Sub2Vec: Feature Learning for Subgraphs

Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan and B. Aditya Prakash

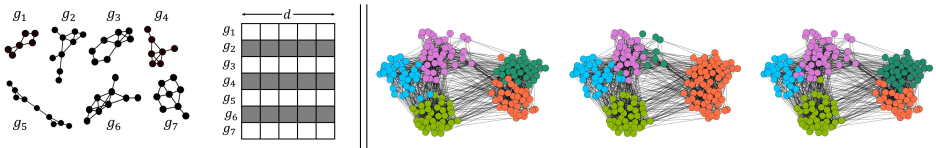
Department of Computer Science, Virginia Tech
Email: [bijaya, yaozhang, naren, badityap]@cs.vt.edu

Abstract. Network embeddings have become very popular in learning effective feature representations of networks. Motivated by the recent successes of embeddings in natural language processing, researchers have tried to find network embeddings in order to exploit machine learning algorithms for mining tasks like node classification and edge prediction. However, most of the work focuses on distributed representations of nodes that are inherently ill-suited to tasks such as community detection which are intuitively dependent on subgraphs. Here, we formulate subgraph embedding problem based on two intuitive properties of subgraphs and propose Sub2Vec, an unsupervised algorithm to learn feature representations of arbitrary subgraphs. We also highlight the usability of Sub2Vec by leveraging it for network mining tasks, like community detection and graph classification. We show that Sub2Vec gets significant gains over state-of-the-art methods. In particular, Sub2Vec offers an approach to generate a richer vocabulary of meaningful features of subgraphs for representation and reasoning.

1 Introduction

Graphs are a natural abstraction for representing relational data from multiple domains such as social networks, protein-protein interaction networks, the World Wide Web, and so on. Analysis of such networks include classification [1], detecting communities [2, 3], and so on. Many of these tasks can be solved using machine learning algorithms. Unfortunately, since most machine learning algorithms require data to be represented as features, applying them to graphs is challenging due to their high dimensionality and structure. In this context, learning discriminative feature representation of subgraphs can help in leveraging existing machine learning algorithms more widely on graph data.

Apart from classical dimensionality reduction techniques (see related work), recent works [4–7] have explored various ways of learning feature representation of nodes in networks exploiting relationships to vector representations in NLP (like word2vec [8]). However, application of such methods are limited to binary and multi-class node classification and edge-prediction. It is not clear how one can exploit these methods for tasks like community detection which are inherently based on subgraphs and node embeddings result in loss of information of the subgraph structure. Embedding 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. Here, we address this gap by studying the problem of learning distributed representations of subgraphs in a low dimensional continuous vector space. Figure 1(a-b) gives an illustration of our framework. Given a set of subgraphs (Figure 1 (a)), we learn a low-dimensional feature representation of each subgraph (Figure 1(b)).



(a) Input subgraphs (b) Embeddings (c) Ground Truth (d) `Node2Vec` (e) `Sub2Vec`

Fig. 1: (a) and (b) An overview of our `Sub2Vec`. Our input is a set of subgraphs $\mathcal{S}$. `Sub2Vec` learns d dimensional feature embedding of each subgraph. (c)-(e) Leveraging embeddings learned by `Sub2Vec` for community detection. (c) Communities in `School` network (different colors represent different communities). (d) Communities discovered via `Node2Vec` deviates from the ground truth, (e) while those discovered via `Sub2Vec` closely matches the ground truth.

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, for community detection, we can first embed the ego-nets of each node using sub2vec, and then apply clustering algorithms like k-means on the embeddings (see Section 4.1 later for more details). Figure 1(c-e) shows a visualization of ground-truth communities in a network (c), communities found by using just node embeddings (d), and those found by our method `Sub2Vec` (e). Clearly our result matches the ground-truth well while the other is far from it. Our contributions are:

- We identify two intuitive properties of subgraphs (Neighborhood and Structural). Then we formulate two novel *Subgraph Embedding* problems, and propose `Sub2Vec`, a scalable subgraph embedding framework to learn features for arbitrary subgraphs that maintains the properties.
- We conduct multiple experiments over diverse datasets to show correctness, scalability, and utility of `Sub2Vec` in several tasks. We get upto a gain of 123.5% in community detection and upto 33.3% in graph classification compared to closest competitors.

