

Persona2vec: a flexible multi-role representations learning framework for graphs

Jisung Yoon^{1,2}, Kai-Cheng Yang², Woo-Sung Jung^{1,3,4} and Yong-Yeol Ahn^{2,5,6}

¹ Department of Industrial and Management Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea

² Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA

³ Department of Physics, Pohang University of Science and Technology, Pohang, Republic of Korea

⁴ Asia Pacific Center for Theoretical Physics, Pohang, Republic of Korea

⁵ Connection Science, Massachusetts Institute of Technology, Cambridge, MA, USA

⁶ Network Science Institute, Indiana University, Bloomington, IN, USA

ABSTRACT

Graph embedding techniques, which learn low-dimensional representations of a graph, are achieving state-of-the-art performance in many graph mining tasks. Most existing embedding algorithms assign a single vector to each node, implicitly assuming that a single representation is enough to capture all characteristics of the node. However, across many domains, it is common to observe pervasively overlapping community structure, where most nodes belong to multiple communities, playing different roles depending on the contexts. Here, we propose *persona2vec*, a graph embedding framework that efficiently learns multiple representations of nodes based on their structural contexts. Using link prediction-based evaluation, we show that our framework is significantly faster than the existing state-of-the-art model while achieving better performance.

Subjects Artificial Intelligence, Data Science, Network Science and Online Social Networks, Social Computing

Keywords Graph embedding, Overlapping community, Social context, Social network analysis, Link prediction

Submitted 24 December 2020

Accepted 22 February 2021

Published 30 March 2021

Corresponding author

Yong-Yeol Ahn, yyahn@iu.edu

Academic editor

Yilun Shang

Additional Information and
Declarations can be found on
page 16

DOI 10.7717/peerj-cs.439

© Copyright

2021 Yoon et al.

Distributed under

Creative Commons CC-BY 4.0

OPEN ACCESS

INTRODUCTION

Graph embedding maps the nodes in a graph to continuous and dense vectors that capture relations among the nodes (*Perozzi, Al-Rfou & Skiena, 2014; Grover & Leskovec, 2016; Tang et al., 2015*). Resulting node representations allow direct applications of algebraic operations and common algorithms, facilitating graph mining tasks such as node classification (*Sen et al., 2008; Perozzi, Al-Rfou & Skiena, 2014*), community detection (*Fortunato, 2010; Yang et al., 2016*), link prediction (*Grover & Leskovec, 2016*), visualization (*Tang et al., 2015*), and computer vision (*Xie et al., 2020*). Most methods map each node to a single vector, implicitly assuming that a single representation is sufficient to capture the full characteristics of a node.

However, nodes often play multiple roles. For instance, people have multiple roles, or “personas”, across contexts (e.g., professor, employee, and so on) (*Ahn, Bagrow &*

Lehmann, 2010; Coscia et al., 2014; Leskovec et al., 2009; Leskovec, Lang & Mahoney, 2010). Similarly, proteins and other biological elements play multiple functionalities (*Palla et al., 2005; Gavin et al., 2006; Ahn, Bagrow & Lehmann, 2010*). Another example is the polysemy of words when their relations are modeled with graphs; many words possess multiple meanings differentiated by the contexts (*Chen, Liu & Sun, 2014; Li & Jurafsky, 2015; Iacobacci, Pilehvar & Navigli, 2015*). Explicit modeling of such multiplicity and overlapping clusters has been fruitful not only for community detection (*Rosvall et al., 2014; Coscia et al., 2014; Epasto, Lattanzi & Paes Leme, 2017*), but also for improving the quality of embedding (*Li & Jurafsky, 2015; Epasto & Perozzi, 2019; Liu et al., 2019*). Yet, with the scarcity of embedding methods embracing this idea, the full potential of this approach has not been properly explored.

In this paper, we propose *persona2vec*, a scalable framework that builds on the idea of ego-splitting (*Epasto, Lattanzi & Paes Leme, 2017*), the process of identifying local structural contexts of a node via performing local community detection on the node's ego-network. For each detected local community (role), we transform each node into multiple personas if there are multiple local communities to which the node belongs. After the split, the original node is replaced by the new persona nodes that inherit the connection from each local community, producing a new persona graph. Instead of separating a node's persona nodes from each other completely (*Epasto & Perozzi, 2019*), we add directed, weighted edges between personas to capture their origin. In doing so, we allow the direct application of the existing graph embedding methods. In addition, we take an approach of considering persona-based learning as fine-tuning of the base graph embedding, achieving both efficiency and balance between information from the original graph and the persona graph. Compared with the previous approach (*Epasto & Perozzi, 2019*), our framework is conceptually simpler to understand and practically easier to implement. Furthermore, it achieves better performance in the link prediction tasks while being much faster. We also would like to clarify that the primary purpose of persona splitting is not about obtaining multiple representations, each of which may be suited for a specific task; it is about teasing out multiple contexts that a single node may possess. In other words, even with a single task, we argue that learning multiple representations for some nodes is highly beneficial.

In sum, we would like to highlight that our approach (1) drastically lowers the barrier for combining existing algorithms with persona splitting, (2) significantly improves the efficiency of the ego-splitting approach, while (3) consistently excelling the previous state-of-the-art model in the link prediction task. Our implementation of *persona2vec* is publicly available at <https://github.com/jisungyoon/persona2vec>.

RELATED WORK

In addition to graph embedding, our work is closely related to the research of identifying overlapping communities in graphs. Various non-embedding methods such as link clustering (*Ahn, Bagrow & Lehmann, 2010; Evans & Lambiotte, 2009*), clique percolation (*Palla et al., 2005*), and mixed membership stochastic blockmodel (*Airoldi et al., 2008*) have been proposed. Another thread of works focuses on using local graph structure to

extract community information (Coscia et al., 2014; Epasto et al., 2015; Epasto, Lattanzi & Paes Leme, 2017). Specifically, Epasto, Lattanzi & Paes Leme (2017) introduce the persona graph method for detecting overlapping communities in graphs, leveraging ego-network partition. The combination of ego-network analysis and graph embedding methods is still rare. An example is SPLITTER (Epasto & Perozzi, 2019), which we use as the baseline in this paper. Instead of constraining the relations between personas with a regularization term, we propose a simpler and more efficient way of adding persona edges to the graph.

Our work is also related to the word disambiguation problem in a word embedding. Recently, word embedding techniques (Mikolov et al., 2013a, 2013b; Pennington, Socher & Manning, 2014) have been extensively applied to various NLP tasks as the vectorized word representations can effectively capture syntactic and semantic information. Although some words have multiple senses depending on the context, the original word embedding methods only assign one vector to each word. Li & Jurafsky (2015) shows that embedding that is aware of multiple word senses and provides a vector for each specific sense does improve the performance for some NLP tasks. For this issue, some utilize the local context information and clustering for identifying word sense (Reisinger & Mooney, 2010; Wu & Giles, 2015; Neelakantan et al., 2015), some resort to external lexical database for disambiguation (Rothe & Schütze, 2015; Iacobacci, Pilehvar & Navigli, 2015; Camacho-Collados, Pilehvar & Navigli, 2016; Chen, Liu & Sun, 2014; Jauhar, Dyer & Hovy, 2015; Pelevina et al., 2017), while some combine topic modeling methods with embedding (Liu, Qiu & Huang, 2015; Liu et al., 2015; Cheng et al., 2015; Zhang & Zhong, 2016). We adopt the idea of assigning multiple vectors to each node in the graph to represent different roles as well as exploiting local graph structure for the purpose.

PROPOSED METHOD: PERSONA2VEC

persona2vec creates a *persona graph*, where some nodes are split into multiple personas. We then apply a graph embedding algorithm to the persona graph to learn the embeddings of the personas (see Fig. 1). Let us explain the method formally. Let $G = (V, E)$ be a graph with a set of nodes V and a set of edges E . $|V|$ and $|E|$ denote the number of nodes and edges respectively. Let $f : v \rightarrow \mathbb{R}^d$ be the embedding function that maps a node v to a d -dimensional vector space ($d \ll |V|$).

