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 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.

CCS CONCEPTS

• Mathematics of computing → Graph algorithms; **• Computing methodologies → Learning latent representations**; **Neural networks**.

KEYWORDS

Heterogeneous networks, network embedding, self-supervised learning, link prediction, semantic association

ACM Reference Format:
Ping Wang[1], Khushbu Agarwal[2], Colby Ham[2], Sutanay Choudhury[2], Chandan K. Reddy[1]. 2021. Self-Supervised Learning of Contextual Embeddings for Link Prediction in Heterogeneous Networks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450060

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 $(Author_1, co-author, Author_2)$ 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

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450060

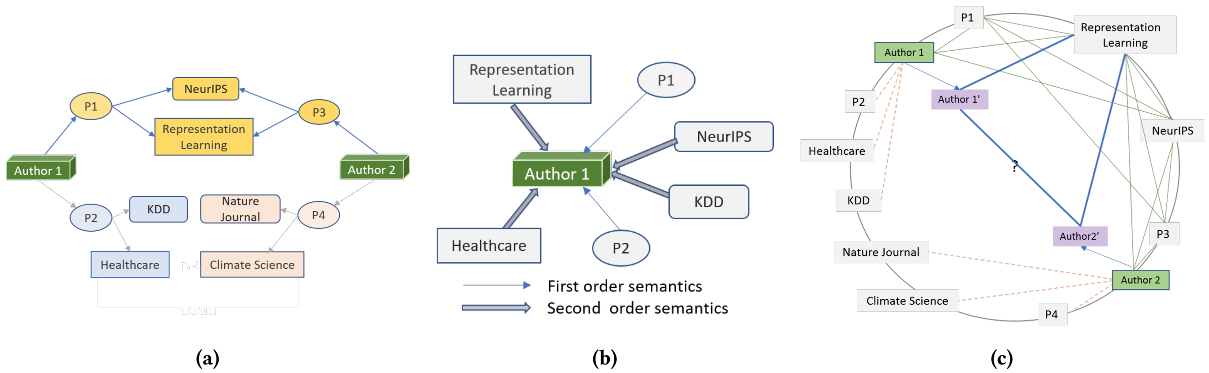


Figure 1: Subgraph driven contextual learning in an academic network. (a) Author nodes publish on diverse topics (participate in diverse contexts) (b) State-of-the-art methods aggregate global semantics for authors based on all published papers (c) Our approach uses context subgraph between authors to contextualize their node embeddings during link prediction.

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_c" 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 Δ_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 Δ_u where $u \in V(g_c)$. 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 metapaths, 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_c 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".

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

based methods [4] seek to learn a representation Γ that minimizes a loss function of the form $||\Gamma^T\Gamma - P_V||^2$, 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 $\sum_{v_i,v_j \in V(G)} -log(p_L(v_j|v_i))$, where $p_L(v_j|v_i)$ is the probability of visiting a node v_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 $G = (V, E, \Sigma_V, \Sigma_E, \lambda_v, \lambda_e)$ 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. λ_v and λ_e are functions mapping the node (or edge) to its node (or edge) type Σ_V and Σ_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 δ is a threshold.

