Relation-Aware Representation Learning in Information Networks

Weizhi Li
JD.com
2900 Lakeside Dr.
Santa Clara, California 95054
weizhi.li@jd.com

Yan Yan*
JD.com
2900 Lakeside Dr.
Santa Clara, California
yan.yan@jd.com

Zitao Liu
Pinterest Inc.
651 Brannan St.
San Francisco, California
zitaoliu@pinterest.com

Wentao Guo
JD.com
6Fl., Tower A, Beichen Century Center
Beijing, China 100101
guowentao@jd.com

Darlene King
University of Texas Southwestern Medical Center
5323 Harry Hines Blvd
Dallas, Texas 75390
adarlenea@msn.com

ABSTRACT

Network embedding is developed in many machine learning applications for mapping a network into low dimensional vector spaces. It has been proved as an effective approach to translate high dimensional structured dependent data into low dimensional independent vector representations. However, learning embeddings from real-world complex networks are not straightforward. First, there are many explicit and implicit relations encoded in the network. Edges from a standard network usually only represent one type of such relations. Moreover, each node in the network is usually associated with a rich set of information, such as text, image, video, etc. How to incorporate both explicit and implicit relations and plenary node information simultaneously while learning the representations still remains unresolved. To address these problems, we propose a novel network embedding technique called *relation-aware network embedding* (RANE) that simultaneously incorporates multiple relations while learning, and takes advantage of node degree changes which reflect the network topology and local structured information. The results demonstrate the benefits of our approach on predictive tasks of link prediction, multi-label classification and node clustering for five standard public data sets and show that RANE outperforms other state-of-the-art representation learning approaches in terms of several evaluation metrics. We release the package of RANE at https://github.com/liwzhi/RANE.

KEYWORDS

Network Embedding, Random Walk, Multi-dimensional Network

1 INTRODUCTION

Network connects hundreds of millions of objects in the world and turns into a vast source to produce important information around our everyday life. For example, millions of Internet users worldwide produce, consume and share posts, photos, videos or other types of contents in online social networks, such as Facebook, Flickr, Twitter, etc. Many machine learning and data mining research works have been done to learn and extract information from network and use the learned knowledge to make inference for unseen problems. Network embedding is one field of research which aims at learning vectorized node representations from networks [4, 5, 9, 11, 17, 24, 27–29]. Once the embeddings are learned, they are served as fundamental signals for many machine learning tasks and have been proven to be the key success of recommendation [6, 8], visualization [18], link prediction [11, 29], node classification [2], etc.

A standard network is usually defined by its nodes and edges where nodes represent entities and edges denote relations among entities. In real-world complex systems, many types of relations among entities exist. For examples, in a Youtube video network, videos are represented as nodes and edges are represented as co-viewing, co-sharing, or co-thumbing up relations among videos. Furthermore, videos are curated into playlists and playlists are created under subscribers. These structured relations can also be represented by edges. In an academic citation network, authors can be viewed as nodes and edges may indicate the corresponding co-authorships, affiliation relations, co-research interest relations, etc. In this work, we categorize all different types of real-world relations into two categories: *explicit relations* and *implicit relations*. Explicit relations represent the major entity relations we want to capture in the network which is in general task dependent. The explicit relations tightly associate with the predictive tasks and directly reflect the relatedness of entities in the real world. The rest belongs to implicit relations, which contain all rich information about the entities themselves and their latent interactions. For example, in the task of recommending relevant videos in Youtube, co-viewing relations among videos are the underlying foundation which serves as the major explicit relations we want to model. At the same time, all other relations like co-sharing, playlists/subscriber structured relations becomes implicit relations and will help improve the recommendation performance.

Although there are a large number of explicit and implicit relations in the real world, how to learn network embeddings from them still remains challenging due to the following three reasons. First, implicit relations in general are weak signals and it's difficult

*correspond author

Conference'17, Washington, DC, USA
2016. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

to make a good balance between explicit relations and implicit relations when learning network embeddings. Briefly, when we overly focus on learning from explicit relations, benefits from implicit relations will vanish. On the other hand, too much attention on implicit relations will deviate the ultimate predictive problem and deteriorate the embedding performance. Second, there are multiple types of implicit relations at the same time. For example, co-sharing relations in Youtube video recommendation scenario are quite different from the playlists/subscriber structured relations. How to create a unified learning framework to incorporate all relations is not obvious. Third, incorporating different types of implicit relations usually cause scalability issues since the learning time complexity grows linearly or even quadratically with the number of relations. Most of existing algorithms of learning network embedding either focus on only learning from explicit relations and completely ignoring implicit relations or don't scale well for many implicit relations of different types.

In this work, we propose a relation-aware network embedding algorithm (RANE) to learn network embedding from both explicit and implicit relations efficiently and effectively. Our approach generates network node embeddings by four steps: firstly, it utilizes the rich node feature vectors to create inferable implicit relations and collects all explicit and implicit relations in the network. Secondly, it computes both first-order and second-order proximity scores for each relation encoded in the network. Thirdly, it conducts the relation-aware random walk on the network and generates meaningful node sequences. Last, it trains the RANE by using skip-gram language models from the node sequences generated in previous step.

Overall this paper makes the following contributions:

- It presents a unified framework to learn network embedding from both explicit and many implicit relations of different types.
- It provides a flexible way to adjust the balance between explicit and implicit relations and it scales well with a large number of relations.
- We evaluate our embedding algorithm and its benefits comprehensively in tasks of link prediction, multi-label classification and node clustering with five public available datasets. We contribute our code to open source projects and make this research more reproducible.

