Makes You Beautiful, Up All Night (One Direction album), Take Me Home (One Direction album)
4422121 and 4345947	Super Mario 3D World, Super Mario Galaxy, Sonic Lost World, Pringles, Action Girlz Racing, Data Design Interactive	Super Mario World, Super Mario World 2: Yoshi's Island, Super Mario Bros. 3, Virtual Boy, Nintendo DS, Kirby Super Star, Yogurt
5032888 and 4592537	Matthew McConaughey, Maggie Q, Theo James, Theo James, Dexter (TV series), Laker Girls, Bayi Rockets, Arctic Monkeys, Dulwich College Beijing	Nicolas Cage, Alan Carr, Liam Neeson, Dale Winton, Craig Price (murderer), Manuel Neuer

4.3 Vandal Early Detection

Our vandal early detection is achieved after each edit is submitted. Although our M-LSTM exploits revert status of the edit, we emphasize that the revert status is inspected by the ClueBot NG in a real time manner. Hence, our M-LSTM can be deployed for real time vandal detection. We evaluate the vandal early detection on the 6427 users who have at least two edits in the testing dataset. Table 4 shows the precision, recall and F1 of our M-LSTM on vandal early detection. We vary the threshold τ from 0.5 to 0.9. Similar to the results of vandal detection shown in Table 1, with increasing τ, a classifier with a higher threshold has more confidence about the prediction, resulting in a higher precision. On the contrary, the recall decreases with the increasing of τ because fewer users will be marked as vandals. The F1 score increases significantly at the 0.005 level with $\tau > 0.6$ and reaches the maximum with $\tau = 0.9$. However, comparing with the results of vandal detection, the vandal early detection has a lower precision but much higher recall. This indicates that we lose some precision but achieve big recall when using partial edit information to do early vandal detection.

Table 4 further shows the average number of edits before the vandals were blocked by the administrators and the ratio of vandals who can be early detected over the whole testing dataset. We can observe that the average number of edits and the ratio of early detected vandals both have a significant decreasing while the threshold $\tau = 0.9$. Note that the ratios of early detected vandals with thresholds from 0.5 to 0.8 are only a little lower than the recall values, which indicates that most of the vandals who are correctly detected are early detected. Overall, setting threshold $\tau = 0.8$ will achieve a balance performance between vandal early detection and accurate prediction.

Table 4. Experimental results on precision, recall and F1 of vandal early detection, the average number of edits before the vandals are blocked, and the ratio of vandals who are early detected.

τ	Precision	Recall	F1	# of edits	% of early detected
0.5	84.10%	99.07%	90.97%	3.50	97.35%
0.6	84.96%	98.99%	91.44%	3.48	96.87%
0.7	85.81%	98.82%	91.86%	3.41	95.94%
0.8	86.88%	98.76%	92.44%	3.33	93.34%
0.9	89.89%	98.34%	93.93%	2.48	72.32%

5 Conclusion and Future Work

In this paper, we have developed a multi-source LSTM model to encode the user behaviors to user embeddings for Wikipedia vandal detection. The M-LSTM is able to simultaneously learn different aspects of user edit information, thus user embeddings accurately capture the different aspects of user behaviors. Our M-LSTM achieves the state-of-the-art results on vandal detection. Furthermore, we show that user embeddings are able to identify collaborative vandal groups with various patterns. Different from existing works which require a list of hand-designed features, our M-LSTM can automatically learn user embeddings from user edits. The user embeddings can be used for a variety of data mining tasks such as classification, clustering, outlier analysis, and visualization. Our empirical evaluation has demonstrated its potential for analyzing collaborative vandals and early vandal detection.

In the future, we plan to incorporate into our M-LSTM more information about user edits, e.g., user-user relations and hyperlink relations among articles. These relations are modeled as graphs and can be naturally incorporated into M-LSTM by using network embedding approaches [26,30]. We also plan to conduct comprehensive evaluations on collaborative vandal detection.

Repeatability. Our software together with the datasets used in this paper are available at https://bitbucket.org/bookcold/vandal_detection.

Acknowledgments. The authors would like to thank anonymous reviewers for their valuable comments and suggestions. The authors acknowledge the support from the 973 Program of China (2014CB340404), the National Natural Science Foundation of China (71571136), and the Research Projects of Science and Technology Commission of Shanghai Municipality (16JC1403000, 14511108002) to Shuhan Yuan and Yang Xiang, and from National Science Foundation (1564250) to Panpan Zheng and Xintao Wu. This research was conducted while Shuhan Yuan visited University of Arkansas.