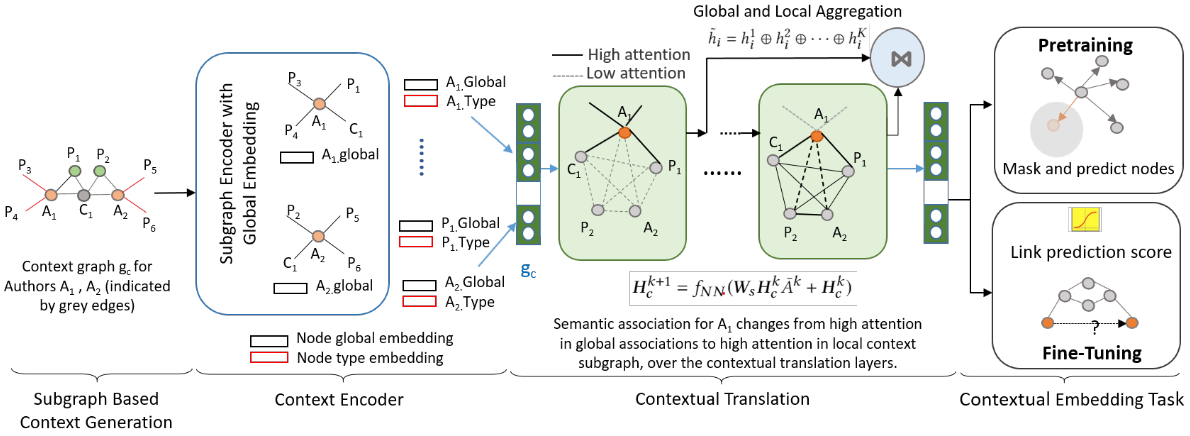

P3
P1
P2
P5
P4
A1
C1
A2
P6
Context graph g_c for Authors A1 , A2 (indicated by grey edges)
Subgraph Encoder with Global Embedding
A1 global
A2 global
A1 Global
A1 Type
P1 Global
P1 Type
A2 Global
A2 Type
Node global embedding
Node type embedding
g_c
High attention
Low attention
Global and Local Aggregation
$\tilde{h}_i = h_i^1 \oplus h_i^2 \oplus \cdots \oplus h_i^K$
$H_c^{k+1} = f_{NN}(W_s H_c^k \bar{A}^k + H_c^k)$
Semantic association for A1 changes from high attention in global associations to high attention in local context subgraph, over the contextual translation layers.
Pretraining
Mask and predict nodes
Link prediction score
?
Fine-Tuning
Subgraph Based Context Generation
Context Encoder
Contextual Translation
Contextual Embedding Task

Figure 2: Overview of SLiCE architecture. Subgraph context is initialized using global features for each node. Each layer in SLiCE shifts the embedding of all nodes in g_c to emphasize the local dependencies in the contextual subgraph. The final embeddings for nodes in context subgraphs are determined as a function of output from last i layers to combine global with local contextual semantics for each node.

Overview of SLiCE. In this paper, the proposed SLiCE framework 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 [37], knowledge base completion [27] or multi-hop reasoning [8, 13].

In our experiments, context subgraphs are generated for each node v 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 $g_c = (v_1, v_2, \cdots, v_{|V_c|})$, 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 $h_i = \boldsymbol{W_e} f_{attr}(v_i)$, where $f_{attr}(.)$ is a function that returns a stacked vector containing the structure-based embedding of v_i and the embedding of its attributes. $\boldsymbol{W_e}$ is the learnable embedding matrix. We represent the input node embeddings in g_c as $\boldsymbol{H_c} = (h_1, h_2, \cdots, h_{|V_c|})$. 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 $\boldsymbol{H_c} \in \mathbb{R}^{d \times |V_c|}$, 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 $\bar{A}$, 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 $\boldsymbol{H}_c^k = (h_1^k, h_2^k, \cdots, h_{|V_c|}^k)$ to be $\boldsymbol{H}_c^{k+1}$.

$$\boldsymbol{H}_c^{k+1} = f_{NN}(\boldsymbol{W}_s \boldsymbol{H}_c^k \bar{A}^k + \boldsymbol{H}_c^k) \tag{1}$$

where f_{NN} is a non-linear function and the transformation matrix $\boldsymbol{W}_s \in \mathbb{R}^{d\times d}$ 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_i and v_j in context subgraph g_c, the corresponding entry $\bar{A}_{ij}^k$ 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 $\bar{A}_{ij}^k$ as follows:

$$\bar{A}_{ij}^k = \frac{exp\left((\boldsymbol{W}_1 h_i^k)^T(\boldsymbol{W}_2 h_j^k)\right)}{\sum_{t=1}^{|V_c|} exp\left((\boldsymbol{W}_1 h_i^k)^T(\boldsymbol{W}_2 h_t^k)\right)} \tag{2}$$

where the transformation matrix $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ 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 $\{h_i^k\}_{k=1,\ldots,K}$ as the contextual embedding $\tilde{h}_i$ for each node as follows.

