ATP: Directed Graph Embedding with Asymmetric Transitivity Preservation

Jiankai Sun

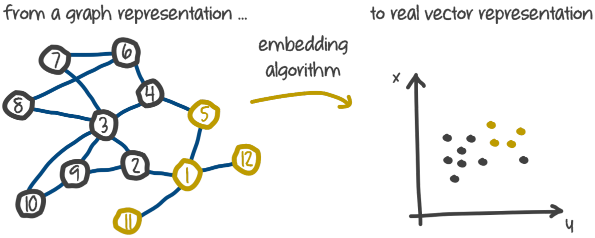
Bortik Bandyopadhyay Armin Bashizade Jiongqian Liang

P. Sadayappan Srinivasan Parthasarathy

The Ohio State University

Background

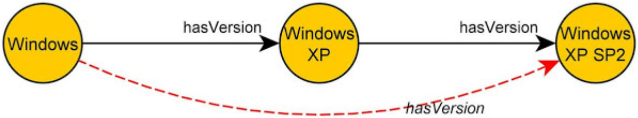
- Graph Embedding: represent vertices of a graph in a low-dimensional space while the structure of the graph can be reconstructed in the vector space
- Classic vector-based machine learning algorithms can leverage embedding results
- Applications: reconstruction, link prediction, vertex recommendation, and outlier detection [Liang et al. SDM'18]



Pic: `https://goo.gl/keJ9Z1`

Background

- Most of the existing graph embedding methods target on un-directed graph
- Recent studies design asymmetric metrics for directed graph embedding
- Transitivity plays a very important role in tasks of graph inference and analysis [Ou et al. KDD 2016]
- Transitivity is asymmetric in directed graphs: $u \to v$ and $v \to w \Rightarrow u \to w$, but not $w \to u$

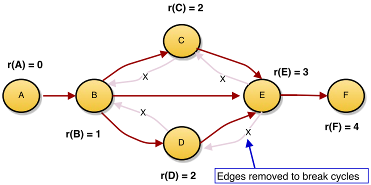


Pic: `https://goo.gl/9mvw1J`

HOPE: Asymmetric transitivity preserving graph embedding [Ou et al. KDD'16]

- Preserving asymmetric transitivity by approximating high-order proximity which are based on asymmetric transitivity
- high-order proximity measurements in graph can reflect the asymmetric transitivity
  - Katz Index
  - Rooted PageRank
  - Common Neighbors
  - Adamic-Adar
- Approximation of high-order proximity measurements: SVD

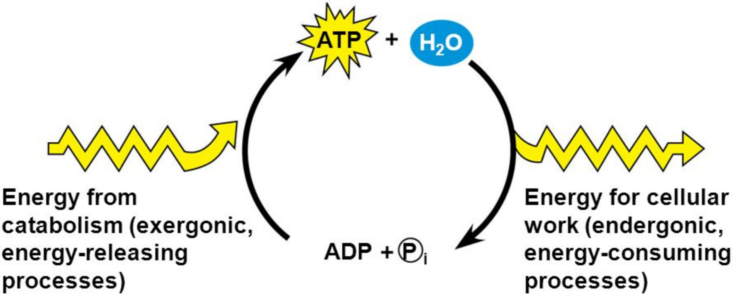
Drawbacks of HOPE



- $KI(B, E)$ is 0.0041, which is smaller than $KI(E, B) = 0.0067$
- $RPR(B, E) = 0.2129$, which is smaller than $RPR(E, B) = 0.2446$
- $AA(B, E) = AA(E, B) = 0.5$, and $CN(B, E) = CN(E, B) = 0$
- $ATP(B, E) = 1.48$ and $ATP(E, B) = 8.18e^{-10}$

