A Time-Aware Inductive Representation Learning Strategy for Heterogeneous Graphs

Bo Yan[*]
University of California, Santa Barbara
boyan@ucsb.edu

Matthew Walker
LinkedIn Corporation
mtwalker@linkedin.com

Krzysztof Janowicz
University of California, Santa Barbara
janowicz@ucsb.edu

ABSTRACT

Graphs are versatile data structures that have permeated a large number of application fields, such as biochemistry, knowledge graphs, and social networks. As a result, different graph representation learning models have been proposed as effective approaches to represent graph components in downstream machine learning tasks, such as node classification and recommendation. However, most representation learning models in graphs do not natively work on heterogeneous (multi-relational) graphs and consequently are not able to learn embeddings for different relations in the graph. In this paper, we extend and improve existing models by enabling an edge-based transformation procedure in order to learn embeddings for different relations in heterogeneous graphs. In addition, we show that by incorporating a sequential model to learn more expressive representations, temporal dynamics in social networks can be captured. Finally, we examine our model within the context of two very disparate heterogeneous graphs, a knowledge graph dataset and a professional social network dataset, to illustrate our point and show the effectiveness of our approach. By learning edge-based transformations, our model yields a Mean Reciprocal Rank score that is **more than 4 times** higher than the homogeneous counterpart for the knowledge graph dataset. By incorporating the temporal dynamics, our model improves the HITS@1 score by **more than 15%** compared with the baseline model for the professional social network dataset.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Machine learning**.

KEYWORDS

Graph Representation Learning, Social Networks, Heterogeneous Graphs, Temporal Dynamics

ACM Reference Format:
Bo Yan, Matthew Walker, and Krzysztof Janowicz. 2019. A Time-Aware Inductive Representation Learning Strategy for Heterogeneous Graphs. In *Proceedings of 15th International Workshop on Mining and Learning with Graphs (MLG'19).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.475/123_4

[*]This work was done when Bo Yan interned at LinkedIn.

1 INTRODUCTION

Representation learning on graphs, namely encoding nodes and edges in a vector space, has drawn an increasing amount of traction in both industry and research recently in parts due to the ubiquity of graphs in terms of diversity and quantity. While social networks are among the most relatable examples of graphs in everyday life, graphs are also the fundamental data structures for many other fields such as the molecular interaction network in biochemistry, the road network in transportation, and knowledge graph in data management. The sheer amount of information encoded in these graphs has also intrigued us — Facebook has more than 2 billion monthly active users (nodes), LinkedIn has more than 590 million members (nodes), and DBpedia (a crowd-sourced and community effort-based knowledge graph) has more then 18 million nodes and 1.7 billion triples, each of which is composed of a source node, an edge, and a target node.[1] In addition to the ubiquity of graphs, another major reason for the rise of graph representation learning is the proven efficacy of bottom-up machine learning algorithms in obtaining low-dimensional representations in languages [8, 18] and vision [15, 16].

Although existing methods may be applied, graph representation learning has its own challenges. Because of the versatility and uniqueness of graphs (e.g., graph isomorphism), there is no regular structure, such as the 1D sequence in natural languages and the 2D grid in images, to facilitate the modeling process. Instead, the structural information is inherently embedded in the various ways in which different nodes are connected with each other. Traditionally, this structural information is represented using summary statistics of the graph, such as clustering coefficients, or handcrafted features that capture the neighborhood structures. In many cases, the spectrum of the graph, i.e., the eigenvalues of the adjacency matrix or Laplacian matrix, has also been used in order to explore the connectivity structure. Besides the challenge of effectively and meaningfully capturing the intrinsic graph structure for better representing the graph components (such as nodes, edges, and subgraphs), more often than not, some extrinsic information associated with these graph components may play a significant role as well. For example, in order to empower friend recommendation features — *People You May Know* — on Facebook or career recommendation features — *Jobs You May Be Interested In* — on LinkedIn, information such as personal background, education, interests, and skills should be taken into consideration when learning representations for the

[1]The statistics for Facebook and LinkedIn are up-to-date as of 2018. The statistics for DBpedia are from the 2016-10 data dump.

friendship graph and the career network. Moreover, the structure of graphs that models information as flexible as careers or friendships evolves over time, e.g., as people move up the career ladder.

In light of this, we propose a time-aware inductive representation learning strategy for heterogeneous graphs that can be used for recommendations on social networks and apply it to the world's largest professional social network – LinkedIn — to recommend new career possibilities, education opportunities, and professional connections. This graph representation learning strategy considers both the intrinsic graph structure and the extrinsic knowledge about nodes and edges in the graph. For the intrinsic graph structure, we sample the local neighborhood nodes to inform the embeddings for center nodes. For extrinsic knowledge, we initialize the node and edge embeddings with prior knowledge, such as using information about a person's skills, education history, and employment history. Because of these mechanisms, our model is inductive, meaning the model is capable of generating embeddings for nodes unseen during the training phase, which is very useful for evolving graphs commonly encountered in social networks like LinkedIn. Moreover, unlike other graph embedding models [6, 11, 14] which are exclusively applied to graphs with one type of nodes and edges and only care about embedding nodes, our model aims to embed both nodes and edges into the same vector space so as to handle heterogeneous graphs. Figure 1 shows examples of a single-relation graph and a heterogeneous graph. Due to its generality, a heterogeneous graph is more expressive in terms of modeling interactions between different types of nodes such as members (users), careers, and institutions. In addition, our method considers the temporal component of the graph. Although the temporal aspect is normally neglected by graph embedding and recommendation models, studies [22, 23] have shown that temporal information is beneficial for modeling the dynamics and progression of professional growth, thus we would like to include such component in our model.

