Multi-scale Attributed Embedding of Networks

Benedek Rozemberczki, Carl Allen and Rik Sarkar
School of Informatics, The University of Edinburgh
{benedek.rozemberczki, carl.allen}@ed.ac.uk,{rsarkar}@inf.ed.ac.uk

***Abstract*—We present algorithms that embed networks into Euclidean feature spaces based on attributes in the neighborhoods of nodes. The algorithms operate in the Skip-gram style, by observing nearby contexts in random walks and then optimizing node representations to align with the observed attributes. Observations from neighborhoods of different sizes are incorporated in our multi-scale approach. Theoretical analysis shows that this approach can be seen as the implicit approximate factorization of a matrix that contains connectivity and attribute information. The attribute-neighborhood perspective over multiple scales allows diverse applications, including latent feature identification across disconnected networks with similar attributes. Experimental results show that the algorithm performs accurately on common social networks, is robust and computationally efficient.**

***Index Terms*—Node embedding, feature extraction.**

I. Introduction

Network embedding is a fundamental technique in network analysis. Representing networks in Euclidean spaces enables various machine learning and optimisation tasks. Thus, network embeddings have been used for community detection, visualization, link prediction and other applications [1], [2], [3]. A popular approach for embedding applies the *skip-gram* style [4] of representation learning on the network link structure. These methods first extract sequences of neighboring nodes using random walks, then treat the probability of proximity of node pairs in these sequences as their mutual similarity. Deepwalk [1], Node2Vec [2], LINE [3] are popular methods that take this approach.

A different approach is to use attributes associated with the nodes. Depending on the network, these attributes can represent their interests, habits, history, and various preferences. Attributes are likely to be closely related to the link structure according to the principle of homophily – that those with similar attributes are likely to be connected, and vice versa. Attributed network embedding methods [5], [6], [7] have leveraged this idea to supplement the link structure information with attribute information. Attributed embeddings find applications in recommender systems, node classification, link prediction, and other problems [8], [9], [10].

In this paper, we take a multi-scale view of attributes. A node's neighborhood can be observed at different scales like classmates, city and country. This idea is shown in Figure 1(a) where nodes u and v share similar neighbors at all scales, but the neighborhoods of w are different.

We can take a more general view by considering the distribution of attributes observed at different scales. Let us say, the attributes of nodes are represented by their shades as in Figure 1. Thus, node x appears in a similar multi-scale

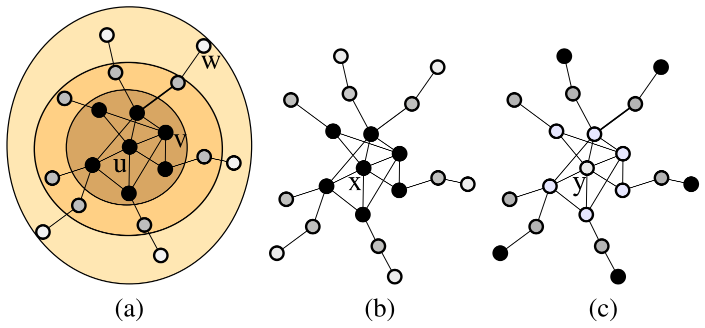

Fig. 1: Attributes at multiple scales. Shades represent attributes. (a) u, v have similar neighborhoods at all scales, w is different. (b) x is in a different network, but can be considered similar to u. (c) y is in a different context due to different distribution of attributes.

attribute context as nodes u and v, while y is in a very different context because the distribution of attributes at different neighborhoods is different. By using general attributes at multiple scales – instead of only node ids as in [11] – we enable a greater variety of possibilities. With this approach, nodes u and x can be inferred to be in a similar "situation" even if they are far apart in the network, or in different networks. The approach thus generalizes existing methods. We also present theoretical analysis of this approach and its properties.

For theoretical understanding, network embedding can be abstractly considered analogous to matrix factorization, where the matrix relates nodes to certain *contexts*. Attributes are natural examples of such contexts, and attributed embedding methods (e.g. [8]) directly factorize a matrix of link and attribute information to embed similar nodes close to each other. From the viewpoint of link structure, neighborhoods can be treated as contexts and nodes structurally close to each other should be embedded close. The factorization view also applies to embeddings using skip-gram optimization [12]; correspondingly, sequence based network embedding methods [1], [2], [3], [13] are shown to implicitly approximate the factorization of a pointwise mutual information (PMI) matrix of node-context pairs [13].

In this paper, we present the first formal description of a skip-gram style embedding algorithm that uses attribute distributions at multiple scales. And we present theoretical analysis of the multi-scale embedding in the matrix factorization perspective. Specifically:

1) We introduce scalable pooled attributed embedding (*AE*)

and its multi-scale version (*MUSAE*) – which are able to encode the presence of generic vertex features in higher order neighbourhoods of nodes.
2) The embedding tasks are formulated as *Skip-Gram* like optimization problems using node-feature pairs generated from first and second-order random walks with attributes.
3) Theoretical proofs show that the algorithm with first-order random walk sampling is equivalent to the approximate implicit factorization of a non-linearly transformed graph adjacency matrix power and feature matrix product.
4) We show that popular feature learning methods *DeepWalk* [1] and *Walklets* [11] are just special cases of our proposed procedures.
5) Experimental results show that *AE* and *MUSAE* on regression problems using real-world Wikipedia page-page networks are accurate compared to existing methods, robust to changes in hyperparameters, and computationally scalable.

Reference implementations of *AE* and *MUSAE* with the Wikipedia networks collected specifically for the paper are available at https://github.com/benedekrozemberczki/MUSAE.

The remainder of the paper is structured as follows. Related works are summarized in Section II. While Section III introduces the pooled and multi-scale attributed node embedding algorithms. In Section IV, we theoretically analyse properties of the algorithms. Experimental evaluation is presented in Section V. The paper concludes in Section VI with a discussion of findings.