$$\tilde{h}_i = h_i^1 \oplus h_i^2 \oplus \cdots \oplus h_i^K \tag{3}$$

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($G, v, m$)  ▷ Generate context subgraph
5      $g_c^v = Encode(g_c^v)$        ▷ Encode context as a sequence
6      $v_m = Random(g_c^v)$     ▷ Mask a node in $g_c^v$ for prediction
7      $G_c^{pre}.Append(g_c^v, v_m)$
8  end
9  $\boldsymbol{H} \leftarrow EmbedFunction(G, d)$       ▷ Learn global embeddings
10 Initialize node embeddings as $\boldsymbol{H}^0 = \boldsymbol{H}$.
11 while not converged do
12     for $g_c^v$ to $G_c^{pre}$ do
13         for $k = 1$ to $K$ do
14             $\boldsymbol{H}_c^{k+1} = f_{NN}(\boldsymbol{W}_s \boldsymbol{H}_c^k \bar{A}^k + \boldsymbol{H}_c^k)$ with $\bar{A}$ calculated
                 by Eq. (2)
15         end
16         Contextual embeddings $\tilde{\boldsymbol{H}}_c = \boldsymbol{H}_c^1 \oplus \boldsymbol{H}_c^2 \oplus \cdots \oplus \boldsymbol{H}_c^K$
17         Update parameters with the contextualized node
             prediction task using the objective function in Eq. (4).
18     end
19 end
```

Given a context subgraph g_c, the obtained contextual embedding vectors $\{\tilde{h}_i\}_{i=1,2,\ldots,|V_c|}$ 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 v_m based on the context g_c in the following form.

$$\theta = \arg\max_{\theta} \prod_{g_c \in G_C} \prod_{v_m \in g_c} p(v_m | g_c, \theta) \tag{4}$$

where θ 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

Algorithm 2: Fine-tuning in SLICE with link prediction.