The remainder of the paper is organized as follows: Section 2 provides a detailed review of existing network embedding learning algorithms. Section 3 introduces the notations and formulates the network embedding problem with multiple relations. In Section 4, we describe our RANE algorithm in details. In Section 5, we conduct experiments in three predictive tasks, including link prediction, multi-label classification and node clustering with multiple datasets to show the effectiveness of the our proposed network embedding learning algorithm in different aspects. We summarize our work and outline potential future extensions in Section 6.

2 RELATED WORK

While network contains compact and convenient information, learning from network still remains quite challenging and stops people to fully utilize such data or benefit from it due to: (1) network algorithms are often computational intensive; (2) low parallelizability for distributive algorithms due to high communication costs; (3) inapplicability to many off-the-shelf machine learning methods due to the violation of sample independence. To tackle these issues, network embedding efforts have been enforced in recent decades that translate nodes into low dimensional vector representations which aim to preserve original network information as much as possible. In general, existing research about learning network embedding can be categorized into the following three categories.

2.1 Learning Network Embedding via Matrix Factorization

A network and its topological information can be easily represented by an adjacency matrix where each element in the matrix denotes the similarity or neighboring measurement between a pair of nodes in the graph. Matrix factorization (MF), which aims to decompose the original matrix into the product of two low-rank matrices, is often viewed as one type of approaches that learn the network embeddings. A number of MF approaches have been applied in different fields and led to various successful applications. In biological networks, Natarajan and Dhillon apply MF to learn latent factors for better explaining the gene-disease associations [23]; in language modeling, Yang et al., introduce another MF algorithm that combines with information from rich texts for the multi-class classification task [30]. In spite of the success of MF, it has several drawbacks when they are applied in learning network embedding: (1) the adjacency matrix only captures two dimensional information which indicates only one relation. Even when tensor or multi-dimensional MF has been developed for capturing more complicated relations, the capability of modeling higher-dimensional relations is still very limited; (2) storing and updating the adjacency matrix are typically computationally expensive. Since factorizing a huge matrix involves matrix approximation for faster computation, large-scale MF usually suffers from substantial accuracy loss; (3) it is difficult to capture local network structure in MF because most MF approaches can only provide a global representation and fail to capture sophisticated local relationships with the low-rank approximation objective in MF approaches.

2.2 Learning Network Embedding via Random Walk

Random walk is one the most popular network embedding approaches in recent years. The walk usually starts from a random node, and by certain strategy, "walk" in the network for a number of steps. Hence, each walk produces a sequence of visited nodes. Node embeddings are learned from these node sequences by using Word2Vec language modeling. Random walk approaches are good for learning node embedding from large scale networks as well as capturing the local network structure. Popular random walk strategies include: Word2Vec [22] which establishes an effective methodology that transforms a sparse high-dimensional word vector into a dense low-dimensional vector embedding; DeepWalk [24] which solves a social relation encoding problem by mapping the walks to sentences and applying the skip-gram algorithm [21] on the sentences; Line [27] which learns the network embeddings by incorporating the information from the first-order and second-order proximities in the network. However, there are some limitations for random walk approaches: (1) it is sensitive to changes from the network topology. A small number of edge changes can affect a significant number of walks which invalidates the learned embeddings; (2) it is weak in terms of capturing multiple relations. Most random walk embedding approaches are based on a single relation in the network and they don't generalize well on networks with many relations. When they are applied to networks with multiple types of relations, both space complexity and time complexity are likely to increase to a large extent.

2.3 Learning Network Embedding via Deep Learning

Deep neural network introduced by [1, 14] has shown its revolutionary power in lots of fields such as natural language processing [7, 20], image recognition [10, 15, 25], recommender systems [6, 8], etc. For network embedding tasks, there are quite a few successful approaches based on deep learning that have been proposed, such as Node2Vec [11], SDNE [28], SDAE [4], etc. Although deep learning based network embedding approaches provide good end-to-end solutions, the most notorious disadvantage for deep learning network embedding is that the proposed embeddings have lost their physical meaning and are difficult for human to understand or explain. Moreover, incorporating multiple relations into deep learning network embedding systems is error-prone. Last, deep neural network themselves are not easy to train or maintain comparing with other types of embedding approaches.

3 PROBLEM STATEMENT

In real-world information network, there are both explicit and implicit relations among netowrk entities, *a.k.a*, nodes. Furthermore, each node is usually associated with a rich feature vector, which stores all available information about the particular node. In this work, we represent a generic network as $\mathcal{G} = \left\langle \mathcal{V}, \mathcal{E}^0, \mathcal{E}^1, \cdots, \mathcal{E}^T \right\rangle$, where $\mathcal{V} = \{v_1, \cdots, v_N\}$ is the set of nodes in $\mathcal{G}$ and $\mathcal{E}^0$ denotes the *explicit relation* and $\mathcal{E}^1, \cdots, \mathcal{E}^T$ are the *implicit relations*. Let $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ be the set of feature vectors corresponding to the node set $\mathcal{V}$. N and T denote the number of nodes and implicit relations in $\mathcal{G}$.