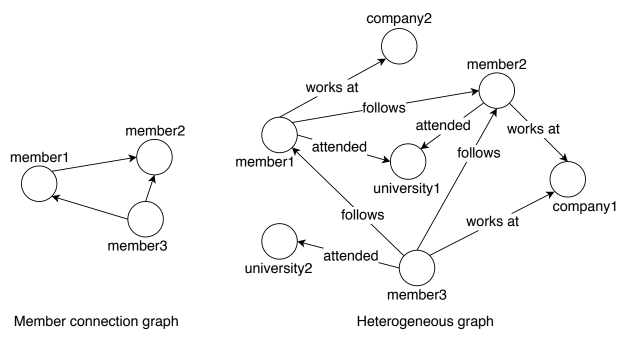

Figure 1: The connection graph on the left shows how member1, member2, and member3 are connected. In addition to expressing the connection information for these 3 members, the heterogeneous graph on the right also provides information about education and employment relations.

Contributions. Below, we identify the main contributions of our work:

- We propose a method for encoding different relation types in heterogeneous graphs as well as techniques to ensure those representations are learned efficiently.
- We propose capturing temporal dynamics within social networks by introducing a sequential model to understand the state evolution when evaluating graph entity representations.
- We evaluate our approach by performing experiments on two real-world datasets, which have different network characteristics, thus demonstrating the effectiveness of our approach across settings. In the evaluation on the knowledge graph dataset, by encoding different relation types, our model can improve the Mean Reciprocal Rank score by **more than 4 times**. In the evaluation on the social network dataset, by capturing the temporal component, our model can improve the HITS@1 score by **more than 15%**.

Outline. The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 describes our heterogeneous graph embedding model in detail. Section 4 presents the two major datasets over which we test our model and shows the evaluation results. Section 5 summarizes the presented research.

2 RELATED WORK

Graph Embeddings. Existing work on graph embeddings can be classified into three categories: factorization-based approach, random walk-based approach, and neighborhood aggregation and convolution-based approach. Inspired by techniques in dimensional reduction, matrix factorization-based approaches are among the early methods for graph embedding learning. Methodologies in this family include Laplacian eigenmaps method [3], graph factorization algorithm [2], GraRep [6], and High-Order Proximity preserved Embedding (HOPE) method [19]. Their major difference lies in the graph proximity measures in their models. In order to improve embedding efficiency and model performance, Random walk-based methods embrace the idea of stochasticity in measuring graph proximity. Perozzi et al. [20] showed that by performing truncated random walks, their DeepWalk model was able to learn latent representations of nodes by capturing local structural information. Node2Vec [10] justified their biased random walk algorithm by providing an intuitive explanation of the relationship between breadth-first search/depth-first search and structural equivalence/homophily in graphs. More recently, neighborhood aggregation and convolution-based approach has drawn a lot of attention for they provide consistent gains over node classification and link prediction benchmarks compared to other approaches [12]. The basic idea behind this approach is to generate node embeddings by aggregating local neighborhood information, which is considered a convolution operation because each node embedding is a function of the neighboring node embeddings. Kipf and Welling [14] developed a semi-supervised graph convolutional network model for node classification tasks. However, their model is not inductive and requires the full graph Laplacian. Hamilton et al. [11] used a sample-and-aggregate strategy (GraphSAGE) to learn node embeddings in graphs in an inductive manner. The GraphSAGE model has some major advantages over previous models: 1) it has a parameter sharing mechanism which allows more efficient learning process, 2) it leverages node attribute information which facilitates the inclusion of extrinsic knowledge about the graph, 3) it is inductive and can be effectively used in evolving graphs. Inspired by their model, we build upon their algorithm and attempt to fix the Achilles' heel of the model, i.e., only able to be applied to homogeneous graphs natively, by extending it to heterogeneous graphs and learn node and edge embeddings simultaneously.

Social Network Modeling. Our work also relates to existing work on modeling social networks. As mentioned before, although both intrinsic graph structure and extrinsic knowledge are important in the modeling process, most work only considered one aspect. For example, Adamic and Adar [1] used link structure and Liben-Nowell and Kleinberg [17] compared different proximity measures, such as graph distance, common neighbors, and Jaccard's coefficient, to predict new interactions in the network. Cetintas et al. [7] extracted keywords from user profiles in order to identify similar people in professional social networks. Xu et al. [23] modeled the career trajectories of different people in the social network by considering temporal information. This work is similar to ours in that it also acknowledges the importance of the temporal component in analyzing career progression. But their work did not model the interaction between different members in the professional social network, thus failing to take into account the intrinsic graph structure. Our model not only considers the temporal aspect, but also embraces the complementary strengths of intrinsic and extrinsic information.

