# People2Vec: Learning Latent Representations of Users Using Their Social-Media Activities

Sumeet Kumar(✉) and Kathleen M. Carley

Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
{sumeetku,kathleen.carley}@cs.cmu.edu

**Abstract.** In most social network studies, it is assumed that nodes are simple and carry no information, and links are explicit ties such as friendship. Which nodes are in which group is determined as a function of these explicit ties. For example, given a set of random walks through the network, it is possible to learn a vector for each node which contains a latent representation of the node. These latent representations have useful properties that can be easily exploited by statistical models for tasks like identifying groups and inferring implicit links. However, most existing representation learning methods ignore node attributes. In many cases, there is a rich body of information and events associated with nodes that also can be used for node clustering and to infer ties. In social media, e.g., an explicit relationship is friendship, and another is the follower-followee relation. Besides, there are the set of messages passed by the users, as well as, their activities in the form of liking or mentioning. What is needed is a way of collectively using both the explicit ties and this rich body of additional information in learning these latent node representations. Combining such data should enable more effective link inference and grouping strategies. In this research, we propose *People2Vec* an algorithm to learn representations that takes into account proximity between users due to their social media activities. We validate our model by experiments on two different social-media datasets and find the model to perform better than prior state-of-the-art approaches.

## 1 Introduction

Online social media platforms are popular for sharing information and allowing users to network with each other. Analyzing such social networks is an active area of research. Significant problems in this field are finding communities, predicting interests of users, and recommending friends and content. Typical end goals are to identify groups, infer links and make network predictions. To achieve these goals, it is first necessary to determine for two social-media users, how socially proximate they are. Two individuals who are connected by a friendship tie or a follower-followee tie are more socially proximate than are those not connected. However, just examining the binary friendship or follower-followee ties is insufficient for assessing there overall social proximity. Users are more

R. Thomson et al. (Eds.): SBP-BRiMS 2018, LNCS 10899, pp. 154–163, 2018.
https://doi.org/10.1007/978-3-319-93372-6_17

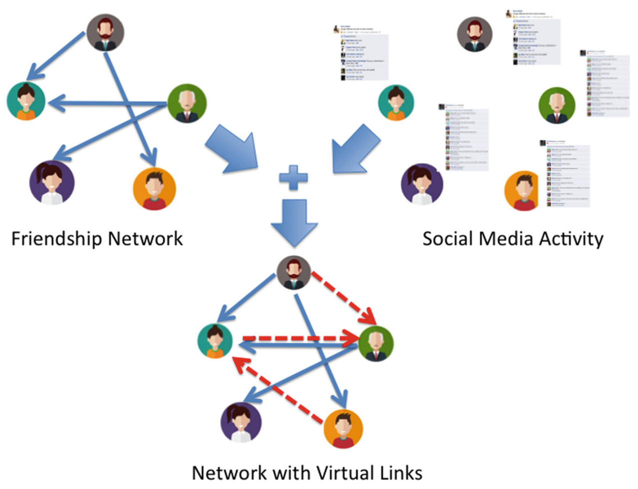

**Fig. 1.** Using just the binary friendship network, the only similarity between the orange and green person is that they are both friends of the brown person. Each person also has social media activity, e.g. a set of messages that they send. Combining these two types of data can generate new weighted virtual links (the red dashed lines) between nodes and reveal hidden connections. (Color figure online)

socially proximate vis-a-vis a topic, the more they have in common. Learned representations using random walks over the network links provide a better feature to find the social proximity of users, but they still miss important cues such as what they say or feel about a topic. We argue, that in addition to the explicit ties among users, the activities and preferences of the social media users could be used to find weighted *virtual links* that can be leveraged to learn more useful node representations (Fig. 1).