Refined ego-splitting

We adopt and refine the ego-splitting method (Epasto, Lattanzi & Paes Leme, 2017; Epasto & Perozzi, 2019). For each node in the original graph, we first extract its ego-graph, remove the ego, and identify the local clusters. Every cluster in the ego-graph leads to a new persona node in the persona graph (see Figs. 1A and 1C). For example, if we consider each connected component as a local community with a connected component algorithm, node C in the original graph belongs to two non-overlapping clusters $\{A, B\}$ and $\{D, E, F\}$ in its ego-graph. Given these two clusters, in the persona graph, C is split into C_1 and C_2 to represent the two roles in respective clusters. C_1 and C_2 inherit the

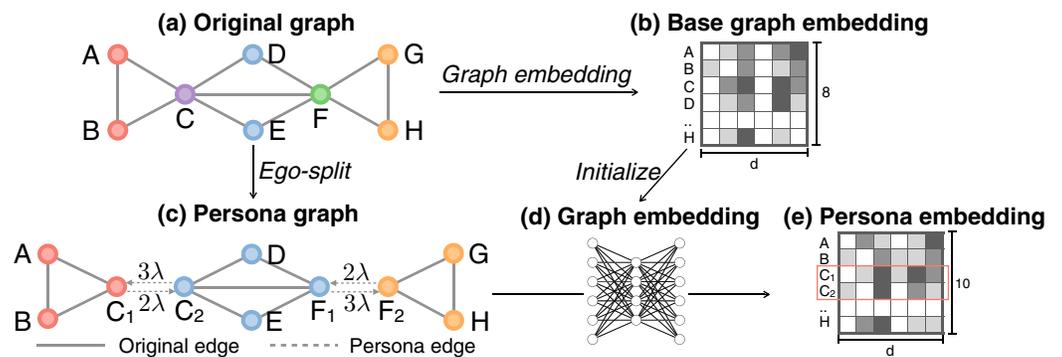


Figure 1 Illustration of persona2vec framework. (A) A graph with an overlapping community structure. (B) Graph embedding of the original graph is obtained first to initialize the persona embeddings. (C) Transform the original graph into a persona graph. Every edge in the original graph is preserved in the persona graph, while new directed persona edges with weight λk_i^o are added between the persona nodes. (D) Graph embedding is applied to the persona graph. (E) The final persona embedding where each persona node has its own vector representation.

Full-size DOI: 10.7717/peerj-cs.439/fig-1

connections of C from both clusters separately (see Fig. 1C). On the other hand, node A only belongs to one ego cluster $\{B, C\}$, so it does not split into multiple personas.

Any graph clustering algorithm can be employed for splitting a node into personas. The simplest algorithm is considering each connected component in the ego-network (sans the ego) as a cluster. This approach is fast and works well on sparse graphs. However, in dense graphs, ego-networks are more likely to form fewer connected components, thus other algorithms such as the Louvain method (Blondel et al., 2008), Infomap (Rosvall & Bergstrom, 2008), and label propagation (Raghavan, Albert & Kumara, 2007) would be more appropriate.

In previous studies, the personas get disconnected without retaining the information about their origin, creating isolated components in the splitting process (Epasto, Lattanzi & Paes Leme, 2017; Epasto & Perozzi, 2019). Because of this disconnectedness, common embedding methods could not be directly applied to the splitted graph. A previous study attempted to address this issue by imposing a regularization term in the cost function to penalize separation of persona nodes originating from the same node (Epasto & Perozzi, 2019).

Here, instead of adopting the regularization strategy, we add weighted *persona edges* between the personas, maintaining the connectedness between them after the splitting (see Fig. 1C). Because the persona graph stays connected, classical graph algorithms and graph embedding methods can now be readily applied without any modification. As we will show later, our strategy achieves both better scalability and better performance.

In the persona graph, we set the weights of the unweighted original edges as 1 and tune the strength of the connections among personas with λ . Persona edges are directed and weighted, with weight λk_i^o , where k_i^o is the out-degree of the persona node after splitting (see Fig. 1C). Assigning weight proportional to k_i^o helps the random walker exploring both the local neighbors and other parts of the graph connected to the other personas regardless of the out-degree k_i^o .

Imagine node u , which is split into n_p personas. Consider one of the personas i with out-degree k_i^o and persona edges with weight w_i . Then the probability p_i that an unbiased random walker at i visits neighbors connected with the original edges at the next step is

$$\frac{k_i^o}{k_i^o + n_p w_i}. \quad (1)$$

If we set constant weight $w_i = \lambda$, then

$$p_i = \frac{k_i^o}{k_i^o + n_p \lambda} = \frac{1}{1 + \frac{n_p}{k_i^o} \lambda}, \quad (2)$$

which depends on k_i^o . A random-walker would not explore its local neighborhood if $n_p \gg k_i^o$, while the opposite happens when $n_p \ll k_i^o$. Instead, assigning the weight proportional to k_i^o , namely $w_i = \lambda k_i^o$, removes such bias because

$$p_i = \frac{k_i^o}{k_i^o + n_p \lambda k_i^o} = \frac{1}{1 + n_p \lambda}, \quad (3)$$

which is independent of k_i^o . Our experiments also show that using the out-degree yields better performance than assigning the identical weight to each persona edge. Our algorithm for refined ego-splitting is described in [Algorithm 1](#). Note that it can be generalized to the directed graphs.

Persona graph embedding

As explained above, any graph embedding algorithm that recognizes edge direction and weight can be readily applied to the persona graph. Although we use Node2vec as the embedding method here, other embedding methods can also be employed. We initialize the persona vectors with the vectors from the original graph before ego-splitting (see [Fig. 1B](#)) to leverage the information from the original graph structure. Persona nodes that belong to the same node in the original graph are thus initialized with the same vector. We then execute the embedding algorithm for a small number of epochs to fine-tune the embedding vectors with the information from the persona graph (see [Fig. 1](#)). Experiments show that usually only one epoch of training is enough.

We find that training the embedding on the persona graphs from scratch fails to yield comparable results. Instead, initializing the embedding with the original graphs, i.e., our present method, consistently improves the performance, suggesting that mixing the structural information from both the original graph and the persona graph is crucial. Our full algorithm is described in [Algorithm 2](#).

Complexity

Space complexity

The persona graph is usually larger than the original graph, but not too large. Node u with degree k_u may be split into at most k_u personas. In the worst case, the number of nodes in the persona graph can reach $O(|E|)$. But, in practice, only a subset of nodes split into

Algorithm 1 Refined ego-splitting for generating the persona graph. Case of the undirected graph.