With the aforementioned notations, our problem can be formally defined as follow:

Definition 3.1. (RANE: Relation-Aware Network Embedding) Given (1) a set of node $\mathcal{V}$ and its corresponding feature set $\mathcal{X}$; (2) an explicit relation $\mathcal{E}^0$ among $\mathcal{V}$; and (3) a set of implicit relations $\mathcal{E}^1, \cdots, \mathcal{E}^T$, we aim to learn a function $\mathcal{F}$ to generate network embeddings $\mathcal{Y}$ for nodes V, i.e.,

$$\mathcal{Y} = \mathcal{F}(\mathcal{V}, \mathcal{E}^0, \mathcal{E}^1, \cdots, \mathcal{E}^T, \mathcal{X}). \tag{1}$$

4 LEARNING RALATION-AWARE NETWORK EMBEDDING

In this work, we develop a novel approach to learn relation-aware network embedding, which is able to (1) fully utilize the similarity learnings, which are computed from node feature vectors; (2) take into account not only explicit relations but also implicit relations in the network and (3) provide a flexible way to balance between explicit and many implicit relations of different types. More specifically, our method works as follows: besides the given (observational) implicit relations, it creates an additional set of implicit latent similarity relations from node feature vectors (Section 4.1). Then it extracts first-order and second-order proximities from both explicit and many implicit relations. The implicit relations are the union of generated implicit relations from node feature vectors (inferable implicit relations) and the implicit relations naturally existed (observational implicit relations) in the network (Section 4.3). After that, we develop a scalable relation-aware random walk algorithm to wisely jump among nodes according to our relation-aware proximity scores which produces more meaningful nodes sequences (Section 4.2). Last, it utilizes skip-gram language models to generate the relation-aware network embeddings (Section 4.4).

4.1 Implicit Relation Generation

Many implicit relations exist in the real-world information network and they can be categorized as *observational implicit relations* and *inferable implicit relations*. Observational implicit relations are those implicit relations naturally appear in the network and can be observed and logged in the data. While inferable implicit relations are the opposite and they are computed and inferred from nodes or other information in the network. For example, in the Youtube network for recommending relevant videos, video co-sharing and playlists/subscriber structured relations belong to observational implicit relations. At the same time, we have rich text and image content about each video and inferable implicit relations can be computed by using the similarity of each pair of nodes in the network.

In this work, we generate the inferable implicit relations by thresholding the similarity between each pair of nodes. Let $\mathcal{E}^k$ be the kth implicit relations and $\mathcal{E}^k(i, j)$ is the indicator variable where $\mathcal{E}^k(i, j)$ is one when there is an implicit relation between node v_i and v_j; otherwise, there is no such relation. $\mathcal{E}^k(i, j)$ is computed as follows:

$$\mathcal{E}^k(i, j) = \begin{cases} 1 & \text{if } Sim(i, j) >= \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

where $Sim(i, j)$ is the function to compute similarity between v_i and v_j. α is the cut-off hyper parameter.

4.2 Relation-Aware Proximity Extraction

Node proximity measures the closeness, *a.k.a.*, relevance between two nodes in a network. How to accurately estimate all pairs of proximities and how to fully utilize them to generate embeddings are the key success to network representation learning.

In this work, in order to utilize all available explicit and implicit relations, we create two generic proximity extraction functions and apply them to each network relation once a time. More specifically, we define a first-order proximity function and a second-order proximity function to capture each node's local structure under each relation. Then for each node, we define a weighting function to combine all relation-specific proximities together.

4.2.1 First-order Proximity Extraction. Let d_i^k be the degree of v_i and B^k be the bin size under the kth relation. In first-order proximity function, we discretize node degrees into buckets and create binning labels for each bucket. Let b_i^k be the binning label of v_i, i.e.,

$$b_i^k = \left\lfloor \frac{d_i^k}{B^k} \right\rfloor \tag{2}$$

Then we define first-order proximity matrix of relation k as $\mathbf{P}^k$ and P_{ij}^k denotes the first-order proximity score between v_i and v_j for relation k as follows:

$$P_{ij}^k = \begin{cases} 1 & \text{if } b_i^k = b_j^k, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

4.2.2 Second-order Proximity Extraction. In this work, we utilize Jaccard similarity coefficient to compute the second-order proximity score for each node pair. Let $\mathcal{N}_i^k$ be the neighbor nodes of v_i in relation k and $|\mathcal{N}_i^k|$ denote the number of nodes in $\mathcal{N}_i^k$. We define second-order proximity matrix of relation k as $\mathbf{Q}^k$ and Q_{ij}^k denotes the second-order proximity score Q_{ij}^k between v_i and v_j for relation k as follows:

$$Q_{ij}^k = \frac{|\mathcal{N}_i^k \cap \mathcal{N}_j^k|}{|\mathcal{N}_i^k| + |\mathcal{N}_j^k| - |\mathcal{N}_i^k \cap \mathcal{N}_j^k|} \tag{4}$$

4.2.3 Relation-Aware Proximity Score. Once all pairs' first-order and second-order proximity scores are computed for both explicit and implicit relations, we construct the relation-aware proximity score R_{ij} by a weighed average of P_{ij}^k and Q_{ij}^k, i.e.,

$$R_{ij} = \frac{\sum_{k=1}^K w_k(\beta P_{ij}^k + (1-\beta)Q_{ij}^k)}{\sum_{k=1}^K w_k} \tag{5}$$

where β is the hyper parameter that adjust the balance between first-order proximity and second-order proximity. K is the total number of relations in the network and w_k is weight, *a.k.a*, importance of the k^{th} relation. There are multiple heuristics to compute w_k from data. For example, w_k might represent the total number of edges in relation k or the sum of all node degrees in relation k.

4.3 Relation-Aware Random Walk

In this section, we design a flexible neighborhood sampling strategy which allows us to fully utilize the explicit and implicit relations in the network. Moreover, our strategy penalizes nodes that are similar to previous visited nodes. We would like to have diverse node sequences to capture the complex network structure, which is also similar in language sentence modeling, a sentence of words that have the same part-of-speech doesn't help much when learning word embeddings. We achieve this by developing a relation-aware random walk procedure that can explore neighborhoods according to the relation-aware proximity score and penalize the neighborhood nodes based on a designated discounting factor.

