CONE: Community Oriented Network Embedding

Hanqing Lu *, Carl Yang #, Kevin Chen-Chuan Chang #

University of Illinois, Urbana Champaign
201 N Goodwin Ave, Urbana, Illinois 61801, USA
{jiyang3, kcchang}@illinois.edu

* *Carnegie Mellon University*
5000 Forbes Ave, Pittsburgh, Pennsylvania 15213, USA
hanqinglu@cmu.edu

***Abstract*—Detecting communities has long been popular in the research on networks. It is usually modeled as an unsupervised clustering problem on graphs, based on heuristic assumptions about community characteristics, such as edge density and node homogeneity. In this work, we doubt the universality of these widely adopted assumptions and compare human labeled communities with machine predicted ones obtained via typical mainstream algorithms. Based on supportive results, we argue that communities are defined by their underlying social patterns and unsupervised learning algorithms based on heuristics is incapable of capturing their various forms. Therefore, we propose to inject supervision into community detection through Community Oriented Network Embedding (CONE), which leverages limited ground-truth communities as examples to learn an embedding model aware of the underlying social patterns. Specifically, a deep architecture is developed by combining recurrent neural networks with random-walks on graphs towards capturing social patterns directed by ground-truth communities. Generic clustering algorithms on the embeddings of other nodes produced by the learned model then effectively reveals more communities that share similar social patterns with the ground-truth ones.**

I. Introduction

One of the most popular topics in network research is to identify communities. On the one hand, as networks are growing larger than ever before, it is efficient and necessary to look into smaller sub-networks, which consist of specific groups of interacting objects (*i.e.*, nodes) and their links (*i.e.*, edges). On the other hand, the knowledge of community structures allows us to better understand the status of an object within a group and the relations between it and its peers, so as to enable multiple benefits including the discovery of functionally related objects [1], the study of interactions between modules [2], the inference of missing contents [3], the prediction of unobserved connections [4] and so on.

Many algorithms aim to solve this problem [5], [6], [7], [8], [9], [10], [11]. However, they are all formulated based on the heuristic assumptions of edge density and node homogeneity, *i.e.*, communities are constructed by densely connected nodes and nodes within one community are all homogeneous in some ways. Most surprisingly, even the ground-truth community labels used for evaluation in many standard datasets are generated by machines rather than humans based on these two assumptions [12]. In this work, we doubt the universality of these widely trusted assumptions.

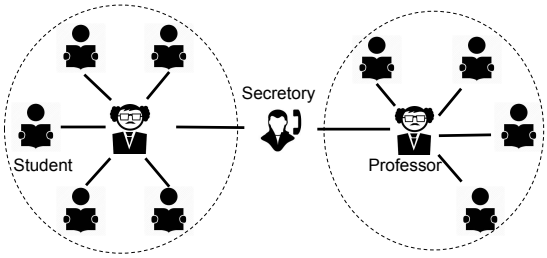


Fig. 1. **Toy example of communities in a CS department**

To see how these two assumptions do not necessarily hold true, consider the scenario in Figure 1, where two groups of students and professors are circled by dashed lines. Each of the two is naturally a valid community as everyone is affiliated to the same research group. However, they are not always densely connected, as it is possible that students work on their individual projects and hardly meet each other. The community is not homogeneous in a specific way either. Students may have similar age range, salary range and come from the same hometown, but these might be quite different from those of the professors. Therefore, unsupervised community detection algorithms based on edge density and node homogeneity can hardly detect communities like these.

Although neither of the density and homogeneity characteristics is prominent in the communities in Figure 1, we observe that there are indeed some interesting *social patterns*. For example, a pattern of *star* is clearly formed within the communities, and the secretory obviously acts as a *bridge* between communities. Those patterns are characterized by both link structures and user contents. For instance, the professor as a star center is distinctive in ages from the students around but may possess similar other contents such as research interest and department. The professor is also densely connected with the students around, but the students may not directly work with each other. On the other hand, the secretory as a bridge shares similar contents like department with all others, while also connects to many of them. Nevertheless, she/he belongs to neither of the two groups. Clearly, communities are defined and should be detected by the underlying social patterns.