II. Related Work

Efficient unsupervised extraction of node features in large networks has seen unprecedented development in recent years. The current paradigm focuses on learning latent space representations of nodes such that those that share neighbours [1], [3], structural roles [14], [15] or attributes are located close together in the embedding space. Our work belongs to this line of research as our goal is to learn similar latent representations for nodes with similar sets of features in their neighbourhoods. Further, we obtain both multi-scale or pooled representations of nodes and features.

Neighbourhood preserving node embedding procedures place nodes with common first, second and higher order neighbours within close proximity in the embedding space. Recent works in the neighbourhood preserving node embedding literature were inspired by the *Skip-Gram* model [16], [4], which generates word embeddings by implicitly factorizing the shifted pointwise mutual information (PMI) matrix [12] obtained from a corpus. This procedure inspired *DeepWalk* [1], a method which generates truncated random walks on a graph to obtain a "corpus" from which the *Skip-Gram* model generates neighbourhood preserving node embeddings. In so doing, *DeepWalk* implicitly factorizes a PMI matrix, which can be shown, based on the underlying first-order Markov process, to correspond to the mean of a set of normalized adjacency matrix powers up to a given order [13]. This pooling of matrices can be suboptimal since it treats neighbours over increasing path lengths (or scales) equally or according to fixed weightings [16], [2], whereas it has been found that an optimal weighting may be task or dataset specific [17]. In contrast, multi-scale node embedding methods such as *LINE* [3], *GraRep* [18] and *Walklets* [11] learn lower-dimensional embeddings for a node from each adjacency matrix power separately and concatenate them to form the final node representation. These unblended representations, having less but *distinct* information at each scale, are known to serve as higher quality unsupervised embeddings for downstream use in a number of settings without increasing the number of free parameters [11].

Attributed node embedding procedures refine ideas from neighbourhood based node embeddings to also incorporate node *features* (equivalently *attributes* or *labels*) [5], [7], [6], [8], [9]. Similarities between both neighbourhood structure and node features contribute to the pairwise node proximity in the embedding space. These models follow quite different strategies to obtain such representations. The most elemental procedure, *TADW* [5], decomposes a convex combination of normalized adjacency matrix powers into a matrix product that includes the feature matrix. A number of models such as *SINE* [10] and *ASNE* [7] implicitly factorize a matrix formed by concatenating the feature and adjacency matrices. Other approaches such as *TENE* [9], formulate the attributed node embedding task as a joint non-negative matrix factorization problem in which node representations obtained from sub-tasks are used to regularize one another. A similar network structure based regularization approach is used in *AANE* [6] where a feature similarity matrix of nodes is decomposed using the alternating direction method of multipliers. The method most similar to our own is *BANE* [8], in which the product of a normalized adjacency matrix power and a feature matrix is explicitly factorized to obtain attributed node representations. Many other attributed node embedding methods exist, but they do not consider the attributes of higher order neighbourhoods [5], [7], [6], [10], [9].

The relationship between our proposed pooled (*AE*) and multi-scale (*MUSAE*) attributed node embedding methods resembles that between graph convolutional neural networks (*GCN*) [19] and multi-scale graph convolutional neural networks (*N-GCN*) [20]. The standard *GCN* layer creates latent space node representations that pool node attributes from arbitrary order neighbourhoods. In contrast, the *N-GCN* architecture creates groups of latent features for each proximity.

III. Attributed Embedding Models

In this section we set up an optimization problem to learn node embeddings using the attributes of nearby nodes that allows embeddings of nodes and features to be learned jointly. The aim is to learn similar latent representations for nodes that occur in neighbourhoods of similar features; and similar representations for features that often occur in similar neighbourhoods of nodes. Let $G = (V, E)$ be an undirected graph of interest where V and E are the sets of vertices and edges respectively; and let F be the set of all possible node features. We define $F_v \subseteq F$ as the subset of features belonging to each node $v \in V$. An embedding of nodes is a mapping $g : V \rightarrow \mathbb{R}^d$ that assigns a d dimensional representation $g(v)$ (or simply g_v) to each node v and is fully described by a matrix $\mathbf{G} \in \mathbb{R}^{|V|\times d}$. Similarly, an embedding of the features (to the same latent space) is a mapping $h : F \rightarrow \mathbb{R}^d$ with embeddings denoted $h(f)$ (or simply h_f), and is fully described by a matrix $\mathbf{H} \in \mathbb{R}^{|F|\times d}$.

A. Attributed Embedding

The *Attributed Embedding* procedure is described by Algorithm 1. We sample n nodes w_1, from which to start attributed random walks on G, with probability proportional to their degree (Line 2). From each starting node, a node sequence of length l is sampled over G (Line 3), where sampling can follow a first or second order random walk [2]. For a given window size t, we iterate over each of the first $l-t$ nodes of the sequence termed *source* nodes w_j (Line 4). For each source node, we consider the following t nodes as *target* nodes (Line 5). For each target node w_{j+r}, we add the tuple (w_j, f) to the corpus $\mathcal{D}$ for each target feature $f \in F_{w_{j+r}}$ (Lines 6 and 7). We also consider features of the source node $f \in F_{w_j}$, adding each (w_{j+r}, f) tuple to $\mathcal{D}$ (Lines 9 and 10). Finally, *Skip-Gram* with negative sampling (SGNS) is run on the generated corpus of tuples with b negative samples (Line 15) to obtain d dimensional node and feature embeddings.

```
   Data: $G = (V, E)$ – Graph to be embedded.
         $\{F_v\}_V$ – Set of node feature sets.
         $n$ – Number of sequence samples.
         $l$ – Length of sequences.
         $t$ – Context size.
         $d$ – Embedding dimension.
         $b$ – Number of negative samples.
   Result: Node embedding $g$ and feature embedding $h$.
1  for $i$ in $1 : n$ do
2      Pick $w_1 \in V$ according to $P(w_1) \sim deg(w_1)/\text{vol}(G)$.
3      $(w_1, w_2, \ldots, w_l) \leftarrow$ Sample Nodes$(G, w_1, l)$
4      for $j$ in $1 : l - t$ do
5          for $r$ in $1 : t$ do
6              for $f$ in $F_{w_{j+r}}$ do
7                  Add tuple $(w_j, f)$ to multiset $\mathcal{D}$.
8              end
9              for $f$ in $F_{w_j}$ do
10                 Add tuple $(w_{j+r}, f)$ to multiset $\mathcal{D}$.
11             end
12         end
13     end
14 end
15 Run SGNS on $\mathcal{D}$ with $b$ negative samples and $d$ dimensions.
16 Output $g_v,\ \forall v \in V$, and $h_f,\ \forall f \in F = \cup_V F_v$.
```