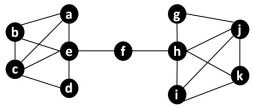
2 Problem Formulation

We begin with the setting of our problem. Let $G(V, E)$ be a graph 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). A graph $g_i(v_i, e_i)$ is said to be a subgraph of a larger graph $G(V, E)$ if $v_i \subseteq V$ 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 $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, typically extracted from the same graph $G(V, E)$. Our goal is to embed each subgraph $g_i \in \mathcal{S}$ into d-dimensional feature space $\mathbb{R}^d$, where $d << |V|$.

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 certain *properties* of each subgraph, defined in the network setting, is maximized in the latent feature space. In this work, we provide a framework to preserve two different properties—namely *Neighborhood* and *Structural*— properties of subgraphs.

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

illustration, consider the following example. In the figure below, 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. 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 $a \rightarrow e \rightarrow d$ indicate that nodes a and d are in neighborhood of each other (despite not being direct neighbors). Hence, the set of all Id-paths captures important connectivity information of the subgraph.

Structural Property: The Structural property of a subgraph captures the *overall* structure of the subgraph as opposed to just the local connectivity information as captured by the Neighborhood property. Several prior works have leveraged degree of nodes and their neighbors to capture structural information in network representation learning [9, 10]. While degree of a node captures its local structural information within a subgraph, it fails in characterizing the similarity between the structures of two nodes in different subgraphs. Note that the nodes of two subgraphs with the same structure but of different sizes will have different degrees. For example, nodes in clique of size 10 have degree of 9, whereas nodes in clique of size 6 have degree 5. Therefore this suggests that instead, the *ratio* of degree to the size of the subgraph, of a node and its neighbors better identifies subgraph structure. Hence we rely on paths in g_i annotated by the ratio of node degrees to the subgraph size. We refer to the set of all such paths as Degree-paths. Degree-paths capture the structure by tracking how the density of edges changes in a neighborhood. While our method is simple, it is effective as shown by the results. One can build upon our framework by leveraging other techniques like rooted subgraphs [11] and predefined motifs [12].

As an example, consider the subgraph g_2 induced by nodes $\{a, b, c, e\}$ and subgraph g_3 induced by nodes $\{f, h, i, g\}$ in the graph shown above. As it is a clique, the ratio of degree to the size of the subgraph for each node in g_2 is 0.75. Hence any Degree-paths of length 3 in g_2 is $0.75 \rightarrow 0.75 \rightarrow 0.75$. Similarly, g_3 is a star and a Degree-path in g_3 from i to g is $0.25 \rightarrow 0.75 \rightarrow 0.25$. The consistent high values in the paths in cliques show that each node in the path is densely connected to the rest of the graph, while the fluctuation in values in stars show that the two spokes in the path are sparsely connected to the rest of the network while the center is densely connected. In practice, since we cannot treat each real value distinctly, we generate labels for each node from a fixed alphabet (see Section 3).

Our Problems: Having defined the Neighborhood and Structural properties of subgraphs, we want to learn vector representations in $\mathbb{R}^d$, such that the likelihood of preserving these properties in the feature space is maximized. Formally the two versions of our Subgraph Embedding problem are:

Problem 1. Given a graph $G(V, E)$, d, and set of $\mathcal{S}$ subgraphs (of G) $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, 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.

Problem 2. Given a graph $G(V, E)$, d, and set of $\mathcal{S}$ subgraphs (of G) $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, learn an embedding function $f : g_i \rightarrow \mathbf{y_i} \in \mathbb{R}^d$ such that the Structural property of each $g_i \in \mathcal{S}$ is preserved.

3 Our Methods

A common framework leveraged by most prior works in network embedding is to exploit Word2vec [8] 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 [4] and Node2vec [5], 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 problems.

3.1 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 [13] 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 Problems 1 and 2. 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* (`Sub2Vec`-DBON) and *Distributed Memory version of Subgraph Vector* (`Sub2Vec`-DM) respectively. We discuss `Sub2Vec`-DM and `Sub2Vec`-DBON in detail in Subsections 3.3 and 3.4.

In addition, another challenge is to generate meaningful context of 'node co-occurrences' which preserve Neighborhood and Structural properties of subgraphs. We tackle this challenge by using our Id-paths and Degree-paths. As discussed earlier, Id-paths and Degree-paths capture the Neighborhood and Structural property respectively. We discuss on efficiently generating samples of Id-paths and Degree-paths next.