Formally, let $\mathbf{v} = \{v_{I_1}, v_{I_2}, \cdots, v_{I_l}\}$ be a sequence of nodes simulated by our relation-aware random walk of fixed length L where $I_1, I_2, \cdots, I_L$ are the node indices and v_{I_j} is the jth visited node in the walk. Node $v_{I_{j+1}}$ is generated by the following distribution:

$$P(v_{I_{j+1}}|v_{I_j}) = \frac{\gamma}{d_{I_j}} \tag{6}$$

where γ is the discounting factor that will decrease the walking probability if the node is very similar to previous visited nodes. γ is defined as follows:

$$\gamma = 1 - \prod_{l=0}^{W-1} \delta(v_{j+1-l}, v_{j-l}) \tag{7}$$

where W is the historical window length that determines how long we look back to compute similarities of current node with past visited nodes in that window. $\delta(v_{j+1-l}, v_{j-l})$ is the probabilistic function which decides whether the penalty is enforced and how much penalty we want to apply. $\delta(v_{j+1-l}, v_{j-l})$ is defined as follows:

$$\delta(v_i, v_j) = \begin{cases} R_{ij} & \text{if } R_{ij} > \lambda, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

where λ is the hyper parameter.

In eq.(7), we design γ such that if there exist one node which is significantly different from all the rest nodes in the sequence window, we continue the walk with uniformly neighborhood sampling; otherwise, we will apply a discount on the probability of walking to that node.

4.4 Embedding Generation via Language Modeling

Skip-gram is a language model that has enlightened many random walk based network embedding approaches. The walking sequences in network are considered as sentences in documents and the underlying assumption is that given a target word, its neighboring words should be statistically dependent on it. Hence, the conditional probability of words that co-occur with that target word within a certain window width should be maximized. Similar to node sequences generated by random walk, the conditional probability of nodes that co-occur should be maximized. More details can be found in [22].

4.5 Summary of the Learning Algorithm

The RANE represenation learning is summarized by Algorithm 1.

5 EXPERIMENTS

Network embeddings can be used in many applications, in the work we conduct three tasks: (1) *link prediction*; (2) *multi-label classification*; (3) *node clustering* to demonstrate the performance of RANE against other baseline methods. Before moving into the experiment details, we first briefly introduce the baseline approaches that have been compared with throughout the experiment section:

- **DeepWalk** [24]: one of the most renowned "walk" type of representation learning approach which translates the walking sequences into sentences and utilizes the language model to solve a joint likelihood optimization problem.

Algorithm 1 RANE: Relation-Aware Network Embedding

```
1:  INPUT:
2:  $\mathcal{X}$: node feature vector set
3:  $\mathcal{E}^0$: explicit relations encoded in the network
4:  $\mathcal{E}^1, \mathcal{E}^2, \cdots, \mathcal{E}^T$: $T$ observational implicit relations
5:  $\alpha$: threshold to determine inferable implicit relations (eq.(4.1)).
6:  $B^k$: bin size to discretize node degrees in $k$th relation (eq.(2)).
7:  $\beta$: coefficient of combining first/second-order proximity score
    (eq.(5)).
8:  $W$: window size of discounting factor (eq.(7)).
9:  $\lambda$: threshold to apply the discount penalty (eq.(8)).
10: procedure LEARNING RANE
11:     // Implicit Relation Generation
12:     Initialize relation set with explicit relation, i.e., $\mathcal{E} = \mathcal{E}^0$.
13:     Add observational implicit relations into $\mathcal{E}$, i.e., $\mathcal{E} =
        \mathcal{E} \cup \{\mathcal{E}^1, \mathcal{E}^2, \cdots, \mathcal{E}^T\}$.
14:     for each similarity metric do
15:         Create inferable implicit relation $e$ by eq.(4.1)
16:         Add $e$ into $\mathcal{E}$, i.e., $\mathcal{E} = \mathcal{E} \cup e\}$.
17:     // Relation-Aware Proximity Extraction
18:     for each node pairs $\langle v_i, v_j \rangle$, $v_i \in \mathcal{V}$, $v_j \in \mathcal{V}$ do
19:         for each relation $e \in \mathcal{E}$ do
20:             Compute first-order proximity $P_{ij}$ by eq.(3).
21:             Compute second-order proximity $Q_{ij}$ by eq.(4).
22:         Compute relation-aware proximity score $R_{ij}$ by eq.(5).
23:     // Relation-Aware Random Walk
24:     Entire node sequence set $\mathcal{S} = \{\}$
25:     for each i = 1, $\cdots$, $N$ do
26:         Generate sequence $\mathbf{s}_i$ at $i$th round by repeating relation-
        aware random walk based on eq.(6). $\mathcal{S} = \mathcal{S} \cup \mathbf{s}_i$.
27:     // Embedding Generation via Language Modeling
28:     Train skip-gram models based on $\mathcal{S}$, i.e., Y = $SkipGram(\mathcal{S})$.
```

- **Node2Vec** [11]: improves on the original random walk strategy for generating sequences by combining breadth-first sampling and depth-first sampling walking preference to visit more unknown samples that are further away from the starting node. In experiment, we set two control parameter in Node2Vec as $p = 1$ and $q = 2$ (shown in [9], for $p = 1$ and $q = 1$ Node2Vec is the equivalent to DeepWalk).
- **LINE** [27]: not only considering the first-order proximity but also the second-order proximity in the network structure, LINE finds the representation that preserves both proximities for each node. LINE-1$^{\text{st}}$ and LINE-2$^{\text{nd}}$ are two variants of LINE implementations which use the *first*- and *second*-order proximity respectively.
- **SDNE** [28]: not only considering the *first*-order and *second*-order relations, SDNE uses the deep learning framework to find the low dimensional representation that preserves higher dimensional non-linear relations and local network structures.

