ZERO-SHOT GRAPH EMBEDDING

ZHENG WANG

International Young Scholars Forum@Renmin University of China

2021/03/29

Outline

- Problem introduction: Zero-shot Graph Embedding (ZGE)
- Our solutions
  - RSDNE [AAAI 2018]
  - RECT [TKDE 2020]
  - ExtendRECT [DASFAA 2021]
- Conclusion
- Q&A

Project page: https://zhengwang100.github.io/project/zero_shot_graph_embedding.html

Background: Graph Embedding

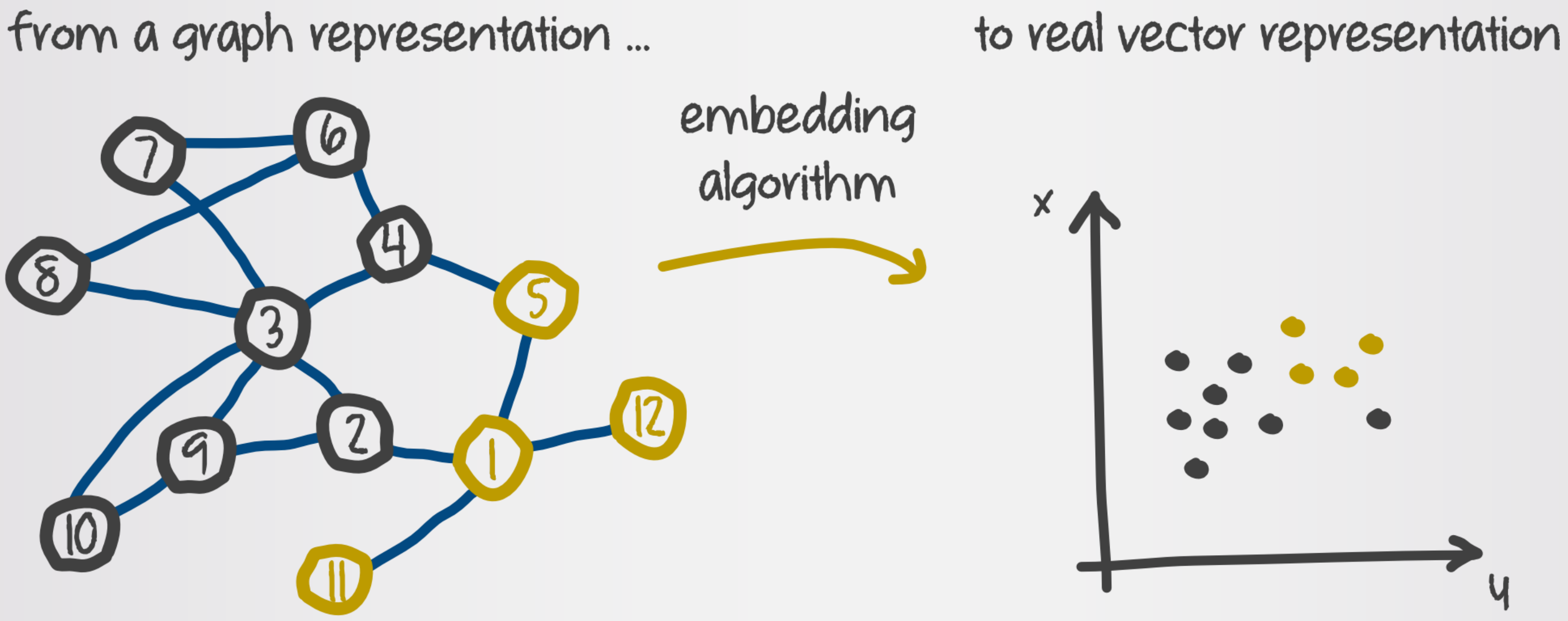


Figure: the aim is to learn low-dimensional latent representation of nodes in a network.

Background: Zero-shot Graph Embedding

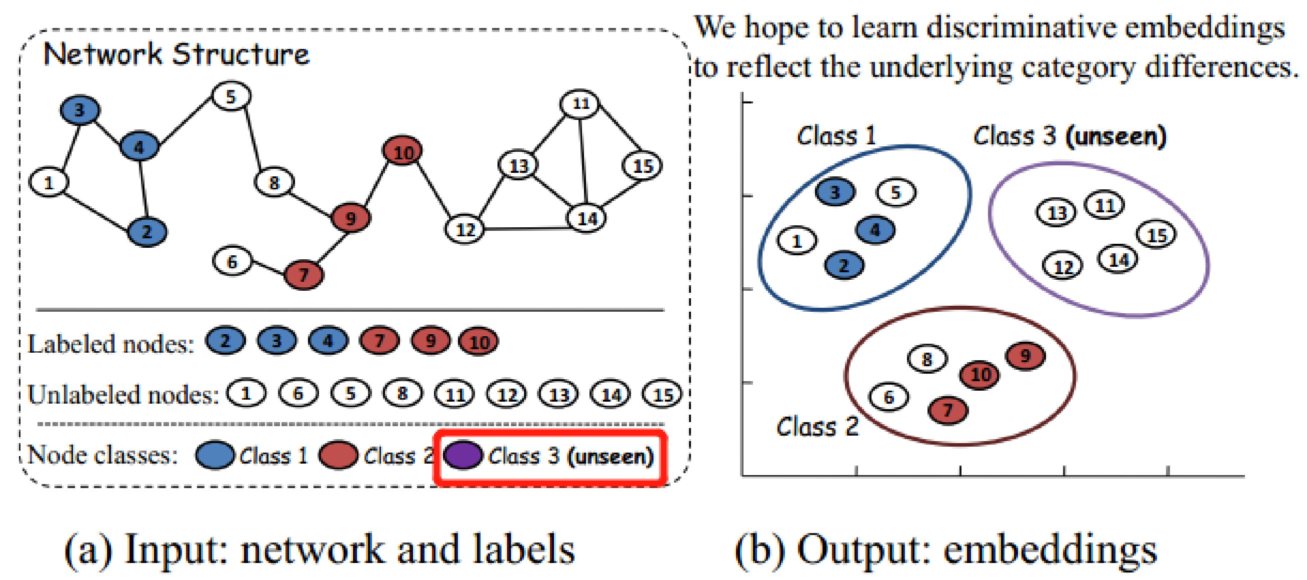


Figure: Illustration of zero-shot graph embed-ding. This graph actually contains three classes of nodes, but only two classes provide labeled nodes, i.e., blue and red nodes. The remaining nodes (including all the nodes of Class 3) are unlabeled.