3 REPRESENTATION LEARNING MODEL

In this section, we describe the basic workflow of our model using an encoder-decoder framework. We further break down each piece in the model and explain the strategy to incorporate relation translation. Finally, we discuss different aggregators and explain ways in which the temporal component could be incorporated.

3.1 Framework

In order to explain different components of our model in a more organized fashion, we follow the general encoder-decoder framework proposed by Hamilton et al. [12]. As shown in Figure 3, this model is an iterative process that generally contains 4 steps: 1) embedding initialization, 2) neighborhood sampling, 3) embedding aggregation, 4) affinity and loss calculation. For our model, the encoder uses sampling and aggregation strategies to encode nodes and edges by considering the neighborhood information; the decoder component calculates the affinity between each node defined using the graph adjacency matrix or the transition matrix in the random walk.

3.2 Encoder

In general, the purpose of the encoder component is to map each node and edge to low dimensional representations. In particular, for a graph $G = (V, E)$, each node $v \in V$ is mapped into a d dimensional vector and each edge type τ is mapped into a square matrix of dimension $d \times d$. The input to the encoder stage is the initialized embeddings for nodes and edges. Because the model takes into account extrinsic knowledge, information not inherently available from the graph structure is usually used to initialize the node embeddings. Depending on the actual dataset used, different initialization embeddings could be applied. For example, for knowledge graphs in which nodes are usually associated with labels, the initial node embeddings can be word embeddings for the node labels; for professional social networks where nodes could be members, the initial node embeddings can be derived from member background information and skill sets. A detailed explanation for embedding initialization with respect to our datasets is in Section 4. For edge embedding initialization, one-hot embeddings are applied.

Two major stages, namely sampling and aggregation, are applied in the encoder part. The intuition is that in order to consider the intrinsic graph structure in the model we gather information about the neighborhood to generate embeddings. However, some nodes in the graph may have thousands of (one-degree) neighbors while others may have very few, which makes indiscriminately aggregating neighboring nodes biased and inefficient. A simple but effective strategy is to sample the neighborhood. Instead of conducting uniform random sampling, for each node we sample its neighboring nodes based on the pairwise connectivity. This is distinct from undirected single-relation graphs, because in directed heterogeneous graphs (or directed multi-relational graphs) we are focusing on each pair of nodes may have multiple edges connecting each other and these edges can be of different directions. Nodes with a large number of edges connecting them are considered having high connectivities. Neighboring nodes that have higher connectivities with respect to center nodes will be sampled more frequently. After sampling the neighboring nodes, the model aggregates the embeddings of neighboring nodes using shared parameters to obtain the embeddings for center nodes. It is important to note that, by sampling neighboring nodes, the model also takes care of the associated edges (relations) in the graph, and in the process of aggregating neighboring nodes the model first applies a linear transformation using edge embeddings. In edge-based transformation, each relation type embeds a different semantic component in the heterogeneous information network. In the example shown in Figure 2, node A

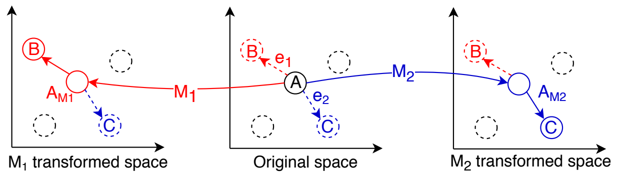

Figure 2: An example for edge-based transformation.

is connected to node B and C via edge e_1 and e_2. Because e_1 and e_2 have different relation types, our model transforms node A embeddings using the respective edge type embeddings M_1 and M_2 before aggregating them.

The pseudocode for the encoding stage is described in Algorithm 1. The input for this algorithm contains several elements. The graph $G(V, E)$ is by default a directed heterogeneous graph where multiple different edge types exist. However, this algorithm can also handle single-relation graphs in which the edge embeddings are set to identity matrices. The neighbor samples $N(v) = \{(u, \tau) | (u, v) \in E$ and edge (u, v) has type $\tau\}$ are the direct results from the sampling stage, where both the neighboring nodes u and the types of the edge τ are tracked for the center node v. Input features $\mathbf{x}_v$ are