```
1  Require: Graph $G$ with nodes set $V$ and edges set $E$, list of
     edges $E_{lp}$ for link prediction, global embeddings $\boldsymbol{H}$,
     pre-trained model parameters $\theta$, context subgraph size $m$,
     No. of translation layers $K$.
2  Fine-tuning dataset $G_c^{fine} \leftarrow \emptyset$
3  for each $e \in E_{lp}$ do
4      $g_c^e$ =GetContext($G$, $e$, $m$)  ▷ Generate context subgraph
5      $g_c^e = Encode(g_c^e)$             ▷ Encode context as a sequence
6      $G_c^{fine}.Append(g_c^e)$
7  end
8  Initialize node embeddings as $\boldsymbol{H}^0 = \boldsymbol{H}$.
9  Set model parameters to pre-trained parameters $\theta$ in
     Algorithm 1.
10 while not converged do
11     for $g_c^e$ to $G_c^{fine}$ do
12         for $k = 1$ to $K$ do
13             $\boldsymbol{H}_c^{k+1} = f_{NN}(\boldsymbol{W}_s \boldsymbol{H}_c^k \bar{A}^k + \boldsymbol{H}_c^k)$ with $\bar{A}$ calculated
                 by Eq. (2)
14         end
15         Contextual embeddings $\tilde{\boldsymbol{H}}_c = \boldsymbol{H}_c^1 \oplus \boldsymbol{H}_c^2 \oplus \cdots \oplus \boldsymbol{H}_c^K$
16         Update fine-tuning parameters using Eq. (5).
17     end
18 end
```

in line 16 and update the model parameters with contextual node prediction task. In a departure from traditional skip-gram methods [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 predicted 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.

$$\mathcal{L} = \sum_{(e_p, g_{cp}) \in D_p} \log(P(e_p | g_{cp}, \theta)) + \sum_{(e_n, g_{cn}) \in D_n} \log(1 - P(e_n | g_{cn}, \theta)) \tag{5}$$

We compute the probability of the edge between two nodes $e = (v_i, v_j)$ as the similarity score $S(v_i, v_j) = \sigma(\tilde{h}_i^T \cdot \tilde{h}_j)$ [1], where $\tilde{h}_i$ and $\tilde{h}_j$ are contextual embeddings of v_i and v_j 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 G. Then, the total number of context subgraphs considered in pre-training stage can be estimated as $|V| * N_{cpn}$ and the cost of iterating over all these subgraphs through multiple epochs will be $O(|V| * N_{cpn})$. 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 $O(|V| * N_{cpn}) \approx O(N_E)$, where N_E is the number of edges in the training dataset. It can also be represented as $N_E = \alpha_T |E|$, where $|E|$ is the total number of edges in graph G and α_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 $O(|E|N_{mc}^2)$. 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.

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

(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 also introduce a new knowledge graph from the publicly available real-world Medical Information Mart for Intensive Care (MIMIC) III dataset[2] 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 is provided below.

- **Amazon[3]:** 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[4]:** includes the relationships between papers, authors, venues, and terms. The edge types include paper_has_term, published_at and has_author.
- **Freebase[5]:** 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**[3]: 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[6].

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 KGEmb[7] for both **TransE** and **RefE**. **node2vec**[8] is implemented by sampling 10 random walks with a length of 80. The original implementation of **metapath2vec**[9] 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 **GAN**[10]. **GATNE-T** is implemented by generating 20 walks with a length of 10 for each node. The results of **CompGCN**[11] are obtained using the multiplicative composition of node and relation embeddings. We adapt the Deep Graph Library (DGL) based implementation[12] of **HGT** and **RGCN** to perform the link prediction task. We use the original implementation of **asp2vec**[13] for the evaluation. For all baselines, the dimension of embedding is set to 128.