Intuitively, a social pattern as we want to leverage should be formed by users with certain contents into specific local structures, and it should appear often in communities. However, while we can think of some intuitive ones like those in

Figure 1, the actual social patterns should be *fuzzy, complex and varying across networks*. It is impractical to enumerate all possible ones and design an unsupervised algorithm to detect communities of all patterns. Moreover, traditional ways of finding and matching patterns on even small networks are notoriously time-consuming and almost impossible on real-world large social networks [13]. To the best of our knowledge, there exists no previous work that aims to leverage such fuzz, complex and varying patterns on networks.

In this work, we propose to inject weak supervision into community detection by learning an embedding model that automatically captures fuzzy social patterns. Instead of taking pre-defined assumptions about community characteristics, it is more reliable to learn from the data what the underlying social patterns look like. Specifically, given a network, we propose to leverage a few labeled communities as examples, and automatically explore and memorize their important social patterns. The model can then easily compute the embeddings on other parts of the network or other similar networks, which can then be leveraged for the detection of unlabeled communities there. For example, in Figure 1, it is intuitive to leverage the star pattern learned from one community to detect the other, and the patterns learned from research groups in the CS department can be well leveraged to detect research groups in other departments, shools, *etc*.

While network embedding has attracted intense research attention recently [14], [15], [16], [17], [18], a supervised embedding directed by social patterns is non-trivial and has never been explored. The task is challenging due to the lack of an efficient way to explore and combine complex user contents, network structures and community labels into a unified learning framework that effectively captures social patterns. In this work, we develop an end-to-end deep architecture of Community Oriented Network Embedding (CONE). The key advantages of CONE over some existing network embedding techniques are as follows:

1) *Content expressive*: to deeply explore and understand high-dimensional noisy contents as signatures of social patterns, we start from content embedding, by decomposing it into multiple nonlinear layers of recurrent neural networks (RNN).
2) *Structure aware*: to leverage links and capture social patterns with local structures, we design a network regularization layer, which essentially generalizes the embedding from single users to local structures according to high-order random-walk transitions on the graph.
3) *Community oriented*: to exploit example communities, we connect a softmax supervision layer into an end-to-end framework to directly require users within the same communities and thus forming some special social patterns to be close in the embedded space. In this way, the supervision can be properly back propagated to the content embedding layer via the network regularization layer, which explores their mutual reinforcement and reliably captures prominent social patterns.
4) *Out-of-sample*: CONE directly learns an embedding model, rather than the embeddings of a specific set of nodes. Therefore, it is able to handle the out-of-sample problem, which addresses the challenges of limited supervision and dynamic network.

II. Motivating Study

In this section, we show the deficiency of traditional unsupervised community detection algorithms by quantitively examining the density and homogeneity assumptions they adopt and their consequences. On the contrary, we demonstrate that there are indeed some communities that are neither dense nor homogeneous, but they share some underlying social patterns.

We conduct a set of analysis using the Facebook dataset described in [8]. This dataset includes 10 ego-networks, consisting of 193 social circles and 4,039 nodes. 10 ego users have manually identified all the communities to which their friends belong. The average number of social circles in each ego-network is 19, with an average community size of 22 users. We trust these manually labeled social circles and use them as the ground-truth communities.

Let a graph $G = \{V, E\}$ represent a network with a vertex set V and an edge set E. A community can be represented as a sub-graph $C = \{V_C, E_C\}$ with $V_C \subset V$ and $E_C \subset E$. $\mathbf{a}_i$ is the content vector of a vertex $v_i \in V$. We evaluate the density and homogeneity of both ground-truth communities and communities detected by the state-of-the-art algorithms.

Metrics. We use a density metric $D(\cdot)$ and a homogeneity metric $H(\cdot)$ defined as follows:

$$D(C) = \frac{2|E_C|}{|V_C|(|V_C|-1)}. \tag{1}$$

The value of $D(C)$ ranges from 0 (a graph without edges) to 1 (a complete clique). It is a standard measure of the density of a network widely used in related literature [12].

$$H(C) = \frac{\sum\limits_{v_i, v_j \in V_C, v_i \neq v_j} \sigma(\frac{\mathbf{a}_i^T \mathbf{a}_j}{avg(\mathbf{a}_i^T \mathbf{a}_j)})}{|V_C|(|V_C|-1)}, \tag{2}$$

where $avg(\mathbf{a}_i^T \mathbf{a}_j)$ is the average of content similarity among all pairs of nodes in the whole network G, and $\sigma(t) = \frac{1-e^{-t}}{1+e^{-t}}$ is the adjusted half sigmoid function in the range of $[0, 1]$. The value of $H(C)$ also ranges from 0 (none of the nodes share the same contents) to 1 (all nodes share the same contents), while the value increases rapidly as the content similarity in C is small compared with the average content similarity in the whole network G.

