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

Jisung Yoon[1,2]**, Kai-Cheng Yang**[2]**, 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 37673, Republic of Korea.**
[2]**Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408, USA**
[3]**Department of Physics, Pohang University of Science and Technology, Pohang 37673, Republic of Korea.**
[4]**Asia Pacific Center for Theoretical Physics, Pohang 37673, Republic of Korea.**
[5]**Network Science Institute, Indiana University, Bloomington (IUNI), IN 47408, USA**
[6]**Connection Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA**

Corresponding author:
Yong-Yeol Ahn[1]

Email address: yyahn@iu.edu

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 models while achieving better performance.

INTRODUCTION

Graph embedding maps the nodes in a graph to continuous and dense vectors that capture relations among the nodes (Perozzi et al., 2014; Grover and 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 et al., 2014), community detection (Fortunato, 2010; Yang et al., 2016), link prediction (Grover and 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 et al., 2010; Coscia et al., 2014; Leskovec et al., 2009, 2010). Similarly, proteins and other biological elements play multiple functionalities (Palla et al., 2005; Gavin et al., 2006; Ahn et al., 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 et al., 2014; Li and Jurafsky, 2015; Iacobacci et al., 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 et al., 2017), but also for improving the quality of embedding (Li and Jurafsky, 2015;

45 Epasto and Perozzi, 2019; Liu et al., 2019). Yet, with the scarcity of embedding methods embracing this
46 idea, the full potential of this approach has not been properly explored.
47   In this paper, we propose `persona2vec`, a scalable framework that builds on the idea of ego-splitting
48 (Epasto et al., 2017), the process of identifying local structural contexts of a node via performing local
49 community detection on the node's ego-network. For each detected local community (role), we transform
50 each node into multiple personas if there are multiple local communities to which the node belongs. After
51 the split, the original node is replaced by the new persona nodes that inherit the connection from each
52 local community, producing a new persona graph. Instead of separating a node's persona nodes from
53 each other completely (Epasto and Perozzi, 2019), we add directed, weighted edges between personas to
54 capture their origin. In doing so, we allow the direct application of the existing graph embedding methods.
55 In addition, we take an approach of considering persona-based learning as fine-tuning of the base graph
56 embedding, achieving both efficiency and balance between information from the original graph and the
57 persona graph. Compared with the previous approach (Epasto and Perozzi, 2019), our framework is
58 conceptually simpler to understand and practically easier to implement. Furthermore, it achieves better
59 performance in the link prediction tasks while being much faster. We also would like to clarify that the
60 primary purpose of persona splitting is not about obtaining multiple representations, each of which may
61 be suited for a specific task; it is about teasing out multiple contexts that a single node may possess. In
62 other words, even with a single task, we argue that learning multiple representations for some nodes are
63 highly beneficial.
64   In sum, we would like to highlight that our approach (1) drastically lowers the barrier for combining
65 existing algorithms with persona splitting, (2) significantly improves the efficiency of the ego-splitting ap-
66 proach, while (3) consistently excelling the previous state-of-the-art model in the link prediction task. Our
67 implementation of `persona2vec` is publicly available at `https://github.com/jisungyoon/`
68 `persona2vec`.

RELATED WORK

70 In addition to graph embedding, our work is closely related to the research of identifying overlapping
71 communities in graphs. Various non-embedding methods such as link clustering (Ahn et al., 2010; Evans
72 and Lambiotte, 2009), clique percolation (Palla et al., 2005), and mixed membership stochastic blockmodel
73 (Airoldi et al., 2008) have been proposed. Another thread of works focuses on using local graph structure
74 to extract community information (Coscia et al., 2014; Epasto et al., 2015, 2017). Specifically, Epasto
75 et al. introduce the persona graph method for detecting overlapping communities in graphs (Epasto
76 et al., 2017), leveraging ego-network partition. The combination of ego-network analysis and graph
77 embedding methods is still rare. An example is `SPLITTER` (Epasto and Perozzi, 2019), which we use as
78 the baseline in this paper. Instead of constraining the relations between personas with a regularization
79 term, we propose a simpler and more efficient way of adding persona edges to the graph.
80   Our work is also related to the word disambiguation problem in a word embedding. Recently, word
81 embedding techniques (Mikolov et al., 2013a,b; Pennington et al., 2014) have been extensively applied to
82 various NLP tasks as the vectorized word representations can effectively capture syntactic and semantic
83 information. Although some words have multiple senses depending on the context, the original word
84 embedding methods only assign one vector to each word. Li *et al.* shows that embedding that is aware of
85 multiple word senses and provides vectors for each specific sense does improve the performance for some
86 NLP tasks (Li and Jurafsky, 2015). For this issue, some utilize the local context information and clustering
87 for identifying word sense (Reisinger and Mooney, 2010; Wu and Giles, 2015; Neelakantan et al., 2015),
88 some resort to external lexical database for disambiguation (Rothe and Schütze, 2015; Iacobacci et al.,
89 2015; Camacho-Collados et al., 2016; Chen et al., 2014; Jauhar et al., 2015; Pelevina et al., 2017), while
90 some combine topic modeling methods with embedding (Liu et al., 2015b,a; Cheng et al., 2015; Zhang
91 and Zhong, 2016). We adopt the idea of assigning multiple vectors to each node in the graph to represent
92 different roles as well as exploiting local graph structure for the purpose.

