TGNet: Learning to Rank Nodes in Temporal Graphs

Qi Song[1] Bo Zong[2] Yinghui Wu[1,3]
Lu-An Tang[2] Hui Zhang[2] Guofei Jiang[2] Haifeng Chen[2]

1 WASHINGTON STATE UNIVERSITY

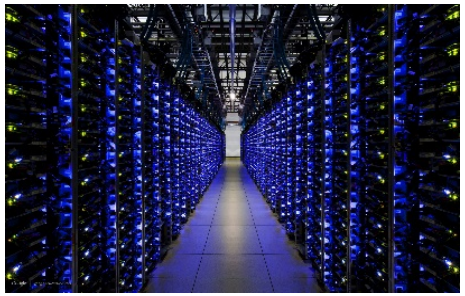
2 NEC Laboratories America
Relentless passion for innovation

3 Pacific Northwest NATIONAL LABORATORY

Node Ranking in temporal graphs

- Temporal graphs have been widely applied to model dynamic networks.

Automated systems

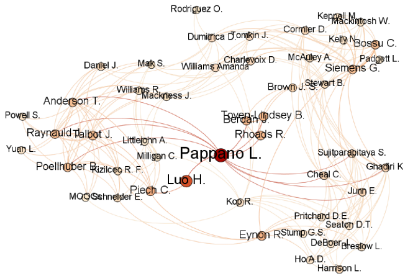


Anomaly detection

Computer networks

Attack detection

Citation networks

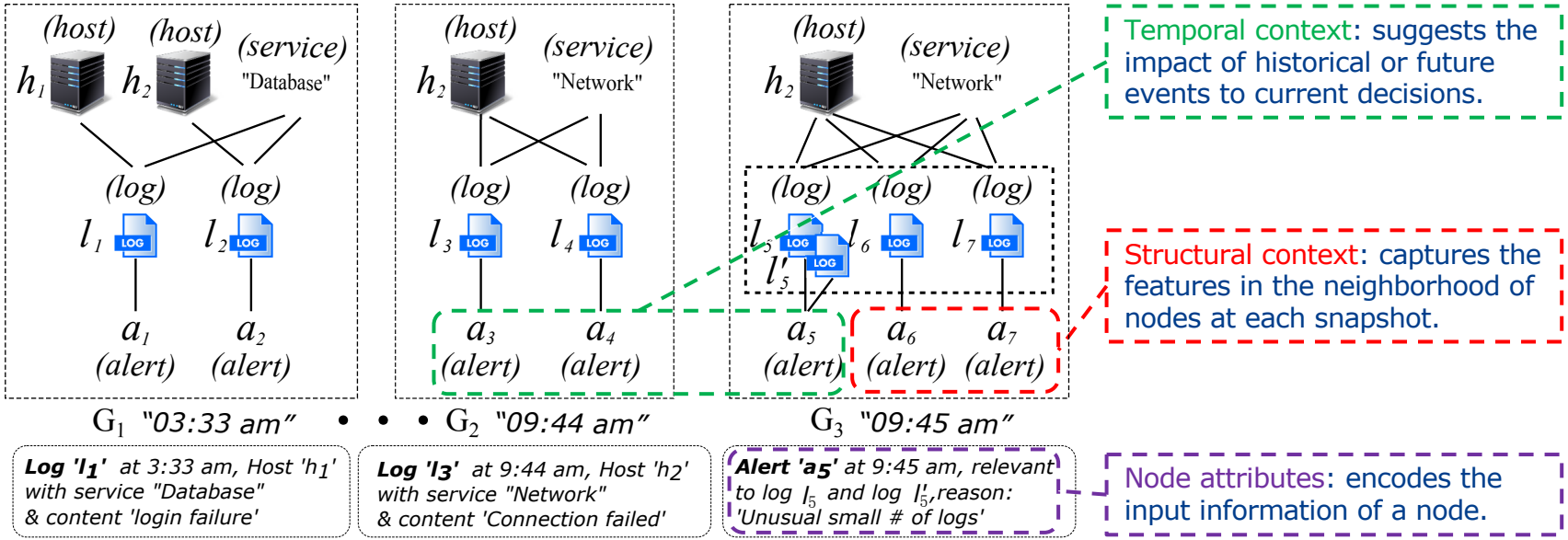


Author ranking

- Ranking models are desired to suggest and maintain the nodes with high priority in dynamic graphs.
- The node ranks can be influenced by rich context from heterogeneous node information, network structures, and temporal perspectives.