Input: Original graph $G(V, E)$; weight parameter λ ; non-overlapping local clustering algorithm \mathcal{C}

Output: Persona graph $G_p(V_p, E_p)$; node to personas mapping $V2P$; persona to local cluster mapping $P2C$

```

1: function REFEGOSPLIT( $G(V, E), \lambda, \mathcal{C}$ )
2:   for each  $v_o \in V$  do
3:      $P_{v_o} \leftarrow \mathcal{C}(v_o)$  ▷ find local clusters of  $v_o$ 
4:     for each  $p \in P_{v_o}$  do
5:       Create  $v_p$ , and add to  $G_p, V2P(v_o)$  ▷ create persona nodes for local clusters
6:        $P2C(v_p) \leftarrow p$ 
7:     for each edge  $(v_i, v_j)$  in  $E$  do
8:        $w \leftarrow$  weight of edge
9:       for each persona node  $v_p$  in  $V2P(v_i)$  do
10:        for each persona node  $v'_p$  in  $V2P(v_j)$  do
11:          if  $v_i \in P2C(v'_p)$  and  $v_j \in P2C(v_p)$  then
12:            Add original edges  $(v_p, v'_p, w), (v'_p, v_p, w)$  to  $E_p$ 
13:    $k_o \leftarrow$  out-degree sequence after adding original edges
14:   for each  $v_o \in V$  do
15:     for each pair  $(v_i, v_j)$  in  $V2P(v_o)$  do
16:       Add persona edges  $(v_i, v_j, k_i^o \times \lambda), (v_j, v_i, k_j^o \times \lambda)$  to  $E_p$ 
17:   return  $G_p(V_p, E_p), V2P, P2C$ 

```

personas, and the number of personas rarely reaches the upper bound. If we look at the persona edges, for a node u with degree k_u , at most $O(k_u^2)$ new persona edges may be added. Thus, the whole persona graph has at most $O(|V| \times k_{\max}^2)$ or $O(|V|^3)$ ($\because k_{\max} \leq |V|$) extra persona edges. If graph's degree distribution follows a power-law distribution $P(k) \sim k^{-\gamma}$, then $k_{\max} \sim |V|^{1/\gamma-1}$. Hence, it could be $O(|V|^{\gamma+1/\gamma-1})$ and it is between $O(|V|^2)$ and $O(|V|^3)$ ($\sim 2 \leq \gamma \leq 3$ in general). However, real graph tends to be sparse and $k_i \ll |V|$. If we further assume $k_i < \sqrt{|E|}$ holds for every node, then $\sum_{n=1}^{|V|} k_n^2 \leq \sum_{n=1}^{|V|} k_n \sqrt{|E|} = 2|E|\sqrt{|E|}$. Under this assumption, the upper bound becomes $O(|E|^{3/2})$. Similarly, with the scale-free condition, the upper bound could be $O(|E||V|^{1/\gamma-1})$, which is between $O(|E||V|^{1/2})$ and $O(|E||V|)$. Again, in practice, the number of persona edges is much smaller than this upper bound. To illustrate, we list the number of nodes and persona edges in the persona graph for the graphs we use in this paper in Table 1. All considered, the extra nodes and edges do not bring too much space complexity burden in practice.

Time complexity

Assessing the time complexity requires consideration of the two steps: ego-splitting and embedding. The ego-splitting algorithm has complexity of $O(|E|^{3/2} + \sqrt{|E|}T(|E|))$ in the worst case, where $|E|$ is the number of edges in the original graph and $T(|E|)$ is the

Algorithm 2 `persona2vec`. Our method for generating persona node embeddings.

Input:

$G(V,E)$, Original graph
 d , embedding dimension
 γ_b , number of walks per node for base embedding
 t_b , random walk length for base embedding
 w_b , window size for base embedding
 γ_p , number of walks per node for persona embedding
 t_p , random walk length for persona embedding
 w_p , window size for persona embedding
 α , learning rate
 REFEGOSPLIT, refined ego-splitting method
 $V2P$, node to personas mapping
 EMBEDDINGFUNC, a graph embedding method e.g. DeepWalk, Node2vec

Output:

Φ_{G_p} , a $N_p \times d$ matrix with d -dimensional vector representations for all N_p persona nodes

```

1: function PERSONA2VEC( $G, d, \gamma_b, t_b, w_b, g_p, t_p, w_p, \text{REFEGOSPLIT}, \text{EMBEDDINGFUNC}, \alpha$ )
2:    $G_p, V2P \leftarrow \text{REFEGOSPLIT}(G)$ 
3:    $\Phi_G \leftarrow \text{EMBEDDINGFUNC}(G, d, \gamma_b, t_b, w_b, \alpha)$ 
4:   for each  $v_o \in V$  do
5:     for each persona node  $v_p$  in  $V2P(v_o)$  do
6:        $\Phi_{G_p}(v_p) = \Phi_G(v_o)$ 
7:    $\Phi_{G_p} \leftarrow \text{EMBEDDINGFUNC}(G_p, \gamma_p, t_p, w_p, \alpha, \Phi_{G_p})$ 
8:   return  $\Phi_{G_p}$ 

```

Table 1 Descriptive statistics of the graphs used in the evaluation. We report the number of nodes $|V|$, number of edges $|E|$, number of nodes in the persona graph $|V_p|$, the ratio of $|V_p|$ over $|V|$, number of persona edges $|E_p|$ added in ego-splitting, and the ratio of $|E_p|$ over $|E|^{3/2}$ which is the upper bound of space complexity.

Dataset	Type	$ V $	$ E $	$ V_p $	$ V_p / V $	$ E_p $	$ E_p /E^{3/2}$
PPI	Undirected	3,863	38,705	16,734	4.34	132,932	0.0175
ca-HepTh	Undirected	9,877	25,998	16,071	1.86	33,524	0.0800
ca-AstroPh	Undirected	17,903	197,301	25,706	1.44	29,102	0.0003
Wiki-vote	Directed	7,066	103,633	21,467	3.04	118,020	0.0035
Soc-epinions	Directed	75,877	508,836	220,332	2.90	3,550,594	0.0098

complexity of detecting the ego clusters in the graph with $|E|$ edges (Epasto, Lattanzi & Paes Leme, 2017). The embedding on the persona graph, which dominates the whole embedding procedure, has complexity $O(|V_p| \gamma \text{ twd}(1 + \log(|V_p|)))$ which is time complexity of Node2vec, where $|V_p|$ is the number of nodes, γ is the number of random walkers, d is the embedding dimension, and w is the window size (Chen et al., 2018).

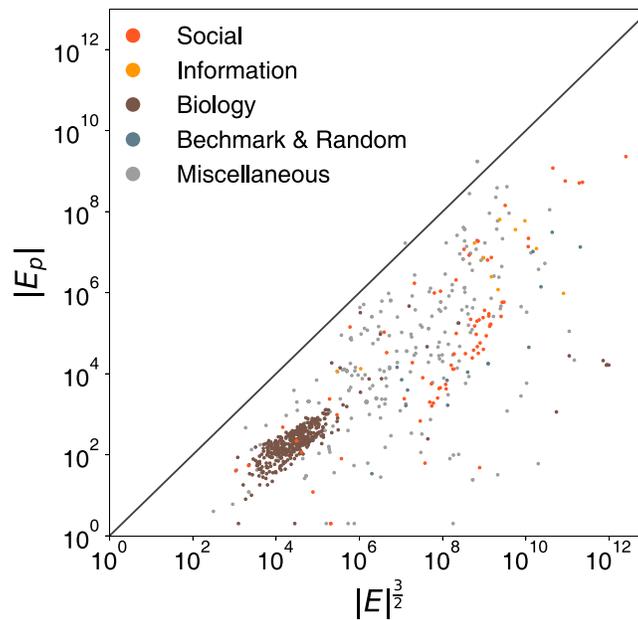


Figure 2 Comparison of the the number of persona edges $|E_p|$ to the practical upper bound $|E|^{3/2}$.
Full-size  DOI: 10.7717/peerj-cs.439/fig-2

The final complexity is $O(|E|^{3/2} + \sqrt{|E|}T(|E|)) + O(|V|\gamma twd(1 + \log(|V|)))$.

Removing the constant factors and assuming close-to-linear local community detection algorithm, the whole process has time complexity of $O(|E|^{3/2})$ with space complexity of $O(|E|^{3/2})$ if $k_i < \sqrt{|E|}$ holds. Complexity can be increased depending on the clustering algorithms on the ego-network.

To test the validity of our assumptions, we sample 1,000 graphs from a public network repository (Rossi & Ahmed, 2015). We apply the refined ego-splitting with connected component algorithms on these samples and report the actual number of persona edges $|E_p|$ with respect to the practical upper bound $|E|^{3/2}$ in Fig. 2, which shows that the actual number of persona edges $|E_p|$ rarely exceeds the tighter upper bound that we propose and is usually orders of the magnitude smaller.