3.2 Subgraph Truncated Random Walks

Both Neighborhood and Structural properties of subgraphs require us to enlist paths in the subgraph (Id-paths for Neighborhood and Degree-paths for Structural). Since there are an unbounded number of paths, it is not feasible to enumerate all of them. Hence, we

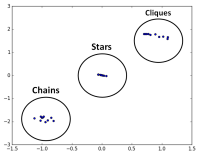
resort to random walks to generate samples of the paths efficiently. Next we describe the random walks for Id-paths and Degree-paths respectively. Note that the random walks are performed *inside* each subgraph g_i separately, and not on the entire graph G.

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 $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$. 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.

Degree-paths: As mentioned in Section 2, we generate labels for each node based on the ratio of degree of a node to the number of nodes in the subgraph for Degree-paths. We formally define the label generation for nodes next.

Let θ_a be the degree of the node n_a. Given a subgraph $g(v_i, e_i)$, we get the ratio $\gamma_a = \frac{\theta_a}{|v_i|}$ for each node $n_a \in v_i$. Note that $\gamma \in [0, 1]$. Roughly, we define various bins of range of values and associate a character with it (e.g. [0,0.2) is 'a', [0.2,0.5) is 'b' and so on). As real networks have very skewed degree distributions, we use logarithmic binning [14] to determine bin-widths. Formally, let Σ be a finite alphabet over distinct characters; and we define a many-to-one relabeling function $\tau : [0, 1] \rightarrow \Sigma$.



After generating the labels for each node, we follow the same procedure as in Id-paths, and do a random walk on each subgraph. The only difference is that we generate a sequence of characters $\alpha \in \Sigma$ for Degree-paths as opposed to node-ids for Id-paths. An example of a 2-dimensional projection of embedding generated based on our relabeling process is shown in the left. In the figure, subgraphs of similar structure are embedded close to each other and well separated from structures.

3.3 `Sub2Vec`-DM

In the `Sub2Vec`-DM model, assuming we want to preserve Neighborhood property of subgraphs, 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 the toy graph shown earlier. 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., $\Pr(c|g_1, \{a, b\})$. Note that if our goal is to preserve the Structural property, we use Degree-paths instead.

Here we give a formal formulation of `Sub2Vec`-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 Sub2Vec-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 Sub2Vec-DM is to maximize:

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

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. It is defined using the softmax function:

$$\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i)) = \frac{e^{\mathbf{W_3}(a) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))}}{\sum_{v \in V} e^{\mathbf{W_3}(v) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(g_i))}} \quad (1)$$

where matrix $\mathbf{W_3}$ is a softmax parameter and $h(\mathbf{x}, \mathbf{y})$ is average or concatanation of vectors $\mathbf{x}$ and $\mathbf{y}$ [13]. In practice, to compute Equation 1, we use negative sampling or hierarchical softmax [8].

3.4 Sub2Vec-DBON

In the Sub2Vec-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 sugraph g_i. Note that Sub2Vec-DBON does not explicitly rely on the embeddings of node-ids as in Sub2Vec-DM. As shown in Section 3.3, 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 3.3: the subgraph g_1 and the node sequence $a \rightarrow b \rightarrow c$ generated by random walks. Now in the Sub2Vec-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 3.3.

Formally, given a subgraph g_i, and the θ drawn from g_i, the objective of Sub2Vec-DBON is the following:

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

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

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

Since computing Equation 2 involves summation over all possible sets of co-occuring nodes, we use approximation techniques such as negative sampling [8].

3.5 Algorithm

Our algorithm Sub2Vec works as follows: we first generate the samples of Id-paths/Degree-paths in each subgraph by running random walks. Then we optimize the SV-DBON/ SV-DM objectives using the stochastic gradient descent (SGD) method [15] by leveraging the random walks. We used the Gensim package for implementation [16]. The complete pseudocode is presented in Algorithm 1.

Algorithm 1 `Sub2Vec`

Require: Graph G, subgraph set $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, length of the context window w, dimension d

walkSet = {}
for each g_i in s **do**
    walk = `RandomWalk` (g_i)
    walkSet[g_i] = walk
end for
f = StochasticGradientDescent(walkSet, d, w)
return f

4 Experiments

We leverage `Sub2Vec`[1] for two applications, namely Community Detection and Graph Classification, and perform a case-study on a real subgraph data. 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 [5], we set dimension of the embedding as 128 unless mentioned otherwise for both parameters.

4.1 Community Detection

Setup. Here we show how to leverage `Sub2Vec` for the well-known community detection problem [2, 3]. 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. One expects many nodes within the same community to have similar neighborhoods. Hence, we can use `Sub2Vec` to embed subgraphs while preserving the Neighborhood property and cluster the embeddings to detect communities.

