ATP: Directed Graph Embedding with Asymmetric Transitivity Preservation

Jiankai Sun, Bortik Bandyopadhyay, Armin Bashizade, Jiongqian Liang, P. Sadayappan, Srinivasan Parthasarathy
Department of Computer Science and Engineering, The Ohio State University, USA
{sun.1306,bandyopadhyay.14,bashizade.1,liang.420,sadayappan.1}.osu.edu,srini@cse.ohio-state.edu

Abstract

Directed graphs have been widely used in Community Question Answering services (CQAs) to model asymmetric relationships among different types of nodes in CQA graphs, e.g., question, answer, user. Asymmetric transitivity is an essential property of directed graphs, since it can play an important role in downstream graph inference and analysis. Question difficulty and user expertise follow the characteristic of asymmetric transitivity. Maintaining such properties, while reducing the graph to a lower dimensional vector embedding space, has been the focus of much recent research. In this paper, we tackle the challenge of directed graph embedding with asymmetric transitivity preservation and then leverage the proposed embedding method to solve a fundamental task in CQAs: how to appropriately route and assign newly posted questions to users with the suitable expertise and interest in CQAs. The technique incorporates graph hierarchy and reachability information naturally by relying on a non-linear transformation that operates on the core reachability and implicit hierarchy within such graphs. Subsequently, the methodology levers a factorization-based approach to generate two embedding vectors for each node within the graph, to capture the asymmetric transitivity. Extensive experiments show that our framework consistently and significantly outperforms the state-of-the-art baselines on three diverse real-world tasks: link prediction, and question difficulty estimation and expert finding in online forums like Stack Exchange. Particularly, our framework can support inductive embedding learning for newly posted questions (unseen nodes during training), and therefore can properly route and assign these kinds of questions to experts in CQAs.

Introduction

Community Question Answering services (CQAs) such as Stack Exchange and Yahoo! Answers are examples of social media sites, with their usage being examples of an important type of computer supported cooperative work in practice. In recent years, the usage of CQAs has seen a dramatic increase in both the frequency of questions posted and general user activity. This, in turn, has given rise to several interesting problems ranging from expertise estimation to question difficulty estimation, and from automated question routing to incentive mechanism design on such CQAs (Fang et al. 2016; Sun et al. 2018a). Previous work (Wang, Jing Liu, and Guo 2014; Sun et al. 2018a) proposed to assign a scalar value to represent question difficulty (and user expertise). However, question difficulty and user expertise can vary in different topics. In Stack Exchange sites, users are required to use tags (a tag is a word or phrase) to describe the topic(s) of the question [1]. Each question can be assigned multi-tags to represent its most relevant topics. For example, in our experiments, the average number of tags per question is 2.82 and 2.96 in Stack Exchange site Apple and Physics respectively. Hence a solely scalar value to represent question difficulty level or user expertise is not thorough.

Some graph embedding methods (Fang et al. 2016; Zhao et al. 2016; Zhao et al. 2017) are then proposed to address the above limitation. The problem of graph embedding seeks to represent vertices of a graph in a low-dimensional vector space in which meaningful semantic, relational and structural information conveyed by the graph can be accurately captured (Ma et al. 2018). Recently, one has seen a surge of interest in developing such methods including ones for learning such representations for directed graphs (while preserving important properties) (Ou et al. 2016), which is the focus of our research. A property of singular importance within a directed graph is asymmetric transitivity, which plays a very important role in tasks of graph inference and analysis (Ou et al. 2016). Question difficulty and user expertise follow the characteristic of asymmetric transitivity. For example, given a question q_1 is easier than q_2 and q_2 is easier than q_3, we can infer that q_1 is easier than q_3 easily. It happens to estimating user expertise too. We can infer that u_1 has more expertise than u_3 based on the fact that u_1 has more expertise than u_2 and u_2 has more expertise than u_3 in a specific domain. In this paper, we tackle the challenge of directed graph embedding with asymmetric transitivity preservation and then leverage the proposed embedding method to solve a fundamental task in CQAs: how to appropriately route and assign newly posted questions to users with the suitable expertise and interest in CQAs.

HOPE (Ou et al. 2016), one of the state-of-art directed graph embedding methods, relies on high-order proximity features (e.g. Adamic Adar (AA), Katz Index (KI), Common Neighbors (CN)) to approximate asymmetric transitivity.

[1] https://stackoverflow.com/help/tagging

Zhou et al. (Zhou et al. 2017) proposed a random walk based graph embedding method named as APP which can implicitly preserve the Rooted PageRank (RPR), another higher-order proximity feature, in the embedding space. However, cycles in directed graphs are very common, and these cycles can undermine the performance of embedding strategies such as HOPE and APP, and hence severely limit the capability of the learned embedding vectors in graph inference and analysis. Figure 1 illustrates the limitation of using high order proximities for preserving asymmetric transitivity.