Some important model parameter settings in implementations are highlighted as follows. For RANE, DeepWalk, and Node2Vec we set $L = 80$ steps as one walking sequences, $W = 10$ as window width, and 150 as the iteration rounds. For SDNE we also set 150 as the iteration rounds, and for LINE we set 500. We utilize all representation learning methods to transform the network vertices into a 128 dimensional vector space and compare the transformed embeddings regarding area under the curve (AUC) for link prediction, $F1$-scores for multi-label classification, and *Calinski-Harabaz* score for node clustering.

5.1 Link prediction

For link prediction, the graph $\mathcal{G}$ is only partially observed. Based on the network embeddings, people predict the connectivity regarding with the remaining unobserved edges by using standard classification algorithms. In this task, we randomly select 50% edges for the representation learning and the remaining 50% for link prediction. Meanwhile we blend the training and testing with unconnected node pairs so the data are balanced. Finally we train a logistic regression classifier for each representation to predict the edge existence. Based on the 128 dimensional embeddings we transform each embedding pair $\langle f(v_i),\ f(v_j) \rangle$ into the following 4 pairwise features that are commonly adopted in link prediction evaluation: (a) average feature: $\frac{f(v_i)+f(v_j)}{2}$; (b) Hadamard transform feature: $f(v_i) * f(v_j)$; (c) $L1$-difference: $|f(v_i) - f(v_j)|$; (d) $L2$-difference: $|f(v_i) - f(v_j)|^2$.

We evaluate all representation learning approaches over the following 3 public datasets:

- *Facebook social network data* (Facebook) [16]: there are 4,039 nodes and 88,234 edges in the network, where each node represents a single user and each edge represents whether there exists the friendship between two corresponding users. The network by itself is undirected.
- *Astro physics collaboration network data* (arXiv) [16]: there are 18,772 nodes and 198,110 edges in the network, where each node represents an author, and each edge represents the co-authorship in the scientific papers that are submitted to the arXiv platform. The network by itself is undirected.
- *Protein-Protein Interaction data* (PPI) [26]: there are 3,890 nodes and 76,584 edges in the network which is a subgraph from a PPI network for Homo Sapiens. Each node represents a particular protein and each edge represents the biological interaction between a pair of proteins. The network by itself is undirected.

The experiment results are reported in Table 1. RANE either beats or is at least as good as other baselines on 3 public datasets. For Facebook data, RANE wins on $L2$-difference features and gets the overall equally best performance as SDNE on $L1$-difference features. For arXiv data, RANE wins all 4 standard features and gets the overall best performance on average features. For PPI data, RANE wins on $L1$-difference and $L2$-difference features and again gets the overall best performance on $L2$-difference features.

5.2 Multi-label classification

Multi-label classification has been well studied and it has been tightly connected with representation learning and network understanding [12]. In this task, we utilize the transformed embedding by each representation learning approach to feed into an one-vs-rest

Ft.	Algorithm	Facebook	arXiv	PPI
(a)	LINE-1st	0.6301	0.5413	0.6353
	LINE-2nd	0.8063	0.6201	0.8635
	SDNE	0.8244	0.7235	0.8499
	DeepWalk	0.7183	0.9792	0.7288
	Node2Vec	0.7618	0.9899	0.8719
	RANE	0.7086	**0.9937**	0.7316
(b)	LINE-1st	0.7383	0.5169	0.5922
	LINE-2nd	0.7822	0.5136	0.8228
	SDNE	0.9497	0.8306	0.7764
	DeepWalk	0.9403	0.8851	0.7195
	Node2Vec	0.9603	0.9805	0.7277
	RANE	0.9468	0.9848	0.7005
(c)	LINE-1st	0.7325	0.5493	0.6277
	LINE-2nd	0.7496	0.5301	0.7883
	SDNE	**0.9902**	0.7265	0.7663
	DeepWalk	0.9884	0.8472	0.9016
	Node2Vec	0.9343	0.9079	0.8449
	RANE	0.9882	0.9559	0.9030
(d)	LINE-1st	0.7369	0.5372	0.6203
	LINE-2nd	0.7647	0.5269	0.7977
	SDNE	0.9871	0.5924	0.7710
	DeepWalk	0.9893	0.8314	0.9024
	Node2Vec	0.9364	0.8561	0.8469
	RANE	**0.9902**	0.9412	**0.9038**

Table 1: AUC for different network embedding approaches on 3 public data sets.

logistic regression classifier with $L2$ regularization for the multi-label classification. For better evaluation on model robustness, we split different proportion of network data into the training set from 10% up to 90% and the rest into the testing set. Regarding with metrics, we use two types of $F1$-score, namely Macro-$F1$ (equal weight *per* class) and Micro-$F1$ (equal weight *per* instance) [28] to justify the multi-label classification performance. Similar as link prediction task, we use 3 public data sets to evaluate the performance for different embedding algorithms:

- *Wikipedia* (Wiki POS) [19]: there are 4, 777 nodes, 184, 812 edges, and 40 different labels, where each node represents a single document, each edge represents the linkage existence between two documents, and different labels represent the structural components of part-of-speech tagging (POS tagging) by using the Stanford POS-Tagger [20].
- *Astro Physics collaboration network* (arXiv) [16]: the same data set that have been tested in previous link prediction task.
- *BlogCatalog* (Blog) [31]: there are 10, 312 nodes, 333, 983 edges and 39 different labels, where each node represents a single blog, each edge represents the friendship among two blog posters and the labels represent the bloggers' interest inferred by other posters' blogs.

