Self-Supervised Learning of Contextual Embeddings for Link Prediction in Heterogeneous Networks

Ping Wang\(^1\), Khushbu Agarwal\(^2\), Colby Ham\(^2\), Sutanay Choudhury\(^2\), Chandan K. Reddy\(^1\)

\(^1\)Department of Computer Science, Virginia Tech, Arlington, VA
\(^2\)Pacific Northwest National Laboratory, Richland, WA

ping@vt.edu,\{khushbu.agarwal,colby.ham,sutanay.choudhury\}@pnnl.gov, reddy@cs.vt.edu

ABSTRACT

Representation learning methods for heterogeneous networks produce a low-dimensional vector embedding (that is typically fixed for all tasks) for each node. Many of the existing methods focus on obtaining a static vector representation for a node in a way that is agnostic to the downstream application where it is being used. In practice, however, downstream tasks such as link prediction require specific contextual information that can be extracted from the subgraphs related to the nodes as provided as input to the task. To tackle this challenge, we develop SLiCE, a framework for bridging static representation learning methods using global information from the entire graph with localized attention-driven mechanisms to learn contextual node representations. We first pre-train our model in a self-supervised manner by introducing higher-order semantic associations and masking nodes, and then fine-tune our model for a specific link prediction task. Instead of training node representations by aggregating information from all semantic neighbors connected via metapaths, we automatically learn the composition of different metapaths that characterize the context for a specific task without the need for any pre-defined metapaths. SLiCE significantly outperforms both static and contextual embedding learning methods on several publicly available benchmark network datasets. We also demonstrate the interpretability, effectiveness of contextual learning, and the scalability of SLiCE through extensive evaluation.

1 INTRODUCTION

The topic of representation learning for heterogeneous networks has gained a lot of attention in recent years [1, 5, 10, 29, 33, 35], where a low-dimensional vector representation of each node in the graph is used for downstream applications such as link prediction [1, 5, 37] or multi-hop reasoning [8, 13, 40]. Many of the existing methods focus on obtaining a static vector representation per node that is agnostic to any specific context and is typically obtained by learning the importance of all of the node’s immediate and multi-hop neighbors in the graph. However, we argue that nodes in a heterogeneous network exhibit a different behavior, based on different relation types and their participation in diverse network communities. Further, most downstream tasks such as link prediction are dependent on the specific contextual information related to the input nodes that can be extracted in the form of task specific subgraphs.

Incorporation of contextual learning has led to major breakthroughs in the natural language processing community [9, 24], in which the same word is associated with different concepts depending on the context of the surrounding words. A similar phenomenon can be exploited in graph structured data and it becomes particularly pronounced in heterogeneous networks where the addition of relation types as well as node and relation attributes leads to increased diversity in a node’s contexts. Figure 1 provides an illustration of this problem for an academic network. Given two authors who publish in diverse communities, we posit that the task of predicting link (Author, co- author, Author) would perform better if their node representation is reflective of the common publication topics and venues, i.e., Representation Learning and NeurIPS. This is in contrast to existing methods where author embeddings would reflect information aggregation from all of their publications, including the publications in healthcare and climate science which are not part of the common context.

Contextual learning of node representations in network data has recently gained attention with different notions of context emerging (see Table 1). In homogeneous networks, communities provide a natural definition of a node’s participation in different contexts referred to as facets or aspects [11, 19, 21, 32, 34]. Given a task such as link prediction, inferring the cluster-driven connectivity between the nodes has been the primary basis for these approaches. However, accounting for higher-order effects over diverse metapaths (defined as paths connected via heterogeneous relations) is demonstrated to be essential in representation learning and link prediction in heterogeneous networks [5, 16, 33, 35]. Therefore, contextual learning methods that primarily rely on the well-defined notion of graph clustering will be limited in their effectiveness for heterogeneous networks where modeling semantic association (via...
meta-paths or meta-graphs) is at least equal or more important than community structure for link prediction.

In this paper, we seek to make a fundamental advancement over these categories that aim to contextualize a node’s representation with regards to either a cluster membership or association with meta-paths or meta-graphs. We believe that the definition of a context needs to be expanded to subgraphs (comprising heterogeneous relations) that are task-specific and learn node representations that represent the collective heterogeneous context. With such a design, a node’s embedding will be dynamically changing based on its participation in one input subgraph to another. Our experiments indicate that this approach has a strong merit with link prediction performance, thus improving it by 10%-25% over many state-of-the-art approaches.

We propose shifting the node representation learning from a node’s perspective to a subgraph point of view. Instead, of focusing on “what is the best representation for a node v”, we seek to answer “what are the best collective node representations for a given subgraph g_v”, and “how such representations can be potentially useful in a downstream application?” Our proposed framework SLiCE (which is an acronym for Self-supervised LearnIng of Contextual Embeddings), accomplishes this by bridging static representation learning methods using global information from the entire graph with localized attention driven mechanisms to learn contextual node representations in heterogeneous networks. While bridging global and local information is a common approach for many algorithms, the primary novelty of SLiCE lies in learning an operator for contextual translation by learning higher-order interactions through self-supervised learning.

**Contextualized Representations:** Building on the concept of translation-based embedding models [3], given a node u and its embedding h_u, computed using a global representation method, we formulate graph-based learning of contextual embeddings as performing a vector-space translation \( \Delta_u \) (informally referred to as shifting process) such that \( h_u + \Delta_u \approx \tilde{h}_u \), where \( \tilde{h}_u \) is the contextualized representation of \( u \). The key idea behind SLiCE is to learn the translation \( \Delta_u \) where \( u \in V(g_v) \). Figure 1(c) shows an illustration of this concept where the embedding of both Author1 and Author2 are shifted using the common subgraph with (Paper P1, Representation learning, NeurIPS, Paper P3) as context. We achieve this contextualization as follows: We first learn the higher-order semantic association (HSA) between nodes in a context subgraph. We do not assume any prior knowledge about important meta-paths, and SLiCE learns important task specific subgraph structures during training (see section 4.3). More specifically, we first develop a self-supervised learning approach that pre-trains a model to learn a HSA matrix on a context-by-context basis. We then fine-tune the model in a task-specific manner, where given a context subgraph \( g_v \) as input, we encode the subgraph with global features and then transform that initial representation via a HSA-based non-linear transformation to produce contextual embeddings (see Figure 2).