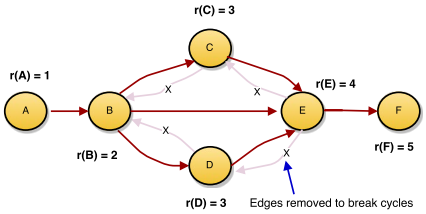

Figure 1: Illustration of the limitation of using high order proximities for preserving asymmetric transitivity due to the existence of cycles in the graph: the KI proximity from B to E (inner product between the source vector u_B^s and the target vector u_E^t, generated by HOPE) represented as $KI(B, E)$ is 0.0041, which is smaller than the KI proximity from E to B (inner product between the source vector u_E^s and the target vector u_B^t) represented as $KI(E, B) = 0.0067$. HOPE will predict the edge direction is from E to B, which is opposite to the real edge (B, E). A similar problem occurs with RPR too, since $RPR(B, E) = 0.2129$, which is smaller than $RPR(E, B) = 0.2446$. Due to the existence of cycles among node B, C, D, and E, CN and AA are the same for node pair (B, E) and (E, B). The AA proximity predicted by HOPE are $AA(B, E) = AA(E, B) = 0.5$, and CN proximity are $CN(B, E) = CN(E, B) = 0$. Hence neither AA or CN can make a confident prediction for the transitivity between B and E. However, with using our framework ATP, we can address above limitation. For example, ATP can predict $ATP(B, E) = 1.48$ and $ATP(E, B) = 8.18e^{-10}$, which strongly indicates that the edge direction is from B to E.

A strong hierarchical structure in the context of directed networks can help explain complex interactions in many real-world phenomena (Tatti 2015), including asymmetric transitivity. Each node can be assigned a ranking score to represent where it stands in the entire network. The relationship among nodes in such a scenario is fully transitive. For example, if a node i has a lower hierarchy than j, and j has a lower hierarchy than k, we then can infer that i must have a lower hierarchy than k. In graphs with strong hierarchical structure, edges are expected to flow from lower hierarchies to higher hierarchies (Gupte et al. 2011; Tatti 2015). However, when transitivity is being predicted leveraging the graph hierarchy alone, without incorporating the inherent graph reachability property, it can sometimes lead to false positive predictions. For example, a lower hierarchy node in a subgraph may not reach a higher hierarchy node in another subgraph which has no connection with the previous subgraph. To redress such problems, one may want to explicitly account for graph reachability as discussed next.

Transitive closure (TC) of a directed graph is a methodology (usually housed in a simple data structure) that makes it possible to answer reachability questions. The TC of a graph $G = (V, E)$ is a graph $G^+ = (V, E^+)$ such that for all v, w in V there is an edge (v, w) in E^+ if and only if there is a non-null path from v to w in G. However, computing TC for large directed graphs with cycles is expensive, while computing TC of directed acyclic graphs (DAGs) is practical (Simon 1988). To leverage the above intuition, we propose to first remove a subset of cycle edges which violate the graph hierarchy to reduce a directed graph to a DAG and then leverage the TC of the reduced DAG to represent graph reachability. We examined several strategies for breaking cycles while preserving the graph hierarchy as much as possible (Herbrich, Minka, and Graepel 2007; Tatti 2015), and found an ensemble approach proposed by Jiankai et al. (Sun et al. 2017) coupling some of these approaches can meet our requirements. Another benefit of breaking cycles is that the reduced DAG has a very strict hierarchy, and each vertex can be assigned a ranking score effectively and efficiently.

A key challenge now is how to incorporate graph hierarchy and reachability in a unified framework to preserve the asymmetric transitivity in the embedding space. To this end, we build an asymmetric matrix $\boldsymbol{M}$, which is a non-linear transformation of a diagonal matrix $\boldsymbol{D}$ and an adjacency matrix $\boldsymbol{A}$. Here, $\boldsymbol{A}$ is the adjacency matrix of the transitive closure of the reduced DAG which implicitly contains graph reachability information, and $\boldsymbol{D}$ is a diagonal matrix storing the nodes' hierarchical ranking score along the diagonal entries of a square matrix. Then a factorization based method is applied to $\boldsymbol{M}$ to generate approximate embeddings. In our experiments, an efficient non-negative matrix factorization (NMF) (Cheng et al. 2017) using Cyclic Coordinate Descent(CCD) (Nisa et al. 2017) with appropriate regularization is leveraged to generate the embedding. Two embedding vectors, source and target vector, are learned for each node to capture the asymmetric transitivity. Through the time complexity analysis of all procedures in our proposed **A**symmetric **T**ransitivity **P**reserving graph embedding framework (**ATP**), we demonstrate that ATP can be applied to large directed graphs efficiently.

We also conducted extensive experiments to verify the usefulness and generality of the learned embedding in various tasks such as link prediction, and question difficulty estimation and expert finding in online CQAs such as Stack Exchange sites. Particularly, ATP can support inductive embedding learning for newly posted questions (unseen nodes during training), and therefore can route and assign these kinds of questions to approximate experts in CQAs, which tackles a fundamental challenge in crowdsourcing.

Related Works

Graph embedding approaches fall into three broad categories classified by Goyal et al. (Goyal and Ferrara 2017):

(1) Factorization based, (2) Random Walk based (Perozzi, Al-Rfou, and Skiena 2014; Yang et al. 2015; Gao et al. 2018), and (3) Deep Learning based (Pan et al. 2016; Dong, Chawla, and Swami 2017; Liang et al. 2018). Our proposed ATP is factorization based, and hence we focus on discussing about factorization based techniques in this section.

Factorization based graph embedding usually solves the graph embedding problem in two steps as follows: (1) represent the connections between nodes in the form of a matrix, and (2) factorize the matrix to get a set of node embedding (Cai, Zheng, and Chang 2017; Goyal and Ferrara 2017). Based on how we construct the input matrix, matrix factorization based approaches are categorized into two types: One is to factorize graph Laplacian, and the other is to directly factorize the node proximity matrix (Cai, Zheng, and Chang 2017). The node proximity is preserved by minimizing the loss during the factorizing the node proximity matrix.

It has been recently shown that many popular random walk based approaches such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015), and node2vec (Grover and Leskovec 2016) can be unified into the matrix factorization framework with closed forms (Qiu et al. 2018). However, these methods ignore the asymmetric nature of the path sampling procedure and train the model symmetrically, which restricts their applications. Since node pairs from two hop away will be regarded as negative labels, LINE can only preserve symmetric second-order proximity when applied to directed graphs (Zhou et al. 2017).