Zero-shot graph embedding (**ZGE**) refers to the process of learning discriminative graph embeddings when labeled data cannot cover all classes (also known as completely-imbalanced label setting).

Why ZGE?

- Hard to collect labels for graph
  - Practical graphs are usually very large
  - Human annotations are costly
- Traditional semi-supervised methods would fail

Social networks

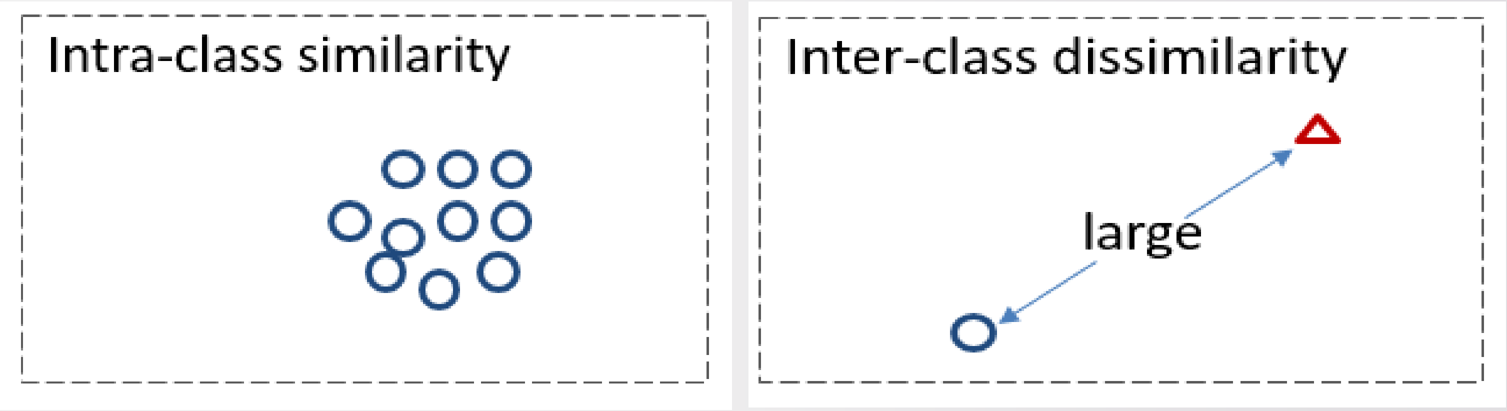
		Accuracy			Relative Accuracy Decline		
Method	*Label*	10%	30%	50%	10%	30%	50%
LSHM	LSHM(b)	0.5007	0.6178	0.6711	-	-	-
	LSHM(-1)	0.4258	0.5887	0.6455	0.1496↓	0.0471↓	0.0382↓
	LSHM(-2)	0.4253	0.5504	0.6027	0.1506↓	0.1091↓	0.1019↓
GCN	GCN(b)	0.7198	0.7473	0.7628	-	-	-
	GCN(-1)	0.6572	0.6937	0.7064	0.0870↓	0.0717↓	0.0739↓
	GCN(-2)	0.4761	0.5085	0.5159	0.3386↓	0.3196↓	0.3237↓

TABLE 1. Classification performance on Citeseer. Here: we use $\mathcal{M}(b)$ and $\mathcal{M}(\text{-}t)$ to denote the method $\mathcal{M}$ using the balanced and completely-imbalanced labeled data with t unseen classes, respectively.

Why traditional semi-supervised methods fail?

Traditional objective functions: $f(graph) + g(labels\ of\ \{seen\ and\ unseen\}\ classes)$

However, as the unseen class nodes are (partly) linked with the seen class ones (i.e., seen and unseen class nodes are correlated), only optimizing over the seen classes is suboptimal for the whole graph.



Minimize intra-class similarity

Maximize intra-class dissimilarity

Figure: the basic idea of traditional semi-supervised methods.

