

Distributed Representations of Subgraphs

Bijaya Adhikari Yao Zhang Naren Ramakrishnan B. Aditya Prakash

Department of Computer Science, Virginia Tech

Email: {bijaya, yaozhang, naren, badityap}@cs.vt.edu

Abstract—There has been a surge in research interest in learning feature representation of networks in recent times. Researchers, motivated by the recent successes of embeddings in natural language processing and advances in deep learning, have explored various means for network embedding. Network embedding is useful as it can exploit off-the-shelf machine learning algorithms for network mining tasks like node classification and link prediction. However, most recent works focus on learning feature representation of nodes, which are ill-suited to tasks such as community detection which are intuitively dependent on subgraphs. In this work, we formulate a novel subgraph embedding problem based on an intuitive property of subgraphs and propose SubVec, an unsupervised scalable algorithm to learn feature representations of arbitrary subgraphs. We demonstrate usability of features learned by SubVec by leveraging them for community detection problem, where SubVec significantly outperforms non-trivial baselines. We also conduct case-studies in two distinct domains to demonstrate wide applicability of SubVec.

I. INTRODUCTION

Graphs are useful as they provide a natural means for representing relational data from various domains such as social networks, co-authorship networks, the World Wide Web, and so on. Some of the common mining tasks on graphs include classification [1], link prediction [2], detecting communities (clustering) [3], [4], and so on. For other data types, these tasks can be solved using various machine learning algorithms. However, we cannot directly apply off-the-shelf machine learning algorithms on graph data, due to their high dimensionality and structural nature. Hence, learning discriminative feature representation of subgraphs can help to leverage existing machine learning algorithms more widely on graph data.

Recent works in network embedding [5], [6] have exploited relationships to vector representations in NLP like word2vec [7] to learn feature representation of nodes in networks. However, application of such methods are limited to node level tasks such as node classification. These methods seem to be unsuitable for subgraph level tasks like community detection as they result in loss of information of the subgraph structure. Embeddings of subgraphs or neighborhoods themselves seem to be better suited for these tasks. Surprisingly, learning feature representation of networks themselves (subgraphs and graphs) has not gained much attention thus far.

To address this issue, we study the problem of learning distributed representations of subgraphs in a low dimensional continuous vector space in this work. Such representations can be very useful in downstream network mining tasks like community detection, anomaly detection, subgraph classification and so on. Figure 1(a-c) gives an illustration of our framework.

Given a set of subgraphs (Figure 1 (b)) of a graph G (Figure 1 (a)), we learn a low-dimensional feature representation of each subgraph (Figure 1(c)).

As shown later, the embeddings of the subgraphs enable us to apply off-the-shelf machine learning algorithms directly to solve subgraph mining tasks. For example, to group subgraphs together, we can apply clustering algorithms like K-Means directly. Figure 2(a-c) shows a visualization of ground-truth communities in a network (a), communities found by using just node embeddings (b), and those found by our method SubVec (c). Clearly our result matches the ground-truth well while the other does not. Our contributions are:

- 1) We formulate novel subgraph embedding problem based on the intuitive Neighborhood property of subgraphs.
- 2) We propose SubVec, a scalable subgraph embedding method to learn features for arbitrary subgraphs that maintains the Neighborhood property.
- 3) We conduct multiple experiments over large diverse real datasets to show correctness and utility of features learnt by SubVec in the community detection task, where we get up to a gain of 123.5% compared to the closest baseline.

The rest of the paper is organized in the usual way. We first formulate and motivate our problem in Section II, and then we present SubVec in Section III. We discuss experiments in Section IV, followed by the related works. Finally, we present discussion and conclusions.

II. PROBLEM FORMULATION

We begin with the setting of our problem. We are given a graph $G(V, E)$ where V is the vertex set, and E is the associated edge-set (we assume unweighted undirected graphs here, but our framework can be easily extended to weighted and/or directed graphs as well). We define $g_i(v_i, e_i)$ as a subgraph of G , where $v_i \subseteq V$ and $e_i \subseteq E$. For simplicity, we write $g_i(v_i, e_i)$ as g_i . As input we require a set of subgraphs $\mathcal{S} = \{g_1, g_2, \dots, g_n\}$. Our goal is to embed subgraphs in \mathcal{S} into d -dimensional feature space \mathbb{R}^d , where $d \ll |V|$.

Main Idea: Intuitively, our goal is to learn a feature representation of each subgraph $g_i \in \mathcal{S}$ such that the likelihood of preserving a *property* of each subgraph, defined in the network setting, is maximized in the latent feature space. In this work, we provide a framework to preserve a natural property — namely *Neighborhood* property—of subgraphs.