Higher order proximity is considered by many traditional similarity measurements, and has been shown to be effective in many real world tasks. HOPE (Ou et al. 2016) proposed to use high-order proximities (AA, CN, RPR, and KI) to approximate asymmetric transitivity. Theoretical analysis shows that APP implicitly preserves the RPR (Zhou et al. 2017). However cycles in directed graphs as shown in Figure 1 can hurt the performance of asymmetric transitivity preserving for HOPE and APP, and hence severely limit the capability of the learned embedding vectors in graph inference and analysis.

Our Framework ATP

We now describe the **4-step work-flow of ATP framework**, which is illustrated in Figure-2 with a toy example, and then present the computational complexity of our methodology. In the **first step** (Section 3.1: Breaking Cycles), an input directed graph G is reduced to a DAG G' by removing a small set of cycle edges which violate the graph hierarchy. Then in the **second step** (Section 3.2: Inferring Graph Hierarchy), each node is assigned a ranking score efficiently based on the hierarchical structure of G'. The **third step** (Section 3.3: Incorporating Hierarchy and Reachability) involves the construction of the proposed novel objective matrix $\boldsymbol{M}$ to incorporate both graph hierarchy and reachability information. Nodes' hierarchical information can be represented using a diagonal matrix $\boldsymbol{D}$, while the transitive closure of G' is represented by $\boldsymbol{A}$. These two matrices ($\boldsymbol{A}$ and $\boldsymbol{D}$) are used to build the matrix $\boldsymbol{M}$ using a non-linear transformation which can preserve hierarchical rankings between local nodes much better than an ordinary linear model. The **final step** (Section 3.4: Generating Asymmetric Transitivity Preserving Graph Embedding from $\boldsymbol{M}$) involves the efficient application of NMF on $\boldsymbol{M}$ to produce two matrices $\boldsymbol{S}$ and $\boldsymbol{T}$, which can be interpreted as asymmetric transitivity preserving source vectors and target vectors of all the nodes in the graph.

Breaking Cycles

Reducing a directed graph $G = (V, E)$ to a DAG $G' = (V, E')$ has two obvious advantages: 1) making it possible for us to compute the transitive closure of G'; 2) inferring the hierarchy of G' becomes easier, since a DAG has a very strict hierarchy.

We examined several strategies for breaking cycles while preserving the graph hierarchy as much as possible, and found an ensemble approach *H_Voting* proposed by Jiankai et al. (Sun et al. 2017) can meet our requirements. *H_Voting* selects the edge with the highest voting score for removal in a fast, scalable, and fully automated way. The voting score of each edge is determined by the severity of their violation, which means that edges that respect the hierarchy receive a score of 0 and score increase linearly as the hierarchy violation becomes more severe. The corresponding hierarchy is inferred by ensembling TrueSkill (Herbrich, Minka, and Graepel 2007) and Social Agony (Gupte et al. 2011; Tatti 2015). Figure 1 illustrates that edges (C,B), (D,B), (E,C) and (E,D) are removed by *H_Voting* to break cycles. Empirically, it has been shown that *H_Voting*, can accurately identify the edges to be removed, even in noisy and large-scale real-world graphs. The time complexity of breaking cycles is $O(E^2)$ in the worst case, which happens in directed complete graphs [2].

Inferring Graph Hierarchy

Given that the graph has been converted to a DAG using the previous step, graph hierarchy can be inferred based on this reduced DAG. Given a graph $G = (V, E)$, inferring graph hierarchy means that we have to construct a function $r : V \to \mathbb{Z}$, which maps each vertex to an integer, representing corresponding vertex's hierarchy in G. The computed graph hierarchy is fully transitive and can be used to infer the asymmetric transitivity in G. One straightforward way to compute a ranking score for each vertex is to use topological sorting. However, topological sorting is non-deterministic. Hence we modify topological sorting algorithm by assigning a ranking score to each vertex in a DAG recursively following the steps below: Step 1) assign the current ranking score o [3] to all nodes with zero in-degree; Step 2) update the target graph by removing all zero in-degree nodes and their corresponding out-going edges; Step 3) update the current ranking score o by increasing 1.

[2] Every pair of distinct vertices is connected by a pair of unique edges (one in each direction).

[3] Ranking score o is initialized to 1.

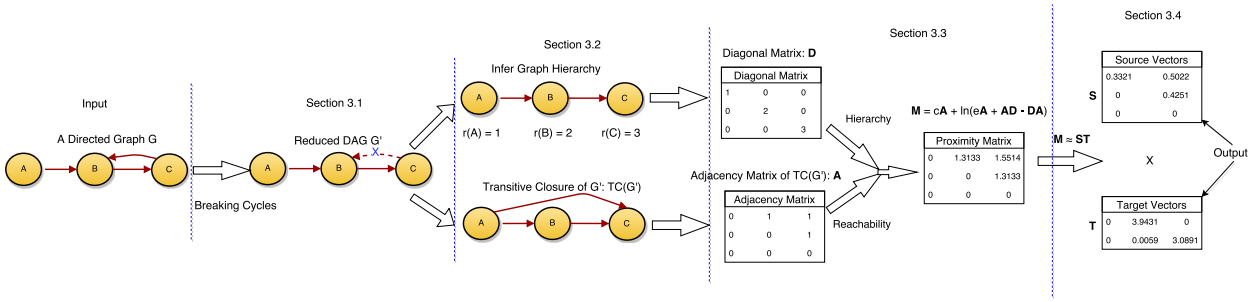

Figure 2: Illustration of **A**symmetric **T**ransitivity **P**reserving (**ATP**) graph embedding framework

Figure 1 shows each node's ranking score inferred by the above procedures. For example, node A's in-degree in the reduced DAG is 0 while its hierarchy is represented as $r(A) = 1$. The time complexity of above procedure is the same as topological sorting, which is $O(|V| + |E|)$.

