Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks

Jingbo Shang[1], Meng Qu[1], Jialu Liu[2], Lance M. Kaplan[3], Jiawei Han[1], Jian Peng[1]
[1]Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA
[2]Google Research, New York City, NY, USA
[3]Sensors & Electron Devices Directorate, Army Research Laboratory, Adelphi, MD, USA
[1]{shang7, mengqu2, hanj, jianpeng}@illinois.edu [2]jialu@google.com [3]lance.m.kaplan.civ@mail.mil

ABSTRACT

Most real-world data can be modeled as heterogeneous information networks (HINs) consisting of vertices of multiple types and their relationships. Search for similar vertices of the same type in large HINs, such as bibliographic networks and business-review networks, is a fundamental problem with broad applications. Although similarity search in HINs has been studied previously, most existing approaches neither explore rich semantic information embedded in the network structures nor take user's preference as a guidance.

In this paper, we re-examine similarity search in HINs and propose a novel embedding-based framework. It models vertices as low-dimensional vectors to explore network structure-embedded similarity. To accommodate user preferences at defining similarity semantics, our proposed framework, ESim, accepts user-defined meta-paths as guidance to learn vertex vectors in a user-preferred embedding space. Moreover, an efficient and parallel sampling-based optimization algorithm has been developed to learn embeddings in large-scale HINs. Extensive experiments on real-world large-scale HINs demonstrate a significant improvement on the effectiveness of ESim over several state-of-the-art algorithms as well as its scalability.

1. INTRODUCTION

A *heterogeneous information network* (*HIN*) is a network that consists of multi-typed vertices connected via multi-typed edges. Modeling data in the real world as heterogeneous information networks (*HINs*) can capture rich data semantics and facilitate various applications [26, 11, 8, 29, 23, 25]. *Similarity search* in HINs is a fundamental problem for mining large HINs and much digital ink has been spilled over it in the community (e.g., [26, 25, 29, 28]). In this paper, we are particularly interested in utilizing HIN to conduct similarity search among the objects of the same type. For example, given a social media network in Yelp with connections between reviews, users and businesses, we can find similar restaurants (i.e., similarity search among businesses), and recommend potential friends with similar preferences (i.e., similarity search among users).

Naturally, people tend to employ different semantics in their network-based search even towards the same application. Take a bibliographic network as an example that consists of papers, authors, venues, and terms. To find similar authors to a given author, some users may weigh more on shared technical terms in papers, while others more on shared publication venues. PathSim [26] proposes to use *meta-paths* to define and guide similarity search in HIN, with good success. A meta-path is represented by a path (i.e., a connected sequence of vertices) at the schema level (e.g., ⟨author−paper−author⟩). However, PathSim has not explored similarities embedded deeply in the HIN structure. When hunting for similar authors, if the given meta-path is ⟨author−paper−venue−paper−author⟩, PathSim can only build bridges for authors publishing papers in the same venues (e.g., "WSDM"), but cannot efficiently explore the semantic similarity between venues (e.g., "WSDM" and "WWW") to improve the search. However, such kind of semantic similarity can be easily implied by embedded semantic structure of the HIN. For an extreme example, if one author only publish in "WSDM" while the other only has publications in "WWW", their PathSim similarity will be 0. Although a bridge can be built between the two similar venues by traversing some long paths, it becomes much more costly and does not seem to be an elegant way compared with the embedding-based approach studied in this paper.

Along another line of study, network-embedding techniques have been recently explored for homogeneous information networks, which treat all vertices and edges as of the same type, represented by LINE [29]. An alternative and better way is to first project the HIN to several bipartite (assuming user-given meta-paths are symmetric) or homogeneous networks and then apply the edge-wise HIN embedding technique, such as PTE [28]. However, the network projection itself is count-based which does not preserve the underlying semantics, and the cost for computing such projected networks is high. For example, taking a user-selected meta-path ⟨author−paper−venue−paper−author⟩ (i.e., shared-venues) to find similar authors, the projection will generate a homogeneous network consisting of only authors, but it loses important network structure information (e.g., venues and papers) [25] and leads to a rather densely connected network (since many authors may publish in many venues). Clearly, direct modeling of the original heterogeneous information network will capture richer semantics than exploiting the projected networks.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Acknowledging the deficiency of the above two types of approaches, we propose ESim, a novel embedding-based similarity search framework, with the following contributions.

- A general embedding-based similarity search framework is proposed for HINs, where an HIN may contain undirected, directed, weighted, and un-weighted edges as well as various types of vertices;
- Our framework incorporates a set of meta-paths as an input from a user to better capture the semantic meaning of user-preferred similarity; and
- It handles large-scale HINs efficiently due to a novel sampling method and a parallel optimization framework.

To the best of our knowledge, this is the first work that proposes a general meta-path guided embedding framework for similarity search in heterogeneous information networks.

2. RELATED WORK

2.1 Meta-Path Guided Similarity Search

The concept of meta-path, which represents a connected sequence of vertices at the schema level, plays a crucial role in typed and structured search and mining in HINs. PathSim [26] defines the similarity between two vertices of the same type by the normalized count of path instances following a user-specified meta-path between any pair of vertices. [26] shows that the PathSim measure captures better peer similarity semantics than random walk-based similarity measures, such as P-PageRank [12] and SimRank [11]. Moreover, [27] shows that user guidance can be transformed to a weighted combination of meta-paths. However, PathSim does not explore the similarity *embedded* in the structure of a HIN. Moreover, PathSim doesn't have the embedding vectors of vertices, which can make the further analysis more efficient, such as clustering.

2.2 Embedding-based Similarity Search

Recently, embedding technique, which aims at learning low-dimensional vector representations for entities while preserving proximities, has received an increasing attention due to its great performance in many different types of tasks. As a special and concrete scenario, embedding of homogeneous networks containing vertices of the same type has been studied recently. LINE [29] and DeepWalk [20] utilize the network link information to construct latent vectors for vertex classification and link prediction. DCA [4] starts from the personalized PageRank but does further decomposition to get better protein-protein interaction predictions in biology networks. However, these homogeneous models cannot capture the information about entity types nor about relations across different typed entities in HINs.

There are also embedding algorithms developed for HINs. For example, Chang et al. propose to incorporate deep neural networks to train embedding vectors for both text and images at the same time [3]. Under a supervised setting, PTE [28] utilizes labels of words and constructs bipartite HINs to learn predictive embedding vectors for words. Embedding techniques have been also applied to knowledge graphs to resolve question-answering tasks [6] and retain knowledge relations between entities [32]. However, these are all specially designed for specific types of networks and tasks and thus difficult to be extended to incorporate user guidance. The vector spaces constructed by different methods have different semantic meanings due to the statistics they emphasize. In many real-world scenarios, it is often difficult to find an appropriate model, and the models have to be revised to fit into the desired usage.