**Our Contributions:** The main contributions of our work are:

- Propose contextual embedding learning for graphs from single relation context to arbitrary subgraphs.
- Introduce a novel self-supervised learning approach to learn higher-order semantic associations between nodes by simultaneously capturing the global and local factors that characterize a context subgraph.
- Show that SLiCE significantly outperforms existing static and contextual embedding learning methods using standard evaluation metrics for the task of link prediction.
- Demonstrate the interpretability, effectiveness of contextual translation, and the scalability of SLiCE through an extensive set of experiments and contribution of a new benchmark dataset.

The rest of this paper is organized as follows. Section 2 provides an overview of related work about network embedding learning and differentiates our work from other existing work. In Section 3, we introduce the problem formulation and present the proposed SLiCE model. We describe the details of experimental analysis and show the comparison of our model with the state-of-the-art methods in Section 4. Finally, Section 5 conclude the discussion of the paper.

### 2 RELATED WORK

We begin with an overview of the state-of-the-art methods for representation learning in heterogeneous network and then follow with a discussion on the nascent area of contextual representation learning. Table 1 provides a summarized view of this discussion.

**Node Representation Learning:** Earlier representation learning algorithms for networks can be broadly categorized into two groups based on their usage of matrix factorization versus random walks or skip-gram-based methods. Given a graph \( G \), matrix factorization
Table 1: Comparison of representative approaches for learning heterogeneous network (HN) embeddings proposed in the recent literature from contextual learning perspective. Other abbreviations used: graph convolutional network (GCN), graph neural network (GNN), random walk (RW), skip-gram (SG). N/A stands for “Not Applicable”.

<table>
<thead>
<tr>
<th>Method</th>
<th>Multi-Embeddings per Node</th>
<th>Context Scope</th>
<th>HN support</th>
<th>Learning Approach</th>
<th>Automated Learning of Meta-Paths/Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>HetGN [36], HetGAT [33]</td>
<td>N</td>
<td>N/A</td>
<td>Y</td>
<td>GNN</td>
<td>N</td>
</tr>
<tr>
<td>GNN [35], HGT [16]</td>
<td>N</td>
<td>N/A</td>
<td>Y</td>
<td>Transformer</td>
<td>Y</td>
</tr>
<tr>
<td>GAN [1], RGCGN [26]</td>
<td>N</td>
<td>N/A</td>
<td>Y</td>
<td>GCN</td>
<td>N</td>
</tr>
<tr>
<td>Polysemy [19], MCNE [32]</td>
<td>Y</td>
<td>Per aspect</td>
<td>N</td>
<td>Extends SG/GCN, GNN</td>
<td>N</td>
</tr>
<tr>
<td>GATTNE [5], CompGCN [29]</td>
<td>Y</td>
<td>Per relation</td>
<td>Y</td>
<td>HN-SG, GCN</td>
<td>N/A</td>
</tr>
<tr>
<td>SPLITTER [11], asp2vec [21]</td>
<td>Y</td>
<td>Per subgraph</td>
<td>Y</td>
<td>Extends RW-SG</td>
<td>N</td>
</tr>
<tr>
<td>SLiCE (proposed)</td>
<td>Y</td>
<td>Self-supervision</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

based methods [4] seek to learn a representation \( \Gamma \) that minimizes a loss function of the form \( ||I\Gamma - P_V||^2 \), where \( P_V \) is a matrix containing pairwise proximity measures for \( V(G) \). Random walk based methods such as DeepWalk [23] and node2vec [12] try to learn representations that approximately minimize a cross-entry loss function of the form \( \sum_{i,j \in V(G)} -\log(P_L(s_j|v_i)) \), where \( P_L(s_j|v_i) \) is the probability of visiting a node \( s_j \) on a random walk of length \( L \) starting from node \( v_i \). Node2vec based approach has been further extended to incorporate multi-relational properties of networks by constraining random walks to ones conforming to specific metapaths [7, 10]. Recent efforts [25] seek to unify the first two categories by demonstrating the equivalence of [23] and [12]-like methods to matrix factorization approaches.

Attention-based Methods: A newer category represents graph neural networks and their variants [18, 26]. Attention mechanisms that learn a distribution for aggregating information from a node’s immediate neighbors is investigated in [31]. Aggregation of attention from semantic neighbors, or nodes that are connected via multi-hop metapaths have been exhaustively investigated over the past few years and can be grouped by the underlying neural architectures such as graph convolutional networks [29], graph neural network [33, 36], and graph transformers [16, 35]. Extending the above methods from meta-paths to meta-graphs [15, 39] as a basis for sampling and learning has emerged as a new direction as well.

Contextual Representation Learning: Works such as [2, 11, 19, 28, 32, 34] study the “multi-aspect” effect. Typically, “aspect” is defined as a node’s participation in a community or cluster in the graph or even being an outlier, and these methods produce a node embedding by accounting for it’s membership in different clusters. However, most of these methods are studied in detail for homogeneous networks. More recently, this line of work has evolved towards addressing finer issues such as inferring the context, addressing limitations with offline clustering, producing different vectors depending on the context as well as extension towards heterogeneous networks [20, 21].

Beyond these works, a number of newer approaches and objectives have also emerged. The authors of [1] compute a node’s representation by learning the attention distribution over a graph walk context where it occurs. The work presented in [5] is a metapath-constrained random walk based method that contextualizes node representations per relation. It combines a base embedding derived from the global structure (similar to above methods) with a relation-specific component learnt from the metapaths. In a similar vein, [29] provides operators to adapt a node’s embedding based on the associated relational context.

Key Distinctions of SLiCE: To summarize, the key differences between SLiCE and existing works are as follows: (i) Our contextualization objective is formulated on the basis of a subgraph that distinguishes SLiCE from [5] and [29]. While subgraph-based representation learning objectives have been superficially investigated in the literature [38], they do not focus on generating contextual embeddings. (ii) From a modeling perspective, our self-supervised approach is distinct from both the metapath-based learning approaches outlined in the attention-based methods section (we learn important metapaths automatically without requiring any user supervision) as well as the clustering-centric multi-aspect approaches discussed in the contextual representation learning category.

3 THE PROPOSED FRAMEWORK

3.1 Problem Formulation

Before presenting our overall framework, we first briefly provide the formal definitions and notations that are required to comprehend the proposed approach.

Definition: (Heterogeneous Graph). We represent a heterogeneous graph as a 6-tuple \( G = (V, E, \Sigma_V, \Sigma_E, \lambda_\Sigma, \lambda_E) \) where, \( V \) (alternatively referred to as \( V(G) \)) is the set of nodes and \( E \) (or \( E(G) \)) denotes the set of edges between the nodes. \( \lambda_\Sigma \) and \( \lambda_E \) are functions mapping the node (or edge) to its node (or edge) type \( \Sigma_V \) and \( \Sigma_E \), respectively.