Incorporating Hierarchy and Reachability

The hierarchical ranking score inferred by the methodology in Section 3.2 reflects where a node stands in the entire network, and it is fully transitive. However, the sole assumption that edges are from lower to higher hierarchy nodes fails to incorporate the inherent graph reachability property, and hence is prone to generating many false positive predictions during down-stream analysis tasks (eg: link prediction using the generated node embeddings). Thus the key challenge is to combine both graph hierarchy and reachability inside a unified framework, which we discuss next.

TC can be thought of as constructing a data structure that makes it possible to answer reachability questions. Instead of computing TC of the original directed graph G, we compute the TC of the reduced DAG G' and represent it as its adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V|\times|V|}$. If node i can reach j in G', then the corresponding element $A_{i,j} = 1$, otherwise $A_{i,j} = 0$.

$\boldsymbol{A}$ contains the information of graph reachability, but it treats the hierarchical difference between any reachable node pairs the same (equal to 1). To emphasize the impact of these non-zero elements in $\boldsymbol{A}$ and leverage graph hierarchy, a simple way is to replace each non-zero element by corresponding node pair's hierarchical difference, which is equivalent to applying a linear function to transform $\boldsymbol{A}$ to $\boldsymbol{L} \in \mathbb{R}^{|V|\times|V|}$. For each non-zero element $A_{i,j} = 1$ in $\boldsymbol{A}$, its corresponding $L_{i,j}$ in $\boldsymbol{L}$ is $\Delta_{i,j} = r(j) - r(i)$ and $\Delta_{i,j} \geq 1$.

Suppose $\boldsymbol{D} \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix, where each non-zero element in the diagonal is $D_{i,i} = r(i)$. Then we have:

$$\boldsymbol{L} = \boldsymbol{AD} - \boldsymbol{DA} \tag{1}$$

Empirically, the maximum value of Δ in $\boldsymbol{L}$ is much larger than the minimum value (which is 1). The high (and varying) range of values in Δ will unfavourably amplify the effect of large hierarchical difference values while damping the effects of smaller values, thereby negatively impacts the transitivity preserving property in local sub-graphs. To overcome this limitation, we seek to reduce the absolute values of Δ to smaller ones, while preserving its important monotonic property. We observe that a simple yet popular harmonic series, which is a non-linear and non-decreasing function, can satisfy our requirements very well and can be used to build the proximity matrix $\boldsymbol{M} \in \mathbb{R}^{|V|\times|V|}$. Each non-zero entry $L_{i,j} \in \boldsymbol{L}$ is transformed to $M_{i,j} = 1 + \frac{1}{2} + ... + \frac{1}{\Delta_{i,j}}$ in $\boldsymbol{M}$. Since a harmonic number $h(\Delta_{i,j}) = \sum_{k=1}^{\Delta_{i,j}} \frac{1}{k}$ can be approximated by $(\gamma + log(\Delta_{i,j}))$ [4], without loss of generality, we represent each non-zero element $M_{i,j} = c + log(e + \Delta_{i,j})$, where c is a constant, and e is the mathematical constant satisfying $log(e + \Delta) > 0$. Figure 2 provides an example of computing $\boldsymbol{M}$, where $c = 0$. After this non-linear transformation, the gap of hierarchical rankings between local nodes can be noticed and well preserved compared to the linear model. Thus, the final matrix $\boldsymbol{M}$ incorporating both graph hierarchy and reachability, is computed as:

$$\boldsymbol{M} = c\boldsymbol{A} + log(e\boldsymbol{A} + \boldsymbol{L}) = c\boldsymbol{A} + log(e\boldsymbol{A} + \boldsymbol{AD} - \boldsymbol{DA}) \tag{2}$$

Constructing the adjacency matrix $\boldsymbol{A}$ is equivalent to computing the transitive closure of G', which has a worst-case time complexity of $O(|V|^2 loglog(|V|))$ (Simon 1988). The time complexity of computing $\boldsymbol{L}$ with Equation 1 is equal to the number of non-zero elements in $\boldsymbol{A}$, which is $O(|V|^2)$ in the worst case. Hence, the time complexity of constructing $\boldsymbol{M}$ is $O(|V|^2 loglog(|V|))$ in the worst case.

Generating Asymmetric Transitivity Preserving Graph Embedding from $\boldsymbol{M}$

We have discussed how to build the non-negative proximity matrix $\boldsymbol{M}$, which is a function of the adjacency matrix of the transitive closure of the reduced DAG and a diagonal matrix which contains graph hierarchy. In this section, we propose to use factorization models to generate asymmetric transitivity preserving embedding for the given directed graph.

The most straightforward way is to apply NMF on $\boldsymbol{M}$ to generate asymmetric transitivity preserving embedding for the given directed graph. NMF of the $(|V| \times |V|)$ matrix

[4] γ is the Euler-Mascheroni constant, log is the Natural logarithm

$\boldsymbol{M}$ generates a low-rank approximation of it: $\boldsymbol{M} \approx \boldsymbol{ST}$, where $\boldsymbol{S} \in \mathbb{R}^{|V|\times k}$ and $\boldsymbol{T} \in \mathbb{R}^{k\times|V|}$, as shown in Figure 2. Each row in $\boldsymbol{S}$ represents a node's out-reach (source) vector, and each column in $\boldsymbol{T}$ represents a node's in-reach (target) vector. k is the dimension size of the source/target embedding space. NMF step is a core part in generating the embedding and we apply a multi-core GPU version of NMF using CCD GPUCCD++ (Nisa et al. 2017) with appropriate regularization to generate the embedding for large graphs efficiently. The time complexity per iteration of GPUCCD++ is $O(k|V|^2)$ in the worst case.