[2] https://mimic.physionet.org/
[3] https://github.com/THUDM/GATNE/tree/master/data
[4] https://github.com/Jhy1993/HAN/tree/master/data
[5] https://github.com/malllabiisc/CompGCN/tree/master/data_compressed
[6] https://github.com/pnnl/SLICE

[7] https://github.com/HazyResearch/KGEmb
[8] https://github.com/aditya-grover/node2vec
[9] 0https://ericdongyx.github.io/metapath2vec/m2v.html
[10] https://github.com/google-research/google-research/tree/master/graph_embedding/watch_your_step
[11] https://github.com/malllabiisc/CompGCN
[12] https://github.com/dmlc/dgl/tree/master/examples/pytorch/hgt
[13] https://github.com/pcy1302/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}.

Type	Methods	micro-F1 Score					AUCROC				
		Amazon	DBLP	Freebase	Twitter	Healthcare	Amazon	DBLP	Freebase	Twitter	Healthcare
Static	**TransE**	50.28	49.60	47.78	50.60	48.42	50.53	49.05	48.18	50.26	49.80
	RefE	51.86	49.60	50.25	48.55	47.96	51.74	48.50	50.41	49.28	50.73
	node2vec	88.06	86.71	83.69	72.72	71.92	94.48	93.87	89.77	80.48	79.42
	metapath2vec	88.86	44.58	77.18	66.73	62.64	95.42	38.41	84.33	72.16	69.11
Contextual	**GAN**	85.47	OOM	OOM	85.01	81.94	92.86	OOM	OOM	92.39	89.72
	GATNE-T	89.06	57.04	OOM	68.16	58.02	94.74	58.44	OOM	72.07	73.40
	RGCN	65.03	28.84	OOM	63.46	56.73	74.77	50.35	OOM	64.35	46.15
	CompGCN	83.42	40.10	65.39	40.75	39.84	90.14	34.04	72.01	39.86	38.03
	HGT	65.77	53.32	OOM	53.13	76.54	68.66	50.85	OOM	59.32	82.36
	asp2vec	94.89	78.82	90.02	88.29	85.46	98.51	92.51	**96.61**	95.00	92.97
	SLICE$_{w/o\ GF}$	67.01	66.02	66.31	67.07	60.88	62.87	57.52	55.31	66.69	63.11
	SLICE$_{w/o\ FT}$	94.99	89.34	90.01	82.19	81.58	98.66	96.07	96.33	90.38	89.51
	SLICE (Ours)	**96.00***	**90.70***	**90.26**	**89.30***	**91.64***	**99.02***	**96.69***	96.41	**95.73***	**94.94***

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' 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

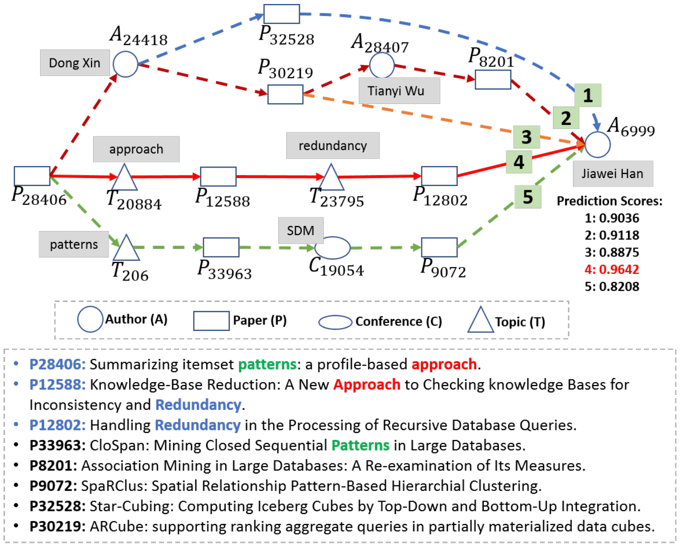
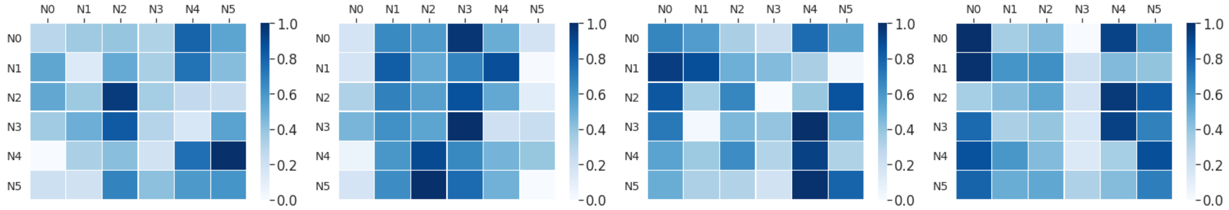

A_{24418}
Dong Xin
P_{32528}
A_{28407}
P_{30219}
Tianyi Wu
P_{8201}
1
2
3
4
5
A_{6999}
Jiawei Han
approach
redundancy
P_{28406}
T_{20884}
P_{12588}
T_{23795}
P_{12802}
patterns
SDM
T_{206}
P_{33963}
C_{19054}
P_{9072}
Prediction Scores:
1: 0.9036
2: 0.9118
3: 0.8875
4: 0.9642
5: 0.8208
Author (A)
Paper (P)
Conference (C)
Topic (T)
P28406: Summarizing itemset patterns: a profile-based approach.
P12588: Knowledge-Base Reduction: A New Approach to Checking knowledge Bases for Inconsistency and Redundancy.
P12802: Handling Redundancy in the Processing of Recursive Database Queries.
P33963: CloSpan: Mining Closed Sequential Patterns in Large Databases.
P8201: Association Mining in Large Databases: A Re-examination of Its Measures.
P9072: SpaRClus: Spatial Relationship Pattern-Based Hierarchial Clustering.
P32528: Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration.
P30219: ARCube: supporting ranking aggregate queries in partially materialized data cubes.

Figure 3: An example from DBLP. Relations in DBLP include paper-author, paper-topic and paper-conference. To predict the paper-author relationship between P_{28406} and A_{6999}, five context subgraphs are generated with a beam-search strategy. The $4th$ context subgraph (along with context subgraph 1, 2 and 3) contain closely related nodes and get high scores, while path 5 containing a generic conference node achieves lowest score.

"P_{28406}", 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 P_{28406} 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 C_{19054} (SIAM Data Mining). As a conference



(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Figure 4: Visualization of the semantic association matrix (after normalization) learnt from different layers on a DBLP subgraph 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.