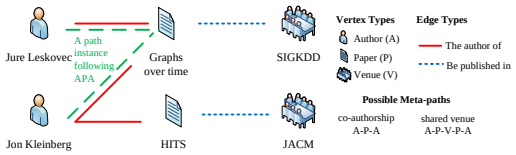


Figure 1: A real small bibliography network for illustrations of the definitions.

2.3 Similarity Search in Vector Spaces

Various similarity metrics have been designed for vector spaces in different tasks, such as cosine similarity [24], Jaccard coefficient [14], and the p-norm distance [5]. In our work, we directly utilize cosine similarity based on embedding vectors of vertices. Many efforts have been paid on optimizing the efficiency of top-k nearest neighbor search [33, 13, 9, 18, 2]. We adopt these existing efficient similarity search techniques to support online queries.

3. PRELIMINARIES

In this section, a series of definitions and notations are presented.

DEFINITION 1. ***Heterogeneous Information Network*** *is an information network where both vertices and edges have been associated with different types. In a heterogeneous information network $G = (V, E, R)$, V is the set of typed vertices (i.e., each vertex has its own type), R is the set of edge types in the network, and $E \subset V \times V \times R$ is the edge set. An* ***edge*** *in a heterogeneous information network is an ordered triplet $e = \langle u, v, r \rangle$, where u and v are two typed vertices associated with this edge and r is the edge type.*

In a general heterogeneous information network, there might be multiple typed edges between the same two vertices and the edge could be either directed or undirected. The definition above naturally supports all these cases. To better explain it, we use a bibliographic network as an example.

EXAMPLE 1. *In the bibliographic network as shown in Fig. 1, the vertex set V consists of three types, {"author", "paper", "venue"}, the edge type set R contains* "the author of the paper" *and* "the paper was published in the venue". *The edge set E consists of concrete edges like $\langle u =$* "Jon Kleinberg", $v =$ "HITS", $r =$ "the author of" $\rangle$, *where* Jon Kleinberg *is one of the authors of the paper* HITS. *For ease of representation, edge type r is denoted as $A-P$ once there is only one edge type between author and paper. Another edge is $\langle u =$* "HITS", $v =$ "JACM", $r =$ "be published in" $\rangle$, *which means the paper was published in the venue* "JACM". *In this case, r is denoted as $P-V$. Note that both edge types are undirected here. That is, $A-P$ and $P-A$ are actually referring to the same edge type. So as $P-V$ and $V-P$.*

We define the concepts of meta-path and sub-meta-path by concatenating edge types in a sequence, as follows.

DEFINITION 2. *In a HIN $G = (V, E, R)$, a* ***meta-path*** *is a sequence of compatible edge types $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$ with length L, $\forall r_i \in R$, and the outgoing vertex type of r_i should match the incoming vertex type of r_{i+1}. For any $1 \le s \le t \le L$, we can induce a* ***sub-meta-path*** *$\mathcal{M}_{s,t} = \langle r_s, r_{s+1}, \ldots, r_t \rangle$.*

Particularly, an edge type r can be viewed as a length-1 meta-path $\mathcal{M} = \langle r \rangle$.

A sequence of edges following $\mathcal{M}$ is called a path instance. Because there might be multiple edges between the same pair of vertices, instead of the vertex sequence, the edge sequence is used to describe a path instance. Formally speaking,

DEFINITION 3. *Given a HIN $G = (V, E, R)$ and a meta-path $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$, any path $\mathcal{P}_{e_1 \rightsquigarrow e_L} = \langle e_1, e_2, \ldots, e_L \rangle$ connecting vertices u_1 and u_{L+1} (i.e., v_L), is a **path instance** following $\mathcal{M}$, if and only if $\forall 1 \leq i \leq L$, the i-th edge is type-r_i, and $\forall 1 \leq i \leq L, v_i = u_{i+1}$.*

We continue the example of the bibliographic network and list some meta-paths and path instances.

EXAMPLE 2. *In the bibliographic network as shown in Fig. 1, a length-2 meta-path $\langle A{-}P, P{-}A \rangle$ (abbrev. as $A{-}P{-}A$) expresses the co-authorship relation. The collaboration between "*Jure Leskovec*" and "*Jon Kleinberg*" on the paper "*Graphs over time*" is a path instance of $A{-}P{-}A$. Similarly, a length-4 meta-path $A{-}P{-}V{-}P{-}A$ captures the shared venues and any two authors published a certain paper in a same venue could be its path instance. Besides, $P{-}V{-}P$ is a sub-meta-path of $A{-}P{-}V{-}P{-}A$.*

DEFINITION 4. ***Meta-path Guided Similarity Search** is a similarity search task on HINs, where the semantic meanings of the similarity are determined by n meta-paths $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n\}$ specified by the user.*

EXAMPLE 3. *In the bibliographic network, to find similar authors, a user may choose two meta-paths $A{-}P{-}V{-}P{-}A$ and $A{-}P{-}A$ as guidance.*

In the subsequent discussion, we focus on the case of a single meta-path $\mathcal{M}$. This is because (1) the principal ideas for exploring multiple weight-assigned meta-paths are essentially the same; (2) in our experiments, the performance gain of the optimal weighted combination of meta-paths is not significant. Moreover, we leave the study on derivation of a weighted combination of multiple meta-paths based on user's high-level guidance (i.e., user providing examples instead of path weights explicitly) to future work. Such a user-guided approach without adopting the embedding framework has been studied in [34, 27, 15].

4. METHODOLOGY

In this section, to incorporate meta-paths, we formulate a probabilistic embedding model inspired from many previous studies. We propose an efficient and scalable optimization algorithm relying on the sampling of path instances following the given meta-path. Our proposed efficient sampling methods are the most crucial steps and thus are discussed separately. In addition, we also provide thorough time complexity analysis.

4.1 Probabilistic Embedding Model Incorporating Meta-Paths

Model Formulation. The basic idea of our approach is to preserve the HIN structure information into the learned embeddings, such that vertices which co-occur in many path instances turn to have similar embeddings.