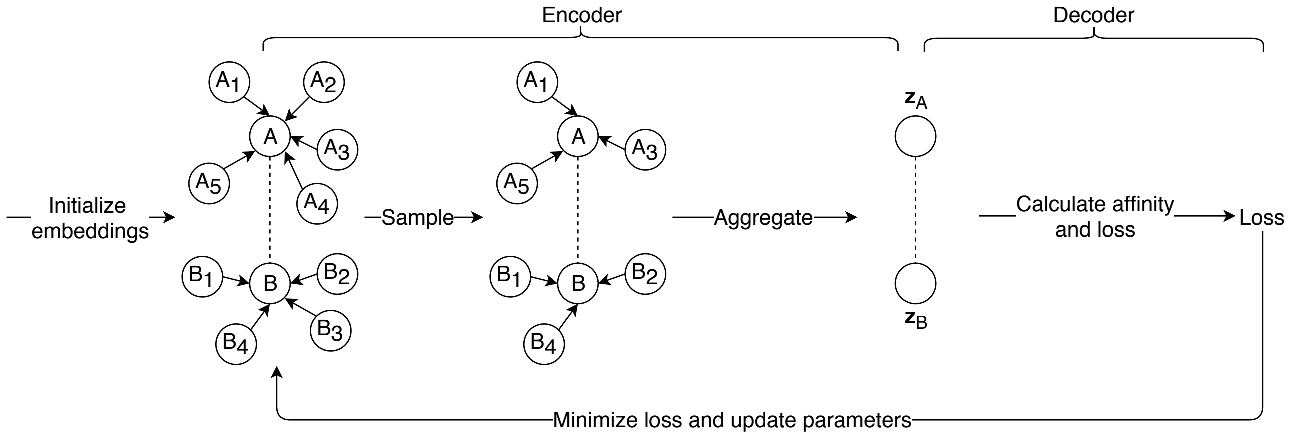

Figure 3: A general encoder-decoder workflow.

the initialized embeddings for node v. K represents the maximum hops from which the center node embedding is aggregated. Since the edges are one-hot encoded, the edge embeddings can be instantiated using Xavier initialization [9]. The weight matrices can also be initialized using the Xavier method and are shared among all nodes for each hop. The model uses non-linear activation functions. For the aggregators AGG, we discuss different options in Section 3.4.

Line 1 of Algorithm 1 shows that the initial node representations $\mathbf{h}_v^0$ are the input features. The maximum hop K determines the depth to which the center node reaches to aggregate information from its local neighborhood. For each node, information is aggregated from its immediate neighbors by first applying a linear transformation on the representation of the previous iteration $\{\mathbf{M}_\tau \mathbf{h}_u^{k-1}, \forall(u,\tau) \in N(v)\}$, which results in the neighborhood embeddings $\mathbf{h}_{N(v)}^k$ of the current iteration k as shown in line 4 of Algorithm 1. After obtaining the neighborhood embeddings at step k, the algorithm concatenates the previous node embeddings $\mathbf{h}_v^{k-1}$ with the neighborhood embedding, multiplies the concatenated embeddings by a shared weight matrix $\mathbf{W}^k$, and applies the nonlinearity (σ function) to transform the representation into the current status $\mathbf{h}_v^k$. The algorithm then goes back to the outer for loop and executes the next step $k+1$. The final node embeddings $Encoder(v) = \mathbf{z}_v$ are obtained by applying an $L2$ normalization on the representations in the last step $\mathbf{h}_v^K$. Intuitively, at each iteration (outer for loop) in Algorithm 1, information about the local (immediate) neighborhood is aggregated towards each node and after K iterations such information is incrementally propagated between different layers of neighborhood which leads to richer representations of nodes.

The hop size K essentially determines the search depth or the extent to which neighborhood information is propagated towards the center node. A smaller K creates a more localized neighborhood but makes the aggregation more efficient while a larger K value entails a more global view of the graph but at the cost of higher computation complexity. The encoder in Figure 3 shows an example where $K = 1$, namely there is only one layer. In this example, node A has immediate neighbors A_1, A_2, A_3, A_4, and A_5 and node B has immediate neighbors B_1, B_2, B_3, and B_4. Suppose in a particular training iteration A_1, A_3, and A_5 are sampled for node A and B_1, B_2, and B_4 are sampled for node B before the aggregation stage. As there is only one layer, the representations for A and B are obtained by applying linear transformation based on the incoming edges and the aggregation operation of the sampled neighboring nodes.

Algorithm 1: $Encoder(v)$

```
Input  : Graph $G(V,E)$; neighbor samples $N(v)$; input
         features $\mathbf{x}_v$; hops $K$; edge embedding $\mathbf{M}_\tau$ for each
         edge (relation) type $\tau$; weight matrices $\mathbf{W}^k$;
         activation function $\sigma$; aggregator $AGG$
Output : Node embedding $\mathbf{z}_v$ for each $v \in V$
1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v$
2 for $k = 1; k \leqslant K; k = k+1$ do
3     foreach $v \in V$ do
4         $\mathbf{h}_{N(v)}^k \leftarrow AGG_k(\{\mathbf{M}_\tau \mathbf{h}_u^{k-1}, \forall(u,\tau) \in N(v)\})$
5         $\mathbf{h}_v^k \leftarrow \sigma\Big(\mathbf{W}^k \cdot CONCAT\big(\mathbf{h}_v^{k-1}, \mathbf{h}_{N(v)}^k\big)\Big)$
6     end foreach
7 end for
8 $\mathbf{z}_v \leftarrow$ L2 normalized $\mathbf{h}_v^K$
```

3.3 Decoder

The purpose of the decoder is to map pairs of node embeddings to real values. These affinity values are calculated using the low dimensional representations of nodes in the embedding space and are expected to quantify the proximity of nodes in the original graph, i.e., if two nodes are close to each other in the original graph, their affinity should be high in the embedding space. This reconstruction mechanism can be formalized in Equation 1:

$$Decoder(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) \approx s_G(v_i, v_j) \tag{1}$$

where s_G is a user-defined graph-based proximity measure between nodes (such as adjacency) on graph $G = (V,E)$ and $\mathbf{z}_{v_i} = Encoder(v_i)$.