Alert ranking in a heterogeneous system alert network.

- A fraction of a real-world heterogeneous temporal information network from data center monitoring.

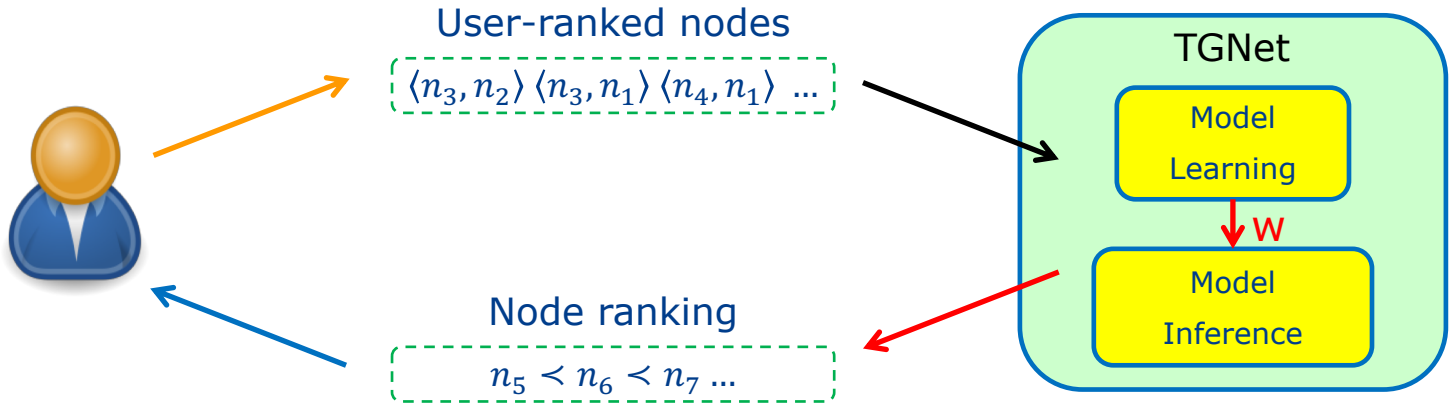


- Learning to rank nodes in temporal graphs is more challenging than its counterpart over static graphs:

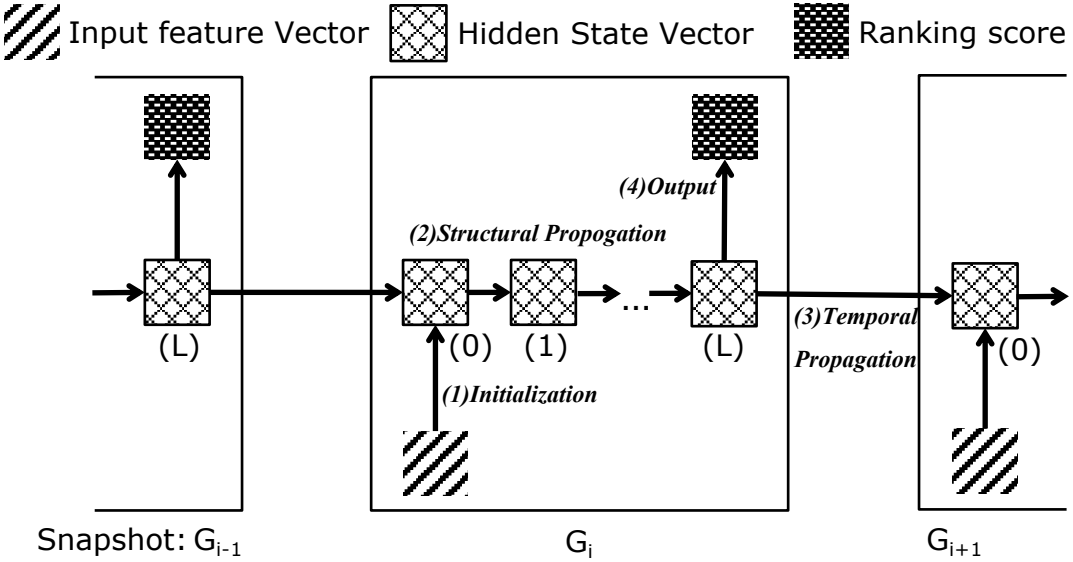
Context learning: structural and temporal context can be used to differentiate the roles that different nodes play in node ranking.	⟹	Automated approach to learn context features for node ranking.
Context dynamics: context features bear constant changes over time.	⟹	Effective models that can capture context evolution over time.
Label sparsity: labels provided by users are sparse and cover a small fraction of nodes	⟹	Effective learning that can be generalized from sparsely labeled data.
Time cost: fast learning and ranking upon the arrival of new nodes.	⟹	Effective optimization and provable performance guarantees.