To preserve the structure of a HIN $G = (V, E, R)$, we first define the conditional probability of vertex v connected to vertex u by some path instances following the meta-path $\mathcal{M}$ as:

$$Pr(v|u, \mathcal{M}) = \frac{\exp(f(u, v, \mathcal{M}))}{\sum_{v' \in V} \exp(f(u, v', \mathcal{M}))} \tag{1}$$

where function f is a scoring function modeling the relevance between u and v conditioned on the meta-path $\mathcal{M}$. In particular, we encode the meta-path through the following formulation inspired from [19, 23]:

$$f(u, v, \mathcal{M}) = \mu_{\mathcal{M}} + \mathbf{p}_{\mathcal{M}}{}^T \mathbf{x_u} + \mathbf{q}_{\mathcal{M}}{}^T \mathbf{x_v} + \mathbf{x_u}{}^T \mathbf{x_v}$$

Here, $\mu_{\mathcal{M}} \in \mathbb{R}$ is the global bias of the meta-path $\mathcal{M}$, $\mathbf{p}_{\mathcal{M}}$ and $\mathbf{q}_{\mathcal{M}} \in \mathbb{R}^d$ are local bias vectors which are d dimensional. $\mathbf{x_u}$ and $\mathbf{x_v} \in \mathbb{R}^d$ are d dimensional embedding vectors for vertices u and v respectively. Under such definition, if the embeddings of two vertices have a larger dot product, the two vertices are likely having a larger relevance score, and thus co-occuring in many path instances. Note that if users want a symmetric score function, $\forall u, v, f(u, v, \mathcal{M}) = f(v, u, \mathcal{M})$, we can restrict $\mathbf{p}_{\mathcal{M}} = \mathbf{q}_{\mathcal{M}}$.

For a better understanding of the scoring function f, we can rewrite it as follows

$$f(u, v, \mathcal{M}) = (\mu_{\mathcal{M}} - \mathbf{p}_{\mathcal{M}}{}^T \mathbf{q}_{\mathcal{M}}) + (\mathbf{x_u} + \mathbf{q}_{\mathcal{M}})^T (\mathbf{x_v} + \mathbf{p}_{\mathcal{M}})$$

where we can see that $\mathbf{p}_{\mathcal{M}}$ and $\mathbf{q}_{\mathcal{M}}$ shift $\mathbf{x_u}$ and $\mathbf{x_v}$ according to the semantic of the meta-path $\mathcal{M}$ while $\mu_{\mathcal{M}}$ adjusts the score to an appropriate range.

For a path instance $\mathcal{P}_{e_1 \rightsquigarrow e_L} = \langle e_1 = \langle u_1, v_1, r_1 \rangle, e_2 = \langle u_2, v_2, r_2 \rangle, \ldots, e_L = \langle u_L, v_L, r_L \rangle \rangle$ following the meta-path $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$, we adopt the following approximation, by approximating the probability of the first vertex.

$$\begin{aligned} Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | \mathcal{M}) &= Pr(u_1 | \mathcal{M}) \times Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | u_1, \mathcal{M}) \\ &\propto C(u_1, 1 | \mathcal{M})^\gamma \times Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | u_1, \mathcal{M}) \end{aligned} \tag{2}$$

where $C(u, i|\mathcal{M})$ represents the number of path instances following $\mathcal{M}$ with the i^{th} vertex being u. γ is a widely used parameter to control the effect of overly-popular vertices, which is usually 3/4 inspired from [16]. In Sec. 4.2.1, we will show an efficient dynamic programming algorithm to compute $C(u, i|\mathcal{M})$.

The conditional probability, $Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | u_1, \mathcal{M})$, is now the last undefined term. The simplest definition is $Pr(v_L | u_1, \mathcal{M})$, which assumes that the probability only depends on the two ends of the path instance and directly applies Eq. (1). However, it omits the intermediate information and is equivalent to projection-based models.

In this paper, we propose two possible solutions as follows, and later show "pairwise" is more effective than "sequential", since it exploits the meta-path guidance in a more thorough way.

- **Sequential** (seq): In this setting, we assume that a vertex is highly relevant to its left/right neighbors in the sequence: $Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | u_1, \mathcal{M}) = \prod_{k=1}^{L} Pr(v_k | u_k, \mathcal{M}_{k,k})$.
- **Pairwise** (pair): In this setting, we assume all vertices in a path instance are highly relevant to each other, and thus the probability of the path instance is defined as $Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L} | u_1, \mathcal{M}) = \prod_{s=1}^{L} \prod_{t=s}^{L} Pr(v_t | u_s, \mathcal{M}_{s,t})$. As a result, vertices co-occur in many path instances turn to have large relevance scores.

Noise-Contrastive Estimation (NCE). Given the conditional distribution defined in Eqs. (1) and (2), the maximum likelihood training is tractable but expensive because com-

puting the gradient of log-likelihood takes time linear in the number of vertices.

Since learning rich vertex embeddings does not require the accurate probability of each path instance, we adopt the NCE for optimization, which has been proved to significantly accelerate the training without cutting down the embedding qualities [7, 17]. The basic idea is to sample some observed path instances associated with some noisy ones, and it tries to maximize the probability of each observed instance while minimize the probability of each noisy one.

Specifically, we reduce the problem of density estimation to a binary classification, discriminating between samples from path instances following the user selected meta-path and samples from a known noise distribution. In particular, we assume these samples come from the mixture.

$$\frac{1}{K+1}Pr^{+}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M}) + \frac{K}{K+1}Pr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})$$

where $Pr^{+}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})$ denotes the distribution of path instances in the HIN following the meta-path $\mathcal{M}$. $Pr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})$ is a noise distribution, and for simplicity we set

$$Pr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M}) \propto \prod_{i=1}^{L+1} C(u_i, i\,|\mathcal{M})^{\gamma}$$

We further assume the noise samples are K times more frequent than positive path instance samples. The posterior probability that a given sample came from the path instance samples following the given meta-path is

$$\frac{Pr^{+}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}{Pr^{+}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M}) + K \cdot Pr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}$$

Since we would like to fit $Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})$ to $Pr^{+}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})$, we simply maximize the following expectation.

$$\begin{aligned}\mathcal{L}_{\mathcal{M}} =& \mathbb{E}_{Pr^{+}}\left[\log \frac{Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}{Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M}) + KPr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}\right] \\ &+ K\,\mathbb{E}_{Pr^{-}}\left[\log \frac{KPr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}{Pr(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M}) + KPr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})}\right]\end{aligned}$$