Neighborhood Property: Intuitively, the Neighborhood property of a subgraph captures the neighborhood information within the subgraph itself for each node in it. For illustration

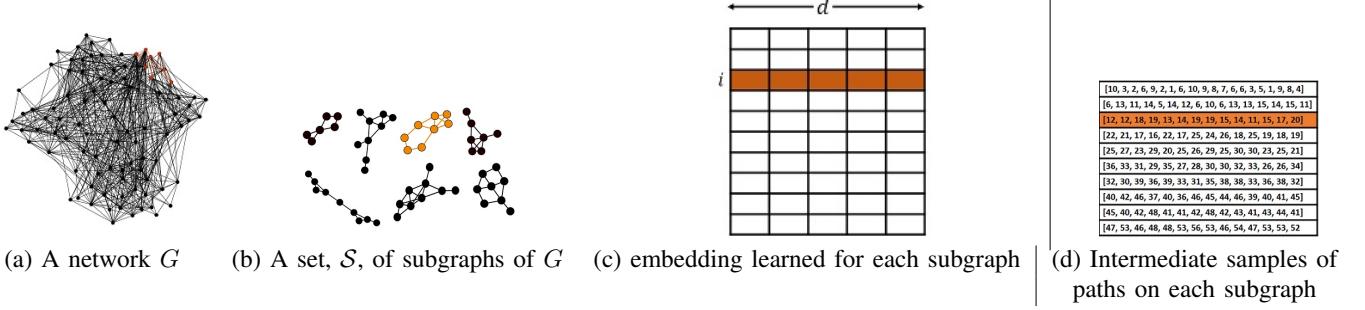


Figure 1: **An overview of our SubVec.** Our input is a set of subgraphs \mathcal{S} drawn from a network G . We obtain d dimensional embedding of subgraphs such that Neighborhood and Structural properties of subgraphs are preserved.

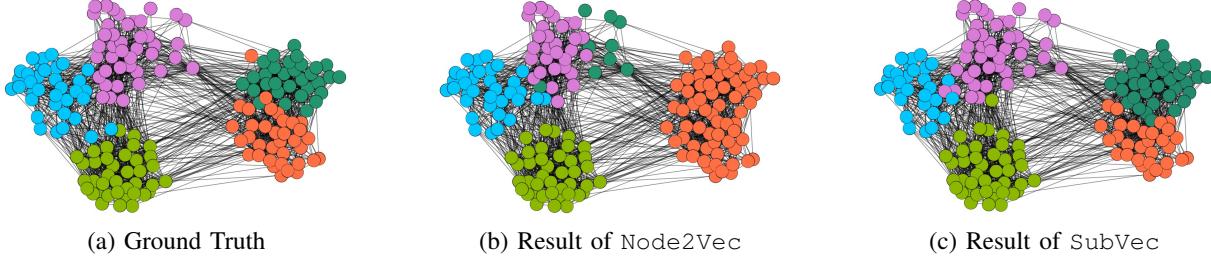


Figure 2: **Communities in HighSchool network** (different colors represent different communities). Layout is kept constant to highlight the results. Communities based on SubVec embeddings matches the ground truth, while the densely connected communities get merged for Node2Vec.

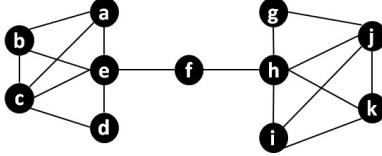


Figure 3: A toy network

consider an example. In Figure 3, let g_1 be the subgraph induced by nodes $\{a, e, c, d\}$. The Neighborhood property of g_1 should be able to capture the information that the nodes a, c are in the neighborhood of node e , that nodes d, e are in the neighborhood of node c and so on. To capture the neighborhood information of all the nodes in a given subgraph, we consider paths annotated by ids of the nodes. We refer to such paths as the *Id-paths* and define the Neighborhood property of a subgraph g_i as the set of all Id-paths in g_i .

The Id-paths capture the neighborhood information in subgraphs and each succession of nodes in Id-paths reveals how the neighborhood in the subgraph is evolving. For example in g_1 described above, the id-path $a \rightarrow c \rightarrow d$ shows that nodes a and c are neighbors of each other. Moreover, this path along with $a \rightarrow e \rightarrow d$ correctly indicate that nodes a and d are in neighborhood of each other (despite not being direct neighbors). Hence, the set of all Id-paths will capture

important connectivity information of the subgraph.