Learning to rank nodes in temporal graphs

- Input: training data $\mathbb{G}$ consists of a temporal graph $\mathcal{G}$ and labeled data $\mathcal{D}$, a ranking function space $\mathbb{M}$, and an error function $j(\cdot)$ that measures the ranking error.
- Output: an optimal function $g^* \in \mathbb{M}$ such that the ranking error $j(g^*, \mathbb{G})$ is minimized.
- Pipeline:



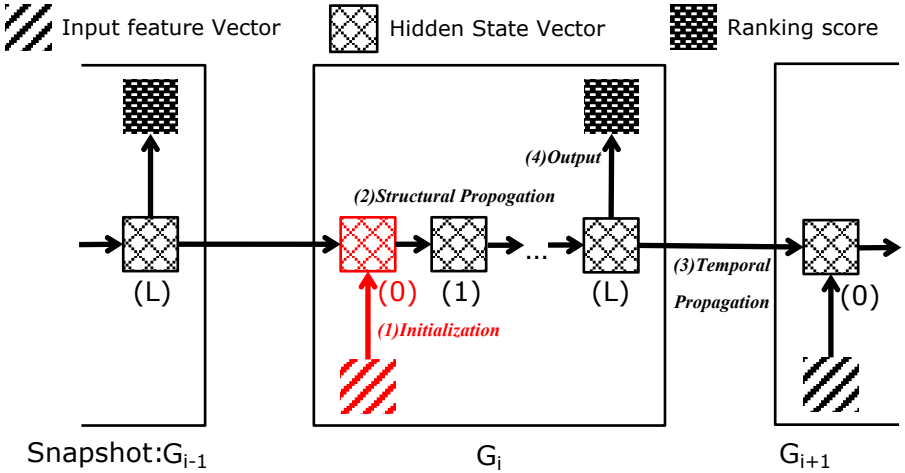
- Four components:
  - Initialization layer: projects input feature vector into hidden state space;
  - Structural propagation layer: exchanges neighborhood information in a snapshot;
  - Temporal propagation layer: propagates temporal influence between snapshots;
  - Output layer: transforms hidden states to ranking scores.

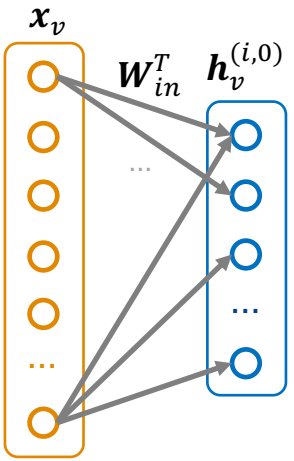


- When a node first occurs:

$$\boldsymbol{h}_v^{(i,0)} = \tanh(\boldsymbol{W}_{in}^T \cdot \boldsymbol{x}_v + \boldsymbol{b}_{in})$$