Algorithm 1: AE sampling and training procedure

B. Multi-scale Attributed Embedding

The proposed *AE* method (Algorithm 1) does not create a multi-scale node embedding since feature sets in neighborhoods at first, second and higher order proximities are not encoded by separate representations. Inspired by multi-scale (unattributed) node embeddings we propose *multi-scale attributed node embeddings* (*MUSAE*). We adapt the previous sampling and optimization problem to accommodate multi-scale representations such that the representation vector for a node $v \in V$ for a specific proximity $r \in \{1, \ldots, t\}$ is now given by a mapping $g^r : G \rightarrow \mathbb{R}^{d/t}$. Similarly, the representation vector of a feature $f \in F$ at the same proximity r is given by a mapping $h^r : F \rightarrow \mathbb{R}^{d/t}$. By simple concatenation we arrive back at d dimensional embeddings for each node and feature.

```
   Data: $\mathcal{G} = (V, E)$ – Graph to be embedded.
         $\{F_v\}_V$ – Set of node feature sets.
         $n$ – Number of sequence samples.
         $l$ – Length of sequences.
         $t$ – Context size.
         $d$ – Embedding dimension.
         $b$ – Number of negative samples.
   Result: Node embeddings $g^r$ and feature embeddings $h^r$ for
           $r = 1, \ldots, t$.
1  for $i$ in $1 : n$ do
2      Pick $w_1 \in V$ according to $P(w_1) \sim deg(w_1)/\text{vol}(G)$.
3      $(w_1, w_2, \ldots, w_l) \leftarrow$ Sample Nodes$(G, w_1, l)$
4      for $j$ in $1 : l - t$ do
5          for $r$ in $1 : t$ do
6              for $f$ in $F_{w_{j+r}}$ do
7                  Add the tuple $(w_j, f)$ to multiset $\mathcal{D}_{\underset{r}{\rightarrow}}$.
8              end
9              for $f$ in $F_{w_j}$ do
10                 Add the tuple $(w_{j+r}, f)$ to multiset $\mathcal{D}_{\underset{r}{\leftarrow}}$.
11             end
12         end
13     end
14 end
15 for $r$ in $1 : t$ do
16     Create $\mathcal{D}_r$ by unification of $\mathcal{D}_{\underset{r}{\rightarrow}}$ and $\mathcal{D}_{\underset{r}{\leftarrow}}$.
17     Run SGNS on $\mathcal{D}_r$ with $b$ negative samples and $\frac{d}{t}$ dimensions.
18     Output $g^r_v,\ \forall v \in V$, and $h^r_f,\ \forall f \in F = \cup_V F_v$.
19 end
```

Algorithm 2: MUSAE sampling and training procedure

The *Multi-Scale Attributed Embedding* procedure is described by Algorithm 2. We again sample n starting nodes w_1 with a probability proportional to node degree (Line 2) and, for each, sample a node sequence of length l over G (Line 3) according to either a first or second order random walk. For a given window size t, we iterate over the first $l - t$ (source) nodes w_j of the sequence (Line 4) and for each source node we iterate through the t (target) nodes w_{j+r} that follow (Line 5). We again consider each target node feature $f \in F_{w_{j+r}}$, but now add tuples (w_j, f) to a *sub-corpus* $\mathcal{D}_{\underset{r}{\rightarrow}}$ (Lines 6 and 7). We add tuples (w_{j+r}, f) to another sub-corpus $\mathcal{D}_{\underset{r}{\leftarrow}}$ for each source node feature $f \in F_{w_j}$ (Lines 9 and 10). Lastly, we run SGNS on each sub-corpus $\mathcal{D}_r = \mathcal{D}_{\underset{r}{\rightarrow}} \cup \mathcal{D}_{\underset{r}{\leftarrow}}$ with b negative samples (Line 16) to obtain d/t dimensional node and feature embeddings for $r = 1, \ldots, t$.

IV. Analysis of Multi-scale attributed embedding as matrix factorization

It has been shown previously that the SGNS loss function is minimized when the embedding matrices approximately factorize a matrix of pointwise mutual information (PMI) derived from pairwise corpus statistics [12]. In particular, for a dictionary of words $\mathcal{D}$, SGNS with b negative samples outputs two embedding matrices $\mathbf{W}, \mathbf{C} \in \mathbb{R}^{d\times n}$ such that $\forall w, c \in \mathcal{D}$:

$$\mathbf{W}_w^\top \mathbf{C}_c \approx \log\left(\tfrac{\#(w,c)|\mathcal{D}|}{\#(w)\#(c)}\right) - \log b \ ,$$

where $\#(w,c), \#(w), \#(c)$ denote counts of word-context pair (w,c), word w and context c; and $\mathbf{W}_w, \mathbf{C}_c \in \mathbb{R}^d$ are columns of $\mathbf{W}$ and $\mathbf{C}$ corresponding to w and c respectively (word embeddings). Considering $\frac{\#(w)}{|\mathcal{D}|}$, $\frac{\#(c)}{|\mathcal{D}|}$, $\frac{\#(w,c)}{|\mathcal{D}|}$ as empirical estimates of $p(w)$, $p(c)$ and $p(w,c)$ respectively shows:

$$\mathbf{W}^\top \mathbf{C} \approx [\text{PMI}(w,c) - \log b]_{w,c} \ ,$$

an implicit matrix factorization where the approximation is due to the low rank of the embedding matrices $\mathbf{W}$ and $\mathbf{C}$.