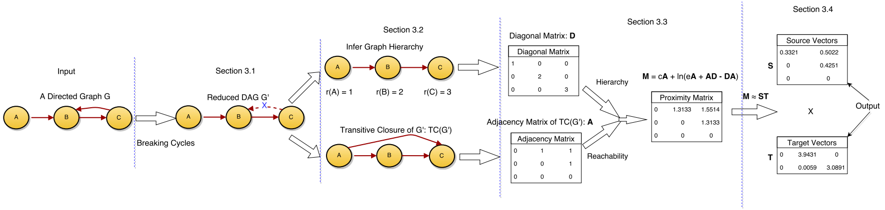
Goal of ATP

leveraging graph hierarchy and reachability to embed graph with the goal of preserving asymmetric transitivity



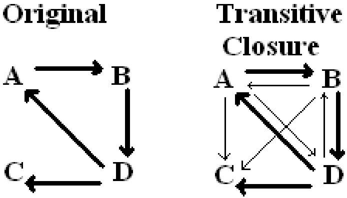
Pic: `https://goo.gl/me4Bqu`

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



Graph Hierarchy and Reachability

- Graph hierarchy has the characteristic of asymmetric transitivity
- Transitive closure (TC) of a directed graph 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.
    - computing TC for large directed graphs with cycles is expensive, while computing TC of directed acyclic graphs (DAGs) is practical



Pic: `https://goo.gl/pzu7MW`

Breaking Cycles

- Breaking cycles in graph G (n nodes, m edges) to get a DAG G' by using H-voting proposed by [Sun. et al. WebSci'17]
- Inferring graph hierarchy easily: assign ranking scores to each node in G (G') based on the structure of G': if i can reach j in G', then $rank_i < rank_j$
- Computing TC of G' is practical

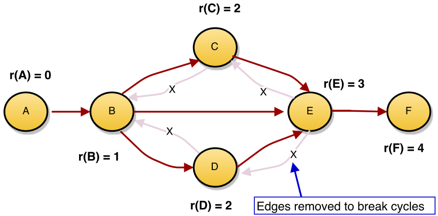
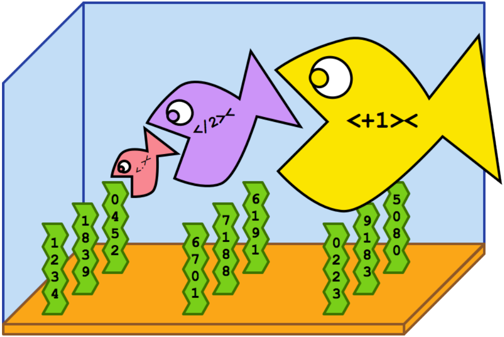


Figure: Break cycles and Infer graph hierarchy

Challenge I:

How to incorporate graph hierarchy and reachability with the goal of preserving asymmetric transitivity?



Pic: `https://goo.gl/xFEVgs`

Adjacency Matrix A

$$A_{i,j} = \begin{cases} 1, & \text{if i can reach j in DAG G'} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

- A is the adjacency matrix representation of the transitive closure of the reduced DAG G'
- All hierarchical differences are the same, which is $\mathbf{1}$

Incorporating graph hierarchy

- Suppose $\boldsymbol{D} \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix, where each non-zero element in the diagonal is $D_{i,i} = r(i)$.
- Incorporate graph hierarchy: $\boldsymbol{L} = \boldsymbol{AD} - \boldsymbol{DA}$

$$L_{i,j} = \begin{cases} r(j) - r(i), & \text{if i can reach j in DAG G'} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

- Limitation: Large hierarchical differences will be favored and the small hierarchical differences will be ignored

Build hierarchical proximity matrix M

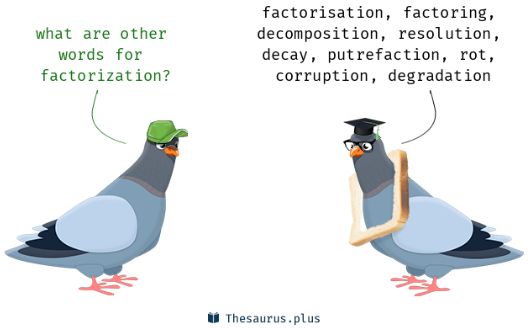
- Non-linear transformation: Harmonic numbers and log

$$M_{i,j} = \begin{cases} 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{r(j) - r(i)}, & \text{if i can reach j in DAG G'} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