Hidden state vector Input feature vector Parameter



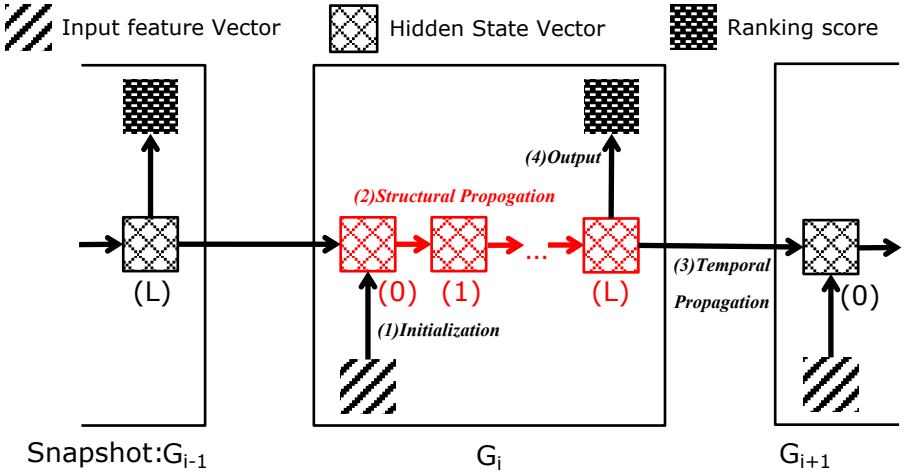


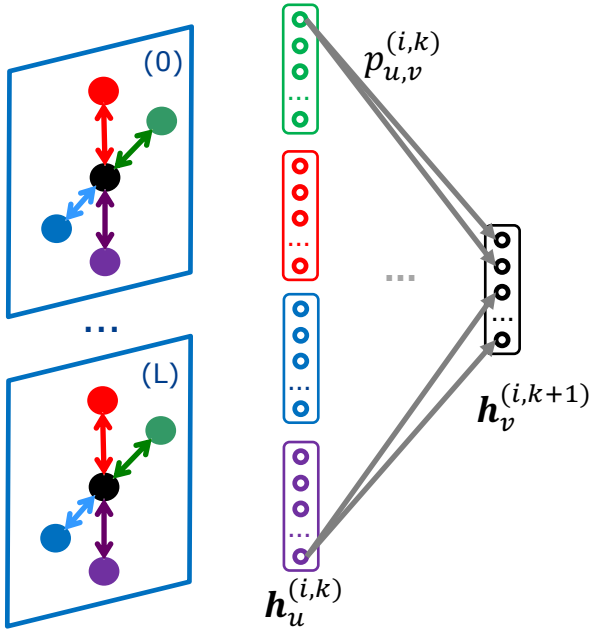
TGNet: Structural propagation layer

- For all nodes within a snapshot: iteratively performing local information propagation (for L times)

$$\boldsymbol{h}_v^{(i,k+1)} = \tanh\Big(\sum_{u \in N_v} p_{u,v}^{(i,k)} \boldsymbol{h}_u^{(i,k)}\Big)$$