References

1. Adler, B.T., de Alfaro, L.: A content-driven reputation system for the wikipedia. In: WWW, pp. 261–270 (2007)
2. Adler, B.T., de Alfaro, L., Mola-Velasco, S.M., Rosso, P., West, A.G.: Wikipedia vandalism detection: combining natural language, metadata, and reputation features. In: CICLing, pp. 277–288 (2011)
3. Akoglu, L., McGlohon, M., Faloutsos, C.: oddball: Spotting anomalies in weighted graphs. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS (LNAI), vol. 6119, pp. 410–421. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13672-6_40
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
5. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. TPAMI **35**(8), 1798–1828 (2013)
6. Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: KDD (2015)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
8. Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., Ghosh, R.: Exploiting burstiness in reviews for review spammer detection. In: ICWSM, pp. 175–184 (2013)
9. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP, pp. 6645–6649 (2013)
10. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD (2016)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
12. Heindorf, S., Potthast, M., Stein, B., Engels, G.: Vandalism detection in wikidata. In: CIKM, pp. 327–336 (2016)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Javanmardi, S., McDonald, D.W., Lopes, C.V.: Vandalism detection in wikipedia: a high-performing, feature-rich model and its reduction through lasso. In: WikiSym, pp. 82–90 (2011)
15. Kumar, S., Spezzano, F., Subrahmanian, V.: Vews: a wikipedia vandal early warning system. In: KDD, pp. 607–616 (2015)
16. Li, J., Zhu, J., Zhang, B.: Discriminative deep random walk for network classification. In: ACL (2016)
17. Lim, E.P., Nguyen, V.A., Jindal, N., Liu, B., Lauw, H.W.: Detecting product review spammers using rating behaviors. In: CIKM, pp. 939–948 (2010)
18. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. JMLR **9**, 2579–2605 (2008)
19. McKeown, K., Wang, W.: Got you!: automatic vandalism detection in wikipedia with web-based shallow syntactic-semantic modeling. In: COLING, pp. 1146–1154 (2010)
20. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)

21. Mola-Velasco, S.M.: Wikipedia vandalism detection through machine learning: feature review and new proposals. arXiv:1210.5560 [cs] (2012)
22. Mukherjee, A., Venkataraman, V., Liu, B., Glance, N.S.: What yelp fake review filter might be doing? In: ICWSM, pp. 409–418 (2013)
23. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: WWW, pp. 83–92 (2006)
24. Papadimitriou, P., Dasdan, A., Garcia-Molina, H.: Web graph similarity for anomaly detection. J. Internet Serv. Appl. **1**(1), 19–30 (2010)
25. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
26. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: KDD, pp. 701–710 (2014)
27. Spirin, N., Han, J.: Survey on web spam detection: principles and algorithms. SIGKDD Explor. Newsl. **13**(2), 50–64 (2011)
28. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: ICDM, pp. 1–8 (2005)
29. Tang, J., Qu, M., Mei, Q.: PTE: Predictive text embedding through large-scale heterogeneous text networks. In: KDD (2015)
30. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
31. West, A.G., Kannan, S., Lee, I.: Detecting wikipedia vandalism via spatio-temporal analysis of revision metadata? In: EUROSEC, pp. 22–28 (2010)
32. Xie, S., Wang, G., Lin, S., Yu, P.S.: Review spam detection via temporal pattern discovery. In: KDD, pp. 823–831 (2012)
33. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: NAACL, pp. 1480–1489 (2016)
34. Ying, X., Wu, X., Barbará, D.: Spectrum based fraud detection in social networks. In: Proceedings of the 27th International Conference on Data Engineering, Hannover, Germany, pp. 912–923. ICDE 2011, 11–16 April 2011 (2011)
35. Yuan, S., Wu, X., Li, J., Lu, A.: Spectrum-based deep neural networks for fraud detection. CoRR abs/1706.00891 (2017)
36. Yuan, S., Wu, X., Xiang, Y.: SNE: signed network embedding. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10235, pp. 183–195. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57529-2_15
37. Zeiler, M.D.: Adadelta: An adaptive learning rate method. arXiv:1212.5701 [cs] (2012)
38. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53