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 association 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 provide the visualization of the semantic association matrix $\bar{A}^k_{ij}$ 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_c, a high value of $\bar{A}^0_{ij}$, indicates a strong global dependency of node v_i on v_j. While a high value of $\bar{A}^k_{ij} (k \geq 1)$ (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 Redundancy 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 layers, 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 embedding 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 positive 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).

Table 4: Comparisons of metapaths learned by SLICE with both predefined and model learned on DBLP dataset for each relation type. Here, P, A, C, and T represent Paper, Author, Conference, and Topic, respectively.

Learning Methods	Paper-Author	Paper-Conference	Paper-Topic
Predefined [35]	APCPA, APA	-	-
SLICE + Shortest Path	TPA, APA, CPA	TPC, APC, TPTPC	TPT, CPT, APT
SLICE + Random	APA, APAPA	TPTPC, TPAPC	TPTPT, APTPT

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 associations and perform interpretable link prediction in heterogeneous networks.

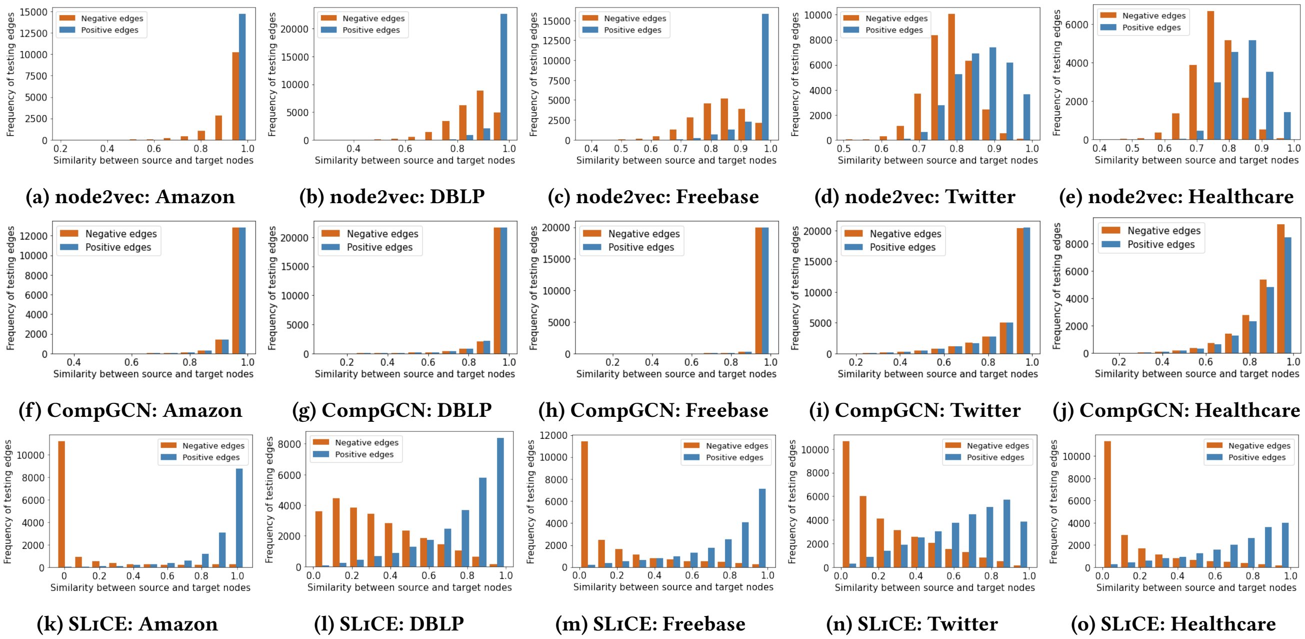
4.4 Effectiveness of Contextual Translation for Link Prediction (RQ3)

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. Second, 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 compare 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 distributions of positive and negative edges significantly. It brings node embeddings in positive edges closer and shifts nodes in negative

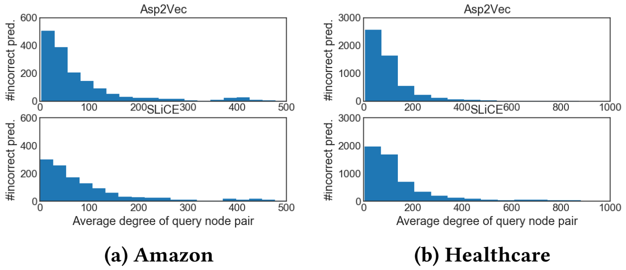


(a) node2vec: Amazon (b) node2vec: DBLP (c) node2vec: Freebase (d) node2vec: Twitter (e) node2vec: Healthcare

(f) CompGCN: Amazon (g) CompGCN: DBLP (h) CompGCN: Freebase (i) CompGCN: Twitter (j) CompGCN: Healthcare

(k) SLICE: Amazon (l) SLICE: DBLP (m) SLICE: Freebase (n) SLICE: Twitter (o) SLICE: Healthcare

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



(a) Amazon (b) Healthcare

Figure 6: Error analysis of contextual translation based link prediction method as a function of degree-based connectivity of query nodes.

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.

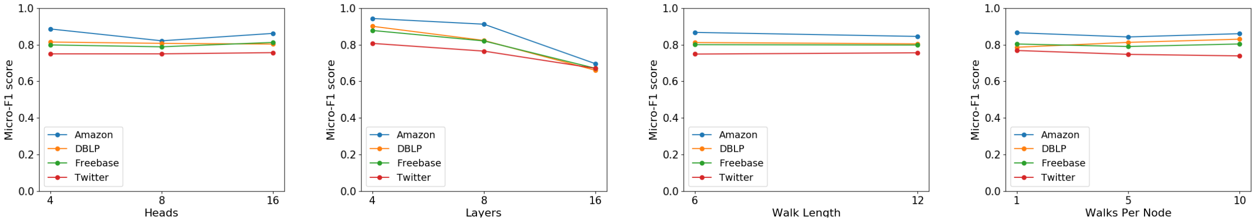
4.4.2 ***Error Analysis of Contextual Methods as a function of Node Properties.*** We investigate the performance of SLICE and the closest performing contextual learning method, asp2vec [21], as a function of query node pair properties. Given each method, we select all query node pairs drawn from both 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 it's 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 asp2vec for such node pairs. This can be attributed to the *aspect-oriented* nature of asp2vec, 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 asp2vec'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.

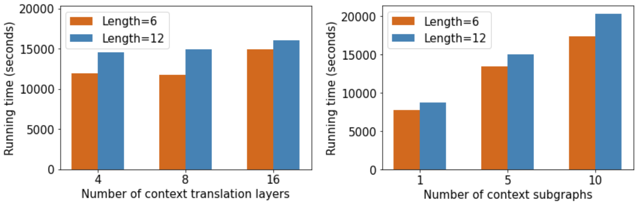
On the other hand, when varying the parameter *number of layers*, we observe that applying four layers of contextual translation provides the best performance on all the datasets; the performance dropped significantly when stacking more layers. This indicates that four contextual translation layers are 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



(a) (b) (c) (d)

Figure 7: Micro-F1 scores for link prediction with different parameters in SLICE on four datasets.



(a) (b)

Figure 8: Analysis of the time complexity of SLICE on Freebase dataset by varying (a) number of translation layers and (b) number of context subgraphs considered for each node.

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.

Table 5: Estimation of the number of context subgraphs for each node in the knowledge graph.