Definition: (Context Subgraph). Given a heterogeneous graph \( G \), the context of a node \( v \) or node-pair \((u, v)\) in \( G \) can be represented as the subgraph \( g_c \) that includes a set of nodes selected with certain criteria (e.g., \( k \)-hop neighbors for \( v \) or \( k \)-hop neighbors connecting \((u, v)\)) along with their related edges. The context of the node or node-pair can be represented as \( g_c(v) \) and \( g_c(u, v) \).

Problem definition. Given a heterogeneous graph \( G \), a subgraph \( g_c \) and the link prediction task \( T \), compute a function \( f(G, g_c, T) \) that maps each node \( v_i \) in the set of vertices in \( g_c \), denoted as \( V(g_c) \), to a real-valued embedding vector \( h_i \) in a low-dimensional space \( d \) such that \( h_i \in \mathbb{R}^d \). We also require that \( S \), a scoring function serving as a proxy for the link prediction task, satisfies the following: \( S(v_i, v_j) \geq \delta \) for a positive edge \( e = (v_i, v_j) \) in graph \( G \) and \( S(v_i, v_j) < \delta \) if \( e \) is a negative edge, where \( \delta \) is a threshold.
During the pre-training phase, the context mainly consists of the following four components: (1) Contextual Subgraph Generation and Encoding: generating a collection of context subgraphs and transforming them into a vector representation. (2) Contextual Translation: for nodes in a context subgraph, translating the global embeddings, which consider various heterogeneous attributes about nodes, relations and graph structure, to contextual embeddings based on the specific local context. (3) Model Pre-training: learning higher order relations with the self-supervised contextualized node prediction task. (4) Model Fine-tuning: the model is then tuned by the supervised link prediction task with more fine-grained contexts for node pairs. Figure 2 shows the framework of the proposed SLiCE model. In the following sections, we will introduce more details layer-by-layer.

### 3.2 Context Subgraphs: Generation and Representation

#### 3.2.1 Context Subgraph Generation

In this work, we generate the pre-training subgraphs \(G_C\) for each node in the graph using a random walk strategy. Masking and predicting nodes in the random walks helps SLiCE learn the global connectivity patterns in \(G\) during the pre-training phase. In fine-tuning phase, the context subgraphs for link prediction are generated using following approaches: (1) **Shortest Path** strategy considers the shortest path between two nodes as the context. (2) **Random** strategy, on the other hand, generates contexts following the random walks between two nodes, limited to a pre-defined maximum number of hops. Note that the context generation strategies are generic and can be applied for generating contexts in many downstream tasks such as link prediction, knowledge base completion or multi-hop reasoning [8, 13].

In our experiments, context subgraphs are generated for each node \(v \in G\) in the graph during pre-training and for each node-pair during fine-tuning. Each generated subgraph \(g_c \in G_C\) is encoded as a set of nodes denoted by \(g_c = (v_1, v_2, \ldots, v_{|V_c|})\), where \(|V_c|\) represents the number of nodes in \(g_c\). Different from the sequential orders enforced on graph sampled using pre-defined metapaths, the order of nodes in this set is not important. Therefore, our context subgraphs are not limited to paths, and can handle tree or star-shaped subgraphs.

#### 3.2.2 Context Subgraph Encoder

We first represent each node \(v_i\) in the context subgraph as a low-dimensional vector representation by \(h_i = W_c \cdot \text{attr}(v_i)\), where \(\text{attr}(\cdot)\) is a function that returns a stacked vector containing the structure-based embedding of \(v_i\) and the embedding of its attributes. \(W_c\) is the learnable embedding matrix. We represent the input node embeddings in \(g_c\) as \(H_c = (h_1, h_2, \ldots, h_{|V_c|})\). It is flexible to incorporate the node and relation attributes (if available) for attributed networks [5] in the low-dimensional representations or initialize them with the output embeddings learnt from other global feature generation approaches that capture the multi-relational graph structure [10, 12, 29, 33, 35].

There are multiple approaches for generating global node features in heterogeneous networks (see ‘related work’). Our experiments show that the node embeddings obtained from random walk based skip-gram methods (RW-SG) produces competitive performance for link prediction tasks. Therefore, in the proposed SLiCE model, we mainly consider the pre-trained node representation vectors from node2vec for initialization of the node features.

#### 3.3 Contextual Translation

Given a set of nodes \(V_c\) in a context subgraph \(g_c\) and their global input embeddings \(H_c \in \mathbb{R}^{d_x \times |V_c|}\), the primary goal of contextual learning is to translate (or shift) the global embeddings in the vector space towards their new positions that indicate the most representative roles of nodes in the structure of \(g_c\). We consider this mechanism as a transformation layer and the model can include multiple such layers according to the higher-order relations contained in the graph. Before introducing the details of this contextual translation mechanism, we first provide the definition of the semantic association matrix, which serves as the primary indicator about the translation of embeddings according to specific contexts.

**Definition: (Semantic Association Matrix)** A semantic association matrix, denoted as \(\tilde{A}\), is an asymmetric weight matrix that indicates the high-order relational dependencies between nodes in the context subgraph \(g_c\).
Note that the semantic association matrix will be asymmetric since the influences of two nodes on one another in a context subgraph tend to be different. The adjacency matrix of the context subgraph, denoted by $A_{gc}$, can be considered as a trivial candidate for $\bar{A}$, which includes the local relational information of context subgraph $g_c$. However, the goal of contextual embedding learning is to translate the global embeddings using the contextual information contained in the specific context $g_c$ while keeping the nodes’ connectivity through the global graph. Hence, instead of setting it to $A_{gc}$, we contextually learn the semantic associations, or more specifically the weights of the matrix $\bar{A}^k$ in each translation layer $k$ by incorporating the connectivity between nodes through both local context subgraph $g_c$ and global graph $G$.