Optimization

Any kind of graph embedding method can be considered, for simplicity, we choose the classical random-walker based embedding method (e.g., Node2Vec, DeepWalk). In the model (Perozzi, Al-Rfou & Skiena, 2014), the probability of a node v_i co-occurring with a node v_j is estimated by

$$p(v_i|v_j) = \frac{\exp(\Phi'_{v_i} \cdot \Phi_{v_j})}{\sum_{k=1}^V \exp(\Phi'_{v_k} \cdot \Phi_{v_j})}, \quad (4)$$

where Φ_{v_i} and Φ'_{v_i} are the ‘input’ and ‘output’ embedding of node i . We use input embedding Φ which is known to be more useful and more widely used. Denominator of Eq. (4) is computationally expensive (Yang et al., 2016; Cao, Lu & Xu, 2016) and there are

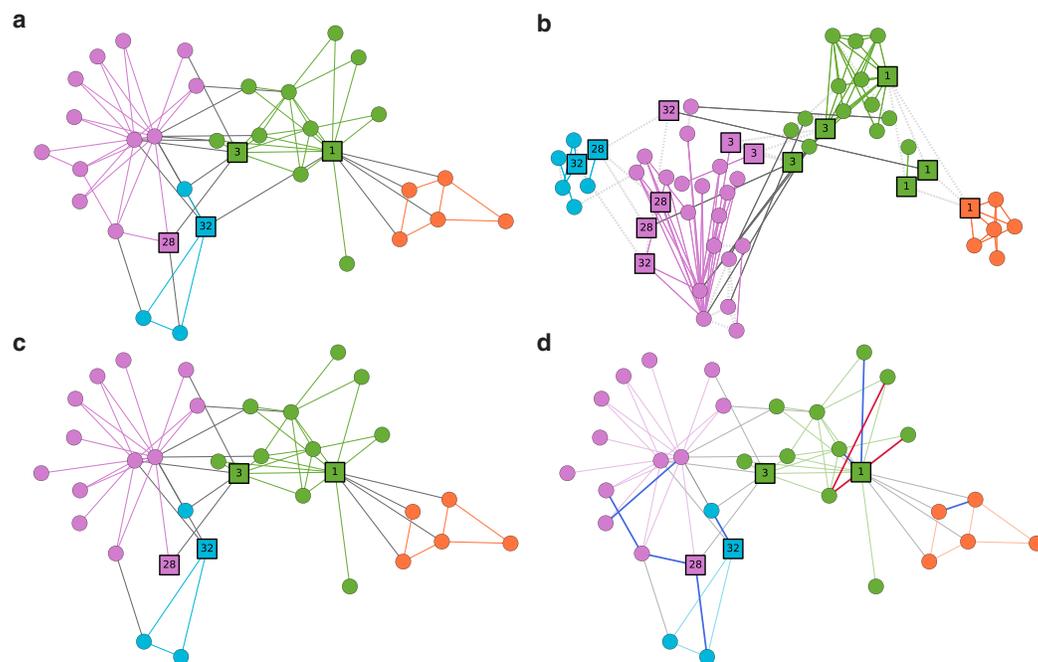


Figure 3 Case Study: Zachary's Karate club network. (A) The Zachary's Karate club network with the force-atlas layout (Zachary, 1977). Nodes are colored by communities detected by the Louvain modularity method (Blondel et al., 2008). (B) The persona graph. Nodes are colored by k-means clusters (MacQueen, 1967) from the embedding vectors. Coordinates of the persona nodes come from the 2-D projection of the embedding with t-SNE (Maaten & Hinton, 2008). Light gray lines represent the persona edges. (C) The network with 20% of edges (16 edges) removed for the link prediction experiment. (D) The network with ten predictions with the highest scores from the link prediction experiment. Blue links represent correctly predicted edges and red edges indicate incorrectly predicted ones.

Full-size DOI: 10.7717/peerj-cs.439/fig-3

two common approximations: hierarchical softmax (Morin & Bengio, 2005) and negative sampling (Mikolov et al., 2013b). We adopt negative sampling not only because it is simpler and popular but also because it shows better performance.

CASE STUDY

Before diving into systematic evaluations, we provide two illustrative examples: Zachary's Karate club network and a word association network.

Case study: Zachary's Karate club network

We use Zachary's Karate club network (Zachary, 1977), a well-known example for the community detection. Nodes represent members of the Karate club, and edges represent ties among the members (see Fig. 3A). Although it is often considered to have two large disjoint communities, smaller overlapping communities can also be seen, highlighted by nodes such as 1, 3, 28, and 32. In Fig. 3B, we present the persona graph of the network. `persona2vec` successfully recognizes these bridge nodes and places their personas in reasonable locations. Take node 1 for example. It splits into four persona nodes, which then end up in two different communities. The orange and green communities are clearly

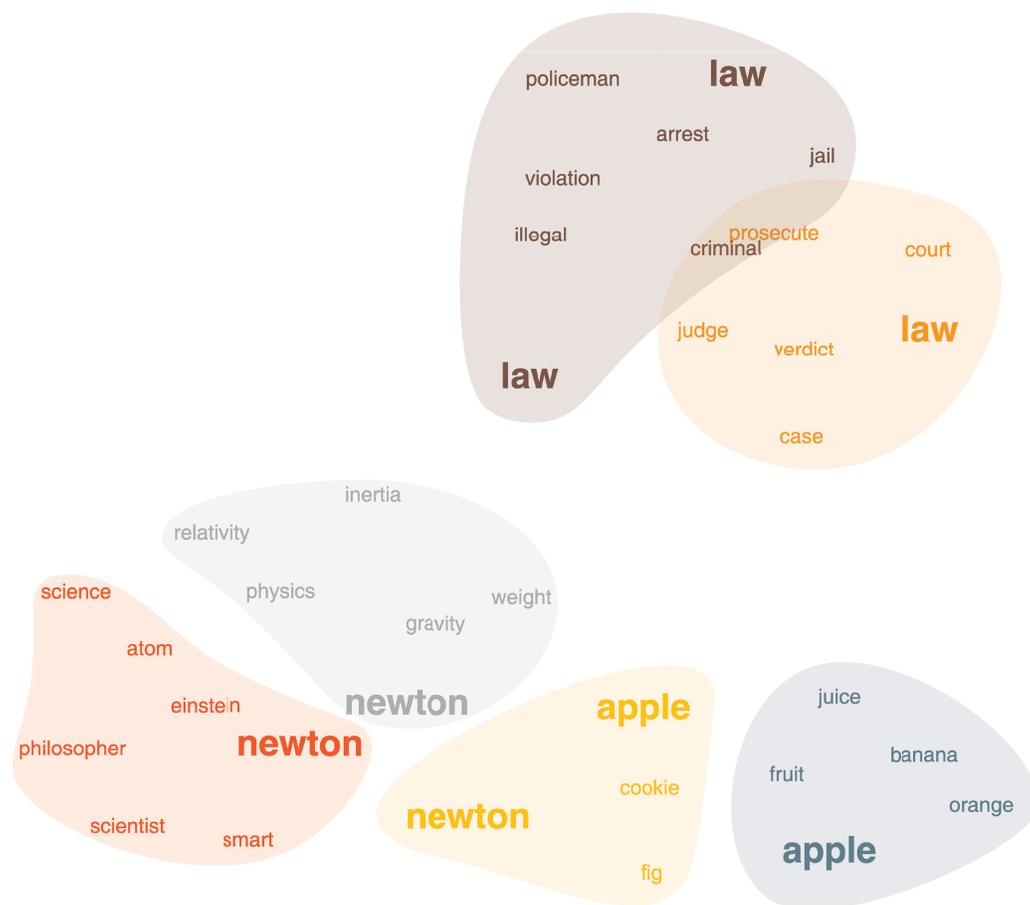


Figure 4 The word association network, clusters around the word “Newton”. Coordinates of the words come from the 2-D projection of the embedding vectors with UMAP (McInnes, Healy & Melville, 2018). Word colors correspond to the clusters obtained by k-means clustering (MacQueen, 1967) on the embedding vectors. Full-size  DOI: 10.7717/peerj-cs.439/fig-4

separated as a result. We also show the ten predictions with the highest scores from the link prediction experiment in Fig. 3D and ensure that the model predicts missing edges well.