Our Problem: Having defined the Neighborhood property of subgraphs, we want to learn vector representations in \mathbb{R}^d , such that the likelihood of preserving the property in the feature space is maximized. Formally, our Subgraph Embedding problem is:

Problem 1: Given a graph $G(V, E)$, d and set of \mathcal{S} subgraphs (of G) $\mathcal{S} = \{g_1, g_2, \dots, g_n\}$, learn an embedding function $f : g_i \rightarrow \mathbf{y}_i \in \mathbb{R}^d$ such that the Neighborhood property of each $g_i \in \mathcal{S}$ is preserved.

III. LEARNING FEATURE REPRESENTATIONS

A common framework leveraged by most prior works in network embedding is to exploit Word2vec [7] to learn feature representation of nodes in the network. Word2vec learns similar feature representations for words which co-appear frequently in the same context. Network embedding methods, such as DeepWalk [5] and Node2vec [6], generate ‘context’ around each node based on random walks and embed nodes using Word2vec. These embeddings are known to preserve various node properties. However, such methods lack the global view of subgraphs, hence they are inherently unable to preserve the properties of entire subgraphs and fail in solving our problem.

A. Overview

A major challenge in solving our problems is to design an architecture which has global view of subgraphs and is able to capture similarities and differences between the properties of entire subgraphs. Our idea to overcome this challenge is to leverage the Paragraph2vec models for our subgraph embedding problems. Paragraph2vec [8] models learn latent representation of entire paragraphs while maximizing similarity between paragraphs which have similar word co-occurrences. Note that these models have the global view of entire paragraphs. Intuitively, such a model is suitable for solving Problem 1. Thus, we extend Paragraph2vec to learn subgraph embedding while preserving distance between subgraphs that have similar ‘node co-occurrences’. We extend both Paragraph2vec models (PV-DBOW and PV-DM). We call our models *Distributed Bag of Nodes version of Subgraph Vector* (SubVec-DBON) and *Distributed Memory version of Subgraph Vector* (SubVec-DM) respectively. We discuss SubVec-DM and SubVec-DBON in detail in Subsections III-C and III-D.

In addition, another challenge is to generate meaningful context of ‘node co-occurrences’ which preserves the Neighborhood property of subgraphs. We tackle this challenge by relying on our Id-paths. As discussed earlier, Id-paths preserves the Neighborhood property by capturing important connectivity information. We discuss more on efficiently generating samples of Id-paths in the next section.

B. Subgraph Truncated Random Walks

The Neighborhood property of subgraphs requires us to enlist paths in the subgraph (i.e, Id-paths). Since there are an unbounded number of paths, it is not feasible to enumerate all of them. Hence, we resort to random walks to generate meaningful samples of the paths efficiently. Next we describe the random walks to generate samples of Id-paths. Note that the random walks are performed *inside* each subgraph g_i separately, and not on the entire graph G .

Random Walk for Id-paths: Our random walk for Id-paths in subgraph $g_i(v_i, e_i)$ starts from a node $n_1 \in v_i$ chosen uniformly at random. We choose a neighbor of n_1 uniformly at random as the next node to visit in the random walk. Specifically, if i^{th} node in the random walk is n_i , then the j^{th} node in the random walk is a node n_j , such that $(n_i, n_j) \in e_i$, chosen uniformly at random among such nodes.

We generate random walk of fixed length l for each subgraph g_i in the input set of subgraphs $\mathcal{S} = \{g_1, g_2, \dots, g_n\}$. At the end of process, for each subgraph $g_i \in \mathcal{S}$, we obtain a random walk of length l , annotated by the ids of the nodes. Figure 1 (d), shows an example.

C. SubVec-DM

In the SubVec-DM model, we seek to predict a node-id u that appears in an Id-path, given other node-ids that co-occur with u in the path, and the subgraph that u belongs to. By co-occurrence, we mean that two ids co-appear in a sliding window of a fixed length w , i.e, they appear within distance

w of each other. Consider a subgraph g_1 (a subgraph induced by nodes $\{a, b, c, e\}$) in Figure 3. Suppose the random walk simulation of Id-paths in g_1 returns $a \rightarrow b \rightarrow c$, and we consider $w = 3$, then the model asks to predict node-id c given subgraph g_1 , and the node’s 2 predecessors (ids a and b), i.e., $\Pr(c|g_1, \{a, b\})$.