Dataset	Amazon	DBLP	Freebase	Twitter	Healthcare
# Nodes ($\|V\|$)	10,099	37,791	14,541	9,990	4,683
# Edges ($\|E\|$)	129,811	170,794	248,611	294,330	205,428
# Contexts (N_{cpn})	7.74	2.71	10.26	17.67	26.32

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 $N_E \approx |V| * N_{cpn}$, 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 $N_E = \alpha_T|E|$ into the approximation (where $|E|$ is the total number of edges in graph G and α_T is the ratio of training split), we can estimate the number of context subgraphs per node as $N_{cpn} = \alpha_T|E|/|V|$. Table 5 shows the estimated numbers (with $\alpha_T = 0.6$) 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

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. 2018. Watch your step: Learning node embeddings via graph attention. In *NeurIPS*.

[2] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty. 2020. Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM.

[5] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *SIGKDD*.

[6] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *Annual Meeting of the Association for Computational Linguistics*.

[7] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*.

[8] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks. In *EACL*.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[10] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[11] Alessandro Epasto and Bryan Perozzi. 2019. Is a single embedding enough? learning node representations that capture multiple social contexts. In *WWW*.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

[13] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *NeurIPS*.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[15] Yu He, Yangqiu Song, Jianxin Li, Cheng Ji, Jian Peng, and Hao Peng. 2019. Hetespaceywalk: a heterogeneous spacey random walk for heterogeneous information network embedding. In *CIKM*.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[19] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a single vector enough? exploring node polysemy for network embedding. In *SIGKDD*.

[20] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *International Conference on Machine Learning*.

[21] Chanyoung Park, Carl Yang, Qi Zhu, Donghyun Kim, Hwanjo Yu, and Jiawei Han. 2020. Unsupervised Differentiable Multi-aspect Network Embedding. In *SIGKDD*.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NeurIPS* (2017).

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

[24] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*. 2227–2237.

[25] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*.

[26] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[27] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*.

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

[29] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *ICLR*.

[32] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: An end-to-end framework for learning multiple conditional network representations of social network. In *SIGKDD*.

[33] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*.

[34] Liang Yang, Yuanfang Guo, and Xiaochun Cao. 2018. Multi-facet network embedding: Beyond the general solution of detection and representation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[35] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph Transformer Networks. In *NeurIPS*.

[36] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*.

[37] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*. 5165–5175.

[38] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *ICLR*.

[39] Wentao Zhang, Yuan Fang, Zemin Liu, Min Wu, and Xinming Zhang. 2020. mg2vec: Learning Relationship-Preserving Heterogeneous Graph Representations via Metagraph Embedding. *IEEE TKDE* (2020).

[40] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.