In this paper we present People2vec, a latent space representation of the user in context as a vector. This representation supports generalization and the identification of similar actors and so groups. This representation is learned from the data and captures the complexities of the situation in which the user is embedded. Our approach is inspired by an analogous problem in language technology which is to learn a representation of a word from the common context of the word in sentences. To that end we draw on the word2vec approach for identifying the context of the node by random traversals of the network similar to [4,8]. However, we move beyond this approach by bringing in user activities as node attributes, which enables the inference of additional links and solves the sparse network problem. Existing network analytic tools, like ORA [2], uses both the network and the attributes on the nodes. By exploiting attribute information to infer the network, People2vec supports a more detailed analysis.

The important contributions of this paper are:

- We propose People2vec, a model to learn node representations that captures similarity in users' attributes and activities, in addition to their friendship links.
- Our model extends the popular random walk approach of learning node representation and brings valuable improvements, yet preserves the simplicity of the approach.

The paper is outlined as follows. First, we discuss prior work in Sect. 2. Then, we introduce our 'People2Vec Model' in Sect. 3. In Sect. 4, we present our experiments on two different datasets, along with a discussion of the results. We finally conclude and discuss the future work.

## 2 Related Work

Some recent advancements in learning node representations are inspired from the improvement in natural language processing. Bengio et al. [1] proposed the distributed representations of words aka 'Word embeddings' which was later used by Collobert and Weston [3] to demonstrate their usefulness in many NLP tasks. Mikolov et al. [6] proposed Word2vec, a Skip-gram model for learning high-quality distributed vector representations using skip-gram model. Such representations capture many syntactic and semantic word relationships e.g. they predict: vec('Berlin') - vec('Germany') + vec('France') = vec('Paris'). The concept of learning representations using skip-gram could be applied to networks as well. These latent representations can be learned in a number of ways; e.g. (a) factorization of social network's adjacency matrix (b) learning functions to find better features. Recently, researchers have tried to train neural-networks to find efficient nonlinear transformations for learning node embeddings. However, unlike words in a language that has plenty of examples to get related contextual words, often networks are sparse. Besides, there is no clear notion of social-context in networks. To incorporate context in networks, researchers have tried random walks [4,8]. Random walks on links in a network help to generate contexts that consist of proximity nodes. Like in language models, such context is then used to build embedding vectors (representations) often by training a shallow neural network. Learning low dimensional representation of nodes in networks allow mapping local structural characteristics to a continuous space representation. Learning these representations of nodes have helped to improve performance in many tasks including node classification and link prediction. Though models proposed in [4,8,9] learn good representation on simple graphs, they mostly explore binary edges, so are best suited for social-networks that have clear links as in friendship and follower-followee relationship. They do not exploit node attributes and preferences which are often very relevant and strong indicators in social networks. This gap is the focus of this research.

## 3 People2Vec Model

We consider the problem of learning node representation in social-networks that captures users' preferences, in addition to their explicit links in the form of friendship or follower-followee relationship. We expect a good solution to have the following two properties:

(a) Users with direct links in a network should be closer in latent representation space. Many existing models exhibit this property.

(b) Users with similar preferences should be closer in latent representation space. The way to measure similarity in preferences should be flexible to allow the model to adapt to different formulations of preferences. For example, in one situation, two users discussing a topic could be similar, and in another situation, two users using same tag could be similar.

We formulate the problem as follows: Let $G(V, E)$ be a network where $v \in V$ are nodes (or users) and $e \in E$ are edges. Let $F : v \rightarrow \mathcal{R}^d$ be the function that learns a $d$ dimensional representation ($z$) of a node. Let $Y$ be a matrix of user preferences that contains a set of preferences for each user, where $Y_i^j$ indicates preferences of node $v_i$ towards $j$th item. The $j$th 'item' could be stance towards a topic or the count of a tag (as in hashtag used by a user). The goal of the algorithm is to learn $z_i$, a low-dimensional representation of user $v_i$ that considers the explicit links in $E$ and also the similarity in $Y$ space.