**Implementation of Contextual Translation:** In the translation layer $k + 1$, the semantic association matrix $\bar{A}^k \in \mathbb{R}^{|V_c| \times |V_c|}$ is updated by the transformation operation defined in Eq. (1). It is accomplished by performing message passing across all nodes in context subgraph $g_c$ and updating the node embeddings $H^k = (h^k_1, h^k_2, \ldots, h^k_{|V_c|})$ to be $H^{k+1}$,

$$H^{k+1} = f_{NN}(W_kH^k\bar{A}^k + H^k) \quad (1)$$

where $f_{NN}$ is a non-linear function and the transformation matrix $W_k \in \mathbb{R}^{d \times d}$ is the learnable parameter. The residual connection [14] is applied to preserve the contextual embeddings in the previous step. This allows us to still maintain the global relations by passing the original global embeddings through layers while learning contextual embeddings. Given two nodes $v_1$ and $v_2$ in context subgraph $g_c$, the corresponding entry $\bar{A}^k_{ij}$ in semantic association matrix can be computed using the multi-head (with $N_h$ heads) attention mechanism [30] in order to capture relational dependencies under different subspaces. For each head, we calculate $\bar{A}^k_{ij}$ as follows:

$$\bar{A}^k_{ij} = \frac{\exp \left( (W^k_i h^k_i)^T (W^k_j h^k_j) \right)}{\sum_{l=1}^{|V_c|} \exp \left( (W^k_i h^k_i)^T (W^k_l h^k_l) \right)} \quad (2)$$

where the transformation matrix $W_1$ and $W_2$ are learnable parameters. It should be noted that, different from the aggregation procedure performed across all nodes in the general graph $G$, the proposed translation operation is only performed within the local context subgraph $g_c$. The updated embeddings after applying the translation operation according to context $g_c$ indicate the most representative roles of each node in the specific local context neighborhood. In order to capture the higher-order association relations within the context, we apply multiple layers of the transformation operation in Eq. (1) by stacking $K$ layers as shown in Figure 2, where $K$ is the largest diameter of the subgraphs sampled in context generation process.

By applying multiple translation layers, we are able to obtain multiple embeddings for each node in the context subgraph. In order to collectively consider different embeddings in the downstream tasks, we aggregate the node embeddings learnt from different layers $(h^k_i)_{k=1,\ldots,K}$ as the contextual embedding $\bar{h}_i$ for each node as follows.

$$\bar{h}_i = h^1_i \oplus h^2_i \oplus \cdots \oplus h^K_i \quad (3)$$

### Algorithm 1: Self-supervised Pre-training in SLiCE.

1. **Require:** Graph $G$ with nodes set $V$ and edges set $E$, embedding size $d$, context subgraph size $m$, No. of translation layers $K$.
2. **Pre-training dataset $G_c^{pre} \leftarrow \emptyset**
3. **for each** $v \in V$ **do**
   4. $g_c^v = $GetContext($G$, $v$, $m$) **do** Generate context subgraph
   5. $g_c^v = $Encode($g_c^v$) **do** Encode context as a sequence
   6. $\bar{v}_m = $Random($g_c^v$) **do** Mask a node in $g_c^v$ for prediction
   7. $G_c^{pre} \leftarrow $Append($g_c^v$, $\bar{v}_m$)
5. **end**
6. $H \leftarrow $EmbedFunction($G$, $d$) **do** Learn global embeddings
7. **Initialize** node embeddings as $H^0 = H$
8. **while** not converged **do**
9. **for** $g_c^v$ **to** $G_c^{pre}$ **do**
10. **for** $k = 1$ **to** $K$ **do**
11. $H^{k+1} = f_{NN}(W_kH^k\bar{A}^k + H^k)$ with $\bar{A}$ calculated by Eq. (2)
12. **end**
13. Contextual embeddings $\hat{H}_c = H^1 \oplus H^2 \oplus \cdots \oplus H^K$
14. Update parameters with the contextualized node prediction task using the objective function in Eq. (4).
15. **end**
16. **end**

Given a context subgraph $g_c$, the obtained contextual embedding vectors $\{\bar{h}_i\}_{i=1,\ldots,|V_c|}$ can be fed into the prediction tasks. In pre-training step, a linear projection function is applied on the contextual embeddings to predict the probability of masked nodes. For fine-tuning step, we apply a single layer feed-forward network with softmax activation function for binary link prediction.

### 3.4 Model Training Objectives

**3.4.1 Self-supervised Contextual Node Prediction.** Our model pre-training is performed by training the self-supervised contextualized node prediction task. More specifically, for each node in $G$, we generate the node context $g_c$ with diameter (defined as the largest shortest pair between any pair of nodes) using the aforementioned context generation methods and randomly mask a node for prediction based on the context subgraph. The graph structure is left unperturbed by the masking procedure. Therefore, the pre-training is learnt by maximizing the probability of observing this masked node $\bar{v}_m$ based on the context $g_c$ in the following form.

$$\theta = \arg\max_{\theta} \prod_{g_c \in G_C} \prod_{\bar{v}_m \in g_c} p(\bar{v}_m | g_c, \theta) \quad (4)$$

where $\theta$ represents the set of model parameters. The procedure for pre-training is given in Algorithm 1. In this algorithm, lines 2-8 generate context subgraphs for nodes in the graph and further applies a random masking strategy to process the data for pre-training. Lines 9-10 learn the pre-trained global node features and initialize them as the node embeddings in SLiCE. In lines 13-15, we apply the contextual translation layers on the context subgraphs, aggregate the output of different layers as the contextual node embeddings.
in line 16 and update the model parameters with contextual node
prediction task. In a departure from traditional skip-gram meth-
ods [21] that predicts a node from the path prefix that precedes it in
a random walk, our random masking strategy forces the model to
learn higher-order relationships between nodes that are arbitrarily
connected by variable length paths with diverse relational patterns.

3.4.2 Fine-tuning with Supervised Link Prediction. The SLiCE model
is further fine-tuned on the contextualized link prediction task by
generating multiple fine-grained contexts for each specific node-pair
that is under consideration for link prediction. Based on the pre-
dicted scores, this stage is trained by maximizing the probability
of observing a positive edge \((e_p)\) given context \((g_{cp})\), while also
learning to assign low probability to negatively sampled edges \((e_n)\)
and their associated contexts \((g_{cn})\). The overall objective is obtained
by summing over the training data subsets with positive edges \((D_p)\)
and negative edges \((D_n)\). Algorithm 2 shows the process of fine-
tuning step. In this algorithm, lines 2-7 generate context subgraphs
of the node-pairs for link prediction task and process the data for
fine-tuning in the same manner described in pre-training. Lines
8-18 perform the fine-tuning with link prediction task.

\[
    \mathcal{L} = \sum_{(e_p,g_{cp}) \in D_p} \log(P(e_p|g_{cp}, \theta)) + \sum_{(e_n,g_{cn}) \in D_n} \log(1 - P(e_n|g_{cn}, \theta))
\]  

We compute the probability of the edge between two nodes \(e = (u, v)\) as the similarity score \(S(u, v) = \sigma(h^1 \cdot \hat{h}_j)\) \([1]\), where \(\hat{h}_j\) and \(\hat{h}_i\) are contextual embeddings of \(u\) and \(v\) learnt based on a
context subgraph, respectively. \(\sigma(\cdot)\) represents sigmoid function.