This result has been extended to node embedding models that apply SGNS to a "corpus" generated by random walks over graphs [13]. In particular, the corpus for *DeepWalk* is derived from a first-order Markov process, which allows joint probability distributions over nodes at different stages of a random walk to be expressed in closed form. A closed form for the factorized PMI matrix, a (weighted) mean of pairwise joint probabilities over random walks, then follows. In a similar manner, we show that our proposed algorithms *AE* adn *MUSAE* can be viewed as implicit matrix factorization. We first consider the matrices factorized in the multi-scale case and then the pooled case which immediately follows.

Notation: Throughout, $\mathbf{A}$ denotes the adjacency matrix and $\mathbf{D}$ the diagonal degree matrix of G, where $\mathbf{D}_{w,w} = \deg(w) = \sum_v \mathbf{A}_{w,v}$. We denote the *volume* of G by $c = \sum_{v,w} \mathbf{A}_{v,w}$. We define the binary attribute matrix $\mathbf{F} \in \{0,1\}^{|V|\times|F|}$ as: $\mathbf{F}_{w,f} = 1 \ \ \textit{if} \ \ f \in F_w, \ \forall w \in V, f \in F$.

Assuming ergodicity of G, we have that $p(w) = \frac{deg(w)}{c}$ is the stationary distribution over nodes $w \in V$, i.e. $\frac{1}{c}\mathbf{D} = diag(p(w))$, where $diag$ indicates a diagonal matrix. Similarly $\frac{1}{c}\mathbf{A}$ corresponds to the stationary joint distribution over consecutive nodes $p(w_j, w_{j+1})$. $\mathbf{F}_{w,f}$ can be considered a Bernoulli parameter describing the probability of observing feature f at a particular node w, $p(f|w)$; and thus $\frac{1}{c}\mathbf{DF}$ as the stationary joint distribution over nodes and features $p(f, w_j)$. To ease notation, we introduce $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, the matrix of conditional distributions $p(w_{j+1}|w_j)$; and $\mathbf{E} = diag(\underline{1}^\top \mathbf{DF})$, a diagonal matrix proportional to the probability of observing each feature at the stationary distribution $p(f)$ (note that $p(f)$ need not sum to 1, whereas $p(w)$ necessarily must).

A. Multi-scale case (MUSAE)

Applying the above results to *MUSAE*, we know that the *Skip-Gram* loss function (Algorithm 2, Line 17) is minimized when the learned embeddings g_v^r, h_f^r satisfy:

$$g_w^{r\,\top} h_f^r \approx \log\left(\tfrac{\#(w,f)_r|\mathcal{D}_r|}{\#(w)_r\#(f)_r}\right) - \log b \quad \forall w \in V, f \in F.$$

Our aim is to express this factorization explicitly in terms of known properties of the graph G and its features.

Lemma 1. *The empirical statistics of node-feature pairs obtained from random walks give unbiased estimates of joint probabilities of observing feature $f \in F$ r steps (i) after; or (ii) before node $v \in V$, as given by:*

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \tfrac{1}{c}(\boldsymbol{D}\boldsymbol{P}^r\boldsymbol{F})_{w,f}$$

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} = \tfrac{1}{c}(\boldsymbol{F}^\top\boldsymbol{D}\boldsymbol{P}^r)_{f,w}$$

Proof. See Appendix. □

Lemma 2. *The empirical statistics of node-feature pairs obtained from random walks give unbiased estimates of joint probabilities of observing feature $f \in F$ r steps either side of node $v \in V$, as given by:*

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_r}{|\mathcal{D}_r|} = \tfrac{1}{c}(\boldsymbol{D}\boldsymbol{P}^r\boldsymbol{F})_{w,f} \ , \tag{1}$$

Proof. See Appendix. □

Marginalizing provides unbiased estimates of the probability of nodes and features at the stationary distribution:

$$\underset{l\to\infty}{\text{plim}}\frac{\#(w)}{|\mathcal{D}_r|} = \frac{deg(w)}{c} = \tfrac{1}{c}\mathbf{D}_{w,w}$$

$$\underset{l\to\infty}{\text{plim}}\frac{\#(f)}{|\mathcal{D}_r|} = \sum_{w|f\in F_w}\frac{deg(w)}{c} = \tfrac{1}{c}\mathbf{E}_{f,f}$$

Theorem 1. *The MUSAE algorithm learns embeddings for $r = 1,\ldots,t$ that approximately factorize the node-feature PMI matrix:*

$$log\left(c\boldsymbol{P}^r\boldsymbol{F}\boldsymbol{E}^{-1}\right) - \log b \ .$$

Proof.

$$\begin{aligned}\tfrac{\#(w,f)_r|\mathcal{D}_r|}{\#(f)_r\#(w)_r} &= \left(\tfrac{\#(w,f)_r}{|\mathcal{D}_r|}\right)/\left(\tfrac{\#(f)_r}{|\mathcal{D}_r|}\tfrac{\#(w)_r}{|\mathcal{D}_r|}\right)\\ &\xrightarrow{p} \left((c\mathbf{D}^{-1})(\tfrac{1}{c}\mathbf{D}\mathbf{P}^r\mathbf{F})(c\mathbf{E}^{-1})\right)_{w,f}\\ &= c(\mathbf{P}^r\mathbf{F}\mathbf{E}^{-1})_{w,f}\end{aligned}$$ □

B. Pooled case (AE)

Lemma 3. *The empirical statistics of node-feature pairs learned by the AE algorithm give unbiased estimates of mean joint probabilities over different path lengths as follows:*

$$\underset{l\to\infty}{plim}\frac{\#(w,f)}{|\mathcal{D}|} = \tfrac{c}{t}\Big(\boldsymbol{D}\big(\sum_{r=1}^{t}\boldsymbol{P}^r\big)\boldsymbol{F}\Big)_{w,f} \tag{2}$$