In order to train the model to learn meaningful representations of nodes and edges, we need to minimize the empirical loss $\mathcal{L}$, i.e., the difference between the affinity $Decoder(\mathbf{z}_{v_i}, \mathbf{z}_{v_j})$ and the graph-based proximity measurement $s_G(v_i, v_j)$ as shown in Equation 2:

$$\mathcal{L} = \sum_{(v_i, v_j) \in D} \ell(Decoder(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}), s_G(v_i, v_j)) \tag{2}$$

where D is a set of training node pairs and ℓ is a loss function that measures the discrepancy between the reconstructed proximity value $Decoder(\mathbf{z}_{v_i}, \mathbf{z}_{v_j})$ and the real proximity value $s_G(v_i, v_j)$. In this work, for the proximity measure, we use the indicator function given by Equation 3:

$$s_G(v_i, v_j) = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

As shown in Figure 3, backpropagation is then used to update the parameters (in this case including the edge embeddings) in order to minimize the loss.

After experimenting with both hinge loss and cross entropy loss, hinge loss shows better empirical performance for our model. In this case, we adopt the negative sampling strategy paired with hinge loss for the optimization as it has also proven to be effective in other graph embedding models [5]. In negative sampling, the positive samples are node pairs that are indeed connected by edges while the negative samples are the ones that are not. In practice, we collect our negative samples for each node by randomly sample from the unigram distribution where nodes with more adjacent nodes are more likely to be selected. Empirically, similar to the Word2Vec model [18], we apply a distortion rate of 0.75 for the final probability calculation as shown in Equation 4:

$$P(v_i) = \frac{deg(v_i)^{3/4}}{\sum_{j=1}^{n} \left(deg(v_j)^{3/4}\right)} \tag{4}$$

where $deg(v_i)$ is the degree of node v_i and there are n nodes in the graph.

We propose to use the dot product between the edge-transformed source node embedding $\mathbf{z}_{v_i}^\top \mathbf{M}_{\tau_{v_i v_j}}$ and the target node embedding $\mathbf{z}_{v_j}$ as the affinity, namely $Decoder(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) = \mathbf{z}_{v_i}^\top \mathbf{M}_{\tau_{v_i v_j}} \mathbf{z}_{v_j}$ where $\mathbf{z}_{v_i}$ is the source node embedding, $\mathbf{z}_{v_j}$ is the target node embedding, and the edge/relation between the source node and target node has type $\tau_{v_i v_j}$. Equation 5 shows the loss function we are using in the model:

$$\mathcal{L} = \sum_{(v_i, v_j) \in D, (v_i, v_j') \in D'_{(v_i, v_j)}} \left(\left[\mathbf{z}_{v_i}^\top \mathbf{M}_{\tau_{v_i v_j}} \mathbf{z}_{v_j'} - \mathbf{z}_{v_i}^\top \mathbf{M}_{\tau_{v_i v_j}} \mathbf{z}_{v_j} + \Delta \right]_+ + \lambda \sum_{k=1}^{m} \sum_{l=1}^{m} \left(\mathbf{M}_{\tau_{v_i v_j}}(k, l) - \mathbf{I}(k, l) \right)^2 \right) \tag{5}$$

where the first term in the summation is the hinge loss, the second term in the summation is the regularization of the edge embeddings, $D'_{(v_i, v_j)}$ defined by Equation 6:

$$D'_{(v_i, v_j)} = \{(v_i, v_j') | (v_i, v_j') \notin E \wedge v_j' \in V\} \tag{6}$$

is the set of negative samples composed of node pairs with target nodes replaced by random nodes, Δ is the margin for hinge loss, λ is a hyperparameter that controls the regularization, $\mathbf{M}_{\tau_{v_i v_j}}(k, l)$ is the element in row k and column l of $\mathbf{M}_{\tau_{v_i v_j}}$, $\mathbf{I}(k, l)$ is the element in row k and column l of an identity matrix, and m is the dimension of the square matrices. Ideally, negative samples should have small affinity values while positive samples should have large affinity values. The regularization restricts the edge embeddings such that they are close to the identity matrix. Stochastic gradient descent is applied to update the parameters needed to obtain the node embeddings as well as the edge embeddings.

Forcing the edge embeddings to be close to the identity matrix implies that the embedding spaces of different node types should not be significantly rotated from one another. Though we did consider and test some other strategies, this approach gave the best results. Standard L1 and L2 regularization approaches on the edge embeddings cause significant distortion in the norm taken in the hinge-loss term. We found some success in enforcing the edge embeddings to have row-wise unit L2 norm. This can be interpreted as finding the principal component in the source space corresponding to a dimension in the target space. However, this technique seems to over constrain the edge embeddings compared to the approach we settled on. For multi-relational, directed graphs, it may be profitable to try a regularizer of the form:

$$\sum_{k=1}^{m} \sum_{l=1}^{m} \left(\mathbf{M}_{\tau_{v_i v_j}} \mathbf{M}_{\tau_{v_j v_i}}(k, l) - \mathbf{I}(k, l) \right)^2 \tag{7}$$

which enforces the edge embedding of one relation is the inverse of the opposite relation. We did not have a suitable use case to test this strategy on.