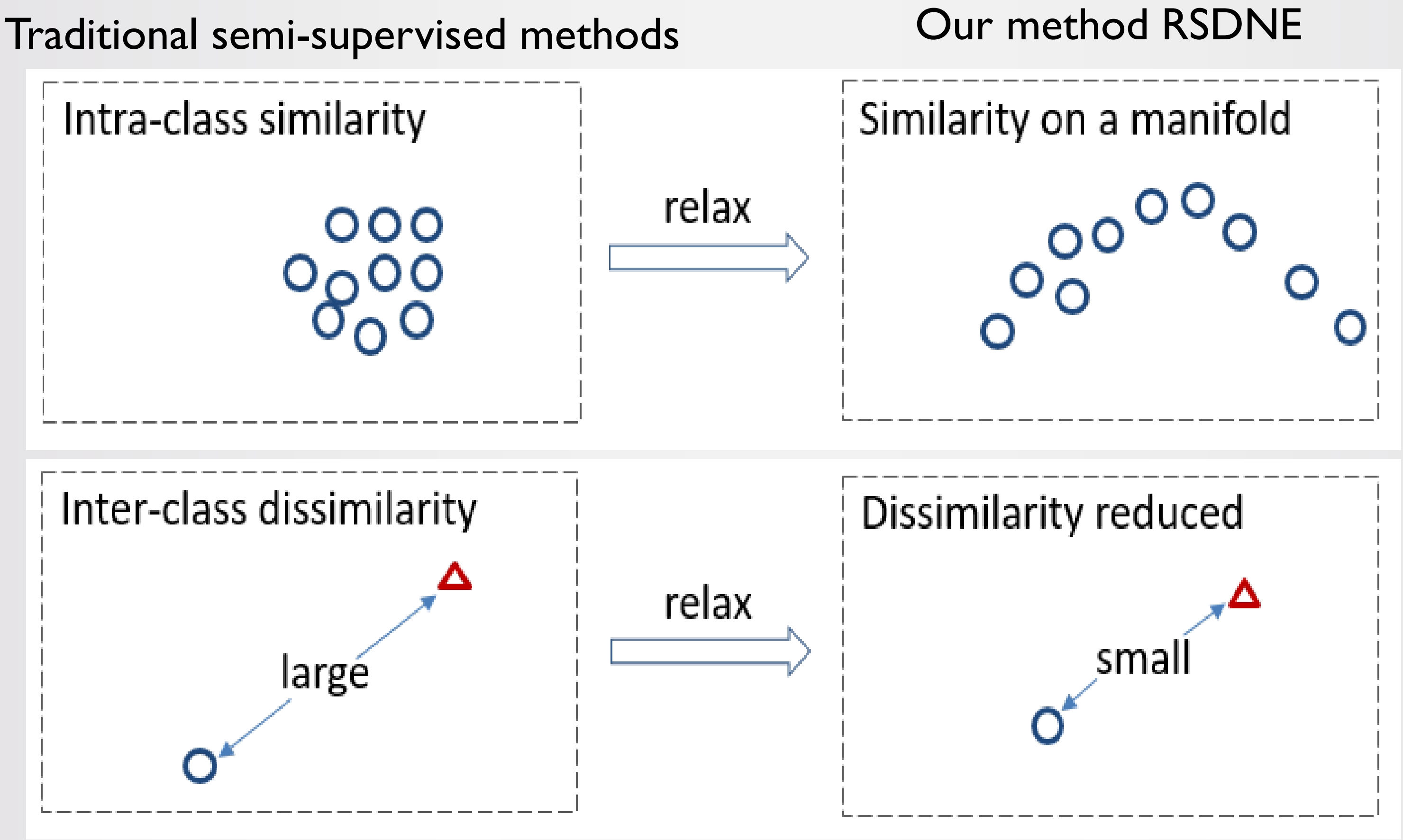
Our solutions

- **RSDNE** [AAAI 2018, CCF-A]
  - The first study on ZGE
  - The first shallow method for ZGE
  - Outperform DeepWalk by 10%-25%
- **RECT** [TKDE 2020, CCF-A]
  - The first deep method for ZGE
  - Can deal with attribute and multi-label graphs
  - Outperform GCN by 30%~300%
- **ExtendRECT** [DASFAA 2021, CCF-B]
  - A deep analysis of RECT
  - Improve RECT by 7%-20%

Our works are all open source.

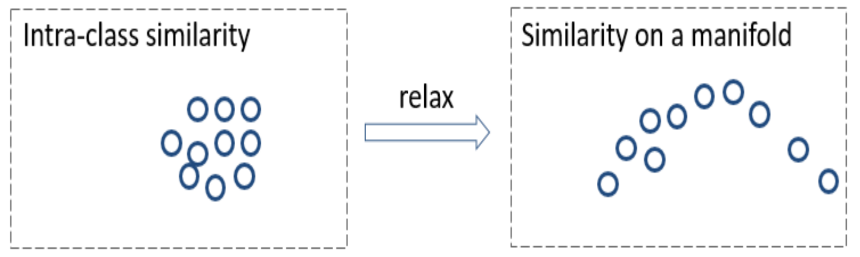
Solution I: a shallow method RSDNE

- The idea is to relax:



Solution I: a shallow method RSDNE

- The idea of RSDNE
  - Relax Intra-class Similarity (matrix S describes the similarity):



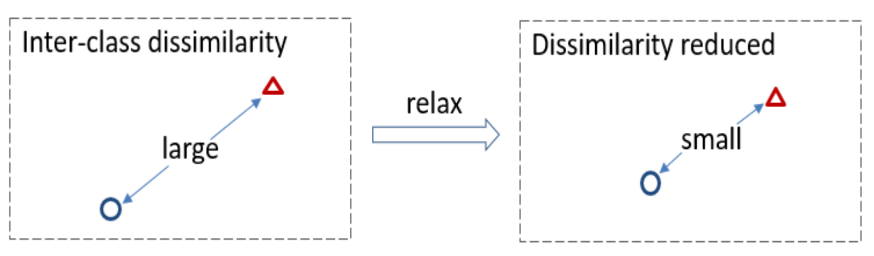
$$\min_{U,S} \ \mathcal{J}_{intra} = \frac{1}{2}\sum_{i,j=1}^{n} \left\|u_i - u_j\right\|_F^2 S_{ij}$$

$$\text{s.t.} \ \forall i \in \mathcal{L}, s_i'\mathbf{1} = k, \ S_{ii} = 0$$

$$\forall i, j \in \mathcal{L}, S_{ij} \in \{0,1\}, \ \text{if } C_i^s = C_j^s$$

$$\forall i, j, S_{ij} = 0, \ \text{if } i \notin \mathcal{L} \text{ or } C_i^s \neq C_j^s$$

- Relax Inter-class Dissimilarity:
  - Remove the known connections (described by matrix M) between the nodes with different labels

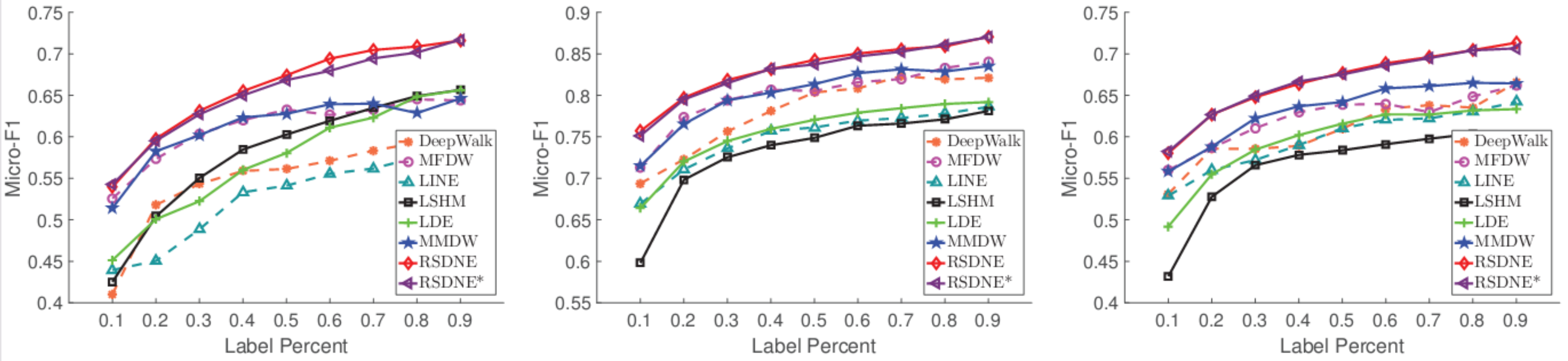


$$W_{ij} = \begin{cases} 0, & \text{if } i, j \in \mathcal{L} \text{ and } C_i^s \neq C_j^s; \\ M_{ij}, & \text{otherwise.} \end{cases}$$