As in Deepwalk [8], we follow the language modeling technique of generating latent representation of words from sentences. In language modeling, given some text corpus W = ($w_0$, $w_1$,..., $w_n$), the goal is to maximize the likelihood $Pr(w_i|w_0, w_1, \ldots w_{i-1}, w_{i+1}, \ldots, w_n)$ over the entire corpus. By analogy, in social networks, we define the likelihood of observing a node $v_i$ given other nodes by $Pr(v_i|v_0, v_1, \ldots v_{i-1}, v_{i+1}, \ldots, v_n)$. In the latent space $F$ of node representations, this can be formulated as maximizing the likelihood of

$$Pr\Big(v_i|\big(F(v_{i-k}), F(v_{i-k+1}), \ldots, F(v_{i-1}), F(v_{i+1}), F(v_{i+k-1}), F(v_{i+k})\big)\Big) \qquad (1)$$

where we only consider $2k$ immediate neighbors on node $v_i$. To efficiently solve such a formulation, we use the skip-gram [6] approach. Taking log the optimization problem can be formulated as:

$$\min_F - \log Pr\Big(\big(v_{i-k}, v_{i-k+1}, \ldots, v_{i+k-1}, v_{i+k}\big)|F(v_i)\Big) \qquad (2)$$

Since a node and its latent vector have symmetry in latent space, the conditional likelihood of a neighbor node $v_j$ given by $Pr(v_j|F(v_i))$ can be approximated as similarity in latent space. As in Deepwalk [8], we use the stochastic gradient descent over neighbor nodes collection generated by random walk to optimize the final objective function. To speed up the training we used hierarchical soft-max [7]. In the proposed model, the neighborhood of a node $v_i$ $(v_0, v_1, \ldots v_{i-1}, v_{i+1}, \ldots, v_n)$ is not limited to nodes reachable by explicit links, and is extended to nodes having similar attributes. We call such links '*virtual links*' (explained next).

## 3.1 Virtual Links from Users' Activities

We define *virtual links* as edges in social-networks that connect users with similar preferences as evident by their involvement on these platforms. These virtual links (like real links) can be used in random-walks to explore node neighborhood, thus enhancing the network information present in the original networks. Moreover, there virtual edges, based on nodes similarities, can also strength the existing linkage. Hence, virtual links enable to learn more meaningful node representations.

There are two possible formulations of virtual-links. A rigid link that is either present or absent, and a weighted link that that gives a probabilistic score of links being present. We go with the probabilistic version of virtual-links as it enables a more flexible learning framework.

The probability of having a virtual link between two users is based on similarity between their activity on social-media platform. As discussed earlier, let's model the activity profile of users as a matrix $Y_i^j$, where $Y_i^j$ indicates preferences of node $v_i$ towards $j$th item. Similarity between users is obtained by measuring similarity between vectors representing users preferences.

The similarity between two users is defined as:

$$Similarity\big(v_k, v_i\big) = Sim\big(Y_k, Y_i)\big) \tag{3}$$

There are a number of ways to measure such similarity. We tried three such similarity measures: (a) Cosine Similarity (b) Hamming Distance (c) Euclidean Distance. On our datasets, we find that 'Cosine Similarity' (CS) performs better than other measures. Hence, we use CS as the similarity measure in rest of the paper.

**Random Walks.** We use random walks to sample neighborhood nodes. The random walks approach provides a more flexible approach over known 'Depth First' and 'Breadth First' sampling as explained in [4]. A random walk starts from a node, say $v_0$, and uses node links to find the next node. In prior studies, the probability of transition from node $v_0$ to $v_i$ is given by:

$$P(v_i|v_0) = \left\{ \begin{matrix} 0, \text{if}(v_0, v_i) \notin E \\ \frac{1}{\sum_k 1}, \text{otherwise} \end{matrix} \right\} \tag{4}$$

where k = Number of links of $v_0$.

**Random Walks over Virtual Links.** In our model, we extend the random walk over nodes to include random walk over virtual links. Figure 2 explains the idea.