PROPOSED METHOD: `PERSONA2VEC`

94 `persona2vec` creates a *persona graph*, where some nodes are split into multiple personas. We then
95 apply a graph embedding algorithm to the persona graph to learn the embeddings of the personas (see
96 Fig. 1). Let us explain the method formally. Let $G = (V, E)$ be a graph with a set of nodes V and a
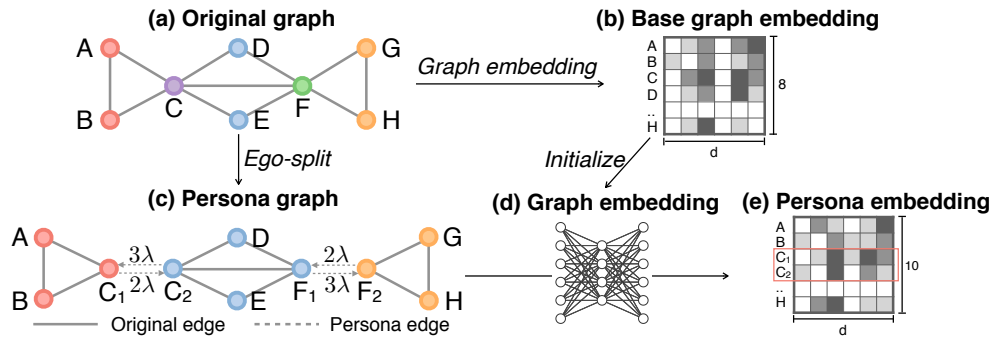
Figure 1. Illustration of `persona2vec` framework. (a) A graph with an overlapping community structure. (b) Graph embedding of the original graph is obtained first to initialize the persona embeddings. (c) Transform the original graph into a persona graph. Every edge in the original graph is preserved 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.

set of edges E. $|V|$ and $|E|$ denote the number of nodes and edges respectively. Let $f : v \to \mathbb{R}^d$ 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 et al., 2017; Epasto and 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 Fig. 1a, c). 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 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 component, thus other algorithms such as the Louvain method (Blondel et al., 2008), Infomap (Rosvall and Bergstrom, 2008), and label propagation (Raghavan et al., 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 et al., 2017; Epasto and 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 and 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 its 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

---

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 $\mathscr{C}$
Output: Persona graph $G_P(V_P,E_P)$; node to personas mapping $V2P$; persona to local cluster mapping $P2C$

1: **function** REFEGOSPLIT$(G(V,E),\lambda,\mathscr{C})$
2:    **for each** $v_o \in V$ **do**
3:       $P_{v_o} \leftarrow \mathscr{C}(v_o)$ ▷ 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** $G_P(V_P,E_P),V2P,P2C$

---

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 edge at the next step is

$$\frac{k_i^{\mathrm{o}}}{k_i^{\mathrm{o}} + n_p w_i}. \tag{1}$$

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

$$p_i = \frac{k_i^{\mathrm{o}}}{k_i^{\mathrm{o}} + n_p \lambda} = \frac{1}{1 + \frac{n_p}{k_i^{\mathrm{o}}} \lambda}, \tag{2}$$

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

$$p_i = \frac{k_i^{\mathrm{o}}}{k_i^{\mathrm{o}} + n_p \lambda k_i^{\mathrm{o}}} = \frac{1}{1 + n_p \lambda}, \tag{3}$$

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.