Suppose we are using the sequential definition as Eq. (3). The loss function derived from NCE becomes

$$\begin{aligned}\mathcal{L}_{\mathcal{M},\text{seq}} =& \mathbb{E}_{Pr^{+}}\left[\log \sigma(\Delta_{e_1 \rightsquigarrow e_L|\mathcal{M}})\right] \\ &+ K\,\mathbb{E}_{Pr^{-}}\left[\log\left(1-\sigma(\Delta_{e_1 \rightsquigarrow e_L|\mathcal{M}})\right)\right]\end{aligned}$$

where $\Delta_{e_1 \rightsquigarrow e_L|\mathcal{M}} = \sum_{i=1}^{L} f(u_i, v_i, \mathcal{M}) - \log\left(K \cdot Pr^{-}(\mathcal{P}_{e_1 \rightsquigarrow e_L}|\mathcal{M})\right)$ and $\sigma(\cdot)$ is the sigmoid function. Note that when deriving the above equation, we used $\exp(f(u, v, \mathcal{M}))$ in place of $Pr(v|u, \mathcal{M})$, ignoring the normalization term in Eq. (1). We can do this because the NCE objective encourages the model to be approximately normalized and recovers a perfectly normalized model if the model class contains the data distribution [7, 17]. The above expectation is also studied in [16], which replaces $\Delta_{e_1 \rightsquigarrow e_L|\mathcal{M}}$ with $\sum_{i=1}^{L} f(u_i, v_i, \mathcal{M})$ for ease of computation and names the method *negative sampling*. We follow this idea and have the approximation as follows.

$$\begin{aligned}\mathcal{L}_{\mathcal{M},\text{seq}} \approx & \sum_{\mathcal{P}_{e_1 \rightsquigarrow e_L} \text{ following } \mathcal{M}} \log \sigma\Big(\sum_{i=1}^{L} f(u_i, v_i, \mathcal{M}_{i,i})\Big) + \\ & \sum_{k=1}^{K} \mathbb{E}_{\mathcal{P}^{k}_{e_1 \rightsquigarrow e_L} \sim Pr^{-}|u_1, \mathcal{M}}\left[\log\left(1-\sigma\Big(\sum_{i=1}^{L} f(u_i^k, v_i^k, \mathcal{M}_{i,i})\Big)\right)\right]\end{aligned} \tag{3}$$

Algorithm 1: ESim Training Framework

Require: HIN $G = (V, E, R)$, a user-specified meta-path $\mathcal{M}$, sampling times t, and negative sampling ratio K
Return: Vertex Embedding Vectors $\mathbf{x}_u, \forall u$
initialize parameters $\mu, \mathbf{p}_{\cdot}, \mathbf{q}_{\cdot}, \mathbf{x}_{\cdot}$
while *not converge* **do**
  for $i = 1$ ***to*** t **do**
    $p^{+} \leftarrow$ a sampled positive path instance following the meta-path $\mathcal{M}$
    Optimize for a path instance p^{+} with label 1.
    $s \leftarrow$ the first vertex on p^{+}
    for $k = 1$ ***to*** K **do**
      $p^{-} \leftarrow$ a sampled negative path instance following the meta-path $\mathcal{M}$ starting from s
      Optimize for a path instance p^{-} with label 0.
return $\mathbf{x}$.

The following loss function under the pairwise setting can be derived from NCE utilizing the same approximation.

$$\begin{aligned}\mathcal{L}_{\mathcal{M},\text{pair}} \approx & \sum_{\mathcal{P}_{e_1 \rightsquigarrow e_L} \text{ following } \mathcal{M}} \log \sigma\Big(\sum_{i=1}^{L}\sum_{j=i}^{L} f(u_i, v_j, \mathcal{M}_{i,j})\Big) + \\ & \sum_{k=1}^{K} \mathbb{E}_{\mathcal{P}^{k}_{e_1 \rightsquigarrow e_L} \sim Pr^{-}|u_1, \mathcal{M}}\left[\log\left(1-\sigma\Big(\sum_{i=1}^{L}\sum_{j=i}^{L} f(u_i^k, v_j^k, \mathcal{M}_{i,j})\Big)\right)\right]\end{aligned} \tag{4}$$

Online Similarity Search. For any interested pairs of vertices u and v, their similarity is defined by the cosine similarity between $\mathbf{x_u}$ and $\mathbf{x_v}$, i.e., $sim(u, v) = \frac{\mathbf{x_u}^T \mathbf{x_v}}{\|\mathbf{x_u}\| \cdot \|\mathbf{x_v}\|}$. We choose the cosine similarity metric instead of the function f in Eq. (1) because the norm of vectors $\mathbf{x_u}$ and $\mathbf{x_v}$ do not help the similarity search task [22]. Moreover, cosine similarity is equivalent to Euclidean distance when $||\mathbf{x_u}|| = ||\mathbf{x_v}|| = 1$, which makes the top-$k$ similar vertices of the given vertex u able to be efficiently solved using approximate nearest neighbors search [18] after normalizations.

4.2 Optimization Algorithm

The number of vertex pairs $\langle u, v\rangle$ that are connected by some path instances following at least one of user-specified meta-paths can be $O(|V|^2)$ in the worst case, which is too large for storage or processing when $|V|$ is at the level of millions and even billions, and thus makes it impossible to directly handle the projected networks over the meta-paths. Therefore, sampling a subset of path instances according to their distribution becomes the best and most feasible choice when optimizing, instead of going through every path instance per iteration. Thus even if the network itself contains a large number of edges, our method is still very efficient. The details of our training framework is shown in Algorithm 1.

Once a path instance following the meta-path $\mathcal{M}$ has been sampled, the gradient decent method is used to update the parameters $\mathbf{x_u}, \mathbf{x_v}, \mathbf{p}_{\mathcal{M}}, \mathbf{q}_{\mathcal{M}}$, and $\mu_{\mathcal{M}}$ one by one. As a result, our sampling-based training framework (Algorithm 1) becomes a stochastic gradient decent framework. The derivations of these gradients are easy and thus are omitted. Moreover, many prior studies have shown that the stochastic gradient descent can be parallelized without any locks. For example, Hogwild [21] provides a general and lock-free strategy for fully parallelizing any stochastic gradient descent algorithms in a shared memory. We utilize this technique to speed up our optimization via multi-threads.

Algorithm 2: Pre-computation of $C(u, i|\mathcal{M})$

Require: HIN $G = (V, E, R)$ and meta-path $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$
Return: $C(u, i|\mathcal{M})$
/* initialization */
for *each vertex* $u \in V$ **do**
  if u *is possibly as the second vertex in* r_L **then**
    $C(u, L+1|\mathcal{M}) \leftarrow 1$
  else
    $C(u, L+1|\mathcal{M}) \leftarrow 0$
/* dynamic programming */
for $i \leftarrow L$ ***to*** 1 **do**
  for *each vertex* $u \in V$ **do**
    $C(u, i|\mathcal{M}) \leftarrow 0$
  for *each type* r_i *edge* $\langle u, v, r_i \rangle$ **do**
    $C(u, i|\mathcal{M}) \leftarrow C(u, i|\mathcal{M}) + C(v, i+1|R)$
Return: $C(u, i|\mathcal{M})$

4.2.1 Efficient Sampling

Given a length-L meta-path $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$, there might be $O(|V|^L)$ different path instances in total. It becomes an obstacle for storing all the path instances while directly sampling over them takes a constant time. We propose to run a dynamic programming algorithm computing auxiliary numbers so that the online sampling part can be done in a constant time.