- the gap of hierarchical rankings between local nodes will be large enough to be noticed and preserved in comparison with the linear model
- a harmonic number $h(\Delta_{i,j}) = \sum_{k=1}^{\Delta_{i,j}} \frac{1}{k}$ can be approximated by $(\gamma + log(\Delta_{i,j}))$
- $\boldsymbol{M} = c\boldsymbol{A} + log(e\boldsymbol{A} + \boldsymbol{L}) = c\boldsymbol{A} + log(e\boldsymbol{A} + \boldsymbol{AD} - \boldsymbol{DA})$

Challenge II:

How to generate asymmetric transitivity preserving graph embedding from M?



Pic: `https://goo.gl/Km56A7`

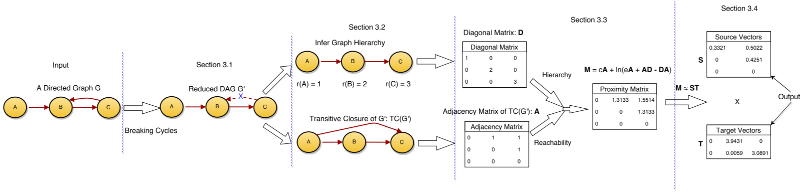
Non-negative Matrix Factorization

- Apply cumf_ccd[1] to do matrix factorization for large matrix M: $M \approx ST$, where $S \in \mathcal{R}^{n \times k}$ and $T \in \mathcal{R}^{k \times n}$
- Each row in S represents a node's latent feature representation of playing the role as a source node
- Each column in T represents a node's latent feature representation of being a target node.

[1] Israt Nisa et al. "Parallel CCD++ on GPU for Matrix Factorization". In: *Proceedings of the General Purpose GPUs*. GPGPU-10. 2017, pp. 73–83.

Time Complexity Analysis

- Breaking cycles: $O(|E|^2)$
- Inferring graph hierarchy: $O(|E| + |V|)$
- Constructing $\boldsymbol{M}$: $O(|V|^2 loglog(|V|))$
- Factorization of $\boldsymbol{M}$: $O(|V|^2 k)$ per iteration (NMF)
- In the worst case: $O(|E|^2)$ (G is a directed complete graph)

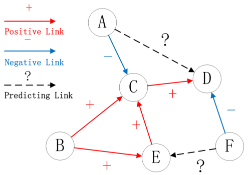


Applications of ATP

- Link prediction
- Question difficulty estimation in community question answering services (CQAs) such as Stack Overflow
- Experts finding (newly posted question routing) in CQAs

Link Prediction

- Given a network with a certain fraction of edges are removed, we would like to predict these missing edges



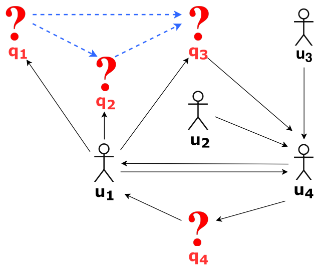
Pic from Liu et al. Entropy 2015: `https://goo.gl/MqH5bE`

- How we select Test Edges for evaluation:
  - Positive Examples: given an edge e in G, removal of e will not disconnect G
  - Negative Example: each node pair (u, v) satisfies the condition that v can reach u, but u cannot reach v in the network