Here we give a formal formulation of SubVec-DM. Let $V = \bigcup_i v_i$ be the union of node-set of all the subgraphs. Let \mathbf{W}_1 be a $|\mathcal{S}| \times d$ matrix, where each row $\mathbf{W}_1(i)$ represents the embedding of a subgraph $g_i \in \mathcal{S}$. Similarly, let \mathbf{W}_2 be a $|V| \times d$ matrix, where each column $\mathbf{W}_2(n)$ is the vector representation of node $n \in V'$. Let the set of node-ids that appear within distance w of a node a be θ_a .

In SubVec-DM, we predict a node-id a given θ_a and the subgraph g_i , from which a and θ_a are drawn. Formally, the objective of SubVec-DM is to maximize:

$$\max_f \sum_{g_i \in \mathcal{S}} \sum_{n \in g_i} \log(\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))),$$

where $\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))$ is the probability of predicting node a given the vector representations of θ_a and g_i . $\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))$ is defined using the softmax function:

$$\Pr(a|m(\theta_a), f(g_i)) = \frac{e^{\mathbf{W}_3(a) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))}}{\sum_{v \in V} e^{\mathbf{W}_3(v) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(g_i))}} \quad (1)$$

where matrix \mathbf{W}_3 is a softmax parameter and $h(\mathbf{x}, \mathbf{y})$ is average or concatenation of vectors \mathbf{x} and \mathbf{y} [8]. In practice, to compute Equation 1, we use negative sampling or hierarchical softmax [7].

D. SubVec-DBON

In the SubVec-DBON model, we want to predict a set θ of co-occurring node-ids in an Id-path sampled from subgraph g_i , given only the subgraph g_i . Note that SubVec-DBON does not explicitly rely on the embeddings of node-ids as in SubVec-DM. As shown in Section III-C, the ‘co-occurrence’ means that two ids co-appear in a sliding window of a fixed length w . For example, consider the same example as in Section III-C: the subgraph g_1 in Figure 3, and the node sequence $a \rightarrow b \rightarrow c$ generated by random walks. Now in the SubVec-DBON model, for $w = 3$, the goal is to predict the set $\{a, b, c\}$ given the subgraph g_1 . This model is parallel to the popular skip-gram model. The matrices \mathbf{W}_1 and \mathbf{W}_2 are the same as in Section III-C.

Formally, given a subgraph g_i , and the θ drawn from g_i , the objective of SubVec-DBON is the following:

$$\max_f \sum_{g_i \in \mathcal{S}} \sum_{\theta \in g_i} \log(\Pr(\theta|\mathbf{W}_1(i))), \quad (2)$$

where $\Pr(\theta|\mathbf{W}_1(i))$ is a softmax function, i.e.,

$$\Pr(\theta|\mathbf{W}_1(i)) = \frac{e^{\mathbf{W}_2(\theta) \cdot \mathbf{W}_1(i)}}{\sum_{\theta' \in G} e^{\mathbf{W}_2(\theta') \cdot \mathbf{W}_1(i)}},$$

Since computing Equation 2 involves summation over all possible sets of co-occurring nodes, we use approximation techniques such as negative sampling [7].

E. Algorithm

Algorithm 1 SubVec

```

Require: Graph  $G$ , subgraph set  $\mathcal{S} = \{g_1, g_2, \dots, g_n\}$ , length of
the context window  $w$ , dimension  $d$ 
1: walkSet = {}
2: for each  $g_i$  in  $\mathcal{S}$  do
3:   walk = RandomWalk( $g_i$ )
4:   walkSet[ $g_i$ ] = walk
5: end for
6:  $f = \text{StochasticGradientDescent}(\text{walkSet}, d, w)$ 
7: return  $f$ 
```

Our algorithm SubVec works as follows: we first generate the samples of Id-paths in each subgraph by running random walks. Then we optimize the SV-DBON/ SV-DM objectives using the stochastic gradient descent (SGD) method [9] by leveraging the random walks. We used the Gensim package for implementation [10]. The complete pseudocode is presented in Algorithm 1.

IV. EXPERIMENTS

We leverage SubVec¹ to solve the well-known community detection problem. We also conduct case studies on two datasets from distinct domains. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB 1066Mhz RAM. We set the length of the random walk as 1000 and following literature [6], we set dimension of the embedding as 128 unless mentioned otherwise for both parameters.

A. Community Detection

Setup. Here we show how to leverage SubVec for the well-known community detection problem [3], [4]. A community in a network is a coherent group of nodes which are densely connected among themselves and sparsely connected with the rest of the network. Hence we expect many nodes within the same community to have similar neighborhoods. Hence, we can use SubVec to embed subgraphs while preserving the Neighborhood property and cluster the embeddings to detect communities.