Pre-computation. As mentioned in Sec. 3, the probability of sampling a path instance following the meta-path $\mathcal{M}$ is only related to $C(u, i|\mathcal{M})$, which represents the count of path instances following the meta-path $\mathcal{M}$ with the i^{th} vertex being u.

First, we need to figure out the boundary case. When $i = L + 1$, for any vertex u, if it is possible to be the next vertex in an edge of r_L (i.e., it could be v_L), we have $C(u, L+1|\mathcal{M}) = 1$. Otherwise, it should be 0.

Then, we derive the following recursion formula when $1 \leq i \leq L$ for any vertex u.

$$C(u, i|\mathcal{M}) = \sum_{v|\langle u,v,r_i \rangle \in E} C(v, i+1|\mathcal{M}) \tag{5}$$

An efficient way to do the summation in this formula is to traverse all type-r_i edges starting from vertex u as shown in Algorithm 2. Its detailed time complex analysis will be presented later.

Online Sampling. Based on the pre-computed $C(u, i|\mathcal{M})$, one can easily figure out an efficient online sampling method for path instances following the user-given meta-path $\mathcal{M}$. The key idea is to sample the vertices on the path instance one by one. That is, the i-th vertex is conditioned on the previous $i-1$ vertices. As shown in Algorithm 3, the sampling pool for the i-th vertex is restricted to the adjacent vertices (via type-r_i edges) of the previous $(i-1)$-th vertex.

However, things are a little different when dealing with the negative path instances. First, the negative path instances are associated with a positive path instance and thus the first vertex is fixed. Second, the remaining vertices on the negative path instances are independent. Therefore, they are all sampled from V based on $\propto C(u, i|\mathcal{M})^\gamma$, $\forall i > 1$.

4.2.2 Weighted Combination

Sometimes, due to the subtle semantic meanings of the similarity, instead of a single meta-path, the weighted combination of n meta-paths could enhance the performance of similarity search. Suppose $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ are the weights ($\forall i, \lambda_i > 0$ and $\sum_{i=1}^n \lambda_i = 1$), the unified loss function becomes the weighted sum over the loss functions of individual meta-paths based on the weights. That is, $\mathcal{L}_{\text{seq}} = \sum_{i=1}^n \lambda_i \mathcal{L}_{\mathcal{M}_i,\text{seq}}$ and $\mathcal{L}_{\text{pair}} = \sum_{i=1}^n \lambda_i \mathcal{L}_{\mathcal{M}_i,\text{pair}}$. The Algorithm 1 can be modified accordingly by first sampling a meta-path $\mathcal{M}$ from $\forall j, Pr(\mathcal{M}_j) = \lambda_j$ in the beginning of the "while" loop.

The weighted combination of meta-paths can be either explicitly specified by users or learned from a set of similar/dissimilar examples provided by users. Such user-guided meta-path generation has been studied in [27] without considering embedding. Because weight learning is beyond the scope of this paper, we leave such an extension to embedding-based similarity search to a future work, and adopt grid searches to obtain the best weighted combination of meta-paths in Sec. 5.3 assuming we have the groundtruth.

Algorithm 3: Sample a positive path instance p^+

Require: HIN $G = (V, E, R)$, meta-path $\mathcal{M} = \langle r_1, r_2, \ldots, r_L \rangle$, $C(u, i|\mathcal{M})$, and weighting factor γ
Return: a positive path instance following $\mathcal{M}$
$u_1 \leftarrow$ a random vertex $\propto C(u_1, 1|\mathcal{M})^\gamma$ from V
for $i = 1$ ***to*** L **do**
  $V_i \leftarrow \{v|\langle u_i, v \rangle \in E_{r_i}\}$
  /* γ is only applied at the first vertex when sampling positive path instances. */
  $v_i \leftarrow$ a random vertex $\propto C(u_i, i|\mathcal{M})$ from V_i
  if $i < L$ **then**
    $u_{i+1} \leftarrow v_i$
return $\mathcal{P}_{e_1 \rightsquigarrow e_L} = \langle e_1 = \langle u_1, v_1, r_1 \rangle, e_2 = \langle u_2, v_2, r_2 \rangle, \ldots, e_L = \langle u_L, v_L, r_L \rangle \rangle$

4.2.3 Complexity Analysis

Pre-computation. For a given length-L meta-path $\mathcal{M}$, we have $u \in V$ and $1 \leq i \leq L + 1$, which means the memory complexity is $O(|V|L)$. For each given i, we only have to consider all type-r_i edges starting from different vertices, which implies the time complexity is $O((|V| + |E|)L)$.

Online sampling. We have adopted the alias method [31] to make the online sampling from any discrete distribution $O(1)$. Therefore, we have to precompute all discrete distributions and restore them in the memory. Given a length-L mete-path $\mathcal{M}$, for the negative sampling, we have $O(L)$ different distributions and each of them is over $|V|$ discrete values; for the positive sampling there are $O(|V|L)$ different distributions but the number of variables depends on the number of edges of the certain type. The total discrete values they have is $O(|E|L)$. In summary, both the time and memory complexities of the preparation of the alias method are $O((|V| + |E|)L)$, while the time complexity of sampling any path instance becomes $O(L)$.

Optimization for a path instance. For a specific path instance $\mathcal{P}_{e_1 \rightsquigarrow e_L}$, the time complexity is $O(dL)$, and $O(dL^2)$ for different loss functions $\mathcal{L}_{\text{seq}}$ and $\mathcal{L}_{\text{pair}}$ respectively.

Overall. In our framework, for Algorithm 1, there are $O(tK)$ times of path optimization per iteration. Suppose there are T iterations before convergence, considering the choices of different loss functions, $\mathcal{L}_{\text{seq}}$ and $\mathcal{L}_{\text{pair}}$, the over-

Table 1: Dataset Statistics. Whole networks in DBLP and Yelp are utilized in training, while grouping labels are only used for evaluation.

Dataset	DBLP	Yelp
Node Types	paper(P), author(A), term(T), venue(V)	review(R), name(N), business(B), word(W)
Edge Types	$P-A$, $P-T$, $P-V$	$B-N$, $R-B$, $R-W$
# of Vertices	2,762,595	1,616,341
# of Edges	103,059,616	76,708,201
Interesting Meta-paths	$A-P-A$, $A-P-V-P-A$, $A-P-T-P-A$	$B-R-W-R-B$, $B-N-B$
Two Grouping Labels	Research Area/Group	Business/Restaurant Type

all time complexity is $O(TtKLd)$ and $O(TtKL^2d)$ respectively. In addition, considering the complicated case of n user-specified meta-paths, $O(n(|V|+|E|)L)$ has to be paid for pre-computations before sampling, where L is the maximum length of given meta-paths.