### 3.5 Complexity Analysis

We assume that \(N_{cpn}\) denotes the number of context subgraphs
generated for each node, \(N_{mc}\) represents the maximum number of
nodes in any context subgraph, and \(|V|\) represents the number of
nodes in the input graph. Then, the total number of context
subgraphs considered in pre-training stage can be estimated as
\(|V| \times N_{cpn}\) and the cost of iterating over all these subgraphs through
multiple epochs will be \(O(|V| \times N_{cpn})\). Since the generated context
subgraphs need to provide us with a good approximation of the
total number of edges in the entire graph, we approximate the total
cost as \(O(|V| \times N_{cpn} |E|) = O(N_E)\), where \(N_E\) is the number of edges in
the training dataset. It can also be represented as \(N_E = \alpha T |E|\), where
\(T\) is the total number of edges in graph \(G\) and \(\alpha T\) represents the
ratio of training split. The cost for each contextual translation layer
in SLiCE model is \(O(N_{mc}^2)\) since the dot product for calculating
node similarity is the dominant computation and is quadratic to
the number of nodes in the context subgraph. In this case, the
total training complexity will be \(O(|E| N_E^2)\). The maximum number of
nodes \(N_{mc}\) in context subgraphs is relatively small and it can be
considered as a constant that does not depend on the size of the
input graph. Therefore, the training complexity of SLiCE is
approximately linear to the number of edges in the input graph.

### 3.6 Implementation Details

The proposed SLiCE model is implemented using PyTorch 1.3 [22].
The dimension of contextual node embeddings is set to 128 in
SLiCE. We used a skip-gram based random walk approach to encode
context subgraphs with global node features. Both pre-training and
fine-tuning steps in SLiCE are trained for 10 epochs with a batch size
of 128 using the cross-entropy loss function. The model parameters
are trained with ADAM optimizer [17] with a learning rate of 0.0001 and
0.001 for pre-training and fine-tuning steps, respectively. The
best model parameters were selected based on the development set.
Both the number of contextual translation layers and number of
self-attention heads are set to 4. We generate context subgraphs by
performing random walks between node pairs by setting the
maximum number of nodes in context subgraphs and the number
of contexts generated for each node to be \([6, 12)\) and \([1, 5, 10]\),
respectively and report the best performance. In the fine-tuning
stage, the subgraph with the largest prediction score are selected as
the best context subgraph for each node-pair. The implementation
of the SLiCE model is made publicly available at this website[1].

### 4 EXPERIMENTS

In this section, we address following research questions (RQs)
through experimental analysis:

1. **RQ1**: Does subgraph-based contextual learning improve
the performance of downstream tasks such as link prediction?
2. **RQ2**: Can we interpret SLiCE’s performance using semantic
features of context subgraphs?
3. **RQ3**: Where does contextualization help in graphs? How do
we quantify the effect of the embedding shift from static to
subgraph-based contextual learning?

[1]https://github.com/pnnl/SLiCE
Table 2: The basic statistics of the datasets used in this paper.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amazon</th>
<th>DBLP</th>
<th>Freebase</th>
<th>Twitter</th>
<th>Healthcare</th>
</tr>
</thead>
<tbody>
<tr>
<td># Nodes</td>
<td>10,099</td>
<td>37,791</td>
<td>14,541</td>
<td>9,990</td>
<td>4,683</td>
</tr>
<tr>
<td># Edges</td>
<td>129,811</td>
<td>170,794</td>
<td>248,611</td>
<td>294,330</td>
<td>205,428</td>
</tr>
<tr>
<td># Relations</td>
<td>2</td>
<td>3</td>
<td>237</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td># Training</td>
<td>126,535</td>
<td>119,554</td>
<td>272,115</td>
<td>282,115</td>
<td>164,816</td>
</tr>
<tr>
<td># Development</td>
<td>14,756</td>
<td>51,242</td>
<td>35,070</td>
<td>32,926</td>
<td>40,612</td>
</tr>
<tr>
<td># Testing</td>
<td>29,492</td>
<td>51,238</td>
<td>40,932</td>
<td>65,838</td>
<td>40,612</td>
</tr>
</tbody>
</table>

(4) RQ4: What is the impact of different parameters and components (including pre-trained global features and fine-tuning procedure) on the performance of SLiCE?

(5) RQ5: Can we empirically verify SLiCE’s scalability?

4.1 Experimental Settings

4.1.1 Datasets used. We use four public benchmark datasets covering multiple applications: e-commerce (Amazon), academic graph (DBLP), knowledge graphs (Freebase), and social networks (Twitter). We use the same data split for training, development, and testing as described in previous works [1, 5, 29]. In addition, we introduce a new knowledge graph from the publicly available real-world Medical Information Mart for Intensive Care (MIMIC) III dataset in the healthcare domain. We generated equal number of positive and negative edges for the link prediction task. Table 2 provides the basic statistics of all datasets. The details of each dataset are provided below.

- **Amazon**: includes the co-viewing and co-purchasing links between products. The edge types, also_bought and also_viewed, represent that products are co-bought or co-viewed by the same user, respectively.

- **DBLP**: includes the relationships between papers, authors, venues, and terms. The edge types include paper_has_term, published_at and has_author.

- **Freebase**: is a pruned version of FB15K with inverse relations removed. It includes links between people and their nationality, gender, profession, institution, place of birth/death, and other demographic features.

- **Twitter**: includes links between tweets users. The edge types included in the network are re-tweet, reply, mention, and friendship/follower.

- **Healthcare**: includes relations between patients and their diagnosed medical conditions during each hospital admission along with relations to procedures and medications received. To ensure data quality, we use a 5-core setting, i.e., retaining nodes with at least five neighbors in the knowledge graph. The codes for generating this healthcare knowledge graph from MIMIC III dataset are also available at 5.

4.1.2 Comparison Methods. We compare our model against the following state-of-the-art network embedding learning methods. The first four methods learn static embeddings and the remaining methods learn contextual embeddings.

- **TransE** [3] treats the relations between nodes as the translation operations in a low-dimensional embedding space.

- **ReFe** [6] incorporates hyperbolic space and attention-based geometric transformations to learn the logical patterns of networks.

- **node2vec** [12] is a random-walk based method that was developed for homogeneous networks.

- **metapath2vec** [10] is an extension of node2vec that constrains random walks to specified metapaths in heterogeneous network.