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$(G, d, \gamma_b, t_b, w_b, \gamma_p, t_p, w_p, \text{REFEGOSPLIT}, \text{EMBEDDINGFUNC}, \alpha)$
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** Φ_{G_P}

Also, training the embedding on the persona graphs from scratch fails to yield comparable results. We find that 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 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 $O(|V|^3)$ ($\because 2 \leq \gamma \leq 3$ 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 complexity of detecting the ego clusters in the graph with $|E|$ edges (Epasto et al., 2017). The embedding on the persona graph, which dominates the whole embedding procedure, has complexity $O(|V_p|\gamma twd(1 + \log(|V_p|)))$ 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).
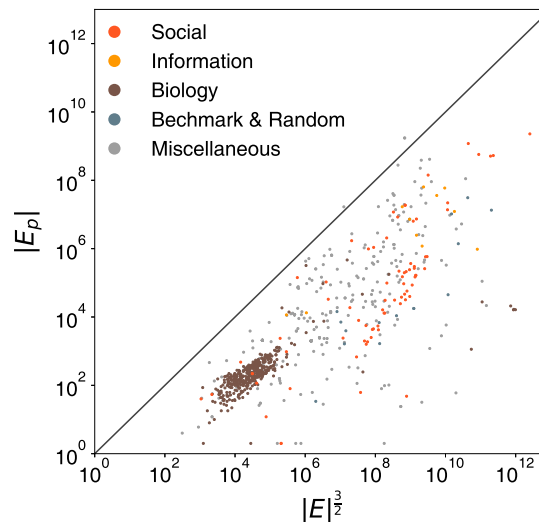
Figure 2. **Comparison of the the number of persona edges $|E_p|$ to the practical upper bound $|E|^{3/2}$.**

Table 1. **Descriptive statistics in 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	29,012	0.0003
wiki-vote	Directed	7,066	103,633	21,476	3.04	118,020	0.0035
soc-epinions	Directed	75,877	508,836	220,332	2.90	3,550,594	0.0098

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 and 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 proposed 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 et al., 2014), the probability of a node v_i co-occurring with a node v_j is estimated by

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

where $\mathbf{\Phi}_{v_i}$ and $\mathbf{\Phi}'_{v_i}$ are the "input" and "output" embedding of node i. We use input embedding $\mathbf{\Phi}$ which is known to be more useful and more widely used. Denominator of eq.4 is computationally expensive (Yang et al., 2016; Cao et al., 2016) and there are two common approximations: hierarchical softmax
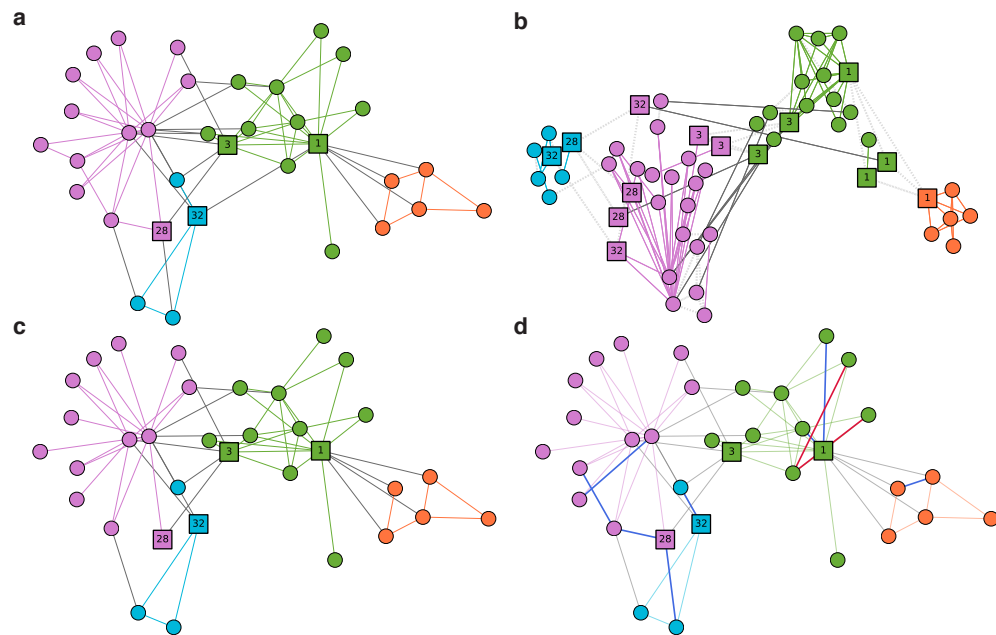
Figure 3. Case Study: Zachary's Karate club network (a) The Zachary's Karate club network with the force-atlas layout (Zachary, 1977). Nodes are colored by communities detected by the Louvain modularity method (Blondel et al., 2008). (b) The persona graph. Nodes are colored by k-means clusters (MacQueen et al., 1967) from the embedding vectors. Coordinates of the persona nodes come from the 2-D projection of the embedding with t-SNE (Maaten and Hinton, 2008). Light grey 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 score from the link prediction experiment. Blue links represent correctly predicted edges and red edges indicate incorrectly predicted ones.