Case study: word association network

Word association network captures how people associate words together (free association task). The dataset was originally assembled from nearly 750,000 responses from over 6,000 peoples. Participants were shown 5,019 words and asked to write down the first word that sprang in mind and all the word pairs were collected with their frequency as the weights. This dataset forms a weighted, directed graph of words that captures their multiple senses. Although it is, in principle, possible to run our method on the original graph, for simplicity, we convert it into an undirected, unweighted graph by neglecting weight and direction (Ahn, Bagrow & Lehmann, 2010). In Fig. 4, we show the persona2vec clusters around the word “Newton”. We use the Louvain method (Blondel et al., 2008) to split the personas of each word. persona2vec successfully captures

multiple contexts of the word “Newton”. For instance, the red persona is associated with “scientists” and “philosopher”, the gray one is linked to the physics, and the yellow one is associated with “apple” (note that there is a cookie called “(Fig) Newton” in the U.S.). Furthermore, `persona2vec` also captures different nuances of the word “law” that are related to the crime (brown cluster) and the legal concepts (orange cluster).

NUMERICAL EXPERIMENT

Link prediction task

To systematically evaluate the performance and scalability of the `persona2vec` framework, we perform a link prediction task using real-world graphs (Grover & Leskovec, 2016; Abu-El-Haija, Perozzi & Al-Rfou, 2017). Link prediction aims to predict missing edges in a graph with partial information, which is useful for many tasks such as suggesting new friends on social networks or recommending products. It has been employed as a primary task to evaluate the performance of unsupervised graph embedding methods (Abu-El-Haija, Perozzi & Al-Rfou, 2017; Zhang et al., 2018).

We follow the task setup from the literature (Grover & Leskovec, 2016; Abu-El-Haija, Perozzi & Al-Rfou, 2017). First, the edge set of an input graph is divided equally and randomly into E_{train} and E_{test} . We then refine E_{test} using a rejection sampling method based on the criterion that, even when we remove all edges in E_{test} , the graph should be connected as a single component. E_{train} is used to train the models, and edges in E_{test} are used as positive examples for the prediction task. Second, a negative edge set $E_{(-)}$ of non-existent random edges with the same size of E_{test} are generated to provide negative examples for testing. The performance of a model is measured by its ability to correctly distinguish E_{test} and $E_{(-)}$ after being trained on E_{train} . We then report ROC-AUC.

Datasets

To facilitate the comparison with the state-of-the-art baseline, we use five graph datasets that are publicly available and previously used (Epasto & Perozzi, 2019; Leskovec & Krevl, 2014). We summarize them as follows.

PPI is a protein-protein interaction graph of *Homo sapiens* (Stark et al., 2006). Nodes represent proteins and edges represent physical interactions between the proteins. *ca-HepTh* is a scientific collaboration graph. It represents the co-authorship among researchers from the Theoretical High Energy Physics field, derived from papers on arXiv. *ca-AstropPh* is also scientific collaboration graph, but from Astrophysics. *wiki-vote* is a voting network, each node is a Wikipedia user and a directed edge from node i to node j represents that user i voted for user j to become an administrator. *soc-epinions* is a voting graph from a general consumer review site [Epinions.com](https://www.epinions.com), each node is a member, and a directed edge from node i to node j means that member i trusted member j .

We use the largest connected component of the undirected graphs and the largest weakly connected component of the directed ones. The statistics of all the graphs are reported in Table 1.

Methods

The state-of-the-art method in this link prediction task is SPLITTER (Epasto & Perozzi, 2019), which also models multiple roles. As reported in the paper, it outperforms various existing algorithms ranging across non-embedding methods like Jaccard Coefficient, Common Neighbors, and Adamic-Adar as well as embedding methods like Laplacian EigenMaps (Belkin & Niyogi, 2002), Node2vec (Grover & Leskovec, 2016), DNGR (Cao, Lu & Xu, 2016), Asymmetric (Abu-El-Haija, Perozzi & Al-Rfou, 2017) and M-NMF (Wang et al., 2017).

Given the state-of-the-art performance of SPLITTER, for simplicity, we compare our framework with SPLITTER using the identical task setup and datasets. In addition, because our method can be considered as an augmentation of a single-role embedding method, and because we use Node2vec as the base embedding method, we also employ Node2vec. We run the link prediction task using the original authors' implementation of Node2vec and SPLITTER. The parameters are also kept consistent with the original paper.

persona2vec and SPLITTER have multiple representations on each node, which leads to non-unique similarity estimations between two nodes. Hence, we define the similarity score of a pair of nodes on persona2vec as the maximum dot-product of embedding vectors between any pair of their personas. We found that, among experiments with three aggregation functions *min*, *max*, *mean*, the highest performance is achieved with *max*, the same with SPLITTER (Epasto & Perozzi, 2019). For SPLITTER, we use maximum cosine similarity, following the author's note in their implementation.

Node2vec (baseline method)

For Node2vec, we set random walk length $t = 40$, the number of walks per node $\gamma = 10$, random walk parameters $p = q = 1$, the window size $w = 5$, and the initial learning rate $\alpha = 0.025$. In the original paper, they learn an additional logistic regression classifier over the Hadamard product of the embedding of two nodes for the link prediction. In general, the logistic regression classifier improves the performance. Here, we report results on Node2vec with both dot products and the logistic regression classifier.

SPLITTER (baseline method)

For SPLITTER, we use the same parameters in the paper (Epasto & Perozzi, 2019) and aforementioned Node2vec baseline. We use Node2vec with random walk parameters $p = q = 1$.

persona2vec (our proposed method)

We set the hyper-parameters of the original graph embedding with $t_b = 40$, $\gamma_b = 10$, $w_b = 5$. For the persona embedding, we set $t_p = 80$, $\gamma_p = 5$, $w_p = 2$ to better capture the micro-structure of the persona graph. The size of the total trajectories is determined by the random walk length t^* times the number of walks per node γ^* , so we keep $t^*\gamma^*$ constant to roughly preserve the amount of information used in the embedding. For both embedding stages, we use $\alpha = 0.025$, and Node2vec with the random walk parameters ($p = q = 1$) as the graph embedding function.