Approach. We propose to use SubVec for community detection by embedding the surrounding neighborhood of each node. First, we extract the neighborhood C_v of each node $v \in V$ from the input graph $G(V, E)$. Then we run SubVec on $\mathcal{S} = \{C_v | v \in V\}$ to learn feature representation of $f(C_v)$ for all $C_v \in \mathcal{S}$. We then use any clustering algorithm (K-Means) to cluster the feature vectors $f(C_v)$. For datasets with overlapping communities (like `Youtube`), we use the Neo-Kmeans algorithm [11] to obtain overlapping clusters. Cluster membership of $f(C_v)$ determines the community membership of node v . The complete pseudocode is in Algorithm 2.

In Algorithm 2, we define the neighborhood of each node to be its ego-network for dense networks (`HighSchool` and `WorkPlace`) and its 2-hop ego-network for sparse networks. The ego-network of a node is the subgraph induced by the

¹Code in Python available at <http://people.cs.vt.edu/~bijaya/codes/SubVec>

Algorithm 2 Community Detection using SubVec

```

Require: A network  $G(V, E)$ , SubVec parameters,  $k$  number
of communities
1: neighborhoodSet = {}
2: for each  $v$  in  $V$  do
3:   neighborhoodSet = neighborhoodSet  $\cup$  neighborhood
      of  $v$  in  $G$ .
4: end for
5:  $\text{vecs} = \text{SubVec}(\text{neighborhoodSet}, w, d)$ 
6:  $\text{clusters} = \text{Clustering}(\text{vecs}, k)$ 
7: return  $\text{clusters}$ 
```

Table I: Information on Datasets for Community Detection.

Dataset	$ V $	$ E $	# Communities	Domain
WorkPlace [12]	92	757	5	contact
Cornell [13]	195	304	5	web
HighSchool [14]	182	2221	5	contact
Texas [13]	187	328	5	web
Washington [13]	230	446	5	web
Wisconsin [13]	265	530	5	web
PolBlogs [15]	1490	16783	2	web
Youtube [16]	1.13M	2.97M	5000	social

node and its neighbors. The 2-hop ego-network is the subgraph induced by the node, its neighbors, and neighbors' neighbors.

Datasets. We use multiple real world datasets from multiple domains like social-interactions, co-authorship, social networks and so on of varying sizes (largest containing $\sim 3M$ edges). See Table I.

- 1) `WorkPlace` is a publicly available social contact network between employees of a company with five departments². Edges indicate that two people were in proximity of each other. Each department is a ground truth community.
- 2) `HighSchool` is a social contact network². Nodes are high school students belonging to one of five different sections and edges indicate that two students were in vicinity of each other. Each section is a ground truth community.
- 3) `Texas`, `Cornell`, `Washington`, `Wisconsin` are networks from the WebKB dataset³. These are networks of webpages and hyperlinks. Each webpage belongs to one of five classes: course, faculty, student, project, and staff, which serve as ground-truth.
- 4) `PolBlogs` is a directed network of hyperlinks between weblogs on US politics, recorded in 2005. We take conservative and liberal blogs as ground-truth communities.
- 5) `Youtube` is a social network, where edges indicate friendship between two users. Ground truth communities are defined by the different user-created groups.

Baselines. We compare SubVec with various traditional community detection algorithms and network embedding based methods. Newman [3] is a well-known community detection algorithm based on betweenness. Louvian [4] is a popular greedy optimization method. DeepWalk and Node2Vec are recent network embedding methods which learn feature

²<http://www.sociopatterns.org/>

³<http://linqs.cs.umd.edu/projects/lbc/>

representations of nodes in the network which we then cluster (in the same way as us) to obtain communities.

Results. We measure the performance of all the algorithms by computing the Average F1 score [16] against the ground-truth.

See Table II. Both versions of SubVec significantly and consistently outperform all the baselines. We achieve a significant gain of 123.5 % over the closest competitor (Node2Vec) for Youttube. We do better than Node2Vec and DeepWalk because intuitively, we learn the feature vector of the neighborhood of each node for the community detection task; while they just do random probes of the neighborhood. Performance of Newman and Louvian is considerably poor in Youttube as these methods output non-overlapping communities. Performance of Node2Vec is satisfactory in sparse networks like Washington and Texas. Node2Vec does slightly better ($\sim 1\%$) than SubVec in PolBlogs—the network consists of homogeneous neighborhoods, which favors it. However, the performance of Node2Vec is significantly worse for dense networks like WorkPlace and HighSchool. On the other hand, performance of SubVec is even more impressive in these dense networks (where the task is more challenging).