(Morin and 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 as we see later.

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 place their personas in reasonable places. 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 separated as a result. We also show the ten predictions with the highest score from the link prediction experiment in Fig. 3d and checked 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
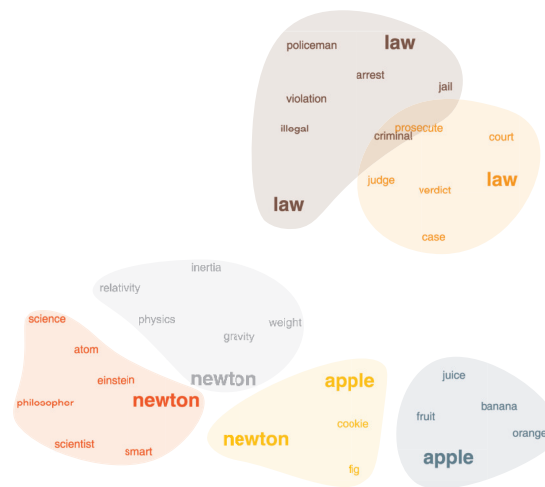
Figure 4. The word association network, clusters around the word "Newton". Coordinates of the words come from the 2-D projection of the embedding vectors with UMAP (McInnes et al., 2018). Word colors correspond to the clusters obtained by k-means clustering (MacQueen et al., 1967) on the embedding vectors.

on the original graph, for simplicity, we convert it into an undirected, unweighted graph by neglecting weight and direction (Ahn et al., 2010). In Fig. 4, we shows 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 grey 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 and Leskovec, 2016; Abu-El-Haija et al., 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 et al., 2017; Zhang et al., 2018).

We follow the task setup from the literature (Grover and Leskovec, 2016; Abu-El-Haija et al., 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 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 E_{test} is 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 as 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 and Perozzi, 2019). 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.

237 *wiki-vote* is a voting network, each node is a Wikipedia user and a directed edge from node i to node j
238 represents that user i voted for user j to become an administrator. *soc-epinions* is a voting graph from a
239 general consumer review site `Epinions.com`, each node is a member, and a directed edge from node i
240 to node j means that member i trusted member j.

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

Methods

244 The state-of-the-art method in this link prediction task is `SPLITTER` (Epasto and Perozzi, 2019),
245 which also models multiple roles. As reported in the paper, it outperforms various exiting algorithms
246 ranging across non-embedding methods like `Jaccard Coefficient`, `Common Neighbors`, and
247 `Adamic-Adar` as well as embedding methods like Laplacian `EigenMaps` (Belkin and Niyogi, 2002),
248 `node2vec` (Grover and Leskovec, 2016), `DNGR` (Cao et al., 2016), `Asymmetric` (Abu-El-Haija et al.,
249 2017) and `M-NMF` (Wang et al., 2017).

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

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

262 **Node2vec (baseline method)**   For `Node2vec`, we set random walk length $t = 40$, the number of walks
263 per node $\gamma = 10$, random walk parameters $p = q = 1$, the window size $w = 5$, and the initial learning rate
264 $\alpha = 0.025$. In the original paper, they learn an additional logistic regression classifier over the Hadamard
265 product of the embedding of two nodes for the link prediction. In general, the logistic regression classifier
266 improves the performance. Here, we report results on `Node2vec` with both dot products and the logistic
267 regression classifier.