Link Prediction Performance: ATP vs State-of-the-art

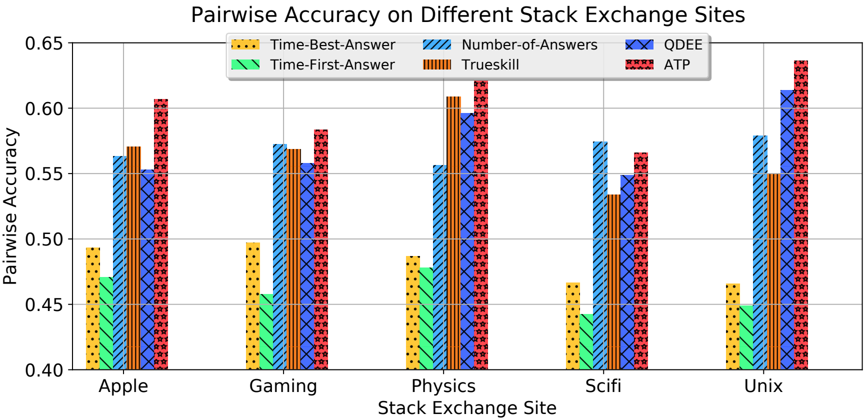
	AUC	Wiki-Vote	Cit-HepPH	GNU	GNM-5K	GNM-30K
LINE	2-nd order	0.4423	0.3310	0.4748	0.4666	0.4109
HOPE	AA	0.7672	0.7385	0.5565	0.5698	0.5747
	CN	0.7860	0.7570	0.5736	0.5640	0.5836
	AI	0.7784	0.7440	0.6159	0.5455	0.5784
SVDM	Harmonic	0.8200	0.7522	0.8166	0.6321	0.6255
	log	0.8215	0.7929	0.8162	0.6325	0.6248
ATP	Constant	0.9123	0.7939	0.8684	0.755	0.7845
	Linear	0.9462	0.8682	0.8893	0.7968	0.8530
	log	**0.9481**	**0.8916**	**0.9314**	**0.8084**	**0.8789**
	Harmonic	0.9478	0.8892	0.9288	0.8077	0.8777

Question Difficulty Estimation and Experts Finding in CQAs



- Datasets: 5 different Stack Exchange sites
- Experimental settings of question difficulty estimation are same as QDEE [Sun et al. ICWSM 2018]
- Experimental settings of cold question routing are same as ColdRoute [Sun et al. ECML PKDD 2018]

Pairwise Accuracy of Question Difficulty Estimation



Cold Question Routing: ATP vs State-of-the-art

		Apple	Gaming	Physics	Scifi	Unix
MRR	CQARank	0.4914	0.4463	0.5315	0.4628	0.5258
	QDEE	0.5579	0.6011	0.524	0.5895	0.5158
	ColdRoute	0.5365	**0.6445**	0.5288	**0.6462**	0.54338
	ATP	**0.574**	0.6242	**0.5814**	0.6405	**0.5756**
P@3	CQARank	0.5855	0.5144	0.699	0.552	0.67
	QDEE	0.7094	0.8019	0.6888	0.7455	0.6566
	ColdRoute	0.6581	0.7796	0.7194	0.7741	0.6869
	ATP	**0.7564**	**0.8179**	**0.7398**	**0.8064**	**0.7205**
Acc.	CQARank	0.5555	0.4979	0.6483	0.5693	0.6134
	QDEE	0.6852	0.737	0.6401	0.711	0.6218
	ColdRoute	0.6324	0.7387	0.6354	0.7369	0.6404
	ATP	**0.7041**	**0.7504**	**0.6895**	**0.7695**	**0.6713**

Conclusion

- Propose to break cycles to make it possible to compute the transitive closure practically
- Incorporate graph hierarchy and reachability information by constructing a novel asymmetric matrix
- Generate two embedding vectors for each node to capture the asymmetric transitivity by factorizing the generated matrix
- Apply ATP to three tasks: link prediction, and question difficulty estimation and expert finding in CQAs
- Support inductive embedding learning for routing newly posted questions (unseen nodes during training), which tackles a fundamental challenge in crowdsourcing

Acknowledgments

- **Acknowledgments:** This work is supported by NSF grants 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.

Q & A