To predict whether there is a directed path from node i to node j, we check the value of $\sigma(\langle\vec{s}_i, \vec{t}_j\rangle)$, where σ is the sigmoid function, $\vec{s}_i$ is node i's source vector and $\vec{t}_j$ is node j's target vector respectively. If $\sigma(\langle\vec{s}_i, \vec{t}_j\rangle) > \alpha$, there is a predicted path from i to j. α is a threshold with range in $[0.5, 1)$. We set $\alpha = 0.5$ in our experiments, which we empirically found to work well.

Complexity Analysis

In this section, we analyze the complexity of the whole framework of ATP, given a directed graph $G = (V, E)$ as input. Fundamental procedures of ATP are breaking cycles, inferring graph hierarchy, constructing $\boldsymbol{M}$, and factorization of $\boldsymbol{M}$. In the worst case, their corresponding time complexity is $O(|E|^2)$, $O(|E| + |V|)$, $O(|V|^2 loglog(|V|))$, and $O(|V|^2 k)$ per iteration (NMF) respectively. By combining them, the time complexity of ATP is $O(|E|^2)$ in the worst case (G is a directed complete graph). The bound is very pessimistic in practice, and the bottleneck part (breaking cycles) can be parallelized since it can perform on each SCC independently to remove cycle edges.

Experiments and Analyses

We apply our graph embedding framework ATP to three diverse tasks: link prediction, and question difficulty estimation and expert finding in CQAs.

Link Prediction

In link prediction, we would like to predict these missing edges given a network with a certain fraction of edges removed. The labeled dataset of edges (or node pairs) consists of positive and negative examples. Given a random edge e, if the removal of this edge will not disconnect the residual network, e will be selected as a positive example. We select $r = 10\%$ edges as positive examples. To generate negative examples, we randomly select an equal number of node pairs from the network which have no edges connecting them [5]. Hence $2r$ edges and node pairs are selected for evaluation.

Datasets used for evaluation are Wiki-Vote[6], GNU[7], Cit-HepPH[8], which were used in prior work (Lai et al. 2017).

[5]Each node pair (u, v) in negative samples satisfies the condition that v can reach u, but u cannot reach v in the network.

[6]https://snap.stanford.edu/data/wiki-Vote.html

[7]https://snap.stanford.edu/data/p2p-Gnutella31.html

[8]https://snap.stanford.edu/data/cit-HepPh.html

Table 1: Comparisons between ATP and the state-of-the-art methods on link prediction, evaluated by AUC

	AUC	Wiki-Vote	Cit-HepPH	GNU
LINE	2-nd order	0.4423	0.3310	0.4748
HOPE	AA	0.7672	0.7385	0.5565
	CN	0.7860	0.7570	0.5736
	AI	0.7784	0.7440	0.6159
SVDM	Harmonic	0.8200	0.7522	0.8166
	log	0.8215	0.7929	0.8162
ATP	Constant	0.9123	0.7939	0.8684
	Linear	0.9462	0.8682	0.8893
	log	**0.9481**	**0.8916**	**0.9314**
	Harmonic	0.9478	0.8892	0.9288

Following existing literature (Grover and Leskovec 2016; Yang et al. 2017; Tran 2018), we use Area Under Curve (AUC) to evaluate the link prediction performance. We compare our method with the most recent work for asymmetric proximity preserving.

- **ATP and its variants**: ATP-Constant ($\boldsymbol{M}$ = $\boldsymbol{A}$), ATP-Linear ($\boldsymbol{M}$ = $\boldsymbol{L}$), ATP-Harmonic, and ATP-log (By default ATP refers to ATP-log). ATP-Harmonic, and ATP-log transforms $\boldsymbol{L}$ to $\boldsymbol{M}$ by harmonic and log function respectively.
- **HOPE**(Ou et al. 2016): As the time complexity of computation of RPR is too high, we only report performances of HOPE-AA, HOPE-CN, and HOPE-KI here.
- **SVDM**: SVD-Harmonic and SVD-log use the same way to build $\boldsymbol{M}$ as ATP-Harmonic and ATP-log respectively. However, unlike ATP, SVD-Harmonic and SVD-log performs Singular Value Decomposition (SVD) as used in HOPE on $\boldsymbol{M}$ and selects the largest k singular values and corresponding singular vectors to construct the embedding.
- **LINE** (Tang et al. 2015): It is worth mentioning that LINE can only preserve symmetric second-order proximity when applied to a directed graph. In our experimental settings, vertex vectors are considered as source vectors, and context vectors are used as target vectors.