Approach. We propose to use `Sub2Vec` 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 $G(V, E)$. For each node v, we extract its neighborhood C_v only once. Hence, we get a set $\mathcal{S} = \{C_v | v \in V\}$ of $|V|$ neighborhoods are extracted from G. Since each C_v is a subgraph, we can leverage `Sub2Vec` to embed each $C_v \in \mathcal{S}$. The idea is that similar C_vs will be embedded together, which can then be clustered to detect communities. We use a clustering algorithm (K-Means) to cluster the feature vectors $f(C_v)$ of each C_v. For datasets with overlapping communities (like `Youtube`), we use the Neo-Kmeans algorithm [17] 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 (`School` and `Work`) and its 2-hop ego-network for others. The ego-network of a node is the subgraph induced by the node and its neighbors. The 2-hop ego-network is the subgraph induced by the node, its neighbors, and neighbors' neighbors.

[1] Code in Python available at: https://goo.gl/Ef4q8g

Algorithm 2 Community Detection using `Sub2Vec`

Require: A network $G(V, E)$, `Sub2Vec` parameters, k number of communities
1: neighborhoodSet = $\{\}$
2: **for each** v in V **do**
3: neighborhoodSet = neighborhoodSet $\cup$ neighbordhood of v in G.
4: **end for**
5: vecs = `Sub2Vec` (neighborhoodSet, w, d)
6: clusters = Clustering(vecs, k)
7: return clusters

Datasets. We use multiple real world datasets from multiple domains like social-interactions, co-authorship, social networks and so on of varying sizes. See Table 1.

Table 1: Information on Datasets for Community Detection (Left) and Graph Classification (Right). # com denotes the number of ground truth communities in each dataset. # nodes denotes the average number of nodes in each graph classification dataset.

Dataset	$\|V\|$	$\|E\|$	# com	Domain
`Work`	92	757	5	contact
`Cornell`	195	304	5	web
`School`	182	2221	5	contact
`Texas`	187	328	5	web
`Wash.`	230	446	5	web
`Wisc.`	265	530	5	web
`PolBlogs`	1490	16783	2	web
`Youtube`	1.13M	2.97M	5000	social

Dataset	# graphs	# classes	# nodes	# labels
`MUT`	188	2	17.9	7
`PTC`	344	2	25.5	19
`ENZ`	600	6	32.6	3
`PRT`	1113	2	39.1	3
`NC1`	4110	2	29.8	37
`NC109`	4127	2	29.6	38

Baselines. We compare `Sub2Vec` with various traditional community detection algorithms and network embedding based methods. `Newman` [2] is a well-known community detection algorithm based on betweenness. `Louvian` [3] is a popular greedy optimization method. `DeepWalk` and `Node2Vec` are recent network embedding methods which learn feature representations of nodes in the network which we then cluster (in the same way as us) to obtain communities.

Table 2: `Sub2Vec` easily out-performs all baselines in all datasets. Average F-1 score is shown for each method. Winners have been bolded for each dataset. G stands for gain obtained by `Sub2Vec` in percentage.

Method	`Work`	`School`	`PolBlogs`	`Texas`	`Cornell`	`Wash.`	`Wisc.`	`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
`Sub2Vec`-DM	**0.77**	**0.93**	0.85	**0.35**	**0.36**	**0.38**	**0.32**	**0.38**
`Sub2Vec`-DBON	0.65	0.57	0.82	**0.35**	0.34	0.37	**0.32**	0.36
G	**20.3**	**17.7**	**-1.2**	**29.6**	**9.1**	**31.0**	**6.6**	**123.5**

Results. We measure the performance of all the algorithms by computing the Average F1 score [18] against the ground-truth. See Table 2. Both versions of `Sub2Vec` signif-

icantly and consistently outperform all the baselines We achieve a significant gain of *123.5 %* over the closest competitor (Node2Vec) for Youtube. 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 Youtube as these methods output non-overlapping communities. Performance of Node2Vec is satisfactory in sparse networks like Wash. and Texas. Node2Vec does slightly better ($\sim 1\%$) than Sub2Vec in PolBlogs—the network consists of homogeneous neighborhoods, which favors it. However, the performance of Node2Vec is significantly worse for dense networks like Work and School. On the other hand, performance of Sub2Vec is even more impressive in these dense networks (where the task is more challenging).