- **GAN** (Graph Attention Networks) [1] is a graph attention network for learning node embeddings based on the attention distribution over the graph walk context.

- **GATNE-T** (General Attributed Multiplex Heterogeneous Network Embedding) [5] is a metapath-constrained random-walk based method that learns relation-specific embeddings.

- **RGCN** (Relational Graph Convolutional Networks) [26] learns multi-relational data characteristics by assigning a different weight for each relation.

- **CompGCN** (Composition-based Graph Convolutional Networks) [29] jointly learns the embedding of nodes and relations for heterogeneous graph and updates a node representation with multiple composition functions.

- **HGT** (Heterogeneous Graph Transformer) [16] models the heterogeneity of graph by analyzing heterogeneous attention over each edge and learning dedicated embeddings for different types of edges and nodes. We adapt the released implementation of node classification task to perform link prediction task.

- **asp2vec** (Multi-aspect network embedding) [21] captures the interactions of the pre-defined multiple aspects with aspect regularization and dynamically assigns a single aspect for each node based on the specific local context.

4.1.3 Experimental Settings. All evaluations were performed using NVIDIA Tesla P100 GPUs. The results of SLiCE are evaluated under the parameter settings described in Section 3.6. The results of all baselines are obtained with their original implementations. Note that for all baseline methods, the parameters not specially specified here are under the default settings. We use the implementation provided in KGEmb7 for both TransE and ReFe. node2vec8 is implemented by sampling 10 random walks with a length of 80. The original implementation of metapath2vec9 is used by generating 10 walks for each node as well. We set the learning rate to be [0.1, 0.01, 0.001] and reported the best performance for GAN10. GATNE-T is implemented by generating 20 walks with a length of 10 for each node. The results of CompGCN11 are obtained using the multiplicative composition of node and relation embeddings. We adapt the Deep Graph Library (DGL) based implementation12 of HGT and RGCN to perform the link prediction task. We use the original implementation of asp2vec13 for the evaluation. For all baselines, the dimension of embedding is set to 128.

---

6https://github.com/HazyResearch/CompGCN
7https://github.com/THUDM/GATNE/tree/master/data
8https://github.com/malllabiisc/CompGCN
9https://ericdongyx.github.io/metapath2vec/m2v.html
10https://github.com/google-research/google-research/tree/master/graph_embedding/watch_your_step
11https://github.com/malllabiisc/CompGCN
12https://github.com/dmlc/dgl/tree/master/examples/pytorch/hgt
13https://github.com/poly1302/asp2vec
Table 3: Performance comparison of different models on link prediction task using micro-F1 score and AUCROC. The symbol “OOM” indicates out of memory. Here, SLiCE_{w/o GF} and SLiCE_{w/o FT} represent two variants of the proposed SLiCE method by removing the Global Feature (GF) initialization and without fine-tuning (FT), respectively. The symbol * indicates that the improvement is statistically significant over the best baseline based on two-sided t-test with p-value $10^{-10}$.

<table>
<thead>
<tr>
<th>Type</th>
<th>Methods</th>
<th>micro-F1 Score</th>
<th>AUCROC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Amazon</td>
<td>DBLP</td>
</tr>
<tr>
<td>Static</td>
<td>TransE</td>
<td>50.28</td>
<td>45.60</td>
</tr>
<tr>
<td></td>
<td>RefE</td>
<td>51.86</td>
<td>49.60</td>
</tr>
<tr>
<td></td>
<td>node2vec</td>
<td>88.06</td>
<td>86.71</td>
</tr>
<tr>
<td></td>
<td>metapath2vec</td>
<td>88.86</td>
<td>84.58</td>
</tr>
<tr>
<td>Contextual</td>
<td>GAN</td>
<td>85.47</td>
<td>OOM</td>
</tr>
<tr>
<td></td>
<td>GATNE-T</td>
<td>89.06</td>
<td>57.04</td>
</tr>
<tr>
<td></td>
<td>RGCN</td>
<td>65.03</td>
<td>28.84</td>
</tr>
<tr>
<td></td>
<td>CompGCN</td>
<td>83.42</td>
<td>40.10</td>
</tr>
<tr>
<td></td>
<td>HGT</td>
<td>65.77</td>
<td>53.32</td>
</tr>
<tr>
<td></td>
<td>asp2vec</td>
<td>94.89</td>
<td>78.82</td>
</tr>
<tr>
<td></td>
<td>SLiCE_{w/o GF}</td>
<td>67.01</td>
<td>66.02</td>
</tr>
<tr>
<td></td>
<td>SLiCE_{w/o FT}</td>
<td>94.99</td>
<td>89.34</td>
</tr>
<tr>
<td></td>
<td>SLiCE (Ours)</td>
<td>96.00*</td>
<td>90.70*</td>
</tr>
</tbody>
</table>

4.2 Evaluation on Link Prediction (RQ1)

We evaluate the impact of contextual embeddings using the binary link prediction task, which has been widely used to study the structure-preserving properties of node embeddings [7, 37].

Table 3 provides the link prediction results of different methods on five datasets using micro-F1 score and AUCROC. The prediction scores for SLiCE are reported from the context subgraph generation strategy (shortest path or random) that produces the best score for each dataset on the validation set. Compared to the state-of-the-art methods, we observe that SLiCE significantly outperforms both static and contextual embedding learning methods by 11.95% and 25.57% on average in F1-score, respectively. We attribute static methods’ superior performance, compared to relation based contextual learning methods (such as GATNE-T, RGCN, and CompGCN), to the ability of capturing global network connectivity patterns. Relation based contextual learning methods limit node contextualization by emphasizing the impact of relations on nodes. We outperform all methods on F1-score, including asp2vec, a cluster-aspect based contextualization method. Asp2vec achieves a marginally better AUCROC score on Freebase, but SLiCE achieves a better F1-score.

SLiCE outperforms asp2vec on all other datasets (in F1-score and AUCROC measure), improving F1-score for DBLP by 13% owing to its ability to learn important metapaths without explicit specification. These results indicate that subgraph based contextualization is highly effective and is a strong candidate for advancing the state-of-the-art for link prediction in a graph network.

4.3 SLiCE Model Interpretation (RQ2)

Here we study the impact of using different subgraph contexts on link prediction performance and demonstrate SLiCE’s ability to learn important higher-order relations in the graph. Our analysis shows SLiCE’s results are highly interpretable, and provide a way to perform explainable link prediction by learning the relevance of different context subgraphs connecting the query node pair.