$$\min_{U} \ \mathcal{J}_{inter} = \frac{1}{2}\sum_{i,j=1}^{n} \left\|u_i - u_j\right\|_F^2 W_{ij}$$

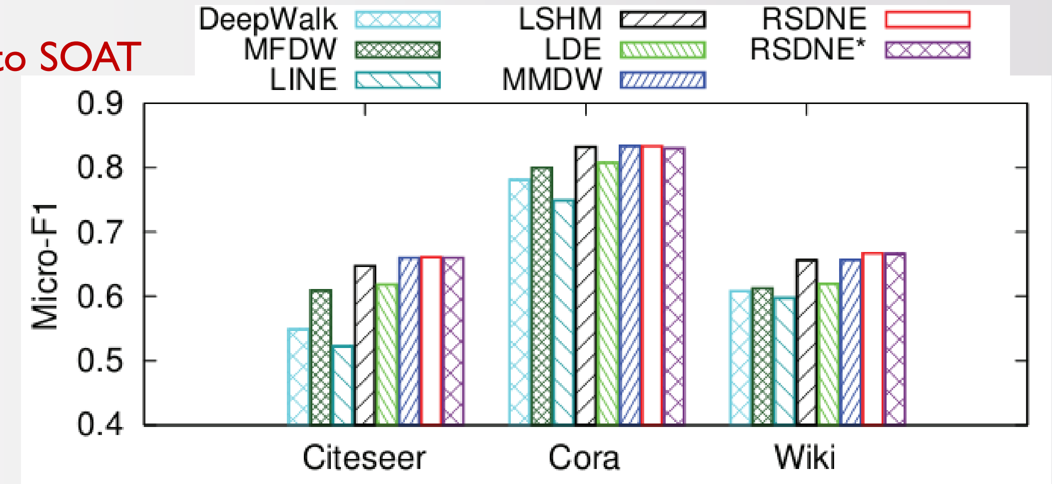
Experiments

- Node classification (Micro-F1)
  - Zero-shot case: ours outperform the best baseline by 7–15%



(a) Citeseer (b) Cora (c) Wiki

  - Balanced (traditional) case: ours obtain comparable performance to SOAT



- Published in [AAAI2018] and open source.

Solution II: a deep method RECT

- Recall RSDNE [AAAI 2018]
  - A shadow method which cannot benefit from the DNNs
  - Can not deal with “Multi-label”
  - Can not utilize node attributes

Solution II: a deep method RECT

- The idea of RECT: an interesting observation
  - Seen classes' features also contain lots of knowledge about unseen classes

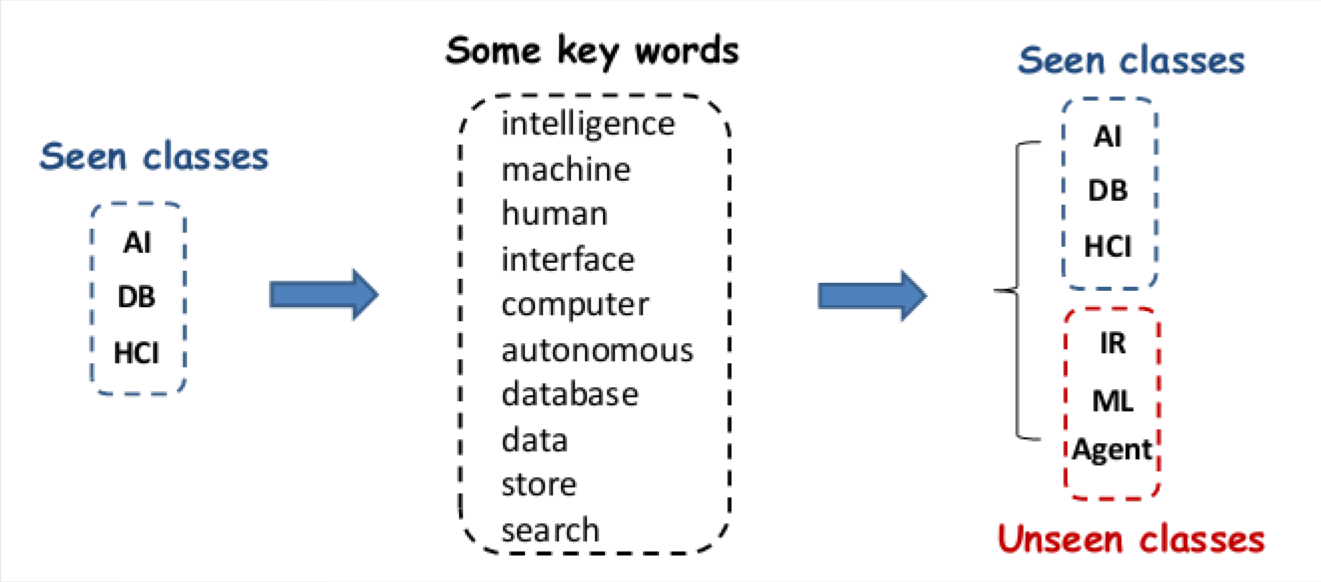


Figure: Some words sampled from the documents of three seen classes (i.e., AI, DB, and HCI) in Citeseer (a paper citation network).