268 **SPLITTER (baseline method)**   For `SPLITTER`, we use the same parameters in their paper (Epasto and
269 Perozzi, 2019) and `Node2vec` baseline. We use `node2vec` with random walk parameters $p = q = 1$.

270 **persona2vec (our proposed method)**   We set the hyper-parameters of the original graph embedding
271 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
272 the micro-structure of the persona graph. The size of the total trajectories is determined by the random
273 walk length t_* times the number of walks per node γ_*, so we keep $t_* \gamma_*$ constant to roughly preserve the
274 amount of information used in the embedding. For both embedding stages, we use the $\alpha = 0.025$, and
275 `node2vec` with the random walk parameters ($p = q = 1$) as the graph embedding function.

Experiment Results

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

281   As expected, larger dimensions lead to better performance, although `persona2vec` achieves rea-
282 sonable results even with tiny embedding dimensions like 8 or 16. We also show how the performance
283 of `persona2vec` varies with λ. For undirected graphs, larger λ is beneficial but the trend saturates
284 quickly. For directed graphs, however, optimal performance is achieved with smaller values of λ. In
285 practice, we suggest starting with $\lambda = 0.5$ as a default parameter because the overall variation brought by
286 λ is not substantial and even when the performance increases with λ, near-optimal performance can be
287 achieved at $\lambda = 0.5$.
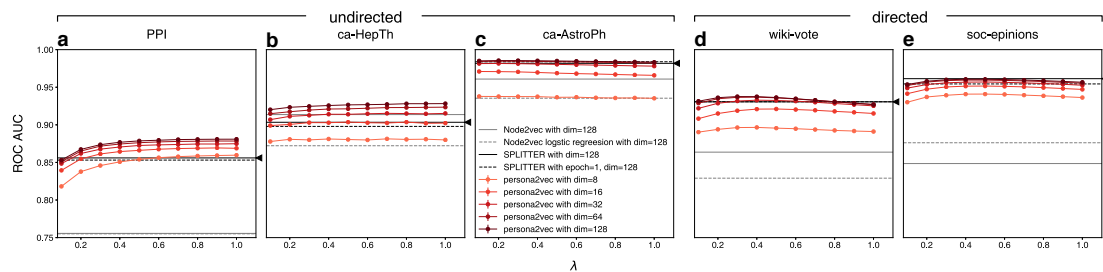
Figure 5. Performance of `persona2vec` in the link prediction task Number of epochs n is set to 1 in all experiments for `persona2vec`. Darker colors represent higher embedding dimensions. The confidence intervals are all within the range of the markers. Given the same number of dimensions, `persona2vec` is always on par or better than `SPLITTER`

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

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	$\mathbf{0.961 \pm 0.001}$
SPLITTER*	0.853	0.898	0.984	0.931	0.954 ± 0.001
persona2vec*	**0.879**	**0.927**	**0.985**	**0.936**	**0.961**
Performance gain	0.294	0.102	0.084	0.242	0.414 ± 0.007

²⁸⁸   When compared with the `SPLITTER` baseline, `persona2vec` shows on par or better performances
²⁸⁹ given the same embedding dimensions across a wide range of λ. We also report the performance
²⁹⁰ summary for `persona2vec` with $\lambda = 0.5$ (our suggested default) compared with the best baselines
²⁹¹ in Table 2, which show that `persona2vec` outperforms the baseline consistently. Also, we report the
²⁹² "performance gains" from `Node2vec`, because we used `Node2vec` as the base embedding method
²⁹³ and `persona2vec` can be considered an augmentation or fine-tuning of the base `Node2vec` vectors
²⁹⁴ with local structural information. As shown, the persona-based fine-tuning significantly improved the
²⁹⁵ performance.

²⁹⁶   Also, we show the performance of both methods across different approximations: hierarchical softmax
²⁹⁷ and negative sampling in Fig. 6. We also found that cosine similarity consistently yields a better result with
²⁹⁸ hierarchical softmax and dot product yields a better result with negative sampling across all methods. So,
²⁹⁹ we use cosine similarity for hierarchical softmax results and use dot product for negative sampling results.
³⁰⁰ We checked that both methods work well across the optimization method. We found that `persona2vec`
³⁰¹ tends to perform better with negative sampling and `SPLITTER` with hierarchical softmax. Nevertheless,
³⁰² `persona2vec` yields the best performance consistently.