4.2 Graph Classification

Setup. Here, we show an application of our method in the Graph Classification task [10, 19]. In the graph classification problem, the data consists of multiple (g_i, Y_i) tuples, where each g_i is a graph and Y_i is its class-label. Moreover, the nodes in each graph g_i are labeled. The goal is to predict the class Y_i for a given graph G_i. Since we can generate a discriminative feature representation of each graph (while preserving either Neighborhood or Structural properties), we can train any off-the-shelf classifier to classify the graphs. In this experiment, we set the dimension of embedding as 300 and set the length of the random walk as 100000.

Approach. Our approach is to learn the embedding of each graph by treating them as a subgraph of a union of all the graphs. First we learn the feature representation of the graphs such that either the Neighborhood (Sub2Vec-N) or Structural (Sub2Vec-S) property is preserved. We then leverage four off-the-shelf classifiers: Decision Tree, Random Forest, SVM, and Multi-Layered Perceptron, to solve the classification task.

Datasets. We test on classic graph classification benchmark datasets. All the datasets are publicly available[2]. List of datasets is presented in Table 1.

Baselines. We used two state-of-the-art methods as our competitors. WL-Kernel [10]: This is a graph kernel method based on the Weisfeiler-Lehman test of graph-isomorphism. DG-Kernel is a deep-learning version of WL-kernel [19], which relies on latent representation of sub-structures of the graphs. It uses the popular skip-gram model.

Table 3: Testing accuracy of graph classification. G is the % gain obtained by Sub2Vec.

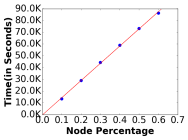
Method	MUT	PTC	ENZ	PRT	NC1	NC109
WL-Kernel	0.80	0.56	0.27	0.72	0.80	0.80
DG-Kernel	0.82	0.60	0.53	0.71	0.80	0.80
Sub2Vec-N	0.74	0.59	**0.89**	0.92	**0.95**	0.90
Sub2Vec-S	**0.85**	**0.62**	0.85	**0.96**	**0.95**	**0.91**
G	**3.7**	**3.3**	**67.9**	**33.3**	**18.8**	**13.8**

[2] http://mlcb.is.tuebingen.mpg.de/Mitarbeiter/Nino/Graphkernels/

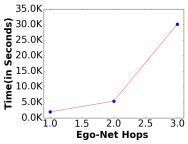
Results. We report the testing accuracy of a 5-fold cross validation. For both Sub2Vec-N and Sub2Vec-S, we run both of our models Sub2Vec-DM and Sub2Vec-DBON. We then train all four classifiers and show the best of them. See Table 3. The results show that both of our methods consistently outperform competitors. The gain of Sub2Vec over the state-of-the-art DG-Kernel is upto a significnt 67.9%. The better performance of Sub2Vec-S over Sub2Vec-N indicates that structural properties in these datasets are more discriminative. This is intuitive as different bonds between the elements results in different structure and also determine the chemical properties of a compound [20]. Interestingly, the Neighborhood property outperforms the Structural property in ENZ dataset. This suggests that, in ENZ dataset, which interestingly also has higher number of classes, the neighborhood property is more important than structural property in determining the graph class.

4.3 Scalability

Here we show the scalability of Sub2Vec with respect to the number and the size of subgraphs. We extract connected subgraphs of Youtube dataset of induced by varying percentage of nodes. We then run Sub2Vec on the set of ego-nets in each resulting network. As shown in Figure (a), Sub2Vec is linear w.r.t number of subgraphs. In Figure (b), we run Sub2Vec 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.



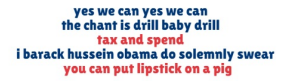
(a) No of Subgraphs



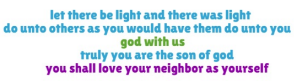
(b) Size of Subgraphs

4.4 Case Studies

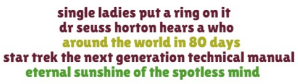
We perform case-studies on MemeTracker[3] dataset to investigate if the embeddings returned by Sub2Vec are interpretable. Here, we run Sub2Vec to preserve the Neighborhood property. The MemeTracker consists of a series of cascades caused by memes spreading on the network of linked web pages. Each meme-cascade induces a subgraph in the underlying network. We first embed these subgraphs in a continuous vector space by leveraging Sub2Vec. We then cluster these vectors to explore what kind of meme cascade-graphs are grouped together and what characteristics of memes determine their similarity and distance to each other. For this case-study, we pick the top 1000 memes by volume, and cluster them into 10 clusters.