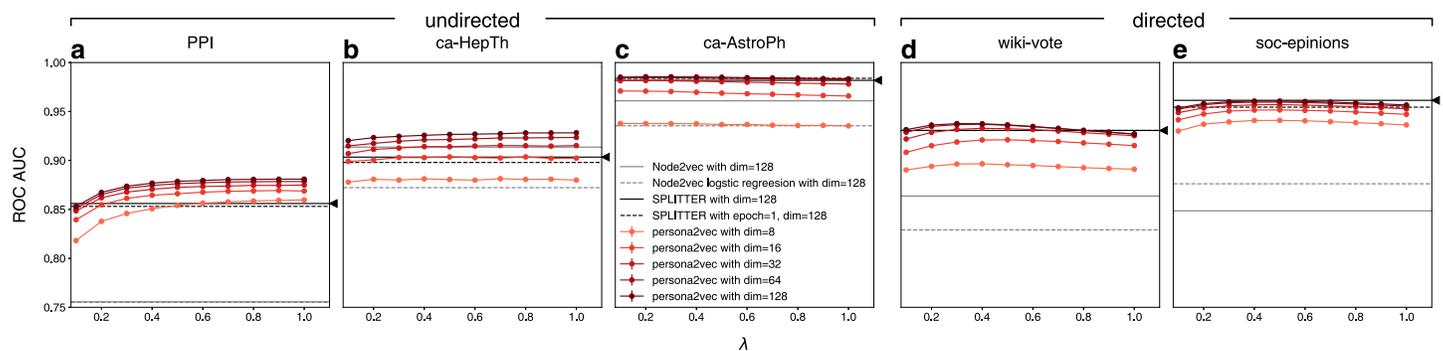


Figure 5 Performance of *persona2vec* in the link prediction task. We report the link prediction performance for each graphs for (A) PPI, (B) ca-HepTh, (C) ca-AstroPh, (D) wiki-vote, and (E) SOC-epinions. Number of epochs n is set to 1 in all experiments for *persona2vec*. Darker colors represent higher embedding dimensions. The confidence intervals are all within the range of the markers. Given the same number of dimensions, *persona2vec* is always on par with or better than SPLITTER. [Full-size !\[\]\(f5a508cc6d05e5d06b117ced927b1acd_img.jpg\) DOI: 10.7717/peerj-cs.439/fig-5](https://doi.org/10.7717/peerj-cs.439/fig-5)

Table 2 Performance of *persona2vec* with $\lambda = 0.5$. All methods use $d = 128$. Node2vec* refers to Node2vec with the logistic regression classifier, SPLITTER* refers to SPLITTER with one epoch, and *persona2vec** refers *persona2vec* with $\lambda = 0.5$, our suggested default. Performance gain is performance difference between *persona2vec** and Node2vec. We omit the standard error which is smaller than 10^{-3} . Bold numbers represent the best performance.

Method	PPI	ca-HepTH	ca_AstroPh	wiki-vote	soc-epinions
Node2vec	0.585	0.825	0.901	0.694	0.547 ± 0.007
Node2vec*	0.662 ± 0.001	0.848	0.914	0.705 ± 0.001	0.767 ± 0.002
SPLITTER	0.856	0.903	0.982	0.931	0.961 ± 0.001
SPLITTER*	0.853	0.898	0.984	0.931	0.954 ± 0.001
<i>persona2vec*</i>	0.879	0.927	0.985	0.936	0.961
Performance_gain	0.294	0.102	0.084	0.242	0.414 ± 0.007

Experiment results

Figure 5 shows the link prediction performance of *persona2vec* in comparison with the baselines. Overall, *persona2vec* yields superior performance across graphs and across a range of hyperparameter choices. We show that augmenting Node2vec by considering personas significantly improves the link prediction performance, evinced by the significant performance gain (see Table 2).

As expected, larger dimensions lead to better performance, although *persona2vec* achieves reasonable results even with tiny embedding dimensions like 8 or 16. We also show how the performance of *persona2vec* varies with λ . For undirected graphs, larger λ is beneficial but the trend saturates quickly. For directed graphs, however, optimal performance is achieved with smaller values of λ . In practice, we suggest starting with $\lambda = 0.5$ as a default parameter because the overall variation brought by λ is not substantial and even when the performance increases with λ , near-optimal performance can be achieved at $\lambda = 0.5$.

When compared with the SPLITTER baseline, *persona2vec* shows on par or better performances given the same embedding dimensions across a wide range of λ . We also

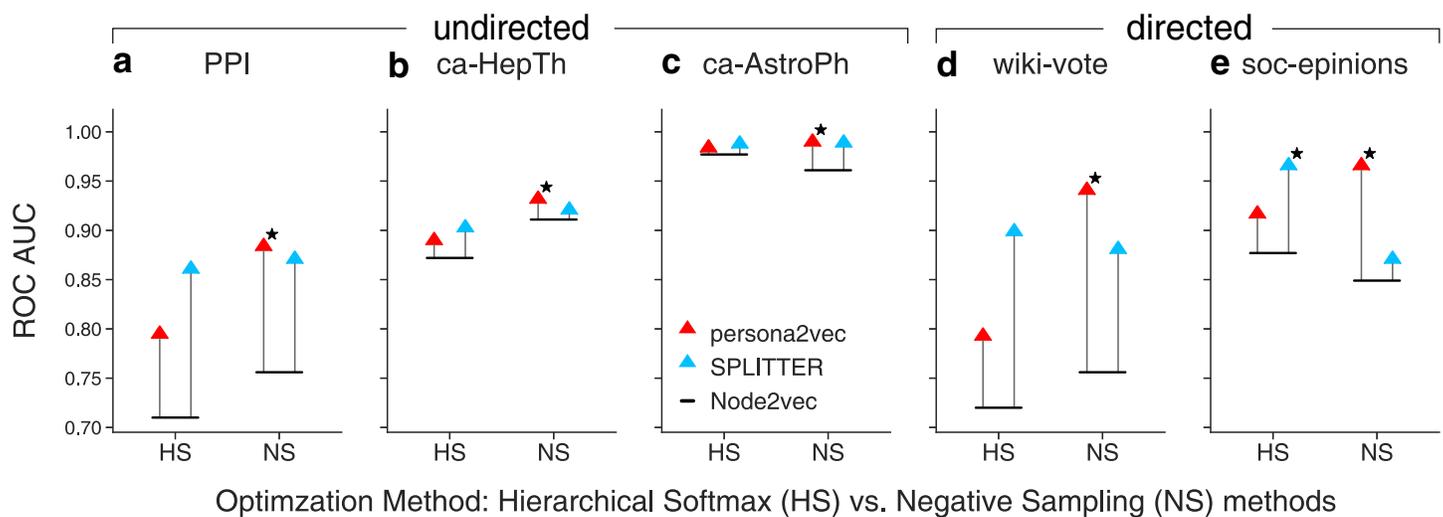


Figure 6 Comparison of link prediction performance between `persona2vec` and `SPLITTER` with different approximations. We report the link prediction performance across optimization methods for each graphs for (A) PPI, (B) `ca-HepTh`, (C) `ca-AstroPh`, (D) `wiki-vote`, and (E) `SOC-epinions`. HS refers to the hierarchical softmax and NS refers to the negative sampling. The star marker indicates the best link prediction performance. [Full-size !\[\]\(365da8a2cbf7f1d19047f927ee6f7f2e_img.jpg\) DOI: 10.7717/peerj-cs.439/fig-6](https://doi.org/10.7717/peerj-cs.439/fig-6)

report the performance summary for `persona2vec` with $\lambda = 0.5$ (our suggested default) compared with the best baselines in [Table 2](#), which shows that `persona2vec` outperforms the baseline consistently. Also, we report the performance gain of `persona2vec` from `Node2vec`, because we used `Node2vec` as the base embedding method and `persona2vec` can be considered as an augmentation or fine-tuning of the base `Node2vec` vectors with local structural information. As shown, the persona-based fine-tuning significantly improves the performance.

We also study the effect of different optimization methods, i.e., hierarchical softmax and negative sampling in [Fig. 6](#). We also find that cosine similarity consistently yields a better result with hierarchical softmax while dot product yields a better result with negative sampling regardless of the embedding methods. So, we use cosine similarity for hierarchical softmax and use dot product for negative sampling. Our experiments suggest that `persona2vec` tends to perform better with negative sampling while `SPLITTER` works better with hierarchical softmax. Nevertheless, `persona2vec` yields the best performance consistently.

In addition to the performance of the link prediction task, we also report the execution time of `persona2vec` and `SPLITTER` to compare their scalability in practice (see [Fig. 7](#)). Note that the reported execution time is from the link-prediction task, with half of the edges removed from the original graph. `SPLITTER` runs the embedding procedures for 10 epochs by default in the original implementation, whereas `persona2vec` only runs for one epoch. For a fair comparison, we also report the results of `SPLITTER` with one epoch of training. When being limited to only one epoch, `SPLITTER`'s performance slightly suffers on three graphs while it goes up or stays stable for the other two.