Proof. By construction, $\mathcal{D} = \sum_r \mathcal{D}_r$, $\#(w,f) = \sum_r \#(w,f)_r$, $|\mathcal{D}_r| = |\mathcal{D}_s| \ \forall \ r,s \in \{1,\ldots,t\}$ and $|\mathcal{D}| = t|\mathcal{D}_s|$. Combining these with Lemma 2, the result follows. □

Theorem 2. *The AE algorithm approximately factorizes the following expression:*

$$\log\left(\tfrac{c}{t}(\sum_{r=1}^{t}\boldsymbol{P}^r)\boldsymbol{F}\boldsymbol{E}^{-1}\right)-\log b\ .$$

Proof. The proof is analogous to that of Theorem 1. □

Remark 1. *DeepWalk is a corner case of AE.*

DeepWalk can be considered a case of AE in which the feature matrix $\mathbf{F}=\mathbf{I}$, the $|V|\times|V|$ identity matrix, i.e. each node has a single unique feature. It follows that $\mathbf{E}=diag(\underline{1}^\top\mathbf{D}\mathbf{I})=\mathbf{D}$ and by Theorem 2 the factorized matrix is:

$$\log\left(\tfrac{c}{t}(\sum_{r=1}^{t}\mathbf{P}^r)\mathbf{D}^{-1}\right)-\log b$$

as observed previously [13].

Remark 2. *Walklets is a corner case of MUSAE.*

The feature matrix $\mathbf{F}$ of *Walklets* is a $|V|\times|V|$ identity matrix as each node has a feature set which contains a unique feature specific to the node. From this it follows that for $r=1,\ldots,t$:

$$\log\left(c\,\mathbf{P}^r\mathbf{D}^{-1}\right)-\log b$$

C. Complexity analysis

Creating linear runtime attributed algorithms that are able to integrate node features from higher order neighbourhoods is unprecedented in the node embedding literature [8]. Under the assumption of a constant number of features per source node and first-order attributed random walk sampling, the corpus generation has a runtime complexity of $\mathcal{O}(n\,l\,t\,x/y)$, where $x=\sum_{v\in V}|F_v|$ the total number of features across all nodes (including repetition) and $y=|V|$ the number of nodes. Using negative sampling, the optimization runtime of a single asynchronous gradient descent epoch on *AE* and the joint optimization runtime of *MUSAE* embeddings is described by $\mathcal{O}(b\,d\,n\,l\,t\,x/y)$. If one does p truncated walks from each source node, the corpus generation complexity is $\mathcal{O}(p\,y\,l\,t\,x)$ and the model optimization runtime is $\mathcal{O}(b\,d\,p\,y\,l\,t\,x)$. Our later runtime experiments in Section V will underpin optimization runtime complexity discussed above.

Corpus generation has a memory complexity of $\mathcal{O}(n\,l\,t\,x/y)$ while the same when generating p truncated walks per node has a memory complexity of $\mathcal{O}(p\,y\,l\,t\,x)$. Storing the parameters of an *AE* embedding has a memory complexity of $\mathcal{O}(y\,d)$ and *MUSAE* embeddings also use $\mathcal{O}(y\,d)$ memory. It is worth noting that with online corpus generation, the memory complexity of models is described by the memory complexity of the parameter storage.

V. Experimental Evaluation

We first introduce the attributed networks (datasets) used to benchmark the predictive performance of *AE* and *MUSAE* embeddigns. We compare against various state-of-the-art high performance neighbourhood preserving ([1], [3], [2], [11]) and attributed ([6], [7], [8]) node embedding methods. We refer to models with first-order random walk sampling as AE and MUSAE [1], while models with second-order random walk sampling [2] are referred to as AE_2 and $MUSAE_2$ We test the robustness of our findings to hyper-parameters. Finally, we investigate how changes in the input size affect the runtime.

TABLE I: Statistics of the Wikipedia page-page networks.

	Chameleon	**Crocodile**	**Squirrel**
$\lvert V\rvert$	2,277	11,631	5,201
$\lvert E\rvert$	31,421	170,918	198,493
$\lvert F\rvert$	2,325	10,935	2,089
$\frac{\sum_{v\in V}\lvert F_v\rvert}{\lvert V\rvert}$	12.805	64.895	17.973
Diameter	11	11	10
Density	0.012	0.003	0.015
Transitivity	0.314	0.026	0.348

A. Wikipedia datasets

The datasets used to evaluate our embedding methods are wikipedia page-page networks collected on three specific topics: chameleons, crocodiles and squirrels. In these networks nodes are articles from the English Wikipedia collected in December 2018, edges are mutual links that exist between pairs of sites. Node features describe the presence of nouns appearing in the articles. For each node we also have the average monthly traffic between October 2017 and November 2018. Table I shows that the networks are heterogeneous in terms of size, density, and clustering.

B. Evaluation Task and Experimental Setup

To demonstrate the predictive ability of the *AE* and *MUSAE* embeddings, we perform a regression task on the evaluation datasets. The task is to predict the log monthly average traffic with an elastic net model that takes the node embeddings as input. All downstream models use mixing parameter $\alpha=0.5$ and regularization coefficient $\lambda=0.01$.

Our embedding models are trained using asynchronous gradient descent with an initial learning rate of 0.025 linearly annealed to zero. Each model is trained for 10 epochs. Random walk sampling methods generated 10 truncated random walks per source node with length 80 and used a symmetric window size of $t=3$. Every test R^2 and standard error is calculated from seeded train-test set splits. The *in-out* and *return* parameters of second-order random walks [2] for *Node2Vec*, AE_2 and $MUSAE_2$ are selected with 5-fold cross validation and grid-search, using the training data from the $\{2^2;2^1;2;2^{-1};2^{-2}\}$ set. To ensure fair comparison, we set the dimension of all embedding models to $d=128$, such that the number of free parameters is constant. For each baseline we use hyper-parameters proposed by the respective author.