We report the multi-label classification results for different representation learning algorithms in Table 2 (PPI data), Table 3 (Wiki POS data), and Table 4 (Blog data). For PPI data, although RANE does not win against SDNE for small training sets (10% & 20%), it quickly catches up when the training set gets bigger (30%-70%). When the training set reaches 80% & 90%, Node2Vec and RANE yield to similar performance. For Wiki POS data, RANE wins cases for smaller training sets (10% - 40%). Later, Node2Vec and DeepWalk catch up when the training set gets big enough (50% - 90%) for both methods to extract meaningful information so to reach comparably good embedding results. For Blog data, which contains most number of edges among 3 networks, RANE performs the best by beating other baselines in all cases (10% - 90%).

Parameter sensitivity: for representation learning, the algorithm robustness is also an important issue. Since both Node2Vec and RANE are walk type of representation learning, we compare both approaches' parameter sensitivity regarding with the following parameters: (1) embedding dimensionality d; (2) the walk length L; the number of walks per node U; (3) the window width W. We utilize PPI data and the same logistic regression classifier to conduct this experiment. As shown in Figure 1 & 2, when the embedding dimension d gets larger, the Micro-$F1$ and Macro-$F1$ for both algorithms drop quickly, which implies the embedded vectors contain redundant information and the logistic regression classifier lean to overfit. Regarding with walking length L and number of walks per node U, both methods are not very sensitive with respect to Macro-$F1$ and Micro-$F1$ performance. For window width W, RANE appears more stable (dropping slower) when W gets larger.

5.3 Node clustering

Clustering, an unsupervised machine learning technique, is helpful in finding unknown relationships in a network, better understanding individual node behavior and its interaction towards the local network structure. In this task, we conduct the experiment on 2 public data set mentioned as before: (1) Protein-Protein Interaction data (PPI); (2) the Wikipedia POS data (Wiki POS). More specifically, after nodes are mapped to a d dimensional vector space via different representation learning approaches, we use K-means [13] algorithm to cluster all samples into different clusters. In terms of metric, we use *Calinski-Harabaz* score [3] s_M to evaluate the clustering quality, which is defined as below:

$$s_M = \frac{tr(B_M)}{tr(W_M)} \times \frac{N-M}{M-1}$$

$$W_M = \sum_{q=0}^{M} \sum_{x \in C_q} (x - c_q)(x - x_q)^\top \quad B_M = \sum_q n_q (c_q - c)(x_q - c)^\top$$

where N is the number of nodes, M is the number of clusters, $tr(\cdot)$ is the trace operator, B_M is called as the between group dispersion matrix and W_M is called as the within-cluster dispersion matrix. C_q is the sample set which defines cluster q; c_q is the centroid of C_q and n_q is the number of the samples in C_q. In general, a high *Calinski-Harabaz* score means that the resulting clusters are dense and well separated. We report the *Calinski-Harabaz* scores for all representation methods in Table 5. For PPI data, comparing with other baselines RANE beats the second best (DeepWalk) for 0.32%. For Wiki POS data, RANE also wins over other baselines by beating the second best (Node2Vec) for 0.98%.

Parameter sensitivity: similar to multi-label classification task, we evaluate how sensitive the clustering quality is over both RANE and Node2Vec parameters under *Calinski-Harabaz* measurement.

	nodes %	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-$F1$	LINE-1st	0.0060	0.0061	0.0040	0.0030	0.0036	0.0025	0.0004	0.0011	0.0001
	LINE-2nd	0.0073	0.0068	0.0054	0.0046	0.0041	0.0032	0.0018	0.0015	0.0011
	SDNE	**0.1154**	**0.1127**	0.1066	0.1012	0.107	0.1012	0.0937	0.0882	0.0633
	DeepWalk	0.0859	0.0789	0.0989	0.0977	0.0967	0.0921	0.0907	0.0844	0.0557
	Node2Vec	0.0862	0.0802	0.0956	0.0881.	0.0929	0.0991	0.0933	**0.0894**	**0.0707**
	RANE	0.0848	0.0968	**0.1159**	**0.1095**	**0.1082**	**0.1018**	**0.0979**	0.0864	0.0680
Macro-$F1$	LINE-1st	0.0046	0.0035	0.0022	0.0019	0.0021	0.0016	0.0003	0.0008	0.0004
	LINE-2nd	0.0052	0.0044	0.0031	0.0027	0.0025	0.0023	0.0014	0.0010	0.0007
	SDNE	**0.0742**	**0.0784**	0.0775	0.0779	0.0801	0.0787	0.073	0.0654	0.0445
	DeepWalk	0.0692	0.0776	0.0905	0.0793	0.0764	0.0714	0.0719	0.0639	0.0431
	Node2Vec	0.0697	0.0627	0.0771	0.0733	0.0771	0.0805	0.0736	**0.0696**	**0.0540**
	RANE	0.0513	0.073	**0.0928**	**0.0878**	**0.0865**	**0.0821**	**0.0775**	0.0690	0.0527