4.3.1 Case Study for Model Interpretation. Figure 3 shows an example subgraph from DBLP dataset between “Jiawei Han” and “P28406”, a paper on frequent itemset mining. Paths 1 to 5 (shown in different legend) show different contexts subgraphs present between the two nodes. We observe that the link prediction score between Jiawei Han and paper P28406 varies, depending on the context subgraph provided to the model.

We also observe that SLiCE assigns high link prediction scores for all context subgraphs (paths 1-4) where all the nodes on the path share strong semantic association with other nodes in the context subgraph. The lowest score is achieved for path-5 which contains a conference node C19054 (SIAM Data Mining). As a conference
publishes papers in multiple topics, we hypothesize that this breaks
the semantic association across nodes, and consequently lowers
the probability of the link compared to other context subgraphs
where all nodes are closely related.

It is important to note, that a node can share a semantic associa-
tion with another node separated by multiple hops on the path, and
thus would be associated via a higher-order relation. We explore
this concept further in the following subsections.

4.3.2 Interpretation of Semantic Association Matrix. We pro-
vide the visualization of the semantic association matrix $A^k_{ij}$ as
defined in Eq. (1) to investigate how the node dependencies evolve
through different layers in SLiCE. Given a node pair $(v_i, v_j)$ in the
context subgraph $g_k$, a high value of $A^k_{ij}$, indicates a strong global
dependency of node $v_i$ on $v_j$. While a high value of $A^k_{ij} (k \geq 1)$
(the association after applying more translation layers) indicates a
prominent high-order relation in the subgraph context.

Figure 4 shows weights of semantic association matrix for the
context generated for node pair (N0: Summarizing itemset patterns:
a profile-based approach (Paper), N1: Jiawei Han (Author)). Nodes
in the context consist of N2: Approach (Topic), N3: Knowledge-base
reduction (Paper), N4: Redundancy (Topic) and N5: Handling Redan-
dancy in the Processing of Recursive Database Queries (Paper). We
observe that at layer 1 (Figure 4a), the association between source
node N0 and target node N1 is relatively low. Instead, they both
assign high weights on N4. However, the dependencies between
nodes are dynamically updated when applying more learning lay-
ers, consequently enabling us to identify higher-order relations. For
example, the dependency of N1 on N0 becomes higher from layer
3 (Figure 4c) and N0 primarily depends on itself without highly
influenced by other nodes in layer 4 (Figure 4d). This visualization
of semantic association helps to understand how the global em-
bedding is translated into the localized embedding for contextual
learning.

4.3.3 Symbolic Interpretation of Semantic Associations via
Metapaths. Metapaths provide a symbolic interpretation of the
higher-order relations in a heterogeneous graph. We analyze the
ability of SLiCE to learn relevant metapaths that characterize posi-
tive semantic associations in the graph. We observe from Table 4
that SLiCE is able to match existing metapaths and also identify
new metapath patterns for prediction of each relation type. For
example, to predict the paper-author relationship, SLiCE learns
three shortest metapaths, including "TPA" (authors publish with
the same topic), "APA" (co-authors) and "CPA"(authors published
in the same conference).

Interestingly, our learning suggests that longer metapath "APCPA",
which is commonly used to sample academic graphs for co-author
relationship, is not as highly predictive of a positive relationship, i.e.,
"all authors who publish in the same conference do not necessarily
publish together". Overall, the metapaths reported in Table 4 are
consistent with the top ranked paths in Figure 3. These metapaths
demonstrate SLiCE’s ability to discover higher order semantic asso-
ciations and perform interpretable link prediction in heterogeneous
networks.

4.4 Effectiveness of Contextual Translation for
Link Prediction (RQ3)
In this section, we study the impact of contextual translation on
node embeddings. First, we evaluate the impact of contextualization
in terms of the similarity (or distance) between the query nodes. Sec-
ond, we analyze the effectiveness of contextualization as a function
of the query node pair properties. The latter is especially relevant
for understanding the performance boundaries of the contextual
methods.

4.4.1 Impact of Contextual Translation on Embedding-based
Similarity. Figure 5 provides the distribution of similarity scores
for both positive and negative edges obtained by SLiCE. We com-
pare against embeddings produced by node2vec [12] which is one
of the best performing static embedding methods (see Table 3)
and CompGCN [29] a relation based contextualization method.
We observe that for node2vec and CompGCN, the distribution
of similarity scores across positive and negative edges overlaps
significantly for all datasets. This indicates that the embeddings
learnt from global methods or relation specific contextualization
cannot efficiently differentiate the positive and negative edges in
link prediction task.

On the contrary, SLiCE increases the margin between the distribu-
tions of positive and negative edges significantly. It brings node
embeddings in positive edges closer and shifts nodes in negative

<table>
<thead>
<tr>
<th>Learning Methods</th>
<th>Paper-Author</th>
<th>Paper-Conference</th>
<th>Paper-Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predefined [35]</td>
<td>TPA, APA, CPA</td>
<td>TPC, APC, TPTPC</td>
<td>TPT, CPT, APT</td>
</tr>
<tr>
<td>SLiCE + Shortest Path</td>
<td>APA, APAPA</td>
<td>TPTPC, TAPAC</td>
<td>TPTPT, APTPT</td>
</tr>
<tr>
<td>SLiCE + Random</td>
<td>APA, APA</td>
<td>APA, APA</td>
<td>APA, APA</td>
</tr>
</tbody>
</table>

Figure 4: Visualization of the semantic association matrix (after normalization) learnt from different layers on a DBLP sub-
graph for link prediction between paper N0 and author N1. An intense color indicates a higher association. Initially (layer 1),
nodes N0 and N1 have low association. In layer 4, SLiCE learns higher semantic association from N1 to N0.
edges farther away in the low-dimensional space. This indicates that the generated subgraphs provide informative contexts during link prediction and enhance embeddings such that it improves the discriminative capability of both positive and negative node-pairs.

4.4.2 Error Analysis of Contextual Methods as a function of Node Properties. We investigate the performance of SLiCE and the closest performing contextual learning method, apsp2vec [21], as a function of query node pair properties. Given each method, we select all query node pairs drawn from both positive and negative samples that are associated with an incorrect prediction. Next, we compute the average degree of the nodes in each such pair. We opt for degree and ignore any type constraint for its simplicity and ease of interpretation. Fig. 6 shows the distribution of these incorrect predictions as a function of average degree of query node pair. It can be seen that, for Amazon and Healthcare datasets, most of the incorrect predictions are concentrated around the query pairs with low and medium values of average degree. However, SLiCE has fewer errors than apsp2vec for such node pairs. This can be attributed to the aspect-oriented nature of apsp2vec, which maps each node to a fixed number of aspects. Since nodes in a graph may demonstrate varying degree of aspect-diversity, mapping a node with low diversity to more aspects (than it belongs to) reduces apsp2vec’s performance. SLiCE adopts a complementary approach, where it considers the subgraph context that connects the query nodes, leading to better contextual representations.