Performance Analysis We can conclude from the performance as shown in Table 1 that:

- Since Harmonic numbers can be approximated by log functions, log and Harmonic transformation can achieve similar performance in both ATP and SVDM. By default, we use log function as our non-linear transformation. It is noticeable that ATP performs better than SVDM. In average, ATP improves over SVDM by 22.01% among all datasets (from 12.45% to 40.67%), which shows the advantage of leveraging NMF to perform transitivity preserving graph embedding.
- ATP and SVDM perform better than HOPE. In average, SVDM improves over HOPE-KI by 13.72% among all datasets, and ATP improves over HOPE-KI from 19.84% to 51.97%. The only difference between SVDM and

Table 2: Statistics of Stack Exchange Sites

Sites	Apple	Gaming	Physics	Scifi	Unix
# nodes	133K	117K	127K	59K	167K
# edges	161K	190K	188K	97K	249K

HOPE is the technique used to build $\boldsymbol{M}$. The results demonstrate the advantage of incorporating graph reachability and hierarchy to construct $\boldsymbol{M}$, in comparison to building a higher order proximity matrix based on AA, CN, and KI.

- Both ATP-Harmonic and ATP-log perform better than ATP-Linear and ATP-Constant as expected, and ATP-Linear performs better than ATP-Constant. For example, ATP-log improves over ATP-Constant by 8.51% in average among all datasets. ATP-log improves over ATP-Linear by 4.74% on the largest dataset GNU. It shows the efficacy of applying non-linear transformation to $\boldsymbol{M}$.

Question Difficulty Estimation and Expert Finding in CQAs

In this section, we start by discussing how to apply ATP to estimate question difficulty and user expertise. We then show how to embed newly posted questions (unseen nodes in the training) inductively and identify best answerers for newly posted questions in CQAs.

Question Difficulty and User Expertise Estimation in CQAs We first talk about how to apply our graph embedding technique into question difficulty and user expertise estimation, which is a central part of automated question routing in CQAs. We select 5 large and popular sites from Stack Exchange [9] for evaluation. More details about the Stack Exchange sites can be found in the Table 2. For each Stack Exchange site, ATP uses the same competition graph as the input as QDEE (Sun et al. 2018a) which assigns solely scalar value to represent question difficulty level and user expertise. In this section, we will show the advantages of learning latent representations for question difficulty (question nodes) and user expertise (user nodes) by leveraging ATP.

Following the same setting as QDEE (Sun et al. 2018a), questions which were provided non-zero bounty scores are selected as our ground truth for evaluation, and pairwise accuracy (*Acc*)[10], as used by previous studies (Wang, Jing Liu, and Guo 2014; Sun et al. 2018a), are used to measure the effectiveness of different estimation techniques. Higher accuracy indicates better performance of the technique. We evaluate ATP and other state-of-the-art methods such as **TrueSkill** (Wang, Jing Liu, and Guo 2014), **Number-Of-Answers** (Yang et al. 2014), **Time-First-Answer**, **Time-Best-Answer** (Huna, Srba, and Bielikova 2016) and **QDEE** (Sun et al. 2018a) on the task of question difficulty estimation.

Question difficulty estimation performance is shown in Figure 3. We can conclude that ATP performs the best on almost all of the Stack Exchange sites. For example, ATP improves over TrueSkill, Number-of-Answer, and QDEE on average by 6.56%, 5.92%, and 5.06% respectively in 5 Stack Exchange sites. TrueSkill suffers from the data sparsity problem. Each question has only one in-edge (from questioner) and one out-edge (to the best answerer), which limits the accuracy of TrueSkill. ATP can leverage graph reachability and hence each question can have interactions with other questions and users, which can overcome the data sparseness.

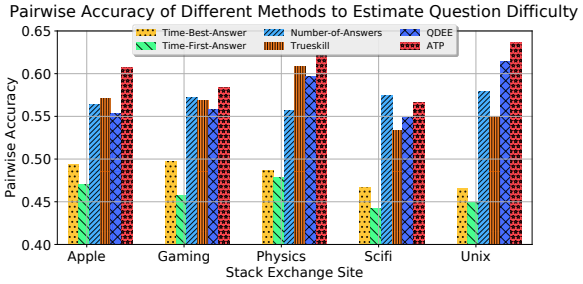

Figure 3: Pairwise accuracy of different approaches to estimate question difficulty

Inductive Embedding Learning for Cold Question Routing in CQAs Usually, there are two types of questions in CQAs – resolved (questions with answers) and newly posted questions (questions that have not received any answers). We refer to these newly posted questions as *cold questions*. The majority of approaches have focused on evaluating content quality after the fact (after questions have been resolved) (Yang et al. 2013). Yet, as the CQAs continue to grow, routing the cold questions to matching experts before answers have been provided has become a critical problem. For example, in Stack Overflow, about 4.8 million questions have not been answered [11].

In this section, we show that ATP can generate quality embedding for new vertices (cold questions) unseen during training, therefore supporting inductive learning in nature. Our idea is to leverage Expertise Gain Assumption (*EGA*) (Sun et al. 2018a) to bridge the gap between cold-start and well-resolved questions asked by the same user. Given a cold question q^* asked by a user u^*, the most recent k questions asked by the same user u^* are $q_1, q_2, ..., q_k$ and their associated embedding are available to us (they can be seen during training). We use the embedding of the question which has the highest difficulty level among $q_1, q_2, ..., q_k$ to approximate q^*'s embedding. A question q_{max} is considered to have the highest difficulty level if $\sigma(\langle \vec{s}_{q_i}, \vec{t}_{q_{max}} \rangle) > \sigma(\langle \vec{s}_{q_{max}}, \vec{t}_{q_i} \rangle)$ for all $i \in [1, k]$ and $q_i \neq q_{max}$. Then $\vec{s}_{q^*} = \vec{s}_{q_{max}}$ and $\vec{t}_{q^*} = \vec{t}_{q_{max}}$. We note that it is possible that the user posing the question is a new user (or one that has not posted a sufficient number of questions). In this case, k well-resolved questions that are closest (i.e. cosine

[9] We used the data dump which is released on June 12, 2017 and is available online at https://archive.org/details/stackexchange

[10] Acc = $\frac{\text{\# correctly predicted question pairs}}{\text{\# all question pairs}}$

[11] https://stackoverflow.com/unanswered

similarity) to q^* in textual descriptions [12], are picked as its nearest neighbors. The source and target embedding of q^* is predicted as the averaged source and target embedding of its nearest neighbors respectively.