To weigh the relative importance of virtual links and to real links, let's introduce a hyper-parameter $\alpha$, a weighing factor. This parameter is tuned based on characteristics on the network under consideration. For random walks over

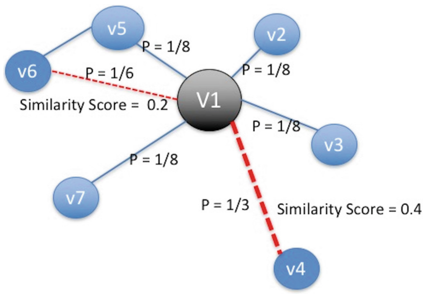

**Fig. 2.** An example of a random walk transition in People2vec. Real links are shown in blue and virtual links are shown in red. The walk originates from node v1. The probability of transition to other nodes is shown in text. Similarity scores for virtual links are obtained using *Sim* function. Here we consider the weighing factor $\alpha = 0.5$, i.e. we weigh the real links and the virtual links equally. For clarity, we have not shown virtual links for nodes already connected via real-links. (Color figure online)

virtual links (see Fig. 2), we use the probability of transition from node $v_0$ to $v_i$ as:

$$P(v_i|v_0) = \left\{ \begin{array}{l} \alpha * Sim'\big(Y(v_0), Y(v_i)\big), \text{if } (v_0, v_i) \notin E \\ (1-\alpha) * \frac{1}{\sum_k 1} + \alpha * Sim'\big(Y(v_0), Y(v_i)\big), \text{if } (v_0, v_i) \in E \end{array} \right\} \quad (5)$$

where k = Number of real links of $v_0$, and $Sim'$ is the normalized similarity score defined as $Sim' = \frac{Sim\big(Y(v_0),Y(v_i)\big)}{\sum_{v_i} Sim\big(Y(v_0),Y(v_i)\big)}$.

### 3.2 People2Vec Algorithm

People2Vec extends the original Deepwalk algorithm [8] by including virtual links. Like in Deepwalk, our algorithm uses random walk to learn node representation. However, in the process of generating walks, the algorithm uses 'virtual links' in addition to the real links, thus considers neighbors that were not accessible in plain random walks. The steps as described in Algorithm 1. Again, similar to Deepwalk, we use skip-gram [6] algorithm to efficiently learn the representations for each node.

## 4 Experiments and Results

In this section, we present the experimental evaluation of our model on two different datasets. The first dataset is a set of blogs and another is a sample of Flickr website. We also evaluate the impact of using different latent-space dimensions.

### 4.1 Datasets

We use two existing datasets (BlogCatalog and Flickr) to evaluate our algorithm. These datasets are publicly available and were used is earlier studies [5].

**Algorithm 1.** The People2Vec algorithm.

```
learnRepresentations
Graph G, VirtualGraph $G_v$
RepresentationDimension d
walks per node r
walk length l
window size w
$walks$ = []
for $iter \in \{1, \ldots, r\}$ do
    for all $node \in V$ do
        $walk$ = randomWalk(G, $G_v$, node, l)
        $walks$.append($walk$)
    end for
end for
SkipGram(F, $walks$, w) (see Ref. [6])
————————————————————————————
randomWalk(Graph G, VirtualGraph $G_v$, Start node u, Length l)
$walk$ = [u]
for $walk\_iter \in \{1, \ldots, l\}$ do
    $currentNode$ = walk[-1]
    $newNode$ = getNeighbor($currentNode$, G, $G_v$)
    $walk$.append($newNode$)
end for
————————————————————————————
getNeighbour(Start node $v_0$, Graph G, VirtualGraph $G_v$)
start at v
$N_r$ = getRealNeighbors(G, $v_0$)
$N_v$ = getVirtualNeighbors($G_v$, $v_0$)
pick neighbor $v_1$ from $[N_v + N_r]$ using $P(v_1|v_0)$ (See Eqn. 5)
```

**Table 1.** Dataset Description

| Dataset | Nodes | Edges | Labels | Attributes |
|---|---|---|---|---|
| BlogCatalog | 5,196 | 171,743 | 6 | 8,189 |
| Flickr | 7,575 | 239,738 | 9 | 12,047 |

**BlogCatalog Dataset.** BlogCatalog is an online community of bloggers. The dataset is created by including keywords used in blog description as attributes [5]. Using those keywords, we generate users preference matrix. In this dataset, the labels used for predicting the classification performance represent bloggers' interests (Table 1).