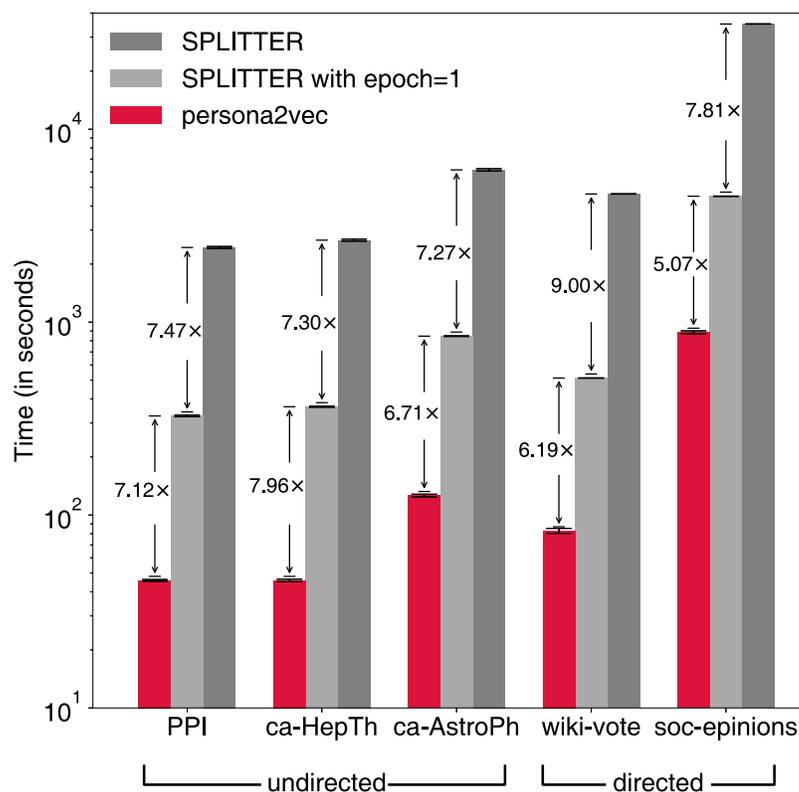


Figure 7 Comparison of elapsed time between persona2vec and SPLITTER. Speed gains by persona2vec are shown. [Full-size !\[\]\(ebfe6d37ad86655679811e032f633da4_img.jpg\) DOI: 10.7717/peerj-cs.439/fig-7](https://doi.org/10.7717/peerj-cs.439/fig-7)

Nevertheless, persona2vec is more efficient—39 to 58 times faster than SPLITTER with 10 epochs and five to eight times faster than SPLITTER with one epoch. The most likely reason behind the drastic difference is the overhead from the extra regularization term in the cost function of SPLITTER, which persona2vec does not need. In sum, persona2vec outperforms the previous state-of-the-art method both in terms of scalability and link prediction performance.

CONCLUSIONS

We present persona2vec, a framework for learning multiple node representations considering the node's local structural contexts. persona2vec first performs ego-splitting, where nodes with multiple non-overlapping local communities in their ego-networks are replaced with corresponding persona nodes. The persona nodes inherit the edges from the original graph and remain connected by newly added persona edges, forming the persona graph. Initialized by the embedding of the original graph, the embedding algorithm applied to the persona graph yields the final representations. Instead of assigning only one vector to every node with multiple roles, persona2vec learns a vector for each of the personas. With extensive link prediction evaluations, we demonstrate that persona2vec achieves the state-of-the-art performance while being able to scale better. Moreover, our method is easy to comprehend and implement without losing any flexibility for incorporating other embedding algorithms, presenting great potential for

applications. The possible combination with various algorithms provides vast space for further exploration. For instance, in a multi-layer network, inter-layer coupling connection can be interpreted as natural persona edges, and `persona2vec` may be applied to tackle the multi-layer link prediction problem.

The graph (relational) structure is ubiquitous across many complex systems, including physical, social, economic, biological, neural, and information systems, and thus fundamental graph algorithms have far-reaching impacts across many areas of sciences. Graph embedding, in particular, removes the barrier of translating methods to the special graph data structure, opening up a powerful way to transfer existing algorithms to the graphs and relational data. Furthermore, given that it is natural to assume that overlapping clusters and their heterogeneous functionality exist in most real networks, multi-role embedding methods may find numerous applications in physical, biological, and social sciences.

ACKNOWLEDGEMENTS

For their comments, we thank Sadamori Kojaku, Alessandro Flammini, Filippo Menczer, Xiaoran Yan, Filipi Nascimento Silva, and Minwoo Ahn.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work is supported by the Air Force Office of Scientific Research under award number FA9550-191-0391. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Air Force Office of Scientific Research: FA9550-191-0391.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- Jisung Yoon conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Kai-Cheng Yang conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.
- Woo-Sung Jung conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.
- Yong-Yeol Ahn conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The preprocessed version of PPI is available at Stanford University: <https://snap.stanford.edu/node2vec/>.

Other graphs (ca-AstroPh, ca-HepTh, wiki-Vote, soc-Epinions1) are also available at the SNAP library:

<http://snap.stanford.edu/data/index.html>.

Code is available at GitHub:

<https://github.com/jisungyoon/persona2vec>.

REFERENCES

- Abu-El-Haija S, Perozzi B, Al-Rfou R. 2017.** Learning edge representations via low-rank asymmetric projections. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*. New York: ACM, 1787–1796.
- Ahn Y-Y, Bagrow JP, Lehmann S. 2010.** Link communities reveal multiscale complexity in networks. *Nature* **466**(7307):761–764 DOI [10.1038/nature09182](https://doi.org/10.1038/nature09182).
- Airoldi EM, Blei DM, Fienberg SE, Xing EP. 2008.** Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* **9**(September):1981–2014.
- Belkin M, Niyogi P. 2002.** Laplacian eigenmaps and spectral techniques for embedding and clustering. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge: MIT Press, 585–591.
- Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. 2008.** Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* **2008**(10):P10008 DOI [10.1088/1742-5468/2008/10/P10008](https://doi.org/10.1088/1742-5468/2008/10/P10008).
- Camacho-Collados J, Pilehvar MT, Navigli R. 2016.** Nasari: integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence* **240**(1):36–64 DOI [10.1016/j.artint.2016.07.005](https://doi.org/10.1016/j.artint.2016.07.005).
- Cao S, Lu W, Xu Q. 2016.** Deep neural networks for learning graph representations. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 1145–1152.
- Chen H, Perozzi B, Hu Y, Skiena S. 2018.** Harp: hierarchical representation learning for networks. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans: AAAI Press.
- Chen X, Liu Z, Sun M. 2014.** A unified model for word sense representation and disambiguation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1025–1035.
- Cheng J, Wang Z, Wen J-R, Yan J, Chen Z. 2015.** Contextual text understanding in distributional semantic space. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 133–142.
- Coscia M, Rossetti G, Giannotti F, Pedreschi D. 2014.** Uncovering hierarchical and overlapping communities with a local-first approach. *ACM Transactions on Knowledge Discovery from Data* **9**(1):6:1–6:27 DOI [10.1145/2629511](https://doi.org/10.1145/2629511).
- Epasto A, Lattanzi S, Mirrokni V, Sebe IO, Taei A, Verma S. 2015.** Ego-net community mining applied to friend suggestion. *Proceedings of the VLDB Endowment* **9**(4):324–335 DOI [10.14778/2856318.2856327](https://doi.org/10.14778/2856318.2856327).