Solution II: a deep method RECT

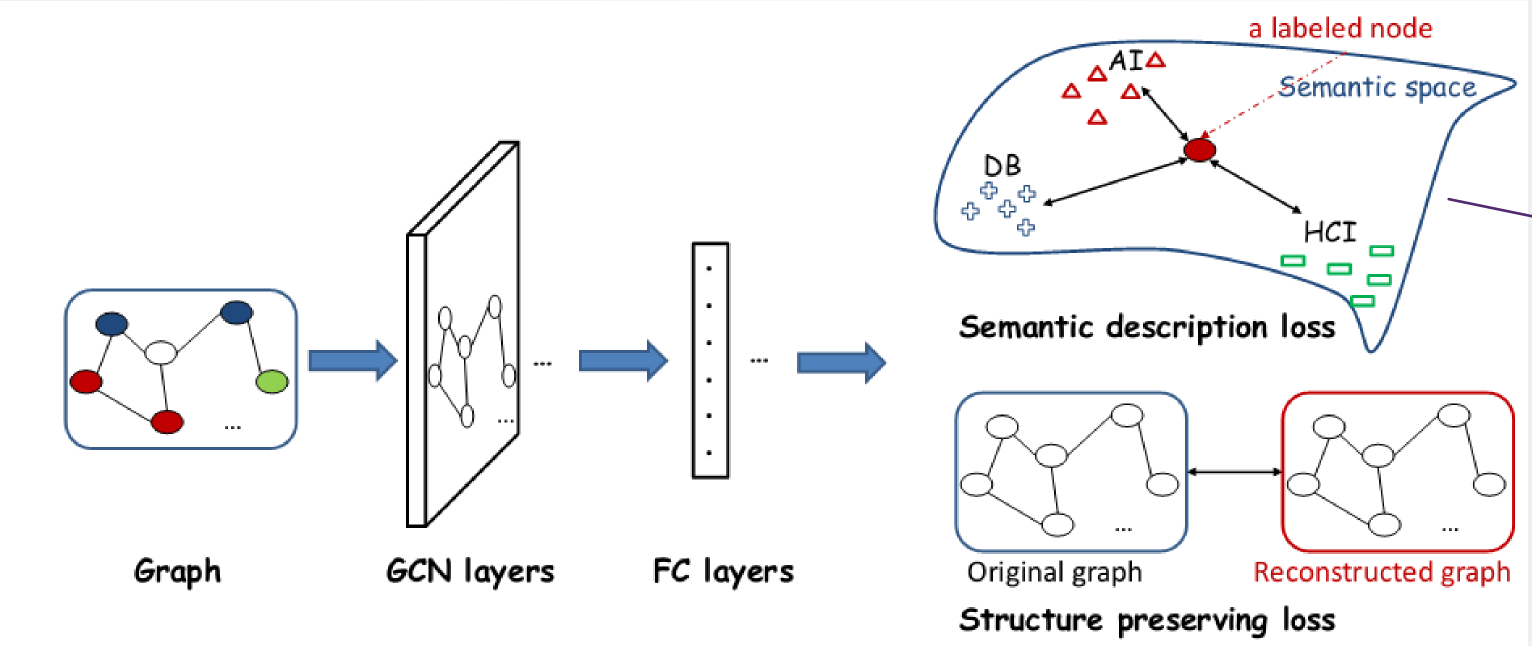


Figure: Architecture overview of RECT.

Highlight :

- Do not need any human annotations or any 3rd-part tools.
- Very easy to implement!

Algorithm RECT (more specifically its supervised part RECT-L)

Require: Graph information (A and X), and label information $\mathcal{L}$

Ensure: The learned graph node embedding results

1: Get semantic knowledge $\hat{y}_c = \mathcal{R}(\{x_i | \forall_i \ C_i^s = c\})$

2: Train a GCN-like model to minimize $\sum_{i \in \mathcal{L}} loss(\hat{y'}_{C_i^s}, \hat{y}_{C_i^s})$

3: **return** The outputs U of the first hidden layer

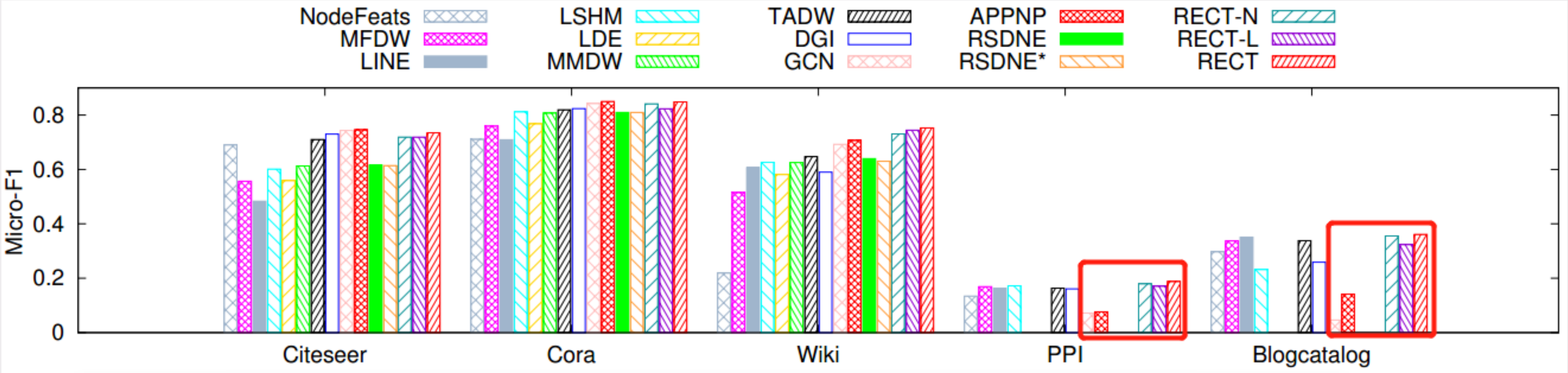
Experiments

Name	Citeseer	Cora	Wiki	PPI	Blogcatalog
Type	Citation graph	Citation graph	Hyperlink graph	Biological graph	Social graph
Nodes	3,312	2,708	2,405	3,890	10,312
Edges	4,732	5,429	17,981	76,584	333,983
Classes	6	7	17	50	39
Features	3,703	1,433	4,973	-	-
Multi-label	No	No	No	YES	YES