4.5 Study of SLiCE (RQ4)

4.5.1 Parameter Sensitivity. In Figure 7, we provide the link prediction performance with micro-F1 score on four datasets by varying four parameters used in SLiCE model, including number of heads, number of contextual translation layers, number of nodes in contexts (i.e., walk length), and the number of (context) walks generated for each node in pre-training. The performance shown in these plots are the averaged performance by fixing one parameter and varying other three parameters.

On the other hand, when varying the parameter number of layers, we observe that applying four layers of contextual translation provides the best performance on all the datasets; the performance dropped significantly when stacking more layers. This indicates that four contextual translation layers are sufficient to capture the complex higher-order relations over various knowledge graphs. Based on these analysis, we set the default values for both number of heads and the number of layers to be 4, and generate one walk for each node with a length of 6 in the pre-training step.

4.5.2 Effect of Pre-trained Global Features (GF). To explore the impact of pre-trained global node features on the performance

Figure 5: Distributions of similarity scores of both positive and negative node-pairs obtained by node2vec, CompGCN, and SLiCE over five datasets.

Figure 6: Error analysis of contextual translation based link prediction method as a function of degree-based connectivity of query nodes.
of SLiCE, we performed an ablation study by analyzing a variant of SLiCE with four contextual translation layers. More specifically, the pre-trained global embeddings are disabled in SLiCE, termed SLiCE$_{w/o}$GF. The results are provided in Table 3. We observe that without initialization using the pre-trained global embeddings, the model performance of SLiCE decreased on all five datasets in both metrics. The reason being that, compared to the random initialization, the global node features are able to represent the role of each node in the global structure of the knowledge graph. By further applying the proposed contextual translation layers, they can collaboratively and efficiently provide contextualized node embeddings for downstream tasks like link prediction.

4.5.3 Effect of Fine-tuning (FT). To investigate the effect of fine-tuning stage on learning the contextual node embeddings, we disable the fine-tuning layer for supervised link prediction task, termed SLiCE$_{w/o}$FT and show the results in Table 3. Compared to the baseline methods, it still achieves competitive performance. We attribute this to the effectiveness of capturing higher-order relations through the contextual translation layers. While compared to the full SLiCE model, the performance of SLiCE$_{w/o}$FT degrades slightly on Amazon, DBLP, and Freebase datasets, but significantly decreases on both Twitter and MIMIC datasets. This can be attributed to the fact that supervised training with the link prediction task is able to learn the fine-grained contextual node embedding for link prediction task.

4.6 SLiCE Model Complexity (RQ5)

Contextual learning methods are known to have high computational complexity. In section 3.5 we observed that the cost of training SLiCE is approximately linear to the number of edges. In this section, we provide an empirical evaluation of the model scalability in view of the prior analysis.

4.6.1 Time Complexity Analysis. In this subsection, we mainly investigate the impact of the following three parameters on the overall time complexity of the SLiCE model: (1) number of contextual translation layers, (2) number of context subgraphs, and (3) length of context subgraph.

- We study the scalability of the SLiCE model when the number of context translation layers are varied and the corresponding plots are provided in Figure 8(a). The x-axis and y-axis represent the number of layers and running time in seconds, respectively. The plots indicate that increasing the number of layers does not significantly increase the training time of the SLiCE model.

- In Figure 8(b), we demonstrate the impact of the number of context subgraphs on the time complexity. Increasing the number of context subgraphs generated for each node in pre-training and each node-pair for fine-tuning raises the number of training edges which further increases the training time of the model.

These two plots empirically verify the analysis about the model complexity, discussed in Section 3.5, that the proposed SLiCE model is approximately linear to the number of edges in the graph and does not depend on other parameters such as the number of contextual translation layers and the number of nodes in the graph. In addition, we also vary the length (number of nodes) of the context subgraph in both plots. The plot shows that even doubling the context length will not significantly increase the running time. This time complexity analysis, combined with the performance results in Table 3 and the parameter sensitivity analysis in Figure 7 can jointly provide the guidelines for parameter selection.

4.6.2 Context Subgraph Sampling Analysis. In the complexity analysis discussed in Section 3.5, we approximated the total number of training edges in the entire graph as $N_E = |V| \cdot N_{cpn}$, where $|V|$ represents the number of nodes in graph $G$ and $N_{cpn}$ denotes the number of context subgraphs generated for each node. This estimation also provides us guidelines for determining the
number of context subgraphs for each node $N_{cpn}$. By incorporating $N_{E} = \alpha |E|$ into the approximation (where $|E|$ is the total number of edges in graph $G$ and $\alpha$ is the ratio of training split), we can estimate the number of context subgraphs per node as $N_{cpn} = \alpha |E|/|V|$. Table 5 shows the estimated numbers (with $\alpha = 0.6$) for the five datasets used in this work. These estimations provide us an approximate range for the value of $N_{cpn}$ during the context generation step. Based on this analysis, in our experiments, we generally consider 1, 5, and 10 for the value of $N_{cpn}$ on all the five datasets in both pre-training and fine-tuning stages.

5 CONCLUSIONS

We introduce SLiCE framework for learning contextual subgraph representations. Our model brings together knowledge of structural information from the entire graph and then learns deep representations of higher-order relations in arbitrary context subgraphs. SLiCE learns the composition of different metapaths that characterize the context for a specific task in a drastically different manner compared to existing methods which primarily aggregate information from either direct neighbors or semantic neighbors connected via certain pre-defined metapaths. SLiCE significantly outperforms several competitive baseline methods on various benchmark datasets for the link prediction task. In addition to demonstrating SLiCE’s interpretability and scalability, we provide a thorough analysis on the effect of contextual translation for node representations. In summary, we show SLiCE’s subgraph-based contextualization approach is effective and distinctive over competing methods.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grant IIS-1838730, Amazon AWS cloud computing credits, and Pacific Northwest National Laboratory under DOE-VA-21831018920.

REFERENCES

[28] Fan-Yun Sun, Meng Qiu, Jordan Hoffmann, Chin-Wei Huang, and Jie Tang. 2019. vGraph: A generative model for joint community detection and node representation learning. In NeurIPS.