SubVec-DM vs SubVec-DBON. In Table II, we observe that SubVec-DM outperforms SubVec-DBON. Recall that the SubVec-DM optimization relies on finding the embeddings of the nodes as well as subgraphs. We conjecture that since SubVec-DM relies on node embedding, based on Id-paths which captures the Neighborhood property, it performs well for the community detection task. Since, SubVec-DBON learns the features of subgraphs directly, without relying on node embeddings, intuitively it should be more useful on very large dense networks, where the node embeddings might not be discriminative enough.

B. Scalability

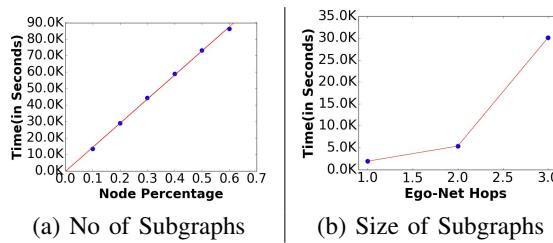


Figure 4: **Scalability w.r.t. number of subgraphs on Youttube and w.r.t size of subgraphs on Astro-PH datasets.**

Here we show the scalability of SubVec with respect to the number and the size of subgraphs. We extract connected subgraphs of Youttube dataset induced by varying percentage of nodes. We then run SubVec on the set of ego-nets in each resulting network. As shown in Figure 4 (a), SubVec is linear w.r.t number of subgraphs. In Figure 4 (b), we run SubVec on 1 to 3 hops ego-nets of Astro-PH dataset. We see a significant jump in the running time when the hop increases from 2 to 3. This is due to the fact that as the hop of ego-net

increases, the size of the subgraph increases exponentially due to the low diameter of real world networks.

C. Case Studies

We perform case-studies on MemeTracker⁴ and DBLP to investigate if the embeddings returned by SubVec are interpretable. MemeTracker consists of a series of cascades caused by memes spreading on the network of linked web pages. Each meme-cascade induces a subgraph in the underlying network. We first embed these subgraphs in a continuous vector space by leveraging SubVec. We then cluster these vectors to explore what kind of meme cascade-graphs are grouped together, what characteristics of memes determine their similarity and distance to each other and so on. For this case-study, we pick the top 1000 memes by volume, and cluster them into 10 clusters using K-Means.

We find coherent clusters which are meaningful groupings of memes based on topics. For example we find cluster of memes related to different topics such as entertainment, politics, religion, technology and so on. Visualization of these clusters is presented in Figure 5. In the entertainment cluster, we find memes which are names of popular songs and movies such as “sweet home alabama”, “somewhere over the rainbow”, “Madagascar 2” and so on. Similarly, we also find a cluster of religious memes. These memes are quotes from the Bible. We also find memes related to politics and religion in the same cluster such as “separation of church and state”. In politics cluster, we find popular quotes from the 2008 presidential election season e.g. Barack Obama’s popular slogan “yes we can” along with his controversial quotes like “you can put lipstick on a pig” in the cluster. We also find Sarah Palin’s quote like “the chant is drill baby drill”. Similarly, we also find different clusters of technology/video games related memes and memes in Spanish language.

For DBLP, we follow the methodology in [17], and extract subgraphs of the coauthorship network based on the keywords contained in the title of the papers. We include keywords such as ‘classification’, ‘clustering’, ‘xml’, and so on. Once we extract the subgraphs, we run SubVec to learn embedding of these subgraphs. We then project the embeddings down to 2-dimensions using t-SNE [18]. See Figure 6. We see that the related keywords such as ‘graphs’, ‘pagerank’, ‘crawling’, and ‘clustering’ appear together. Classification related keywords such as ‘boosting’, ‘svm’, and ‘classification’ are grouped together. These meaningful groups of keywords highlight the fact that SubVec results in meaningful embeddings.

V. RELATED WORK

Network Embedding. The network embedding problem has been well studied. Most of work seeks to generate low dimensional feature representation of nodes. Earliest work in dimensionality reduction includes Laplacian Eigenmap [19], IsoMap [20], locally linear embedding [21], and spectral techniques [22], [23], [24]. However, these methods are slow

⁴snap.stanford.edu

Table II: SubVec **easily out-performs all baselines in all datasets**. Average F-1 score is shown for each method. Winners have been bolded for each dataset.