Parallelization. It has been proved that stochastic gradient descent can be fully parallelized without locks [21]. If we use k threads, although the pre-computations remain the same, the time complexity of training can be k times faster.

5. EXPERIMENTS

In this section, we evaluate our proposed model ESim, comparing with several state-of-the-art methods on two real-world large-scale HINs both quantitatively and qualitatively. Also, we evaluate the sensitivity of parameters and the efficiency of ESim.

5.1 Datasets

Table 1 shows the statistics of two real-world large-scale HINs: **DBLP** and **Yelp**. The first dataset, *DBLP*, is a bibliographic network in computer science, which includes papers (P), authors (A), venues (V), and terms (T) as four types of vertices and takes P as the center in a star network schema. There are 3 types of undirected edges: $P-A$, $P-V$, and $P-T$. The interesting meta-paths that may be specified by users are the co-authorship meta-path $A-P-A$, the shared venue meta-path $A-P-V-P-A$, and the shared term meta-path $A-P-T-P-A$. We have the following two groupings labeled by human experts.

- **Research Area.** We use the 4-area grouping in *DBLP* labeled by human experts, which was used when evaluating PathSim [26]. There are 3,750 authors from 4 diverse research domains of computer science: "*data mining*", "*database*", "*machine learning*" and "*information retrieval*".
- **Research Group.** This grouping in *DBLP* is also labeled by human experts and is more fine-grained comparing to the *Research Area* grouping. There are 103 authors from 4 research groups: "*Christos Faloutsos*", "*Jiawei Han*", "*Michael I. Jordan*", and "*Dan Roth*".

The second dataset, *Yelp*, is a social media network of Yelp, released in Yelp Dataset Challenge[1]. This network includes businesses (B), words in business names (N), reviews of businesses (R), and words in reviews (W) as vertices. There are 3 different types of undirected edges: $B-N$, $B-R$, and $R-W$. The interesting meta-paths that may be specified by users are the shared review word meta-path $B-R-W-R-B$ and the shared name word meta-path $B-N-B$. We have the following two groupings extracted from the meta-data provided in *Yelp* dataset.

[1]https://www.yelp.com/academic_dataset

- **Business Type.** There are various business types in the *Yelp* dataset, such as "*restaurants*", "*hotels*", "*shopping*", and "*health and medical*". Businesses with multiple categories have been discarded to avoid ambiguity. To keep the results from being biased by some dominating types, we randomly sample 881 businesses from each of these four types as labeled data, because the 4-th popular type contains that many businesses.
- **Restaurant Type.** Since the majority of businesses in the *Yelp* dataset are restaurants, we look deep into them by dividing them into different types of cuisines. More specifically, we have sampled 270 restaurants from 5 cuisines respectively: "*Chinese*", "*Japanese*", "*Indian*", "*Mexican*", and "*Greek*". As a result, there are in total 1350 labeled restaurants in our labeled dataset.

5.2 Experimental Setting

Meta-path. We select different meta-paths for different datasets to see how the meta-paths will reflect the user-preferred similarity and affect the performance. In addition, we run grid searches against different groupings to obtain the best weights of different meta-paths for ESim models.

Compared Algorithms and Notations. We select the previous state-of-the-art algorithm in the meta-path guided similarity search problem, PathSim, which has been reported to beat many other simiarity search methods, for example, SimRank [11], P-PageRank [10], random walk, and pairwise random walk. In addition, we also consider (heterogeneous) network embedding methods, such as LINE and PTE, which beat other embedding methods like graph factorization [1] and DeepWalk [20]. More details about these methods are as follows.

- PathSim [26] is a meta-path guided similarity search algorithm which utilizes the normalized count of path instances following the user selected meta-path between any pair of vertices. When the meta-path involves text (e.g., $A-P-T-P-A$), PathSim becomes a *text-based similarity* — the cosine similarity using bag-of-words.
- LINE [29] is an embedding algorithm specifically designed for homogeneous networks, which considers both first and second order information in a network (i.e., the neighbors and the neighbors of the neighbors). By treating all vertices and edges in the HIN as homogeneous ones, we can directly apply LINE and denote the model with first order only as **LINE-1st** and the model using the second order information as **LINE-2nd** respectively. One can also project the HIN to a weighted homogeneous network based on the user selected meta-path and apply LINE. However, based on our experiments, the results are always worse than PTE and thus omitted.
- PTE [28] decomposes a HIN to a set of edgewise bipartite networks and then learn embedding vectors. To adapt this method to our settings, the way with the best performance we discovered is to project the HIN to a weighted bipartite HIN based on the user selected meta-path. For example, if the selected meta-path is the shared venue meta-path $A-P-V-P-A$ in the bibliographic network, we construct a bipartite HIN consisting of A and V, where the weight of edges between any pair of a type-A vertex and a type-V vertex equals to the numbers of path instances following $A-P-V$ between them.

ESim refers to our proposed meta-path guided embedding model. Considering the choice of loss functions $\mathcal{L}_{\mathcal{M},\text{seq}}$ and

Table 2: AUC Evaluation on DBLP dataset.

Meta-path	Model	Research Area	Research Group
N/A	LINE-1st	52.32%	52.89%
	LINE-2nd	51.82%	51.53%
$A-P-A$	PathSim	52.07%	76.75%
	PTE	50.90%	77.07%
	ESim-seq	52.97%	73.97%
	ESim-pair	52.87%	**81.46%**
$A-P-V-P-A$	PathSim	80.51%	72.60%
	PTE	74.40%	65.87%
	ESim-seq	77.06%	74.83%
	ESim-pair	**83.58%**	73.07%
$A-P-T-P-A$	PathSim	55.22%	61.16%
	PTE	68.38%	73.28%
	ESim-seq	61.17%	65.73%
	ESim-pair	69.18%	74.96%
Best Weighted Combination (grid search)	ESim-pair	**83.81%** 0.1 $A-P-A$ + 0.9 $A-P-V-P-A$	**82.27%** 0.9 $A-P-A$ + 0.1 $A-P-V-P-A$

$\mathcal{L}_{\mathcal{M},\text{pair}}$, the corresponding model are denoted as ESim-seq and ESim-pair respectively.