- Node classification (Micro-F1)
  - Zero-shot case: RECT outperforms GCN by 30%~300%

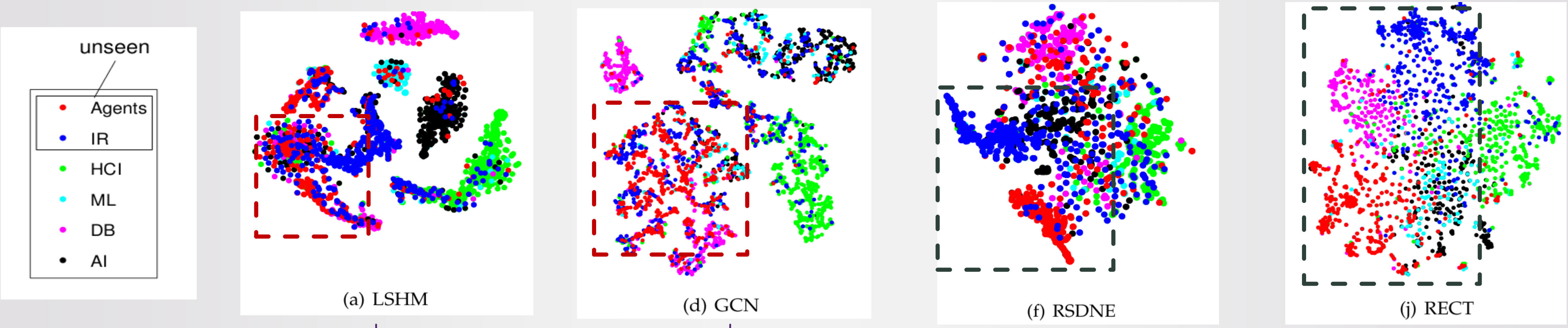
Information		X	A		A, $\mathcal{L}$			A, X		A, X, $\mathcal{L}$		A, $\mathcal{L}$		A, X	A, X, $\mathcal{L}$	
Data \ *Method*		NodeFeats	MFDW	LINE	LSHM	LDE	MMDW	TADW	DGI	GCN	APPNP	RSDNE	RSDNE*	RECT-N	RECT-L	RECT
Citeseer	10%	0.6535	0.4810	0.4448	0.4253	0.4515	0.5141	0.6844	0.7014	0.5640	0.5944	0.5395	0.5426	0.6975	0.6601	**0.7083**
	30%	0.7006	0.5793	0.4959	0.5504	0.5224	0.6020	0.7187	0.7293	0.5889	0.6274	0.6313	0.6271	0.7301	0.7154	**0.7403**
	50%	0.7161	0.6096	0.5084	0.6027	0.5805	0.6278	0.7276	0.7377	0.5995	0.6356	0.6741	0.6683	0.7359	0.7294	**0.7475**
Cora	10%	0.6508	0.6699	0.6678	0.5981	0.6641	0.7149	0.7978	0.7996	0.6436	0.7068	0.7569	0.7513	0.8187	0.7617	**0.8197**
	30%	0.7214	0.7908	0.7220	0.7254	0.7449	0.7939	0.8245	0.8350	0.6696	0.7347	0.8184	0.8147	0.8524	0.8208	**0.8561**
	50%	0.7589	0.8164	0.7373	0.7487	0.7705	0.8135	0.8361	0.8366	0.6786	0.7607	0.8426	0.8372	0.8550	0.8331	**0.8615**
Wiki	10%	0.1741	0.3570	0.5586	0.4319	0.4920	0.5582	0.5899	0.5423	0.6616	0.6189	0.5803	0.5822	0.7028	0.7006	**0.7180**
	30%	0.2212	0.5579	0.6170	0.5658	0.5846	0.6224	0.6669	0.6005	0.6952	0.6463	0.6477	0.6493	0.7363	0.7534	**0.7580**
	50%	0.2616	0.6303	0.6434	0.5838	0.6158	0.6419	0.6845	0.6274	0.7033	0.6578	0.6772	0.6751	0.7457	0.7704	**0.7711**
PPI	10%	0.0980	0.1447	0.1391	0.0306	-	-	0.1379	0.1433	0.0469	0.0439	-	-	0.1518	0.1537	**0.1659**
	30%	0.1390	0.1799	0.1693	0.0626	-	-	0.1724	0.1671	0.0449	0.0458	-	-	0.1873	0.1773	**0.1956**
	50%	0.1660	0.1833	0.1816	0.0891	-	-	0.1809	0.1715	0.0438	0.0410	-	-	0.1960	0.1834	**0.2065**
Blogcatalog	10%	0.2683	0.3192	0.3311	0.1632	-	-	0.3302	0.2371	0.0271	0.1121	-	-	0.3372	0.3076	**0.3399**
	30%	0.2984	0.3436	0.3504	0.2357	-	-	0.3409	0.2654	0.0316	0.1364	-	-	0.3571	0.3261	**0.3627**
	50%	0.3249	0.3485	0.3600	0.2803	-	-	0.3431	0.2741	0.0492	0.1365	-	-	0.3621	0.3321	**0.3692**

  - Balanced (traditional) case: RECT obtains comparable (and sometimes much superior) performance to SOAT