Our task of cold question routing is to select the user who has the highest possibility to be selected as the best answerer for a newly posted question. Given the testing question set $\mathcal{Q}_t$, the predicted ranking list of all potential answerers for a test question q^* is R^{q^*} for all $q^* \in \mathcal{Q}_t$. The ranking score of a potential answerer u for the cold question q^* is computed as $\sigma(\langle \vec{s}_{q^*}, \vec{t}_u \rangle)$. The answerer who has the highest ranking score will be selected as the best answerer for q^*.

We compare ATP with state-of-the-art methods (**BoW** (Figueroa and Neumann 2013), **Doc2Vec** (Dong et al. 2015), **LDA** (Ji et al. 2012), **CQARank** (Yang et al. 2013)), **QDEE** (Sun et al. 2018a), and **ColdRoute** (Sun et al. 2018b), based on several popular evaluation criteria such as Mean Reciprocal Rank (**MRR**) (Zhu et al. 2014), **Precision**@3 (Zhao et al. 2017; Sun et al. 2018b), and **Accuracy** (Zhao et al. 2017; Sun et al. 2018b). We followed the same settings proposed by Jiankai et al. (Sun et al. 2018b) to select cold questions for evaluation.

Table 3 shows the performance of different approaches on the task of cold question routing, evaluated by MRR, Precision@3 and Accuracy. Jiankai et al. (Sun et al. 2018b) reported that ColdRoute performed consistently better than BoW, Doc2Vec and LDA. To save space, we omitted their performance Table 3. Based on the results, we can make the following observations:

- ATP performs the best overall evaluation metrics in almost all Stack Exchange sites. For example, ATP improves upon routing metric Accuracy over ColdRoute by 6.14%, since ColdRoute fails to take the interactions between questions (asked by the same asker) into consideration. ATP improves upon routing metric MRR over QDEE by 7.59%, which indicates that incorporating graph reachability and representing user expertise and question difficulty as a feature vector can help ATP identify matching experts for cold questions more accurately and robustly than the state-of-the-art methods.
- ATP performs better than CQARank. The reason is that CQARank's Q&A graph contain more noise than the competition graph used by ATP. The direction of edges in CQARank's Q&A graph is from the asker to the answerer. The underlying assumption is that askers have lower expertise than corresponding answerers. However, Wang et al. (Wang, Jing Liu, and Guo 2014) shows that the expertise of the asker is not assumed to be lower than the expertise score of a non-best answerer, since such a user may just happen to see the question and responded that, rather than knowing the answer well. These kinds of answers do not show corresponding answerers' expertise are higher than the asker's expertise. The generated noise edges in CQARank's Q&A graph can undermine CQARank's performance on experts finding for cold questions.

[12]Each question can be represented as a feature vector by LDA (Ji et al. 2012)

Table 3: Comparisons between ATP and the state of the art methods on cold question routing in CQAs, evaluated by MRR, Precision@3 (P@3), and Accuracy.

		Apple	Gaming	Physics	Scifi
MRR	CQARank	0.4914	0.4463	0.5315	0.4628
	QDEE	0.5579	0.6011	0.524	0.5895
	ColdRoute	0.5365	**0.6445**	0.5288	**0.6462**
	ATP	**0.574**	0.6242	**0.5814**	0.6405
P@3	CQARank	0.5855	0.5144	0.699	0.552
	QDEE	0.7094	0.8019	0.6888	0.7455
	ColdRoute	0.6581	0.7796	0.7194	0.7741
	ATP	**0.7564**	**0.8179**	**0.7398**	**0.8064**
Acc.	CQARank	0.5555	0.4979	0.6483	0.5693
	QDEE	0.6852	0.737	0.6401	0.711
	ColdRoute	0.6324	0.7387	0.6354	0.7369
	ATP	**0.7041**	**0.7504**	**0.6895**	**0.7695**

Conclusion

In this paper, we have proposed a novel asymmetric transitivity preserving directed graph embedding framework (ATP). Our scalable embedding technique incorporates both graph hierarchy and reachability information by constructing a novel asymmetric matrix, which is a non-linear transformation of an adjacency matrix (graph reachability) and a diagonal matrix (graph hierarchy). An efficient factorization based approach is used to generate two embedding vectors for each node to capture the asymmetric transitivity.

With incorporating graph hierarchy and reachability, ATP can perform better than the state-of-the-art in various tasks such as link prediction, and question difficulty estimation and cold question routing in CQAs. And we have proposed several approaches to combine both graph hierarchy and reachability inside a unified framework, and empirically the non-linear transformation works the best.

As extension of current study, we plan to apply our model to other applications such as community detection in dynamic networks (Wang et al. 2018) and exception-tolerant abduction (Zhang, Mathew, and Juba 2017) in attributed networks (Liang et al. 2018). We also would like to address the problem of routing newly posted questions (item cold-start) to newly registered users (user cold-start) in CQAs, with hoping to increase the expertise of the entire community.

Acknowledgments This work is supported by NSF grants EAR-1520870, CCF-1645599, CCF-1629548, IIS-1550302, and CNS-1513120, and a grant from the Ohio Supercomputer Center (PAS0166). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their sponsors.