Default Parameters. The parameter γ controlling the effect of overly-popular vertices is set to 3/4 inspired from [16]. The dimension of the vertex embedding vectors, d, is set to 50. The negative sampling ratio K is set to 5, whereas the sampling times t is set to 1 million by default. The number of working cores is set to 16. Talking about the initialization of global bias, local bias vectors, and embedding vectors, we assign all parameters as a random real value uniformly in $[-1, 1]$. The learning rate in stochastic gradient descent is initialized as 0.25 and later linearly decreased.

Machine. The following experiments on execution time were all conducted on a machine equipped two Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz with 20 physical cores in total. Our framework is fully implemented in C++[2].

5.3 AUC Evaluations

Although it is hard to obtain the labels of the detailed rankings among all pairs of vertices, it is relatively easy to give an estimation based on the labels of the vertex groupings $l(\cdot)$. Considering the ranking problem for each individual vertex u, if we rank the other vertices based on the similarity scores, it is very natural that we expect the vertices from the same group (similar ones) are at the top of the ranking list whereas the dissimilar ones are at the bottom of the list. More specifically, we define the AUC score as follows. For a better similarity metric, the AUC score should be larger.

$$AUC = \frac{1}{|V|}\sum_{u\in V}\frac{\sum_{v,v'\in V\wedge l(u)=l(v)\wedge l(u)\neq l(v')} \mathbb{1}_{sim(u,v)>sim(u,v')}}{\sum_{v,v'\in V\wedge l(u)=l(v)\wedge l(u)\neq l(v')} 1}$$

Note that the models that generate similarity measures are learned from the whole dataset, whereas the AUC metric is calculated only in the subset of vertices where we have group labels. The subset is usually small because computing AUC needs pairwise similarities among the subset.

We have the following observations from Tables 2 and 3. First, about single meta-path we have:

- *User-guidance is crucial.* The choice of the user-selected meta-path is really important and affects the performance of user-guided models significantly. For example, $A-P-V-P-A$ works the best in the *Research Area* grouping, where the shared venue is more telling. However, collaborations are more important in *Research Groups* grouping, and thus $A-P-A$ works the best. In the *Yelp* dataset, although business names carry some semantics, words in review are more telling. In both groupings, $B-R-W-R-B$ always better fits the user-preferred similarity. These phenomena imply that a better fit meta-path will lead models such as ESim to better performance.
- ESim-*pair performs better than* ESim-*seq*, because the pairwise loss function exploits the meta-path guidance in a more thorough way. We will focus on ESim-pair in later experiments.
- *User-guided models perform better.* As long as the selected meta-path is reasonable (e.g., $A-P-V-P-A$ in the *Research Area* grouping , $A-P-A$ in the *Research Group* grouping, and $B-R-W-R-B$ in the *Yelp* dataset), ESim, PathSim, and PTE always perform better than LINE, which proves the importance of following the guidance from user-selected meta-paths.
- ESim-*pair performs the best* with significant advantages over PathSim and PTE, which demonstrates the power of embedding techniques and proper usage of network structures respectively.

Table 3: AUC Evaluation on Yelp dataset.

Meta-path	Model	Business Type	Restaurant Type
N/A	LINE-1st	78.89%	82.39%
	LINE-2nd	80.20%	70.20%
$B-R-W-R-B$	PathSim	83.49%	74.66%
	PTE	85.67%	83.77%
	ESim-seq	84.39%	78.62%
	ESim-pair	**89.22%**	**88.73%**
$B-N-B$	PathSim	53.77%	55.26%
	PTE	61.69%	63.18%
	ESim-seq	62.53%	61.27%
	ESim-pair	59.61%	59.39%

Meta-path Combination. We choose the best performing meta-paths $A-P-A$ and $A-P-V-P-A$ in the *DBLP* dataset and run grid searches for best weights to achieve highest AUC scores in *Research Area* and *Research Group* groupings respectively. Because the best performing meta-path in *Yelp* dataset is always $B-R-W-R-B$, any combination is useless in this case. Note that this grid search against grouping labels shows an upper bound of the highest possible AUC score, which can be rarely achieved without knowing labels. As shown in Table 2, the improvement of best weighted meta-paths is marginal. Therefore, weighted combination might be necessary to achieve the best performance but the choice of meta-paths is more important.

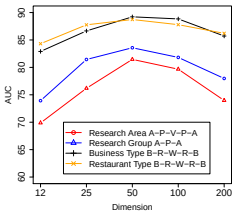
5.4 Visualizations

With embedding vector for each vertex, we can show meaningful visualizations, which layout the vertices of the same type in a two-dimensional space, and check whether the boundaries between different groups are relatively clear.

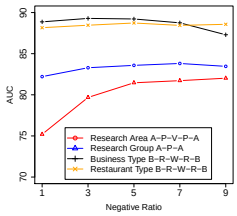
Taking the *DBLP* dataset as an example, we visualize the vertices with different colors regarding to their group labels in the *Research Area* grouping and the *Research Group* grouping. Their embedding vectors are projected to a 2-D space using the t-SNE package [30], which is a nonlinear dimensionality reduction technique and well suited for projecting high-dimensional data into a low dimensional space.

Laying out these vertex embedding vectors is challenging, especially for the vertices in the four closely related research areas: "*data mining*", "*database*", "*machine learning*" and "*information retrieval*". For different embedding-based meth-

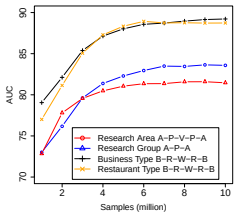
[2]The source code will be published in the author's GitHub after acceptance.



(a) Varying d.

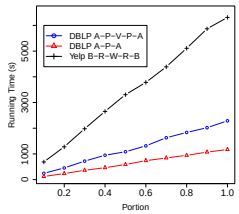


(b) Varying K.

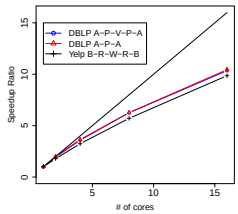


(c) Varying total samples.

Figure 2: Parameter Sensitivity of ESim-pair.

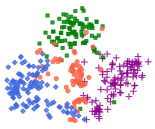


(a) Varying network sizes.



(b) Varying cores.

Figure 3: Efficiency of ESim-pair.

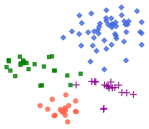


(a) ESim-pair

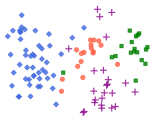
(b) PTE

(c) LINE-1st

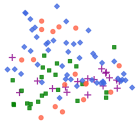
Figure 4: Visualization of embedding vectors of 10% random sampled authors in *DBLP Research Area* grouping when $\mathcal{M}$=$A-P-V-P-A$. Colors correspond to research areas: *Database*, *Data Mining*, *Information Retrieval*, and *Machine Learning*.