Hidden state vector — Structural influence factor



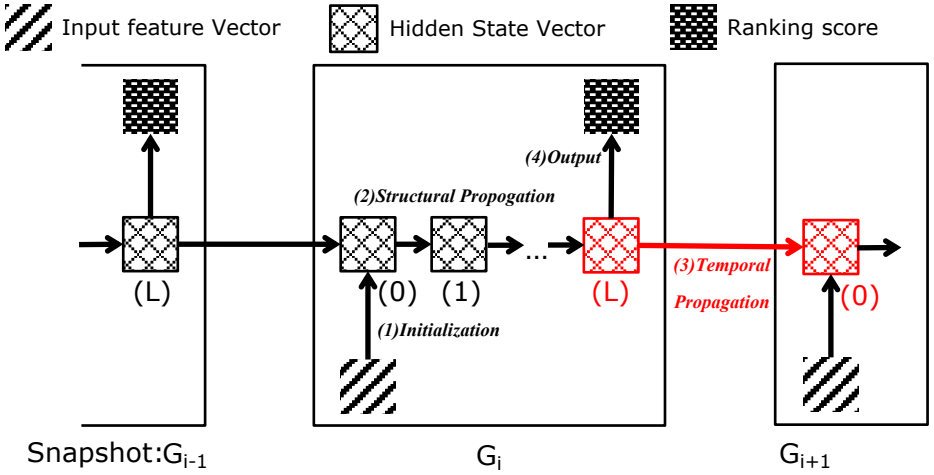


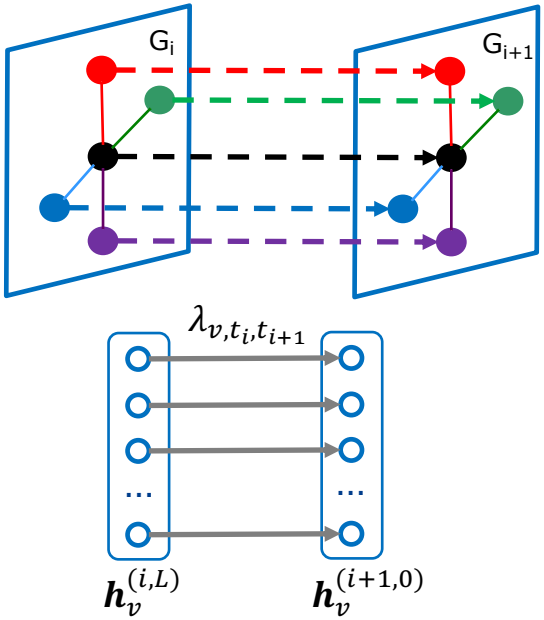
TGNet: Temporal propagation layer

- For nodes exist in two continuous snapshots:

$$\boldsymbol{h}_v^{(i+1,0)} = \lambda_{v,t_i,t_{i+1}} \boldsymbol{h}_v^{(i,L)}$$

Hidden state vector — Temporal influence factor



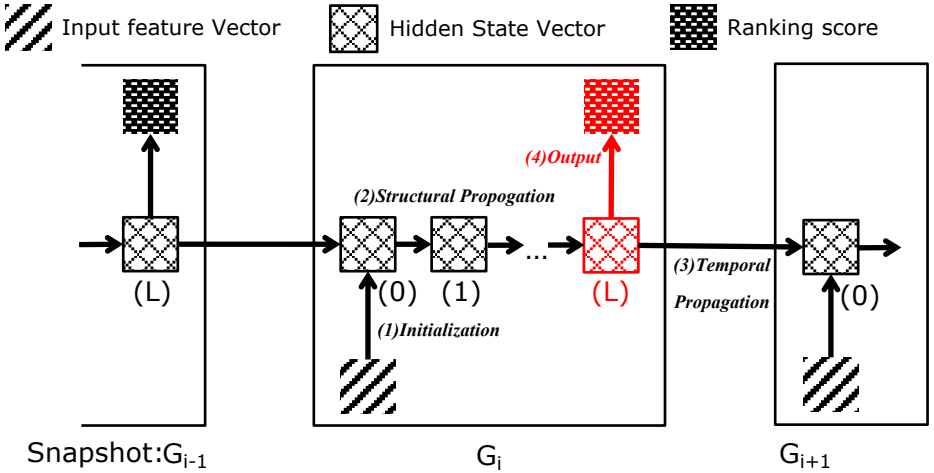


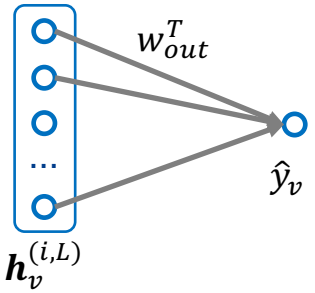
TGNet: Output layer

- For nodes in the last snapshot they appear:

$$\hat{y}_v = \sigma(w_{out}^T \cdot \boldsymbol{h}_v^{(i,L)} + b_{out})$$

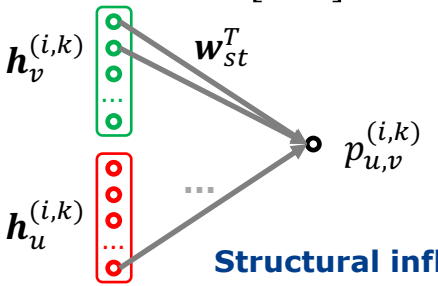
Ranking value — Hidden state vector — Parameter





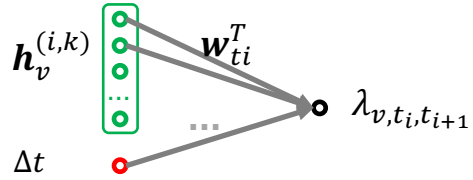
- Edge-centric: parameters are associated to edges;
  - Can not deal with context changes, daunting for users to choose edge types.
- Node-centric (influence network):
  - The influence between two nodes is conditioned by their contexts;
  - The node context is determined by its hidden state.