Experiments

- 2D visualization of embedding results



Traditional SSL methods

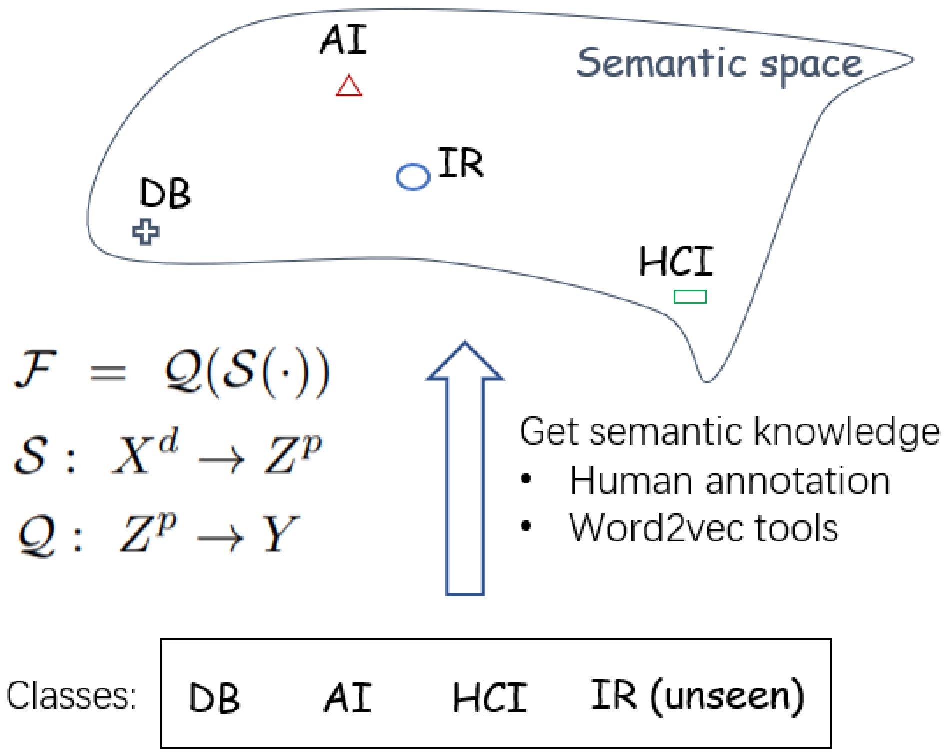
- Published in [TKDE 2020] and open source.

Solution III: ExtendRECT

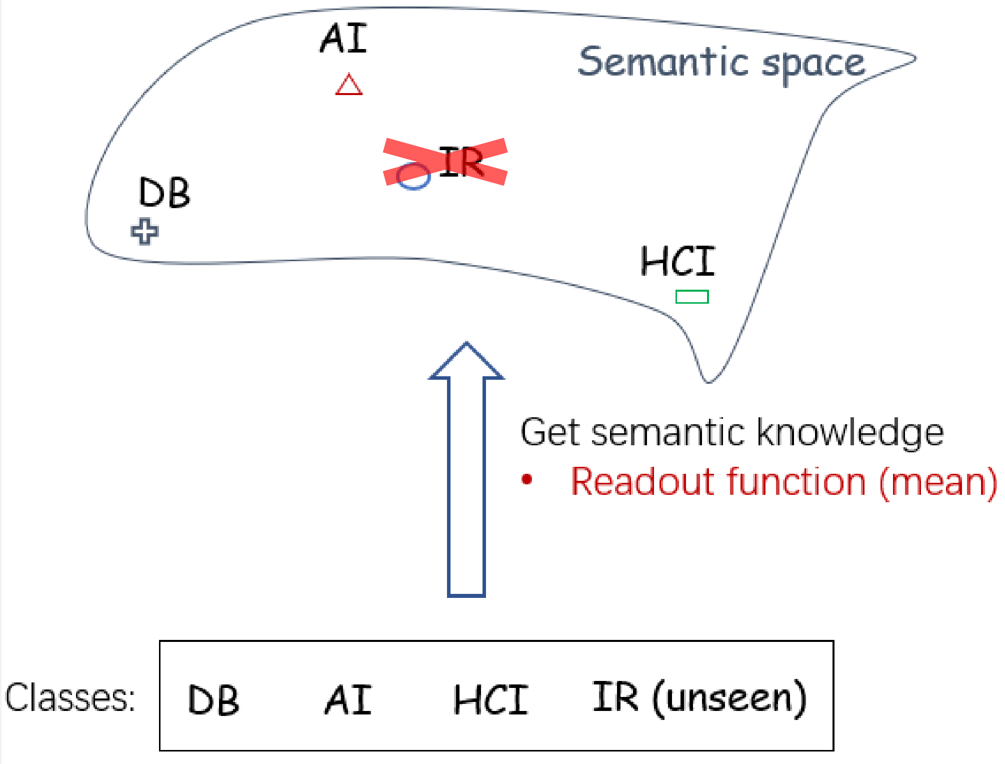
- Recall RECT [TKDE 2020]
  - Working mechanisms are not clear
  - Needs lots of training data

Why RECT works: RECT-L v.s. ZSL Methods

ZSL methods (in CV)



RECT-L (in graph)



ZSL methods have the semantic knowledge of IR (generated from some auxiliary information), while RECT-L knows nothing about IR.

Why RECT works

Remark 1 (The Difference Between RECT-L and ZSL Methods). In the semantic space of ZSL methods, class prototypes are described by human annotation or third-part resources; while in the semantic space of RECT-L, class prototypes are described by their mean feature vectors. In addition, in RECT-L, the knowledge of relationship between unseen classes and semantic space points is unknown.

Remark 2 (The Reasonability of RECT-L). As shown above, RECT-L actually learns a prototypical model with the labeled data of seen classes, reflecting its reasonability on seen classes. On the other hand, as shown in Remark 1, the learned prototypical model maps the data from the raw-input space into a semantic space, like ZSL methods. As validated by lots of ZSL methods, this enables the success of transferring supervised knowledge of seen classes to unseen classes, indicating its reasonability on unseen classes.