3.4 Aggregators

While there are many options for choosing the aggregators in Algorithm 1, we examine two of them — mean aggregator and Long Short-Term Memory (LSTM) aggregator — in our model. These two candidates have their own advantages and disadvantages. Because, unlike other machine learning tasks which usually operate on n dimensional lattices (such as sentences, images, or 3D volumes) [11], a graph usually does not have an ordered structure, aggregators that are able to handle arbitrarily ordered node inputs are usually desired. Mean aggregator in this sense is a good candidate for generic cases in graph representation learning, however, because of its simplicity, its representational capacity is limited. On the contrary, the LSTM aggregator creates a sequential ordering of nodes in the neighborhood. This ordering is artificial in some cases in that it is not permutation invariant but in other cases it is indeed useful in that for some applications sequential information (e.g., temporal component) plays an important role. For example, Xu et al. [23] have shown that by considering temporal information a better professional similarity measure can be developed based on career trajectories using LinkedIn data. Because LSTM aggregator can model long term sequential dependencies, it has a higher representational capacity than the mean aggregator.

The idea of the mean aggregator is to calculate the element-wise mean of the edge-transformed neighbor samples $\{\mathbf{M}_\tau \mathbf{h}_u^{k-1}, \forall (u, \tau) \in N(v)\}$. By substituting AGG with the mean aggregator, line 4 of Algorithm 1 can be written as:

$$\mathbf{h}_{N(v)}^k \leftarrow \frac{\sum_{(u,\tau) \in N(v)} \mathbf{M}_\tau \mathbf{h}_u^{k-1}}{|N(v)|} \tag{8}$$

where $|N(v)|$ is the cardinality of $N(v)$, namely the number of neighbor samples for each node. By applying this mean aggregator and removing the edge transformation term $\mathbf{M}_\tau$, this algorithm is almost equivalent to the graph convolutional network model proposed by Kipf and Welling [14] except that the "skip connection" created by the concatenation in the algorithm could lead to significant improvements [11]. In Section 4, we will show that the edge transformation term $\mathbf{M}_\tau$ is essential for learning more expressive embeddings for heterogeneous graphs.

While ordering the neighboring nodes may make the algorithm not permutation invariant, there are many cases where such ordering is meaningful and necessary to uncover the implicit structure of the underlying data. In order to illustrate our point, we focus on the temporal structure on the heterogeneous graph from LinkedIn (one of the datasets we are using in the experiment which will be explained in detail in Section 4). For a professional network that has relations about member connections, employment, and education, each link is associated with a timestamp, e.g., the time when two members are connected, when a member takes on a new position, and when a member graduates from a university. In order to provide better career recommendations using the graph embedding model, it is important to consider the temporal dynamics of a member's professional connections, career progressions, and educational achievements. For instance, a member majored in mathematics in undergraduate study may end up being interested in a more applied field such as machine learning which is evident in his recent connections, new intern jobs, and graduate focus. A good career recommendation system should consider this subtle but important dynamics in the member's professional interests and recommend a position in machine learning related fields rather than a position as an economist because the member had an internship in the financial sector during the undergraduate study. LSTM maintains a cell state and controls the information flow using forget gate, input gate, and output gate [13]. Because of these features, we employ the LSTM aggregator (Figure 4) to encode the temporal information in the network. The output O is then assigned to the

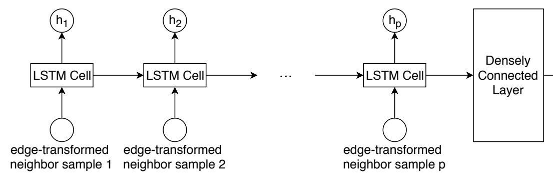
h1
h2
hp
LSTM Cell
LSTM Cell
...
LSTM Cell
Densely Connected Layer
O
edge-transformed neighbor sample 1
edge-transformed neighbor sample 2
edge-transformed neighbor sample p

Figure 4: The edge-transformed samples are from $\{\mathbf{M}_\tau \mathbf{h}_u^{k-1}, \forall(u,\tau) \in N(v)\}$ and they are ordered chronologically.

neighborhood embeddings and line 4 of Algorithm 1 can be written as $\mathbf{h}_{N(v)}^k \leftarrow O$.

4 EXPERIMENT AND RESULTS

In this section, we conduct experiments on two datasets for our model and show the evaluation results: one from a general knowledge graph and one from a heterogeneous professional social network. These two datasets are used to examine closely and separately the two major improvements in our model, namely the edge-based transformation and the temporal component.

4.1 Datasets

FB15K[2] is a subset of the large collaborative knowledge graph Freebase. It contains 14,951 nodes and 1,345 relation (edge) types. The dataset is partitioned into a training set with 483,142 triplets (source node, relation, target node), a validation set of size 50,000, and a test set with 59,071 triplets. Unlike the citation data, Reddit data, or the protein-protein interactions data that are typically used graph embedding learning models [11, 14] which ignore relation types, FB15K has more than 1,000 different relation types which make it unrealistic to learn comprehensive graph representations without considering these different relations. We examine the performance of our model compared with different baseline models using the FB15K dataset in order to verify whether incorporating edge-based transformation is beneficial to the representation learning process.