Table 2: Micro-$F1$ and Macro-$F1$ for different embedding algorithms on Homo sapiens, PPI data

	nodes %	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-$F1$	LINE-1st	0.2206	0.2242	0.1994	0.2115	0.207	0.2142	0.2163	0.2173	0.2373
	LINE-2nd	0.1856	0.1836	0.1895	0.2093	0.1974	0.2098	0.2159	0.2186	0.2118
	SDNE	0.3582	0.343	0.3348	0.3187	0.3078	0.3085	0.3125	0.2994	0.2787
	DeepWalk	0.3785	0.3813	0.3714	0.3620	0.3675	0.3620	0.3570	0.3483	**0.3251**
	Node2Vec	0.3831	0.3776	0.3762	0.3708	0.3651	0.3542	0.3597	0.3385	0.3225
	RANE	**0.4112**	**0.4049**	**0.3960**	**0.3846**	**0.3717**	**0.3745**	**0.3633**	**0.3522**	0.3113
Macro-$F1$	LINE-1st	0.0114	0.0116	0.0105	0.0117	0.0113	0.0125	0.0142	0.0176	0.0219
	LINE-2nd	0.011	0.0101	0.0103	0.0116	0.011	0.0131	0.0149	0.018	0.0228
	SDNE	0.0386	0.037	0.0344	0.0334	0.0300	0.0291	0.0297	0.0276	0.0233
	DeepWalk	0.0596	0.0579	0.0553	0.0584	0.0558	**0.0563**	0.0516	0.0427	**0.0387**
	Node2Vec	0.0679	0.0677	0.0678	0.0645	**0.0577**	0.0529	**0.0519**	**0.0478**	0.0358
	RANE	**0.0697**	**0.0836**	**0.0735**	**0.0678**	0.0546	0.0538	0.0484	0.0454	0.0315

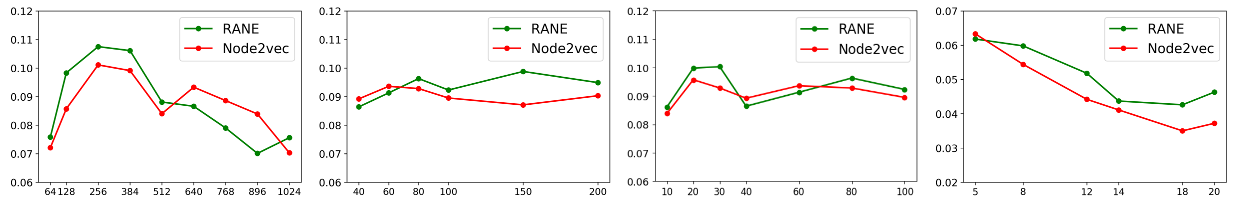
Table 3: Micro-$F1$ and Macro-$F1$ for different embedding algorithms on Wiki POS data

	nodes %	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-$F1$	LINE-1st	0.1240	0.1222	0.1183	0.1161	0.1156	0.1113	0.1107	0.1108	0.1045
	LINE-2nd	0.1250	0.1251	0.1235	0.1190	0.1131	0.1073	0.1011	0.1053	0.1040
	SDNE	0.1147	0.1152	0.1093	0.1106	0.109	0.1094	0.1013	0.0971	0.0873
	DeepWalk	0.1124	0.1120	0.1067	0.0989	0.0968	0.089	0.0785	0.0771	0.0704
	Node2Vec	0.1224	0.1274	0.1371	0.1381	0.1371	0.1399	0.1359	0.1472	0.1440
	RANE	**0.1818**	**0.1817**	**0.1891**	**0.1838**	**0.1819**	**0.1834**	**0.1763**	**0.1759**	**0.1625**
Micro-$F1$	LINE-1st	0.0128	0.0125	0.0136	0.0136	0.0148	0.0156	0.0159	0.0164	0.0163
	LINE-2nd	0.0125	0.0137	0.0151	0.0158	0.0155	0.0155	0.0151	0.0156	0.0168
	SDNE	0.0471	0.049	0.0479	0.0475	0.0460	0.0459	0.0435	0.0413	0.0379
	DeepWalk	0.0132	0.0159	0.0187	0.0179	0.0211	0.0226	0.0210	0.0249	0.0244
	Node2Vec	0.0649	0.0675	0.0747	0.0778	0.0788	0.0830	0.0788	0.0821	0.0756
	RANE	**0.0907**	**0.0932**	**0.1011**	**0.1040**	**0.1028**	**0.1016**	**0.0968**	**0.0932**	**0.0800**

Table 4: Micro-$F1$ and Macro-$F1$ for different embedding algorithms on Blog data

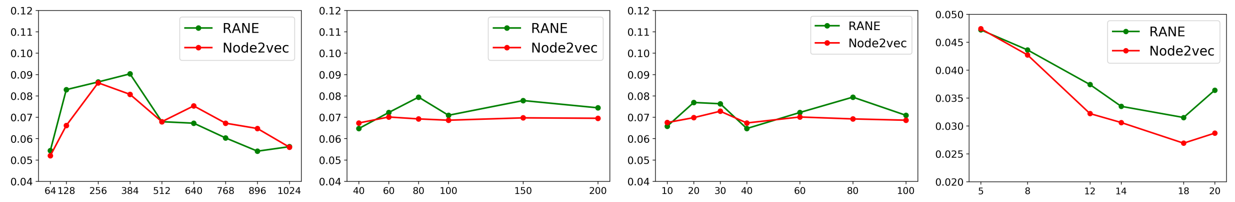
Shown in Figure 3, clustering quality drops quickly for both approaches when embedding dimension d getting larger. When walking steps L gets bigger and the window width W gets wider, the clustering quality for both methods seems getting slightly better. With respect to number of walks U, clustering quality is not as sensitive as over other parameters.

Further, for better visualization we adopt t-SNE [18] on the PPI data (Figure 4) and Wiki POS data (Figure 5) to demonstrate the clustering quality and get a basic idea how well each embedding approach behaves (Since LINE-1st, LINE-2nd, and SDNE are not as well performed under *Calinski-Harabaz* scores, we do not present their t-SNE visualizations for saving some space). As shown, RANE separates nodes in both cases better than Node2Vec and DeepWalk with respect to density as well as separability, which also explains why RANE performs better for node clustering task in general.