- Epasto A, Lattanzi S, Paes Leme R. 2017.** Ego-splitting framework: From non-overlapping to overlapping clusters. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17*. New York: ACM, 145–154.
- Epasto A, Perozzi B. 2019.** Is a single embedding enough? Learning node representations that capture multiple social contexts. In: *The World Wide Web Conference*. New York: ACM, 394–404.
- Evans TS, Lambiotte R. 2009.** Line graphs, link partitions, and overlapping communities. *Physical Review E* **80(1)**:016105 DOI [10.1103/PhysRevE.80.016105](https://doi.org/10.1103/PhysRevE.80.016105).
- Fortunato S. 2010.** Community detection in graphs. *Physics Reports* **486(3–5)**:75–174 DOI [10.1016/j.physrep.2009.11.002](https://doi.org/10.1016/j.physrep.2009.11.002).
- Gavin A-C, Aloy P, Grandi P, Krause R, Boesche M, Marzioch M, Rau C, Jensen LJ, Bastuck S, Dümpelfeld B, Edelmann A, Heurtier M-A, Hoffman V, Hoefert C, Klein K, Hudak M, Michon A-M, Schelder M, Schirle M, Remor M, Rudi T, Hooper S, Bauer A, Bouwmeester T, Casari G, Drewes G, Neubauer G, Rick JM, Kuster B, Bork P, Russell RB, Superti-Furga G. 2006.** Proteome survey reveals modularity of the yeast cell machinery. *Nature* **440(7084)**:631–636 DOI [10.1038/nature04532](https://doi.org/10.1038/nature04532).
- Grover A, Leskovec J. 2016.** Node2vec: scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*. New York: ACM, 855–864.
- Iacobacci I, Pilehvar MT, Navigli R. 2015.** Sensembed: learning sense embeddings for word and relational similarity. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Stroudsburg: Association for Computational Linguistics, 95–105.
- Jauhar SK, Dyer C, Hovy E. 2015.** Ontologically grounded multi-sense representation learning for semantic vector space models. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg: Association for Computational Linguistics, 683–693.
- Leskovec J, Krevl A. 2014.** SNAP datasets: Stanford large network dataset collection. Available at <http://snap.stanford.edu/data>.
- Leskovec J, Lang KJ, Dasgupta A, Mahoney MW. 2009.** Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* **6(1)**:29–123 DOI [10.1080/15427951.2009.10129177](https://doi.org/10.1080/15427951.2009.10129177).
- Leskovec J, Lang KJ, Mahoney M. 2010.** Empirical comparison of algorithms for network community detection. In: *Proceedings of the 19th International Conference on World Wide Web, WWW '10*. New York: ACM, 631–640.
- Li J, Jurafsky D. 2015.** Do multi-sense embeddings improve natural language understanding? *arXiv*. Available at <http://arxiv.org/abs/1506.01070>.
- Liu N, Tan Q, Li Y, Yang H, Zhou J, Hu X. 2019.** Is a single vector enough? exploring node polysemy for network embedding. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York: Association for Computing Machinery, 932–940.
- Liu P, Qiu X, Huang X. 2015.** Learning context-sensitive word embeddings with neural tensor skip-gram model. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. Palo Alto: AAAI Press.
- Liu Y, Liu Z, Chua T-S, Sun M. 2015.** Topical word embeddings. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence, 25–30 January 2015, Austin Texas, USA*.

- Maaten Lvd, Hinton G. 2008.** Visualizing data using t-sne. *Journal of Machine Learning Research* **9(November)**:2579–2605.
- MacQueen J. 1967.** Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Oakland, CA, USA. Vol. 1. 281–297.
- McInnes L, Healy J, Melville J. 2018.** Umap: uniform manifold approximation and projection for dimension reduction. Available at <https://arxiv.org/abs/1802.03426>.
- Mikolov T, Chen K, Corrado G, Dean J. 2013a.** Efficient estimation of word representations in vector space. Available at <https://arxiv.org/abs/1301.3781>.
- Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. 2013b.** Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*. 3111–3119.
- Morin F, Bengio Y. 2005.** Hierarchical probabilistic neural network language model. In: *Artificial Intelligence and Statistics (AISTATS'05)*. Vol. 5. 246–252.
- Neelakantan A, Shankar J, Passos A, McCallum A. 2015.** Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv*. Available at <https://arxiv.org/abs/1504.06654>.
- Palla G, Derényi I, Farkas I, Vicsek T. 2005.** Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435(7043)**:814–818 DOI [10.1038/nature03607](https://doi.org/10.1038/nature03607).
- Pelevina M, Arefyev N, Biemann C, Panchenko A. 2017.** Making sense of word embeddings. Available at <https://arxiv.org/abs/1708.03390>.
- Pennington J, Socher R, Manning C. 2014.** Glove: global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.
- Perozzi B, Al-Rfou R, Skiena S. 2014.** Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*. New York: ACM, 701–710.
- Raghavan UN, Albert R, Kumara S. 2007.** Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* **76(3)**:036106 DOI [10.1103/PhysRevE.76.036106](https://doi.org/10.1103/PhysRevE.76.036106).
- Reisinger J, Mooney RJ. 2010.** Multi-prototype vector-space models of word meaning. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 109–117.
- Rossi RA, Ahmed NK. 2015.** The network data repository with interactive graph analytics and visualization. In: *AAAI*.
- Rosvall M, Bergstrom CT. 2008.** Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America* **105(4)**:1118–1123 DOI [10.1073/pnas.0706851105](https://doi.org/10.1073/pnas.0706851105).
- Rosvall M, Esquivel AV, Lancichinetti A, West JD, Lambiotte R. 2014.** Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications* **5(1)**:4630 DOI [10.1038/ncomms5630](https://doi.org/10.1038/ncomms5630).
- Rothe S, Schütze H. 2015.** Autoextend: extending word embeddings to embeddings for synsets and lexemes. *arXiv*. Available at <https://arxiv.org/abs/1507.01127>.
- Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. 2008.** Collective classification in network data. *AI Magazine* **29(3)**:93 DOI [10.1609/aimag.v29i3.2157](https://doi.org/10.1609/aimag.v29i3.2157).

- Stark C, Breitkreutz B-J, Reguly T, Boucher L, Breitkreutz A, Tyers M. 2006.** Biogrid: a general repository for interaction datasets. *Nucleic Acids Research* **34**:D535–D539.
- Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. 2015.** Line: large-scale information network embedding. In: *Proceedings of the 24th International Conference on World Wide Web, WWW '15, Republic and Canton of Geneva, Switzerland, International World Wide Web Conferences Steering Committee*. 1067–1077.
- Wang X, Cui P, Wang J, Pei J, Zhu W, Yang S. 2017.** Community preserving network embedding. In: *Thirty-First AAAI Conference on Artificial Intelligence*.
- Wu Z, Giles CL. 2015.** Sense-aware semantic analysis: a multi-prototype word representation model using wikipedia. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*.
- Xie G-S, Liu L, Zhu F, Zhao F, Zhang Z, Yao Y, Qin J, Shao L. 2020.** Region graph embedding network for zero-shot learning. In: *European Conference on Computer Vision*. Cham: Springer, 562–580.
- Yang L, Cao X, He D, Wang C, Wang X, Zhang W. 2016.** Modularity based community detection with deep learning. In: *IJCAI 16*. New York: AAAI Press, 2252–2258.
- Zachary WW. 1977.** An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**(4):452–473 DOI [10.1086/jar.33.4.3629752](https://doi.org/10.1086/jar.33.4.3629752).
- Zhang H, Zhong G. 2016.** Improving short text classification by learning vector representations of both words and hidden topics. *Knowledge-Based Systems* **102**(1):76–86 DOI [10.1016/j.knosys.2016.03.027](https://doi.org/10.1016/j.knosys.2016.03.027).
- Zhang Z, Cui P, Wang X, Pei J, Yao X, Zhu W. 2018.** Arbitrary-order proximity preserved network embedding. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York: ACM, 2778–2786.