(a) ESim-pair



(b) PTE



(c) LINE-1st

Figure 5: Visualization of embedding vectors of all authors in *DBLP Research Group* grouping when $\mathcal{M}$=$A-P-A$. Colors correspond to research groups: *Christos Faloutsos*, *Jiawei Han*, *Michael I. Jordan*, and *Dan Roth*.

ods (i.e., ESim, PTE, and LINE), we choose to visualize their variants holding the best AUC performance, i.e., ESim-pair using $A-P-V-P-A$, PTE using $A-P-V-P-A$, and LINE-1st. As shown in Fig. 4, we visualize 10% random samples of 3,750 authors from 4 research domains of computer science. Better embedding vectors should lead to a clearer figure where the boundaries between different colored points should be clean and almost not interfering with each other. Based on the visualizations, one can easily observe that ESim-pair using $A-P-V-P-A$ provides the best embedding vectors judged from this criterion.

Similarly, the visualizations of the *Research Groups* grouping based on the models with best AUC performance are shown in Fig. 5. Our proposed model ESim-pair using $A-P-A$ clearly beats PTE using $A-P-A$ and LINE-1st.

The significant improvements over PTE and LINE observed via visualization are consistent with the previous evaluations.

5.5 Parameter Sensitivity

We select the best performing models ESim-pair to study the parameter sensitivity, such as using $A-P-V-P-A$ in the *Research Area* grouping, using $A-P-A$ in the *Research Group* grouping, and using $B-R-W-R-B$ in both *Business Type* and *Restaurant Type* groupings. We vary the parameter values and see how the AUC performance changes.

Based on the curves in Fig. 2(a), we can observe that setting the dimension (d) of vertex embedding vectors as 50 is reasonable, because too small d cannot sufficiently capture the semantics, while too large d may lead to some overfitting problem. Fig. 2(b) indicates that the AUC scores are not sensitive to the negative sample ratio K and $K = 5$ is a good choice. As shown in Fig. 2(c), as more samples are optimized during training, the AUC scores keep an increasing trend and finally converge.

5.6 Scalability and Parallelization

We investigate the efficiency of ESim by considering both the scalability and the parallelization as shown in Fig. 3(a). We try different portions of network sizes (i.e., $|V| + |E|$) in the two networks and run our best performing models, i.e., ESim-pair using $A-P-V-P-A$ and using $A-P-A$ on the *DBLP* dataset, as well as ESim-pair using $B-R-W-R-B$ on the *Yelp* dataset. Based on these curves, the running time is linear to the size of networks while the longer meta-path costs a little more time, which are consistent with our previous theoretical time complexity analysis. We vary the number of working cores and run our models on the *DBLP* and *Yelp* datasets. The results are plotted in Fig. 3(b). The speedup is quite close to linear, which shows that ESim is quite scalable to the number of working cores.

6. CONCLUSIONS

In this paper, we propose a general embedding-based similarity search framework for heterogeneous information networks (HINs). Our proposed model, ESim, incorporates given meta-paths and network structures to learn vertex embedding vectors. The similarity defined by the cosine similarity between vertex embeddings of the same type has demonstrated its effectiveness, outperforming the previous state-of-the-art algorithms on two real-world large-scale HINs. The efficiency of ESim has also been evaluated and proved to be scalable.

There are several directions to further extend this work. First, instead of similarities between vertices of the same type, one can also explore the relevances between vertices of different types. Second, a mechanism could be developed to automatically learn and extract a set of interesting meta-paths or their weighted combinations from user-provided rankings or preferences. Third, similarity is the fundamental operation for mining and exploring HINs. This study on similarity measure, defined in HINs based on meta-path guided embedding, and its efficient computations will impact other searching and mining problems in HINs. For example, it is necessary to re-examine clustering, classification and prediction functions in HINs by reconsidering the similarity measures defined based on meta-path guided embedding. Also, mining outliers in networks can be formulated as finding a small subset of vertices with extremely low similarities to other vertices or clusters.

7. REFERENCES

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *WWW*, pages 37–48, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. volume 51, pages 117–122, New York, NY, USA, Jan. 2008. ACM.

[3] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *SIGKDD*, pages 119–128, New York, NY, USA, 2015. ACM.

[4] H. Cho, B. Berger, and J. Peng. Diffusion component analysis: Unraveling functional topology in biological networks. In *Research in Computational Molecular Biology*, pages 62–64. Springer, 2015.

[5] P. L. Duren. *Theory of Hp spaces*, volume 38. IMA, 1970.

[6] K. Gu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *EMNLP 2015*, 2015.

[7] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *JMLR*, 13(1):307–361, 2012.

[8] D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. In *SIGKDD*, pages 387–396, New York, NY, USA, 2015. ACM.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, New York, NY, USA, 1998. ACM.

[10] G. Iván and V. Grolmusz. When the web meets the cell: using personalized pagerank for analyzing protein interaction networks. *Bioinformatics*, 27(3):405–407, 2011.

[11] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *SIGKDD*, pages 538–543, New York, NY, USA, 2002. ACM.

[12] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279, New York, NY, USA, 2003. ACM.

[13] N. Katayama and S. Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *SIGMOD*, pages 369–380, New York, NY, USA, 1997. ACM.

[14] M. Levandowsky and D. Winter. Distance between sets. *Nature*, 234(5323):34–35, 1971.

[15] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, pages 754–764. International World Wide Web Conferences Steering Committee, 2015.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[17] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *ICML*, 2012.

[18] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2, 2009.

[19] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, New York, NY, USA, 2014. ACM.

[21] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[22] A. M. Schakel and B. J. Wilson. Measuring word significance using distributed representations of words. *arXiv preprint arXiv:1508.02297*, 2015.

[23] J. Shang, T. Chen, H. Li, Z. Lu, and Y. Yu. A parallel and efficient algorithm for learning to match. In *ICDM*, pages 971–976. IEEE, 2014.

[24] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[25] Y. Sun and J. Han. Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explor. Newsl.*, 14(2):20–28, Apr. 2013.

[26] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB 2011*, 2011.

[27] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *SIGKDD*, pages 1348–1356, New York, NY, USA, 2012. ACM.

[28] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *SIGKDD*, pages 1165–1174. ACM, 2015.

[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[30] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

[31] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, Sept. 1977.

[32] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, 2016.

[33] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[34] X. Yu, Y. Sun, B. Norick, T. Mao, and J. Han. User guided entity similarity search using meta-path selection in heterogeneous information networks. In *CIKM*, pages 2025–2029, New York, NY, USA, 2012. ACM.