**Flickr Dataset.** Flickr is popular website that hosts videos and images. Users can follow each other, thus, forming a network. They can join different groups which is used as labels. To get the users' preference matrix, tags by users are used [5].

## 4.2 Baseline Algorithms and Model Optimization

We measure the performance of People2Vec against several state-of-the-art algorithms [4,8,9]. **DeepWalk** [8] uses uniform random walks on networks to learn embeddings as in language modeling techniques like word2vec. **LINE** [9] algorithm preserves both local and global network structures and uses edge sampling approach for optimization. **Node2Vec** [4] extends Deepwalk by combining depth-first and breadth-first search in their sampling strategy.

We use social-media tags to create a nodes' preference vectors. Preference vectors of different nodes are then compared using cosine similarity measures to create weighted virtual links which are later used to learn node representations. Because similarity between nodes generates $O(n^2)$ possible edges, which could result in a very dense graph so we use a threshold to reduce the number of virtual-edges used in learning embeddings. The exact threshold is of lesser importance as People2vec random-walk prefers node transitions to higher similarity nodes, and thus ignores less similar nodes more often. The dimensions of representations used are 64 and 128. For a fair comparison, all algorithms used $walklength = 10$ and $walkcount = 40$. For all models, we use one-vs-rest logistic regression as used in [8]. We trained all the models on an Ubuntu Linux machine with 64 GB ram and eight core Intel i7 processor with 4.00 GHz processing speed.

## 4.3 Experimental Results

We present the results of the experiments. Table 2 shows the top classification performance for the two datasets. Figure 3 shows the trend of F1-score (macro) for different train and test ratio. The plot show that most algorithms have a reasonable performance right from the smallest training percentage (10%). There are small improvements as we increase the training data percentages. Peopl2vec performed better than rest of the algorithms. In general, embeddings with 128 dimension perform better than 64 dimension embeddings. People2Vec is an exception for which scores were very similar. Figure 4 shows the results for the Flickr dataset. Again for most algorithms 128 dimensional embedding performed better than 64 dimensional. Like before, Peopl2vec performed better than all other algorithms.

**Table 2.** Best F1 macro score

| Method | BlogCatalog | Flickr |
|---|---|---|
| DeepWalk | 0.73 | 0.58 |
| Node2vec | 0.72 | 0.58 |
| LINE | 0.73 | 0.58 |
| People2Vec | **0.83** | **0.75** |

We don't compare our results with LANE [5] (BlogCatalog: 0.90 best F1 score, Flick: 0.90 best F1 score) as their implementation uses matrix factorization

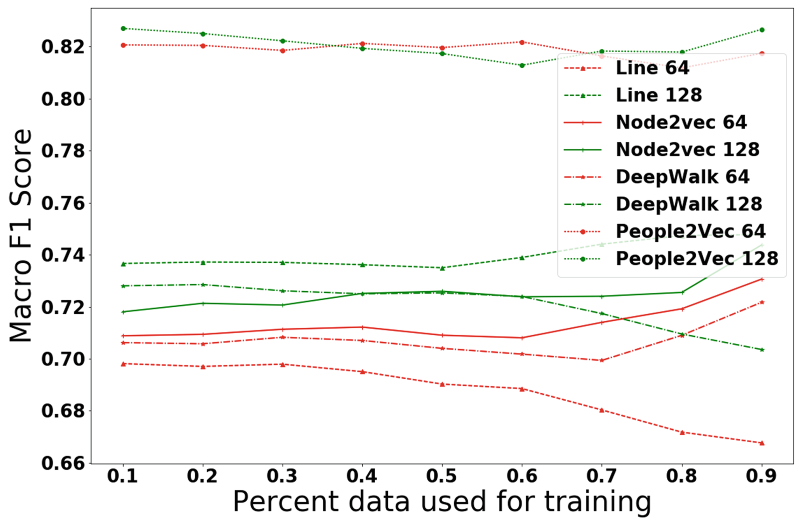

**Fig. 3.** We tried different algorithms to learn the class labels (a proxy of community) of nodes in BlogCatalog graph. In this plot, we compare mean F1 score for different algorithms.