$$\boldsymbol{p}_v^{(i,k)} = softmax(\boldsymbol{w}_{st}^T \cdot \begin{bmatrix} \boldsymbol{h}_u^{(i,k)} \\ \boldsymbol{h}_v^{(i,k)} \end{bmatrix} + b_{si})$$



Structural influence

$$\lambda_{v,t_i,t_{i+1}} = \sigma(\boldsymbol{w}_{ti}^T \cdot \begin{bmatrix} \boldsymbol{h}_v^{(i,k)} \\ \Delta t \end{bmatrix} + b_{ti})$$



Temporal influence

- Inference cost: $O(|\mathcal{G}|)$
- Neighbor sampling: randomly sample a subset of the neighbors of each node.

- Objective function:

$$J(\Theta, \mathbb{G}) = \sum_{\langle u,v \rangle \in \mathcal{D}} E(\Theta, u, v) + \beta_1 R_1(\Theta) + \beta_2 R_2(\Theta, \mathcal{G})$$

Parameters

Loss function: measure the error made by Θ

$$E(\Theta, u, v) = 1 - (\hat{y}_v - \hat{y}_u)$$

$\mathcal{L}_2$ regularization: penalize model complexity

$$R_1(\Theta) = \|\theta\|^2$$

Graph smoothness: penalize high dissimilarity between a node and its neighbors

$$R_2(\Theta, \mathcal{G}) = \sum_{i=1}^{s} \sum_{\langle u,v \rangle \in E_i} \left\| \boldsymbol{h}_u^{(i,L)} - \boldsymbol{h}_v^{(i,L)} \right\|_2^2$$

- Gradient Descent
  - Iteratively compute error resulting from current Θ and update Θ by their gradients.
- Learning cost: $O(|\mathcal{G}||\mathcal{D}|)$

Experiment Setting

- Dataset
  - System log data (SLD)
  - ISCX Intrusion detection data (IDS)
  - Microsoft academic graph (MAG)
  - Amazon co-purchasing network (APC)

Dataset	#Snap-shots	# of nodes	# of edges	# of Training pairs
SLD	19.4K	61.4K	258.1K	20K
IDS	66.7K	5.7M	11.5M	100K
MAG	12K	2.5M	16.2M	100K
APC	1.5K	2.1M	9.5M	100K

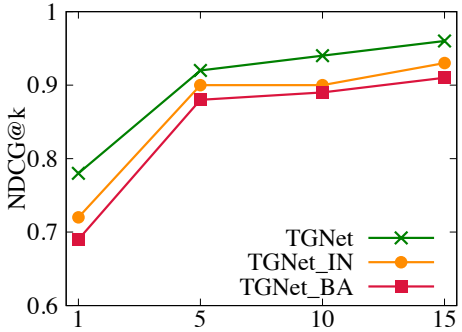
- Baseline:
  - TGNet_BA, TGNet_IN
  - Supervised PageRank (SPR)
  - Dynamic PageRank (DPR)
  - SVMRank
  - Alert fusion for intrusion detection (LRT)
  - Network anomaly detection (BGCM)
- Metric:
  - Top-*k* version of Normalized Discounted Cumulative Gain (NDCG@k)
  - *k* is determined by domain experts

Experiment Results

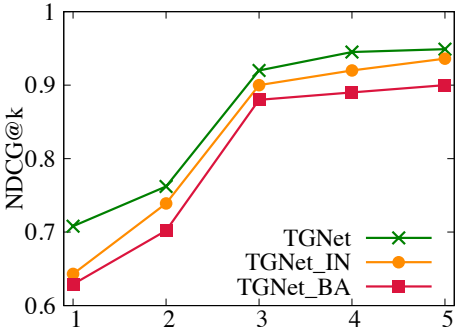
- Accuracy comparison

	BGCM	LRT	SVMRank	SPR	DPR	TGNet_BA	TGNet_IN	TGNet
SLD	0.35	0.72	0.74	0.78	0.56	0.88	0.90	**0.92**
IDS	0.32	0.59	0.70	0.72	0.67	0.77	0.85	**0.87**
MAG	-	-	0.56	0.69	0.61	0.79	0.86	**0.91**
APC	-	-	0.52	0.55	0.45	0.73	0.82	**0.88**