Method	WorkPlace	HighSchool	PolBlogs	Texas	Cornell	Washington	Wisconsin	Youtube
Newman	0.32	0.34	0.58	0.17	0.33	0.21	0.16	0.04
Louvian	0.25	0.31	0.50	0.20	0.20	0.13	0.19	0.01
DeepWalk	0.40	0.48	0.80	0.25	0.32	0.29	0.29	0.15
Node2Vec	0.64	0.79	0.86	0.27	0.33	0.28	0.30	0.17
SubVec-DM	0.77	0.93	0.85	0.35	0.36	0.38	0.32	0.38
SubVec-DBON	0.65	0.57	0.82	0.35	0.34	0.37	0.32	0.36
Gain of SubVec [%]	20.3	17.7	-1.2	29.6	9.1	31.0	6.6	123.5

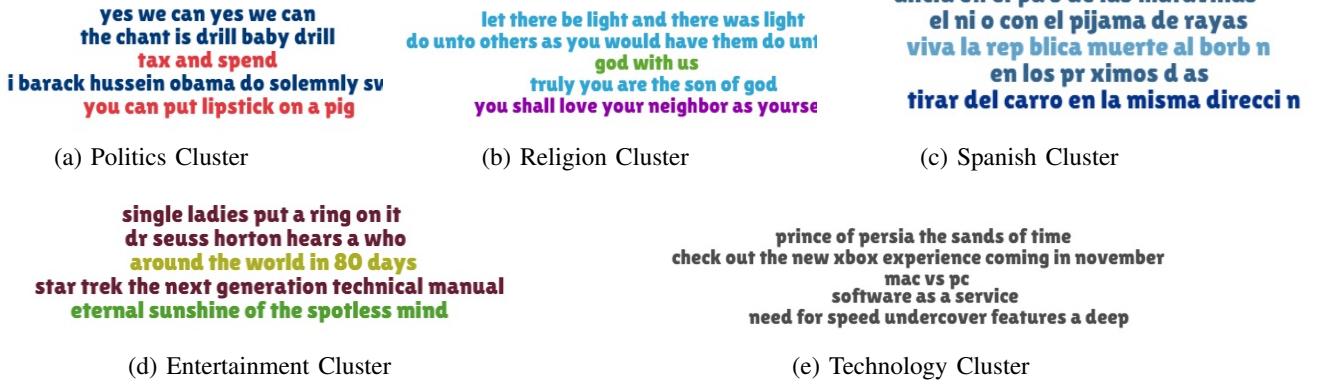


Figure 5: Different Clusters of Memes for the MemeTracker dataset.

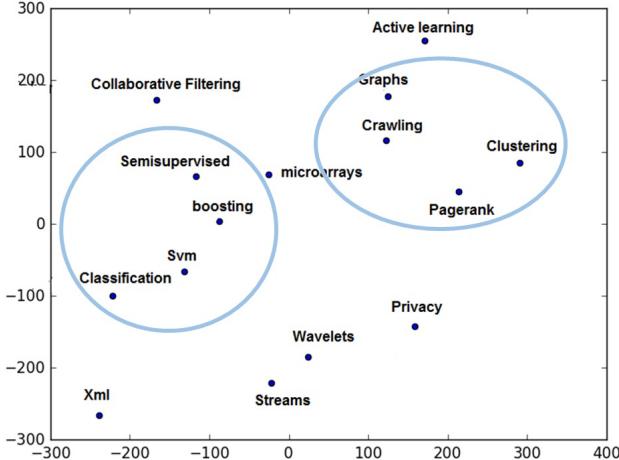


Figure 6: 2D projection of feature vectors learnt by SubVec of subgraphs of DBLP induced by different keywords.

and do not scale to large networks. Recently, several deep learning based network embeddings algorithms were proposed to learn feature representations of nodes [5], [25], [26], [6]. Perozzi et. al [5] proposed DeepWalk, which extends skip-Gram model [7] to networks and learns feature representation based on contexts generated by random walks. Grover et. al. proposed a more general method, Node2Vec [6], which generalizes random walks to generate various contexts. SDNE [25] and LINE [26] learn feature representation of nodes while

preserving first and second order proximity. Node2vec was shown to outperform DeepWalk and LINE in link prediction. However, all of them learn low dimensional feature vector of nodes, while our goal is to embed subgraphs.

Some of the closely related works in literature include [27], [28], [29]. Risen and Bunke propose to learn vector representations of graphs based on edit distance to a set of pre-defined prototype graphs [27]. Yanardag et. al. [28] and Narayanan et al. [29] learn vector representation of the subgraphs using the Word2Vec [7] by generating "corpus" of subgraphs where each subgraph is treated as a word. The above work focuses on some specific subgraphs like graphlets and rooted subgraphs. None of them embed subgraphs with arbitrary structure. In addition, we interpret subgraphs as paragraphs, and leverage the PV-DBOW/DM models [8].