³⁰³   In addition to the performance of the link prediction task, we also report the execution time of
³⁰⁴ `persona2vec` and `SPLITTER` to compare their scalabilities in practice (see Fig. 7). Note that the
³⁰⁵ reported execution time is on the link-prediction task, with half of the edges removed from the original
³⁰⁶ graph. `SPLITTER` runs the embedding procedures for 10 epochs by default in the original implementation,
³⁰⁷ whereas `persona2vec` only runs for one epoch. For a fair comparison, we also report the results of
³⁰⁸ `SPLITTER` with one epoch of training. When being limited to only one epoch, `SPLITTER`'s performance
³⁰⁹ slightly suffers on three graphs while it goes up or stays stable for the other two.

³¹⁰   Nevertheless, `persona2vec` is more efficient—39 to 58 times faster than `SPLITTER` with 10
³¹¹ epochs and five to eight times faster than `SPLITTER` with one epoch, while consistently outperforming
³¹² both. The most likely reason behind the drastic difference is the overhead from the extra regularization
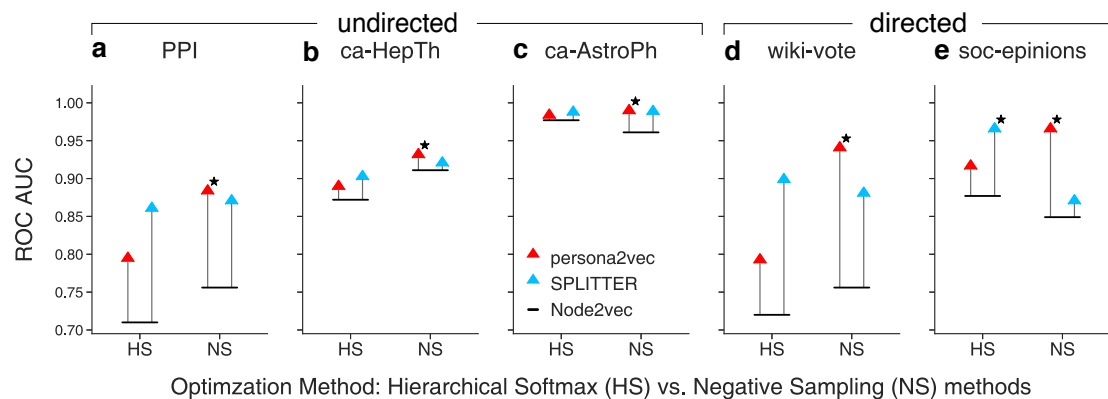
Figure 6. Comparison of link prediction performance between `persona2vec` and `SPLITTER` with different approximations. HS refers to the hierarchical softmax and NS refers to the negative sampling. The star marker indicates the best link prediction performance.
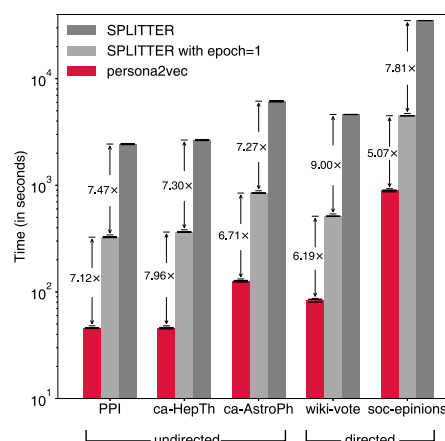
Figure 7. Comparison of elapsed time between `persona2vec` and `SPLITTER`. Speed gains by `persona2vec` are shown.

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 vectors 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 the multi-layer network, inter-layer coupling connection can be interpreted as natural persona edges, `persona2vec` may be applied to tackle the multi-layer link prediction problem.

As we know, 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 overlapping clusters and their heterogeneous functionality in most real networks, multi-role embedding methods may find numerous applications in physical, biological, and social sciences.

ACKNOWLEDGMENTS

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

Data Availability

All datasets used in this paper are publicly available. For *PPI*, we use the prepossessed version from the `node2vec` project web page (Grover and Leskovec, 2016), `https://snap.stanford.edu/node2vec/Homo_sapiens.mat` while other graphs are downloaded from the SNAP library homepage (Leskovec and Krevl, 2014), `http://snap.stanford.edu/data/index.html` (ca-AstroPh, ca-HepTh, wiki-Vote, soc-Epinions1).