C. Benchmark Embedding Methods

We compare *AE* and *MUSAE* to the following neighbourhood preserving and attributed node embedding procedures:

- **Doc2Vec** [16], [4]: implicitly factorizes a node-feature matrix.

- **DeepWalk** [1]: implicitly factorizes a weighted mean of normalized adjacency matrix powers.
- **LINE$_2$** [3]: encodes first and second order graph proximity in separate embedding dimensions.
- **Node2Vec** [2]: decomposes a node pair co-occurrence matrix from second-order random walks, encoding both structural and neighbourhood node information.
- **Walklets** [11]: higher order multi-scale neighbourhood preserving embedding (we use window size $t = 4$ and dimension $d/t = 32$ for each proximity r).
- **TADW** [5]: decomposes a weighted sum of normalized adjacency powers as a feature matrix and basis matrices.
- **AANE** [6]: decomposes a similarity matrix of node features with graph-based regularization.
- **ASNE** [7]: implicitly factorizes a concatenation of the adjacency and feature matrices.
- **BANE** [8]: binary factorization of an arbitrary order adjacency matrix power and feature matrix product.
- **TENE** [9]: joint non-negative matrix factorization of the adjacency and feature matrices.

TABLE II: Average test R^2 values and standard errors on the Wikipedia traffic prediction tasks. Bold numbers denote the best results on each page-page network.

	Chameleon	**Crocodile**	**Squirrel**
Doc2Vec	0.506 $\pm$0.002	0.640 $\pm$0.001	0.152 $\pm$0.002
DeepWalk	0.375 $\pm$0.004	0.553 $\pm$0.001	0.170 $\pm$0.002
LINE$_2$	0.381 $\pm$0.003	0.586 $\pm$0.001	0.232 $\pm$0.002
Node2Vec	0.414 $\pm$0.003	0.574 $\pm$0.001	0.174 $\pm$0.002
Walklets	0.426 $\pm$0.003	0.625 $\pm$0.001	0.249 $\pm$0.002
TADW	0.527 $\pm$0.003	0.636 $\pm$0.001	0.271 $\pm$0.002
AANE	0.458 $\pm$0.003	0.687 $\pm$0.001	0.287 $\pm$0.002
ASNE	0.440 $\pm$0.003	0.572 $\pm$0.001	0.228 $\pm$0.002
BANE	0.464 $\pm$0.003	0.617 $\pm$0.001	0.168 $\pm$0.002
TENE	0.494 $\pm$0.008	0.701 $\pm$0.001	0.228 $\pm$0.01
AE	0.550 $\pm$0.002	0.673 $\pm$0.001	0.275 $\pm$0.002
AE$_2$	0.559 $\pm$0.003	0.678 $\pm$0.002	0.282 $\pm$0.002
MUSAE	0.571 $\pm$0.003	**0.715** $\pm$0.001	**0.316** $\pm$0.003
MUSAE$_2$	**0.586** $\pm$0.003	**0.718** $\pm$0.001	**0.317** $\pm$0.002

D. Results

The results of our evaluation are summarized in Table II. We report average test R^2 and standard error of the predictive performance over 100 train-test splits. Our key observation are: (i) that *MUSAE* outperforms all benchmark neighbourhood preserving and attributed node embedding methods, with the strongest *MUSAE* variant outperforming the best baseline by between 2.4% and 11.2% (test R^2); (ii) that attributed node embedding methods *TENE*, *TADW* and *AANE* are competitive with *AE* on certain datasets (*Crocodiles* and *Squirrels*); (iii) that vanilla *MUSAE* significantly outperforms *AE* by between 3.8% and 18.5% (test R^2); and (iv) the benefit of the multi-scale approach for models using second-order random walks can be as large as 12.4%. In particular, these findings empirically support the intuition that learning weights over different scales, is preferable to pooling information, essentially presupposing the weights. The results also demonstrate that using second-order random walks can improve performance of embeddings, but appears dataset specific (here only on the *Chameleons* dataset).

E. Sensitivity analysis

We now test the robustness of our proposed methods to hyperparameter changes. Specifically, we investigate how changes in dimension, epoch number, window size and truncated random walk length affect the predictive performance of embeddings. We evaluate the performance of *AE* and *MUSAE* variants on the *Chameleons* dataset with the same train-test split ratio. Each data point in the plots of Figure 2 represents the average test R^2 over 100 seeded experimental runs. We use the experimental settings described in Subsection V-B except for dimension (referred to as *combined dimension* to emphasize comparability across pooled and multi-scale methods). We use $d = 96$ in order to vary the window size without affecting the number of free parameters.

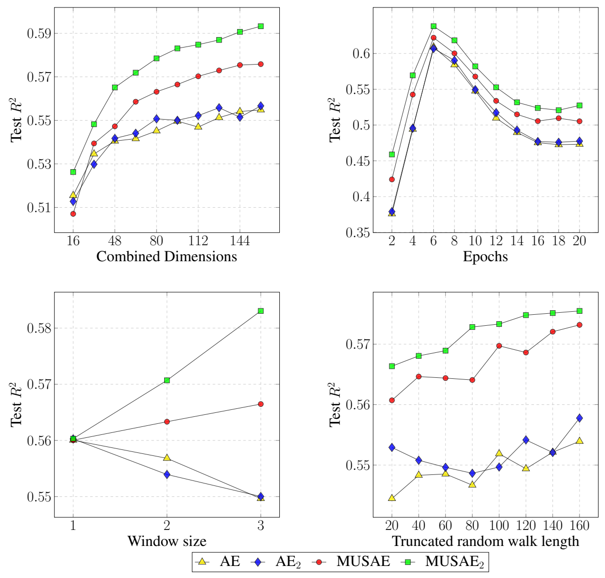

Fig. 2: Predictive performance sensitivity to hyperparameters.