ExtendRECT: how to improve RECT

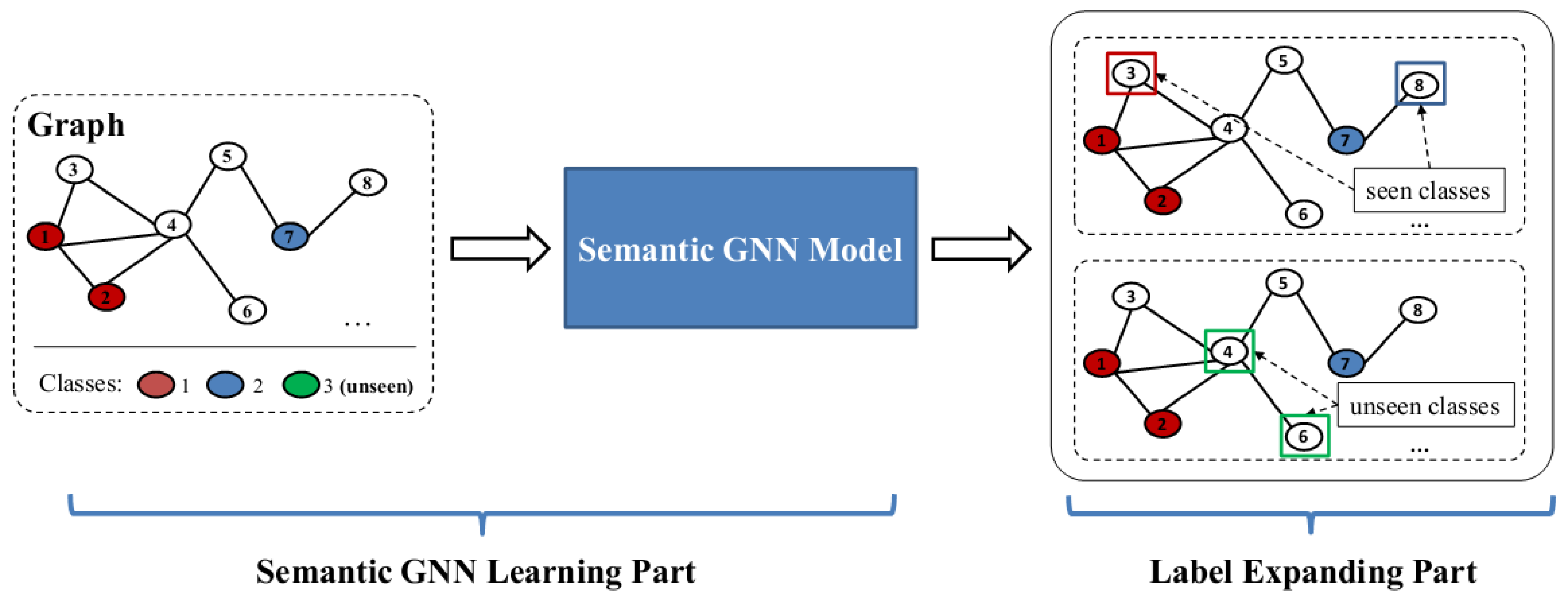


Figure: An overview of the proposed method. In the semantic GNN learning part, we learn a semantic graph embedding model. In the label expanding part, we expand the labeled node sets of seen classes and unseen classes.

Experiments

- Node classification (Micro-F1)
  - Our method outperforms RECT-L by 7%~12%

	Citeseer			Cora			Pubmed		
	1%	3%	5%	1%	3%	5%	1%	3%	5%
DeepWalk	0.1941	0.2935	0.3713	0.1972	0.3401	0.4916	0.3766	0.5879	0.6350
LSHM	0.1779	0.2143	0.2648	0.1284	0.1295	0.2233	0.3331	0.3591	0.3965
RSDNE	0.2291	0.3066	0.4035	0.2465	0.3869	0.5167	0.4193	0.6219	0.6862
GCN	0.4194	0.5211	0.5478	0.4756	0.5984	0.6266	0.6067	0.6479	0.6664
APPNP	0.4192	0.5397	0.5692	0.4921	0.6380	0.6791	0.6036	0.6287	0.6514
TEA	0.2554	0.3564	0.4010	0.2996	0.4966	0.5770	0.4953	0.5848	0.6431
RECT-L	0.4506	0.5754	0.6204	0.4964	0.6564	0.7325	0.6679	0.7495	0.7668
Ours_{SL}	0.5001	0.6004	0.6326	0.5288	0.6748	0.7374	0.7206	0.7622	0.7586
Ours_{SUL}	**0.5343**	0.6228	0.6497	0.5125	0.6761	0.7275	0.6641	0.7419	0.7336
Ours_{SUL^*}	0.5281	0.6226	0.6500	0.4984	0.6636	0.7208	0.6612	0.7406	0.7309
$\text{Ours}_{SL\text{-}SUL}$	0.5297	**0.6229**	0.6513	0.5450	**0.6963**	**0.7515**	0.7224	0.7704	0.7688
$\text{Ours}_{SL\text{-}SUL^*}$	0.5293	0.6226	**0.6518**	**0.5474**	0.6919	0.7507	**0.7353**	**0.7752**	**0.7730**

- Published in [DASFAA 2021] and open source.

Related Publications

- **Zheng Wang**, Xiaojun Ye, Chaokun Wang, etc. RSDNE: Exploring Relaxed Similarity and Dissimilarity from Completely-imbalanced Labels for Network Embedding. (AAAI 18). CCF-A.
- **Zheng Wang**, Xiaojun Ye, Chaokun Wang, Jian Cui, and Philip S. Yu. Network Embedding with Completely-imbalanced Labels. (TKDE 20). CCF-A.
- **Zheng Wang**, Chaokun Wang, Zhigong Gong and et al. Expanding Semantic Knowledge for Zero-shot Graph Embedding. (DASFAA 21). CCF-B.

Datasets and codes can be found in this project page:
https://zhengwang100.github.io/project/zero_shot_graph_embedding.html

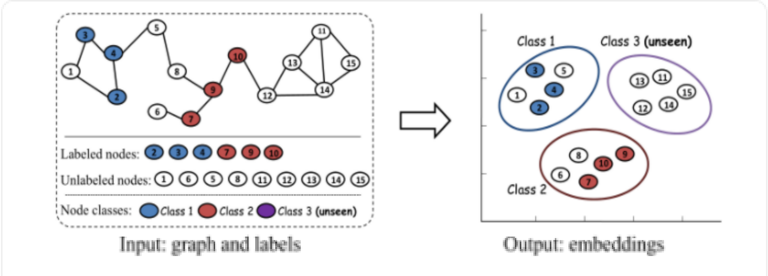


Figure: Illustration of zero-shot graph embedding. This graph actually contains three

Zero-shot Graph Embedding (ZGE) refers to the process of learning discriminative graph embeddings when labeled data cannot cover all classes.

DETAIL

NEW! Our method RECT has been officially recommended in the famous GNN library DGL.

Further work

- Design new GNN models for ZGE problem
- Design new DB platforms to support this task
- Design new AI-DB platforms to support data mining

Thanks for your time.