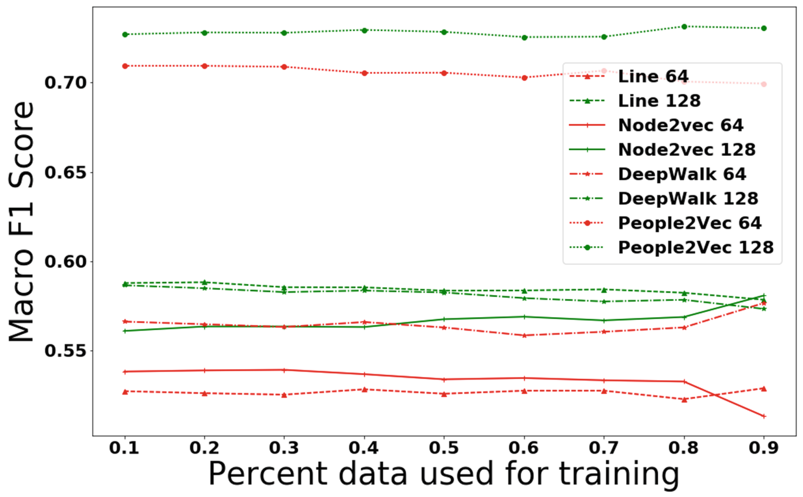

**Fig. 4.** We tried different algorithms to learn the class labels (a proxy of community) of nodes in Flickr graph. In this plot, we compare mean F1 score for different algorithms.

approach, which is a very different from random-walk based approaches compared in this work. However, we observe similar performance gains over the baselines e.g. on BlogCatalog dataset, LANE improved from 0.81 (Deepwalk) to 0.90 (LANE), which is similar to ours, which improved from 0.73 (Deepwalk) to 0.83 (People2vec).

## 5 Conclusions and Future Work

We proposed People2Vec, a model to learn the representation of nodes in complex networks. We used nodes similarity to construct virtual links between nodes. Virtual links enhance the original graph with additional information that uses users' attributes and preferences. People2Vec considers these virtual edges while building random-walk paths, thus, exploits similarity of nodes in addition to real links. Experiments on two real-world datasets reveal that using People2Vec to learn representations substantially improves node classification performance. On the BlogCataog dataset, we observe an improvement of 13% (F1 score) over other state-of-the-art algorithms to learn embeddings. On the Flickr dataset, the performance improved by 25% on F1-score.

People2Vec is straightforward and intuitive, yet learns better node representations. The approach is also very general, hence can easily be extended to other types of data on users. In future, we plan to investigate the usage of sentiment and emotions in social media posts, to learn node representations that consider the stance of users towards topics. We would also like to explore confidence levels on the results for sparse graphs.

**Acknowledgments.** This work was supported in part by the MURI Award No. N000140811186, MURI Award No. N000141712675 and the Center for Computational Analysis of Social and Organization Systems (CASOS). The views and conclusions contained in this document are those of the authors only.

## References

1. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
2. Carley, K.M.: ORA: A Toolkit for Dynamic Network Analysis and Visualization. Springer, New York (2017). https://doi.org/10.1007/978-1-4614-7163-9_309-1
3. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. pp. 160–167. ACM (2008)
4. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 855–864. ACM (2016)
5. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. pp. 731–739. ACM (2017)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013) arXiv preprint arXiv:1301.3781
7. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Aistats, vol. 5, pp. 246–252. Citeseer (2005)
8. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
9. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077. International World Wide Web Conferences Steering Committee (2015)