Figure 2 (top left) shows that increasing the combined dimension improves embedding performance. More interestingly, we see that multi-scale embeddings improve at an increased rate as dimension increases, i.e. whereas the pooled algorithms appear to level out, the multi-scale versions show continual gains. We also see (top right) that the *AE* and *MUSAE* models overfit after 6 epochs (on the *Chameleons* dataset) and earlier stopping significantly improves on the performance reported in Table II. Varying the window size (bottom left) shows two empirical regularities: (i) with higher window-sizes, the pooled models fail to encode the higher order information usefully resulting in lower predictive performance; and (ii) multi-scale models benefit materially from a higher window size. Finally, increasing the truncated random walk length (bottom right) tends to improve performance of embeddings, which may be because samples become more independent over longer random walks, in line with our theoretical analysis (Section IV).

F. Scalability

In order to show the efficacy of our proposed algorithms we run a series of experiments on synthetic graphs where we are able to manipulate the input size directly. Specifically, we look at the effect of changing the number of vertices and the number of features per vertex. Our detailed experimental setup was as follows. Each point in Figure 3 is the mean runtime obtained from 100 experimental runs on Erdos-Renyi graphs. The base graph that we manipulated had 2^{11} nodes, 2^3 edges per node and 2^3 unique features per node uniformly selected from a feature set of 2^{11}. Our experimental settings were the same as the ones described in Subsection V-B except for the number of epochs. We only did a single training epoch with asynchronous gradient descent on each graph. We tested the runtime with 1,2 and 4 cores and included a dashed line as the linear runtime reference in each subfigure.

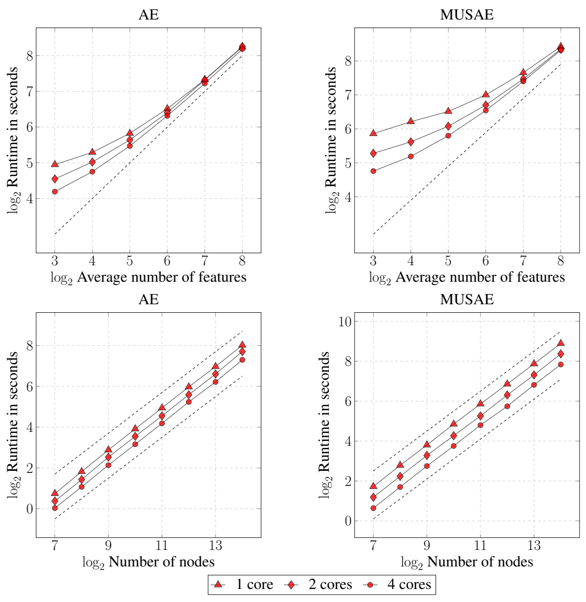

Fig. 3: Optimization time as a function of average feature count / number of vertices.

We observe that doubling the average number of features per vertex doubles the runtime of the AE and MUSAE algorithm. Moreover, the number of cores used during the optimization does not decrease the runtime when the number of unique features per vertex compared to the cardinality of the feature set itself is large. When we look at the change in the vertex set size we also see a linear behaviour. Doubling the input size simply results in a doubled optimization runtime. In addition, if one interpolates linearly from these results it comes that a network with 1 million nodes, 8 edges per node, 8 unique features per node can be embedded with *MUSAE* on commodity hardware in less than 5 hours. This interpolation assumes that the standard parameter settings proposed in Subsection V-B and 4 cores were used for optimization.

VI. Discussion and Conclusion

We investigated attributed node embedding and propose efficient pooled (*AE*) and multi-scale (*MUSAE*) attributed node embedding algorithms with linear runtime. And we proved that these algorithms implicitly factorize probability matrices of features appearing in the neighbourhood of nodes. Two widely used neighbourhood preserving node embedding methods [1], [11] are in fact simplified cases of our models. On several datasets (Wikipedia page-page networks) we found that representations learned by our methods, in particular *MUSAE*, materially outperform neighbourhood based node embedding methods ([1], [2]), multi-scale algorithms ([3], [11]) and recently proposed attributed node embedding procedures ([5], [7], [6], [8], [9]).

Our proposed embedding models are differentiated from other methods in that they encode feature imformation from higher order neighborhoods. The most similar previous model *BANE* [8] encodes node attributes from higher order neighbourhoods but has non-linear runtime complexity and the product of adjacency matrix power and feature matrix is decomposed explicitly.

Our sensitivity analysis shows that multi-scale attributed node embeddings offer significant performance improvement over pooled methods in downstream tasks. Furthermore, the multi-scale embeddings take greater advantage of increased dimensionality and of an increased context.

Appendix

Lemma 1. *The empirical statistics of node-feature pairs obtained from random walks give unbiased estimates of joint probabilities of observing feature $f \in F$ r steps (i) after; or (ii) before node $v \in V$, as given by:*

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{c}(\boldsymbol{D}\boldsymbol{P}^r\boldsymbol{F})_{w,f}$$

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} = \frac{1}{c}(\boldsymbol{F}^\top\boldsymbol{D}\boldsymbol{P}^r)_{f,w}$$

Proof. The proof is analogous to that given for Theorem 2.1 in [13]. We show that the computed statistics correspond to sequences of random variables with finite expectation, bounded variance and covariances that tend to zero as the separation between variables within the sequence tends to infinity. The Weak Law of Large Numbers (S.N.Bernstein) then guarantees that the sample mean converges to the expectation of the random variable. We first consider the special case $n = 1$, i.e. we have a single sequence $w_1, ..., w_l$ generated by a random walk (see Algorithm 1). For a particular node-feature pair