References

Cai, H.; Zheng, V. W.; and Chang, K. C. 2017. A comprehensive survey of graph embedding: Problems, techniques and applications. *arXiv* abs/1709.07604.

Cheng, P.; Wang, S.; Ma, J.; Sun, J.; and Xiong, H. 2017. Learning to recommend accurate and diverse items. In *WWW*.

Dong, H.; Wang, J.; Lin, H.; Xu, B.; and Yang, Z. 2015. Predicting best answerers for new questions: An approach leveraging distributed representations of words in community question answering. In *FCST*.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. Metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*.

Fang, H.; Wu, F.; Zhao, Z.; Duan, X.; Zhuang, Y.; and Ester, M. 2016. Community-based question answering via heterogeneous social network learning. In *AAAI*.

Figueroa, A., and Neumann, G. 2013. Learning to rank effective paraphrases from query logs for community question answering. In *AAAI*.

Gao, M.; Chen, L.; He, X.; and Zhou, A. 2018. Bine: Bipartite network embedding. In *SIGIR*.

Goyal, P., and Ferrara, E. 2017. Graph embedding techniques, applications, and performance: A survey. *arXiv* abs/1705.02801.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *KDD*.

Gupte, M.; Shankar, P.; Li, J.; Muthukrishnan, S.; and Iftode, L. 2011. Finding hierarchy in directed online social networks. In *WWW*.

Herbrich, R.; Minka, T.; and Graepel, T. 2007. Trueskill™: A bayesian skill rating system. In *NIPS*.

Huna, A.; Srba, I.; and Bielikova, M. 2016. Exploiting content quality and question difficulty in CQA reputation systems. In *NetSci-X*.

Ji, Z.; Xu, F.; Wang, B.; and He, B. 2012. Question-answer topic model for question retrieval in community question answering. In *CIKM*.

Lai, Y.-A.; Hsu, C.; Chen, W.; Yeh, M.; and Lin, S. 2017. Prune: Preserving proximity and global ranking for network embedding. In *NIPS*.

Liang, J.; Jacobs, P.; Sun, J.; and Parthasarathy, S. 2018. Semi-supervised embedding in attributed networks with outliers. In *SIAM SDM*.

Ma, Y.; Ren, Z.; Jiang, Z.; Tang, J.; and Yin, D. 2018. Multi-dimensional network embedding with hierarchical structure. In *WSDM*.

Nisa, I.; Sukumaran-Rajam, A.; Kunchum, R.; and Sadayappan, P. 2017. Parallel ccd++ on gpu for matrix factorization. In *GPGPU*.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*.

Pan, S.; Wu, J.; Zhu, X.; Zhang, C.; and Wang, Y. 2016. Tri-party deep network representation. In *IJCAI*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*.

Qiu, J.; Dong, Y.; Ma, H.; Li, J.; Wang, K.; and Tang, J. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*.

Simon, K. 1988. An improved algorithm for transitive closure on acyclic digraphs. In *Theoretical Computer Science*.

Sun, J.; Ajwani, D.; Nicholson, P. K.; Sala, A.; and Parthasarathy, S. 2017. Breaking cycles in noisy hierarchies. In *WebSci*.

Sun, J.; Moosavi, S.; Ramnath, R.; and Parthasarathy, S. 2018a. QDEE: Question Difficulty and Expertise Estimation in Community Question Answering Sites. In *ICWSM*.

Sun, J.; Vishnu, A.; Chakrabarti, A.; Siegel, C.; and Parthasarathy, S. 2018b. Coldroute: effective routing of cold questions in stack exchange sites. In *ECML PKDD*.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*.

Tatti, N. 2015. Hierarchies in directed networks. In *ICDM*.

Tran, P. V. 2018. Learning to make predictions on graphs with autoencoders. *arXiv* abs/1802.08352.

Wang, Y.; Bandyopadhyay, B.; Chakrabarti, A.; Sivakoff, D.; and Parthasarathy, S. 2018. Spread sampling for graphs: Theory and applications. In *MLG*.

Wang, Q.; Jing Liu, B. W.; and Guo, L. 2014. A regularized competition model for question difficulty estimation in community question answering services. In *EMNLP*.

Yang, L.; Qiu, M.; Gottipati, S.; Zhu, F.; Jiang, J.; Sun, H.; and Chen, Z. 2013. CQArank: Jointly model topics and expertise in community question answering. In *CIKM*.

Yang, J.; Ke, T.; Alessandro, B.; and Geert-Jan, H. 2014. Sparrows and owls: Characterisation of expert behaviour in stackoverflow. In *UMAP*.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network representation learning with rich text information. In *IJCAI*.

Yang, C.; Sun, M.; Liu, Z.; and Tu, C. 2017. Fast network embedding enhancement via high order proximity approximation. In *IJCAI*.

Zhang, M.; Mathew, T.; and Juba, B. 2017. An improved algorithm for learning to perform exception-tolerant abduction. In *AAAI*.

Zhao, Z.; Yang, Q.; Cai, D.; He, X.; and Zhuang, Y. 2016. Expert finding for community-based question answering via ranking metric network learning. In *IJCAI*.

Zhao, Z.; Lu, H.; Zheng, V. W.; Cai, D.; He, X.; and Zhuang, Y. 2017. Community-based question answering via asymmetric multi-faceted ranking network learning. In *AAAI*.

Zhou, C.; Liu, Y.; Liu, X.; Liu, Z.; and Gao, J. 2017. Scalable graph embedding for asymmetric proximity. In *AAAI*.

Zhu, H.; Chen, E.; Xiong, H.; Cao, H.; and Tian, J. 2014. Ranking user authority with relevant knowledge categories for expert finding. In *WWW*.