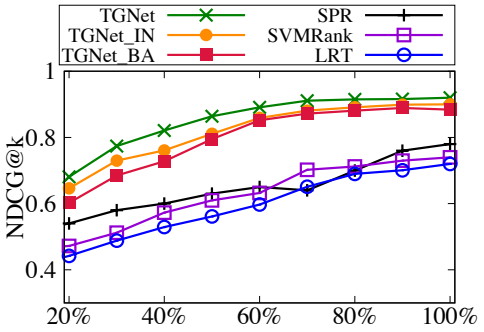
- Accuracy varying parameters



Varying d_h (number of hidden state dimension): TGNet achieves good accuracy with small d_h.



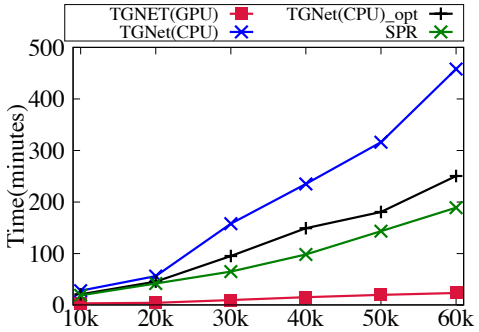
Varying L (# of structural propagation iteration): prediction performance tends to converge when L is small.



Varying labeled data size: TGNet needs the least amount of labeled data to achieve the highest accuracy.

Experiment Results

- Learning cost

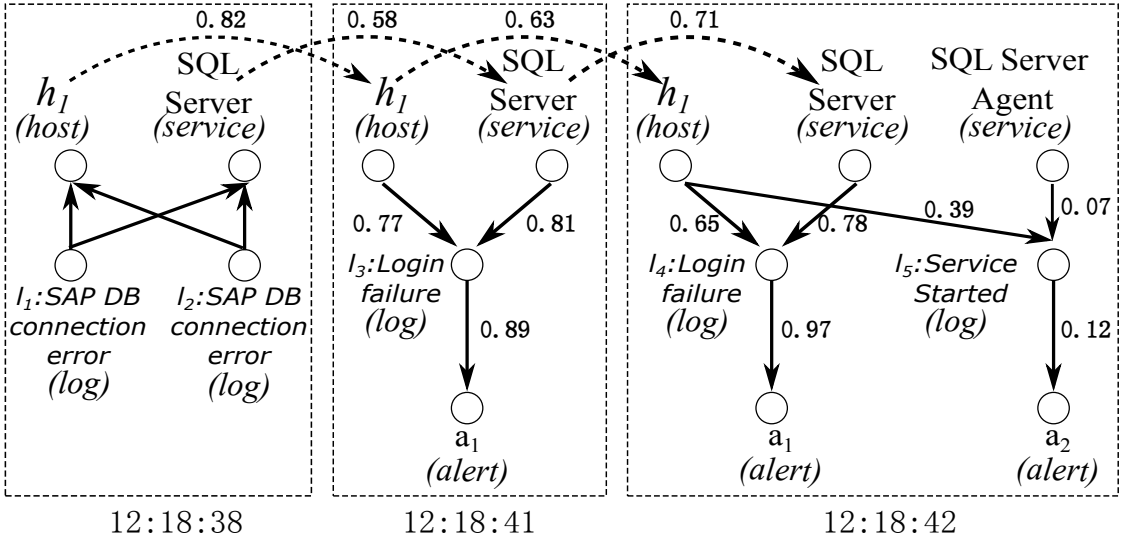


Varying graph size on synthetic SLD: training is 10 times faster than SPR

- Case study

1. Critical alert 'a_1' starts at 12:18:41, ends at 12:18:42, with reason 'Excessive number of login failure message', relevant to log "l_3,l_4".

2. Ignore alert 'a_2' starts at 12:18:42, ends at 12:18:42, with reason 'Number of service started message below threshold', relevant to log "l_5".



Conclusion

- TGNet: a novel graph neural network network
  - Explicitly tackles challenges in node ranking for temporal graphs;
  - Influence network: boost the capability of modeling context dynamics;
  - Graph smoothness: cope with label sparsity;
  - End-to-end learning algorithm with optimization techniques.

Sponsored by:

Thank you!