(a) # dimension d **(b) walk length L** **(c) # walks per node U** **(d) window width W**

Figure 1: Parameter Sensitivity Study: Micro-$F1$ on PPI data



(a) # dimension d **(b) walk length L** **(c) # walks per node U** **(d) window width W**

Figure 2: Parameter Sensitivity Study: Macro-$F1$ on PPI data

Algorithm	PPI data	Wiki POS data
LINE-1st	26.0379	21.0849
LINE-2nd	26.4460	21.3307
SDNE	74.7855	24.6970
DeepWalk	77.3388	28.5728
Node2Vec	77.2953	28.9533
RANE	**77.5942**	**29.233**

Table 5: Calinski-Harabaz score

5.4 Scalability

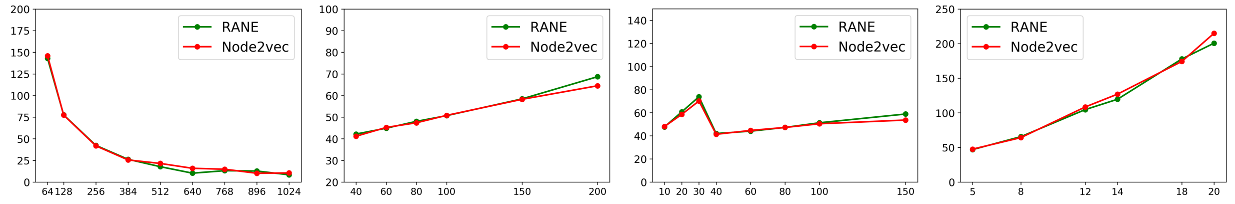
In real applications, networks are typically in large-scale. A practical network like "Youtube users", "e-commerce customers" could easily reach to *hundreds of millions* in nodes and *tens of billions* in edges. Loading the whole network into memory for *walk* generation could be resource costly. This issue can be solved via parallelized computing. The experiment throughout this work is done by a quad-core Intel Core i7 2.6GHz, 16Gb memory Linux machine. Algorithm by itself, RANE scales very well and is good for large-scale network representation learning since (1) negative sampling in walk generation is efficient in terms of the space complexity $O(N)$ as well as constant time complexity; (2) the stochastic gradient ascent during optimization also scales well with the network size.

6 CONCLUSION

In this paper, we presented our relation-aware network embedding learning algorithm. In contrast to the traditional network embedding methods, our approach is able to utilize rich feature vectors of nodes and capture both explicit and implicit relations in the network. We conducted comprehensive experiments on three classic prediction tasks including link prediction, multi-label classification, and node clustering on five public datasets. Our experimental results demonstrate that our RANE approach can outperform other well-known network embedding models.

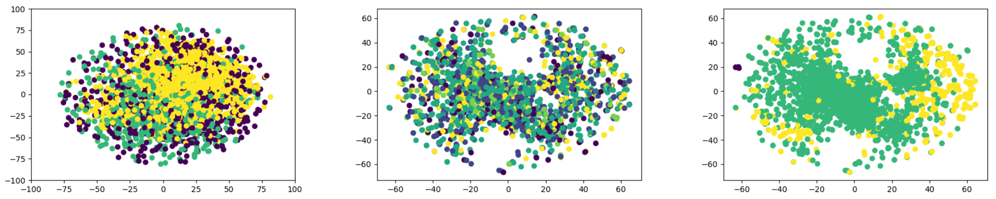
REFERENCES

[1] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478.
[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.
[3] Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3, 1 (1974), 1–27.
[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations.. In *AAAI*. 1145–1152.
[5] Ting Chen and Yizhou Sun. 2017. Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 295–304.
[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[7] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.
[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.
[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 315–323.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
[12] Yuhong Guo and Suicheng Gu. 2011. Multi-label classification using conditional dependency networks. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22. 1300.
[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. Unsupervised learning. In *The elements of statistical learning*. Springer, 485–585.



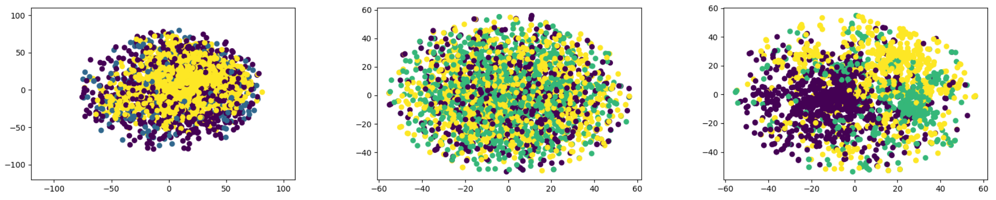
(a) # dimension d **(b) walk length L** **(c) # walks per node U** **(d) window width W**

Figure 3: Parameter Sensitivity Study: Calinski-Harabaz score on PPI data



(a) DeepWalk **(b) Node2Vec** **(c) RANE**

Figure 4: t-SNE Clustering visualization on PPI data



(a) DeepWalk **(b) Node2Vec** **(c) RANE**

Figure 5: t-SNE Clustering visualization Wiki POS data

[14] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[16] Jure Leskovec and Andrej Krevl. 2015. {SNAP Datasets}:{Stanford} Large Network Dataset Collection. (2015).

[17] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-Dimensional Network Embedding with Hierarchical Structure. (2018).

[18] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[19] Matt Mahoney. 2011. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html* (2011).

[20] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[23] Nagarajan Natarajan and Inderjit S Dhillon. 2014. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* 30, 12 (2014), i60–i68.

[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[25] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[26] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. *Nucleic acids research* 34, suppl_1 (2006), D535–D539.

[27] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[28] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[29] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding.. In *AAAI*. 203–209.

[30] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.

[31] Reza Zafarani and Huan Liu. 2009. Social computing data repository at ASU, 2009. *URL http://socialcomputing. asu. edu* (2009).