Code Availability

Code used in this analysis can be found at `https://github.com/jisungyoon/persona2vec`

REFERENCES

Abu-El-Haija, S., Perozzi, B., and 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, pages 1787–1796, New York, NY, USA. ACM.

Ahn, Y.-Y., Bagrow, J. P., and Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764.

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2016). Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Cao, S., Lu, W., and Xu, Q. (2016). Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Chen, H., Perozzi, B., Hu, Y., and Skiena, S. (2018). Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, Hilton New Orleans Riverside, New Orleans, Louisiana, USA. AAAI press.

Chen, X., Liu, Z., and 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)*, pages 1025–1035.

Cheng, J., Wang, Z., Wen, J.-R., Yan, J., and Chen, Z. (2015). Contextual text understanding in distributional semantic space. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 133–142. ACM.

Coscia, M., Rossetti, G., Giannotti, F., and Pedreschi, D. (2014). Uncovering hierarchical and overlapping communities with a local-first approach. *ACM Trans. Knowl. Discov. Data*, 9(1):6:1–6:27.

Epasto, A., Lattanzi, S., Mirrokni, V., Sebe, I. O., Taei, A., and Verma, S. (2015). Ego-net community mining applied to friend suggestion. *Proceedings of the VLDB Endowment*, 9(4):324–335.

Epasto, A., Lattanzi, S., and 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, pages 145–154, New York, NY, USA. ACM.

Epasto, A. and Perozzi, B. (2019). Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*, pages 394–404, New York, NY, USA. ACM, ACM.

Evans, T. S. and Lambiotte, R. (2009). Line graphs, link partitions, and overlapping communities. *Phys. Rev. E*, 80:016105.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75–174.

Gavin, A.-C., Aloy, P., Grandi, P., Krause, R., Boesche, M., Marzioch, M., Rau, C., Jensen, L. J., Bastuck, S., Dümpelfeld, Birgit, E. 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, J. M., Kuster, B., Bork, P., Russell, R. B., and Superti-Furga, G. (2006). Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631.

Grover, A. and 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, pages 855–864, New York, NY, USA. ACM.

Iacobacci, I., Pilehvar, M. T., and 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)*, pages 95–105.

Jauhar, S. K., Dyer, C., and 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*, pages 683–693.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2009). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123.

Leskovec, J., Lang, K. J., and Mahoney, M. (2010). Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 631–640, New York, NY, USA. ACM.

Li, J. and Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? *preprint arXiv:1506.01070*.

Liu, N., Tan, Q., Li, Y., Yang, H., Zhou, J., and 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*, pages 932–940.

Liu, P., Qiu, X., and Huang, X. (2015a). Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015b). Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *preprint arXiv:1802.03426*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space. *preprint arXiv:1504.06654*.

Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814.

Pelevina, M., Arefyev, N., Biemann, C., and Panchenko, A. (2017). Making sense of word embeddings. *preprint arXiv:1708.03390*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perozzi, B., Al-Rfou, R., and 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, pages 701–710, New York, NY, USA. ACM.

Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106.

Reisinger, J. and Mooney, R. J. (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*, pages 109–117. Association for Computational Linguistics.

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *AAAI*.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Rosvall, M., Esquivel, A. V., Lancichinetti, A., West, J. D., and Lambiotte, R. (2014). Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications*, 5:4630.

Rothe, S. and Schütze, H. (2015). Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *preprint arXiv:1507.01127*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. (2006). Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017). Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Wu, Z. and Giles, C. L. (2015). Sense-aaware 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., and Shao, L. (2020). Region graph embedding network for zero-shot learning. In *European Conference on Computer Vision*, pages 562–580. Springer.

Yang, L., Cao, X., He, D., Wang, C., Wang, X., and Zhang, W. (2016). Modularity based community detection with deep learning. In *IJCAI*, volume 16, pages 2252–2258, New York, New York, USA. AAAI Press.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.

Zhang, H. and Zhong, G. (2016). Improving short text classification by learning vector representations of both words and hidden topics. *Knowledge-Based Systems*, 102:76–86.

Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., and Zhu, W. (2018). Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2778–2786. ACM.