Other Subgraph Problems. There has been a lot of work on subgraph related problems like subgraph discovery. Finding the largest clique is a well-known NP-complete problem [30]. Lee et al. surveyed dense subgraph discovery algorithms for several subgraphs including clique, K-core, K-club, etc [31]. Perozzi et al. studied the attributed graph anomaly detection by exploring the neighborhood subgraph of a nodes [32]. Different from the above works, we seek to find feature representations of subgraphs.

VI. DISCUSSION AND CONCLUSION

In this paper, we formulated novel subgraph embedding problem which seeks to preserve Neighborhood property of subgraphs. We propose scalable SubVec algorithm, which

learns feature representation of subgraphs based on samples of local connectivity information, to solve the problem. We demonstrate that SubVec gives meaningful interpretable embeddings of arbitrary subgraphs by leveraging it to solve the community detection problem. We show via our experiments that SubVec outperforms traditional algorithms as well as node-level embedding algorithms for extracting communities from networks. Similarly, we also demonstrate usability of SubVec by conducting case studies on two datasets from different domains.

Embedding subgraphs while preserving other properties are of great interest as well. One such property is based on the *structure* of subgraphs. For example, in Figure 3, subgraphs induced by nodes $\{a, b, c, e\}$ and $\{h, i, j, k\}$ have the same structural property as both are cliques of size four. Embedding subgraphs while preserving structure based property could be leveraged for tasks like graph classification, graph isomorphism detection and so on. We leave the task of embedding subgraphs based on the structure induced properties of subgraphs and leveraging them for graph mining as future work.

Acknowledgements This paper is based on work partially supported by the National Science Foundation (IIS-1353346, DGE-1545362, and IIS-1633363), the National Endowment for the Humanities (HG-229283-15), ORNL (Task Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the respective funding agencies.

REFERENCES

- [1] S. Bhagat, G. Cormode, and S. Muthukrishnan, “Node classification in social networks,” in *Social network data analytics*. Springer, 2011, pp. 115–148.
- [2] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” *Journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
- [3] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
- [4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
- [5] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.
- [6] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.
- [7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *NIPS*, 2013, pp. 3111–3119.
- [8] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *ICML*, vol. 14, 2014, pp. 1188–1196.
- [9] O. Bousquet and L. Bottou, “The tradeoffs of large scale learning,” in *Advances in neural information processing systems*, 2008, pp. 161–168.
- [10] R. Rehurek and P. Sojka, “Software framework for topic modelling with large corpora,” in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.
- [11] J. J. Whang, I. S. Dhillon, and D. F. Gleich, “Non-exhaustive, overlapping k-means,” in *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015, pp. 936–944.
- [12] M. Genois, C. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat, “Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers,” *Network Science*, vol. 3, pp. 326–347, 9 2015.
- [13] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.
- [14] J. Fournet and A. Barrat, “Contact patterns among high school students,” *PLoS ONE*, vol. 9, no. 9, p. e107878, 09 2014.
- [15] L. A. Adamic and N. Glance, “The political blogosphere and the 2004 us election: divided they blog,” in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.
- [16] J. Yang and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 587–596.
- [17] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila, “Finding effectors in social networks,” in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1059–1068.
- [18] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [19] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *NIPS*, vol. 14, no. 14, 2001, pp. 585–591.
- [20] J. B. Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
- [21] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [22] F. R. Bach and M. I. Jordan, “Learning spectral clustering,” in *NIPS*, vol. 16, 2003.
- [23] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.
- [24] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, “Community preserving network embedding,” 2017.
- [25] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1225–1234.
- [26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1067–1077.
- [27] K. Riesen and H. Bunke, *Graph classification and clustering based on vector space embedding*. World Scientific Publishing Co., Inc., 2010.
- [28] P. Yanardag and S. Vishwanathan, “Deep graph kernels,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374.
- [29] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Samithan, “subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs,” *arXiv preprint arXiv:1606.08928*, 2016.
- [30] J. Hästads, “Clique is hard to approximate within n1,” in *Proc. 37th Symp. on Found. Comput. Sci*, 1996, pp. 627–636.
- [31] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, “A survey of algorithms for dense subgraph discovery,” in *Managing and Mining Graph Data*. Springer, 2010, pp. 303–336.
- [32] B. Perozzi and L. Akoglu, “Scalable anomaly ranking of attributed neighborhoods,” in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 207–215.