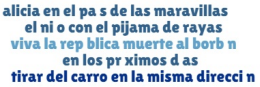
(a) Politics



(b) Religion



(c) Entertainment



(d) Spanish

[3] snap.stanford.edu

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 above. In the entertainment cluster, we find memes which are names of popular songs and movies such as "sweet home alabama", "Madagascar 2" and so on. Similarly, we also find a cluster of religious memes. These memes are quotes from the Bible. In the 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. Interestingly, we find that all the memes in Spanish language were clustered together. This indicates that memes in different language travel though separate websites, which matches with the reality as most webpages use one primary language.

5 Related Work

The network embedding problem, which seeks to generate low dimensional feature representation of nodes, has been well studied. Early work includes [21–23]. However, these methods are slow and do not scale to large networks. Recently, several deep learning based network embeddings algorithms were proposed. DeepWalk [4] and Node2Vec [5] learn feature representation based on contexts generated by random walks. SDNE [6] and LINE [7] learn feature representation of nodes while preserving first and second order proximity. Other recent works include [24, 9, 25]. However, all of them node embeddings, while our goal is to embed subgraphs.

The most similar network embedding literature includes [12, 19, 11]. Risen and Bunke [12] propose to learn vector representations of graphs based on edit distance to a set of pre-defined prototype graphs. Yanardag et al. [19] and Narayanan et al. [11] learn vector representation of the subgraphs using the Word2Vec by generating "corpus" of subgraphs where each subgraph is treated as a word. The above works focuses on some specific subgraphs like graphlets and rooted subgraphs. None of them embed subgraphs with arbitrary structure.

6 Conclusions and Discussion

We focus on the embedding problem for a set of subgraphs by formulating two intuitive properties (Neighborhood and Structural). We developed `Sub2Vec`, a scalable embedding framework which gives interpretable embeddings such that these properties are preserved. We also demonstrate via detailed experiments that `Sub2Vec` outperforms traditional algorithms as well as node-level embedding algorithms in various applications, more so in challenging settings. Extending our framework to handle more properties of interest would be fruitful. For example, one may think of a 'Positional property' which relates to the *position or role* of nodes in the subgraph w.r.t. the overall graph.

Acknowledgements This paper is based on work partially supported by the NSF (CAREER-IIS-1750407, DGE-1545362, and IIS-1633363), the NEH (HG-229283-15), ORNL (Task Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift

References

1. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social network data analytics. Springer (2011) 115–148
2. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12) (2002) 7821–7826
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment (2008)
4. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD, ACM (2014) 701–710
5. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD, ACM (2016) 855–864
6. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: SIGKDD, ACM (2016) 1225–1234
7. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, ACM (2015) 1067–1077
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. (2013) 3111–3119
9. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: KDD 2017, ACM (2017) 385–394
10. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.v., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(Sep) (2011) 2539–2561
11. Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., Saminathan, S.: subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. arXiv preprint arXiv:1606.08928 (2016)
12. Riesen, K., Bunke, H.: Graph classification and clustering based on vector space embedding. World Scientific Publishing Co., Inc. (2010)
13. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. Volume 14. (2014) 1188–1196
14. Milojević, S.: Power law distributions in information science: Making the case for logarithmic binning. Journal of the American Society for Information Science and Technology (2010)
15. Bousquet, O., Bottou, L.: The tradeoffs of large scale learning. In: NIPS. (2008) 161–168
16. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: LREC 2010 Workshop on New Challenges for NLP Frameworks, Citeseer (2010)
17. Whang, J.J., Dhillon, I.S., Gleich, D.F.: Non-exhaustive, overlapping k-means. In: SDM, SIAM (2015) 936–944
18. Yang, J., Leskovec, J.: Overlapping community detection at scale: a nonnegative matrix factorization approach. In: WSDM, ACM (2013) 587–596
19. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: SIGKDD, ACM (2015) 1365–1374
20. Carey, F.A., Sundberg, R.J.: Advanced Organic Chemistry: Part A: Structure and Mechanisms. Springer Science & Business Media (2007)
21. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: NIPS. Volume 14. (2001) 585–591
22. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500) (2000) 2319–2323
23. Bach, F.R., Jordan, M.I.: Learning spectral clustering. In: NIPS. Volume 16. (2003)
24. Cheng, K., Li, J., Liu, H.: Unsupervised feature selection in signed social networks. In: KDD 2017, ACM (2017) 777–786
25. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. (2017) 203–209