The second dataset we are using is the subgraph of the LinkedIn graph. This subgraph is a heterogeneous network that contains 3 different relations: member connection, employment, and education. This dataset was generated by selecting LinkedIn members who have self-selected their location as the San Francisco Bay Area. The member connection graph was then pruned by finding the k-core graph with $k = 50$ and subsampled for 0.1% of connections. After this reduction, all employment and education relations for remaining members were added. Timestamps for all relations are available. We are left with approximately 500,000 nodes of all types and 2 million edges of all relation types. Relations are sorted by timestamp, and the first 80% chronologically form the training set, the next 10% are the validation set, and the final 10% are the test set. As mentioned in Section 3.4, this LinkedIn heterogeneous graph is used to test the effectiveness of explicitly incorporating the temporal component in our model.

4.2 Experiment

Unlike previous graph embedding models which use node classification to evaluate the performance, we use the link prediction task to examine our models. This task provides a direct test of the embedding similarity rather than a test of an additional task. For every testing sample (source node, relation type, target node), the target node is removed. In order to predict the target node, an approximation is calculated by applying an edge transformation on the source node using the learned embeddings. This approximation is then compared to each of the candidate nodes in our dataset using cosine similarity. These candidate nodes can be ranked based on the similarity scores with respect to the approximation. We calculate the Mean Reciprocal Rank (MRR) based upon the position at which the true target node appears in the ranking. In addition, we calculate the HITS@1 and HITS@5 scores which measure the percentages that the true target node appears in the first one and first five nodes in the ranking. In contrast to recommendation systems that are purely based on finding nearest neighbors [24], this link prediction task naturally becomes a recommendation task on the LinkedIn graph and particularly it translates to recommending new professional connections, positions, or educational opportunities in the future. For example, by predicting the target node given a member and the employment relation, the system is essentially answering the question of discovering the most plausible job that

[2]https://everest.hds.utc.fr/doku.php?id=en:transe

the member could take on in the future that would make the current LinkedIn graph more complete. Because there are millions of nodes in the LinkedIn dataset, we randomly sample 19 false target nodes to make a candidate size of 20 for each test in the LinkedIn graph experiment.

In order to initialize the node embeddings, for FB15K data, since each node (entity) in the graph has a description, we take advantage of such information. We use the node labels and keywords provided by Xie et al. [21] and adopt the fastText [4] model to obtain word embeddings because it handles out-of-vocabulary words and considers the morphology of words by viewing each word as a bag of character n-grams. The model makes use of the distributional semantics assumption, i.e. encoding words by considering the contextual words. However, unlike the skip-gram model [18], the fastText model is designed to optimize the binary classification task of predicting the presence (or absence) of context words given target words. The model learns the embeddings by finding the parameterization settings that minimize the following loss function:

$$\sum_{t=1}^{T}\left[\sum_{c\in C_t}\ell\Big(score_{w_t,w_c}\Big)+\sum_{n\in\mathcal{N}_{t,c}}\ell\Big(-score_{w_t,n}\Big)\right] \quad (9)$$

where we denote the logistic loss $\ell : x \mapsto \log(1+\exp^{-x})$, $score_{w_t,w_c}$ denotes the scoring function for the target/center word w_t and the context word w_c, n is a negative context word from a set of negative examples $\mathcal{N}_{(t,c)}$ sampled from the vocabulary, the context C_t is a set of word indices surrounding word w_t, and T is the maximum index of a sequence of words $w_1, ..., w_T$. In order to encode the morphology of words, each word w is represented as a set of character n-grams $\mathcal{G}_w \subset \{1, ..., r\}$ where r is the size of our dictionary of n-grams. For each n-gram g, there is a vector representation $\mathbf{o}_g$ associated with it. Then the scoring function is defined as $score_{w_t,w_c} = \sum_{g\in\mathcal{G}_{w_t}} \mathbf{o}_g^\top \mathbf{v}_{w_c}$ where $\mathbf{v}_{w_c}$ is the vector representation for word w_c. The initial node embeddings are the average embeddings for the node labels and keywords in the FB15K data. Joining the FB15K data with the initial embeddings and converting to our input format is carried out on a single machine.

In the LinkedIn data, the nodes are initialized by internally generated embeddings. These embeddings were created for specific entities, such as company, school, skill, and job title by minimizing the KL-divergence of the observed co-occurence probability and the probability implied by the sigmoid of the inner product of the embeddings of two entities. To generate embeddings for members, we perform a mean pooling over each of the different entity types and then concatenating the title and skill embeddings representing each type for each member. Our initial member embeddings have dimension 50, and we generate 50 dimensional embeddings for schools and companies.

The models are implemented and trained using version 1.9 of TensorFlow on a single NVIDIA K80 GPU. The hidden and output dimension of the model is 50. The edge embedding matrices are 50 by 50. We performed hyperparameter tuning using a grid search on the learning rate, regularization coefficient (λ in Equation 5), dropout rate, and the hinge-loss margin (Δ in Equation 5). As mentioned earlier, we also experimented with different objectives and regularizers.