(w, f), we let Y_i, $i \in \{1, \ldots, l-t\}$, be the indicator function for the event $w_i = w$ and $f \in F_{i+r}$. Thus, we have:

$$\frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{l-t}\sum_{i=1}^{l-t} Y_i, \tag{3}$$

the sample average of the Y_is. We also have:

$$\mathbb{E}[Y_i] = \frac{deg(w)}{c}(\mathbf{P}^r\mathbf{F})_{w,f} = \frac{1}{c}(\mathbf{D}\mathbf{P}^r\mathbf{F})_{w,f}$$

$$\begin{aligned}
&\mathbb{E}[Y_iY_j] = \text{Prob}[w_i = w, f \in F_{i+r}, w_j = w, f \in F_{j+r}] \\
&= \underbrace{\frac{deg(w)}{c}}_{p(w_i=w)} \underbrace{\underbrace{\mathbf{P}^r_{:w}}_{p(w_{i+r}|w_i=w)} \underbrace{diag(\mathbf{F}_{:f})}_{p(f\in F_{i+r}|w_{i+r})} \underbrace{\mathbf{P}^{j-(i+r)}_{:w}}_{p(w_j=w|w_{i+r})}}_{p(w_j=w,f\in F_{i+r}|w_i=w)} \underbrace{\mathbf{P}^r_{w:}\mathbf{F}_{:f}}_{p(f\in F_{j+r}|w_j=w)}
\end{aligned}$$

for $j > i + r$. This allows us to compute the covariance:

$$\begin{aligned}
\text{Cov}(Y_i, Y_j) &= \mathbb{E}[Y_iY_j] - \mathbb{E}[Y_i]\,\mathbb{E}[Y_j] \\
&= \frac{deg(w)}{c}\mathbf{P}^r_{w:}diag(\mathbf{F}_{:f})\underbrace{\left(\mathbf{P}^{j-(i+r)}_{:w} - \frac{deg(w)}{c}\underline{1}\right)}_{\text{tends to 0 as } j-i\to\infty}\mathbf{P}^r_{w:}\mathbf{F}_{:f}, \quad (4)
\end{aligned}$$

where $\underline{1}$ is a vector of ones. The difference term (indicated) tends to zero as $j - i \to \infty$ since then $p(w_j = w|w_{i+r})$ tends to the stationary distribution $p(w) = \frac{deg(w)}{c}$, regardless of w_{i+r}. Thus, applying the Weak Law of Large Numbers, the sample average converges in probability to the expected value, i.e.:

$$\frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{l-t}\sum_{i=1}^{l-t} Y_i \xrightarrow{p} \frac{1}{l-t}\sum_{i=1}^{l-t}\mathbb{E}[Y_i] = \frac{1}{c}(\mathbf{D}\mathbf{P}^r\mathbf{F})_{w,f}$$

A similar argument applies to $\frac{\#(w,f)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}$, with expectation term $\frac{1}{c}(\mathbf{F}^\top\mathbf{D}\mathbf{P}^r)_{f,w}$. In both cases, the argument readily extends to the general setting where $n > 1$ with suitably defined indicator functions for each of the n random walks (see [13]). $\square$

Lemma 2. *The empirical statistics of node-feature pairs obtained from random walks give unbiased estimates of joint probabilities of observing feature $f \in F$ r steps either side of node $v \in V$, as given by:*

$$\underset{l\to\infty}{plim}\frac{\#(w,f)_r}{|\mathcal{D}_r|} = \frac{1}{c}(\boldsymbol{D}\boldsymbol{P}^r\boldsymbol{F})_{w,f}\ , \tag{1}$$

Proof.

$$\begin{aligned}
\frac{\#(w,f)_r}{|\mathcal{D}_r|} &= \frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_r|} + \frac{\#(w,f)_{\overleftarrow{r}}}{|\mathcal{D}_r|} \\
&= \frac{1}{2}\Big(\frac{\#(w,f)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} + \frac{\#(w,f)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}\Big) \\
&\xrightarrow{p} \frac{1}{2}\big(\tfrac{1}{c}(\mathbf{D}\mathbf{P}^r\mathbf{F})_{w,f} + \tfrac{1}{c}(\mathbf{F}^\top\mathbf{D}\mathbf{P}^r)_{f,w}\big) \\
&= \frac{1}{2c}\big(\mathbf{D}\mathbf{P}^r\mathbf{F} + \mathbf{P}^{r\top}\mathbf{D}\mathbf{F}\big)_{w,f} \\
&= \frac{1}{2c}\big((\mathbf{D}\mathbf{P}^r + (\mathbf{A}^\top\mathbf{D}^{-1})^r\mathbf{D})\mathbf{F}\big)_{w,f} \\
&= \frac{1}{2c}\big((\mathbf{D}\mathbf{P}^r + \mathbf{D}(\mathbf{D}^{-1}\mathbf{A}^\top)^r)\mathbf{F}\big)_{w,f} \\
&= \frac{1}{c}(\mathbf{D}\mathbf{P}^r\mathbf{F})_{w,f}\ .
\end{aligned}$$

The final step follows from the symmetry of $\mathbf{A}$, but shows how the Lemma can be readily extended to directed graphs. $\square$

References

[1] 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.*, 2014.

[2] 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*, 2016.

[3] 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*, 2015, pp. 1067–1077.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[5] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *IJCAI*, 2015, pp. 2111–2117.

[6] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 633–641.

[7] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[8] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1476–1481.

[9] S. Yang and B. Yang, "Enhanced network embedding with text information," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 326–331.

[10] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Sine: Scalable incomplete network embedding," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 737–746.

[11] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: Online learning of multi-scale network embeddings," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 258–265.

[12] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in neural information processing systems*, 2014, pp. 2177–2185.

[13] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 459–467.

[14] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17, 2017, pp. 385–394.

[15] N. K. Ahmed, R. Rossi, J. B. Lee, X. Kong, T. L. Willke, R. Zhou, and H. Eldardiry, "Learning role-based graph embeddings," *arXiv preprint arXiv:1802.02896*, 2018.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[17] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 9198–9208.

[18] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[20] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," *arXiv preprint arXiv:1802.08888*, 2018.