4.3 Results

Table 1 shows the evaluation results on the FB15K dataset. Two baselines are used in this evaluation. The initial embedding baseline model uses the initial node embeddings as the final embeddings, which means that no training is involved. The second baseline is the model that excludes edge-based transformation which means that different relation types are treated the same in the learning process. We compare these baseline models with our proposed model using both mean aggregator and LSTM aggregator. This evaluation shows that our model with mean aggregator yields the best results on FB15K. The LSTM aggregator is slightly worse than the mean aggregator because we are not able to model any sequential pattern in the FB15K dataset, which implies that a permutation invariant model (e.g., using the mean aggregator) is a better choice in this case. Our model using the mean aggregator has an MRR score that is **more than 4 times** higher than the model without edge-based transformation. This improvement verifies that incorporating edge-based transformation is beneficial to the representation learning process. Table 2 shows the evaluation results on the LinkedIn dataset. For MRR and HITS@1 metrics, our model yields better results. In particular, our model with the LSTM aggregator has the best results for these two metrics: the MRR score is increased by **more than 5%** and the HITS@1 score is increased by **more than 15%** compared with the baseline. This is because our model with the LSTM aggregator not only considers different relation types but also takes into account the temporal dynamics on the LinkedIn graph. For the slight decrease in HITS@5, we suspect it is because 1) there are only 3 different relations in the LinkedIn dataset (compared with 1,345 different relations in FB15K) and 2) nodes associated with different relations are not balanced (e.g., there are much more nodes associated with professional connection relation than with employment or education relation). These two facts about the LinkedIn dataset might be the culprit for the slight decrease in performance. In general, however, the results from Table 1 and Table 2 show that incorporating edge-based transformation is beneficial to the representation learning process and explicitly incorporating temporal component is helpful for capturing the dynamics in the graph.

Model	MRR	HITS@1	HITS@5
Initial Embeddings	0.053	0.022	0.078
W/O Edge-based Transformation	0.093	0.007	0.180
Our Model W/ Mean Aggregator	**0.425**	**0.317**	**0.547**
Our Model W/ LSTM Aggregator	0.333	0.240	0.431

Table 1: Results for FB15K dataset

Model	MRR	HITS@1	HITS@5
W/O Edge-based Transformation	0.489	0.316	**0.701**
Our Model W/ Mean Aggregator	0.504	0.356	0.671
Our Model W/ LSTM Aggregator	**0.516**	**0.366**	0.688

Table 2: Results for LinkedIn dataset

5 CONCLUSIONS

In this paper, we introduced a graph representation learning model that works on heterogeneous graphs by taking into account edge-based transformations. We showed that this approach provides a substantial improvement over one that ignores the relational context.

We further proposed the idea of explicitly considering the temporal dynamics of the graph by using the LSTM aggregator in our model. When temporal information is relevant and available, this approach shows improvement in identifying new links in the graph.

By applying the model on both FB15K and LinkedIn datasets, we demonstrated the effectiveness of our time-aware inductive representation learning strategy for two very different classes of heterogeneous graphs. The FB15K dataset has a very high number of relations compared to the number of nodes and edges in the graph, a characteristic which is common among knowledge graphs. However, as knowledge graphs are often hand-curated, the overall number of nodes and edges is generally small. The LinkedIn dataset, on the other hand, has a very large number of nodes and edges, but the number of relations is very low. These characteristics are common of web-scale social networks. Our results show our approach is useful for graphs at very different places in the graph size-relation density spectrum.

We propose different prongs on future work. Within the application space, exploring the transferability of the embeddings learned here to supervised utilities of high interest. As we have learned embeddings for schools, companies, LinkedIn members and the relations between all three, these are relevant to predicting possible connections between members, likely applicants to jobs, qualified applicants to jobs, interesting members to follow and others. Incorporating interactions within social networks (e.g. messages, comments on posts) as relation types is another interesting avenue. This would allow us to understand if certain interaction channels are preferred for engaging users in different directions along the graph and therefore how to predict which channel for notifying members about activities and opportunities within the network.

Another path to explore is developing an inductive approach to generating edge embeddings. Our model, while an inductive method for generating node embeddings, is still transductive on the edge embeddings. Adapting the method to inductively generate edge embeddings would be extremely in useful in situations where new relations are regularly discovered or created. In the context of social networks, the deployment of new features introducing ways for users to interact is an example of how new relations can enter the graph.

ACKNOWLEDGMENTS

The authors would like to thank Shipeng Yu, Myunghwan Kim, Yafei Wang, Yiou Xiao, and other members in the Growth and Ecosystem Relevance team at LinkedIn for their support, comments, and suggestions.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 37–48.

[3] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*. 585–591.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

[5] 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.

[6] 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, 891–900.

[7] Suleyman Cetintas, Monica Rogati, Luo Si, and Yi Fang. 2011. Identifying similar people in professional social networks with discriminative probabilistic models. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 1209–1210.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[16] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 609–616.

[17] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[19] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[21] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions.. In *AAAI*. 2659–2665.

[22] Huang Xu, Zhiwen Yu, Hui Xiong, Bin Guo, and Hengshu Zhu. 2015. Learning career mobility and human activity patterns for job change analysis. In *2015 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1057–1062.

[23] Ye Xu, Zang Li, Abhishek Gupta, Ahmet Bugdayci, and Anmol Bhasin. 2014. Modeling professional similarity by mining professional career trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1945–1954.

[24] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 974–983. https://doi.org/10.1145/3219819.3219890