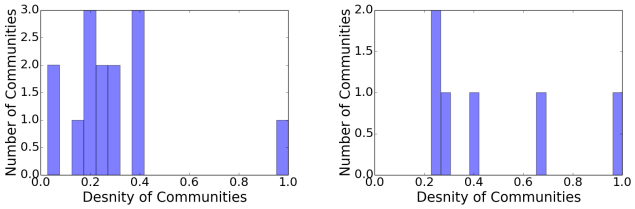
Compared algorithms. We study three mainstream community detection algorithms within the state-of-the-art: MinCut [7] (based on network modularity), CESNA [10] (based on probabilistic generative model) and InfoMaps [19] (based on information theory).

Density and homogeneity analysis.

Assumption 1: nodes within the same community are densely connected.

Conflicting evidence: In Figure 2, we present $D(C)$ computed on the ground-truth communities as well as the communities detected by the state-of-the-art algorithms in ego-network 686 as an example. It can be observed that ground-truth communities do not always have a high density. In fact, it is possible that there are more low-density communities than high-density ones as in (a). The evidence directly contradicts with Assumption 1.

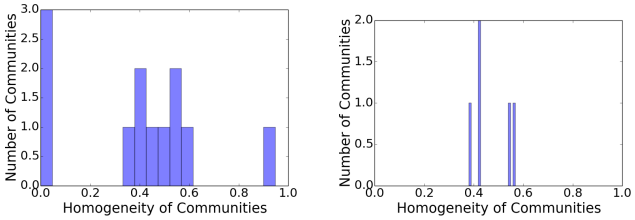
Based on this inaccurate assumption, traditional unsupervised community detection algorithms tend to detect communities with high density. As an example, the distributions of $D(C)$ computed on the detected communities in the same ego-network are presented in Figure 2(b)-(d). It can be observed that most of the detected communities have a high density ($D(C) > 0.5$), which is inconsistent with the density distribution of the ground-truth communities. It suggests that the inaccurate assumption of edge density indeed leads to unreliable community detection results.



(a) Ground-truth (b) MinCut

(c) CESNA (d) InfoMap

Fig. 2. **Community density in ego-net 686.**



(a) Ground-truth (b) MinCut

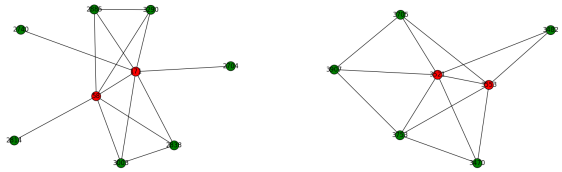
(c) CESNA (d) InfoMaps

Fig. 3. **Community homogeneity in ego-net 698.**

Assumption 2: nodes within the same community are homogeneous in some ways.

Conflicting evidence: As an example, the distribution of $H(C)$ computed on the ground-truth communities in ego-network 698 is contradictory to Assumption 2. As shown in Figure 3(a), few communities are indeed homogeneous (*e.g.*, the one with homogeneity higher than 0.8), while the majority are much less homogeneous. Given $\sigma(1) \simeq 0.46$, the results indicate that most communities have a homogeneity similar to that of the whole network.

We conduct the same homogeneity analysis on communities detected by the same group of algorithms. As can be seen in Figure 3(c)-(d), since CESNA and InfoMaps assume node homogeneity in communities, they do detect communities of slightly higher homogeneity, which diverge from the ground-truth ones. MinCut does not assume node homogeneity.



(a) Cmt 1 in ego-net 1684 (b) Cmt 15 in ego-net 3437

Fig. 4. **Examples of real communities with social patterns.**

Real community social pattern analysis. After showing that nodes within the same communities are not always densely connected or homogeneous in some ways, we look into the structures of ground-truth communities to find evidence for the existence of social patterns, so as to further motivate our novel approach of community detection.

In Figure 4, among many patterns and examples, we show two communities that clearly share a *co-star* pattern, where almost all members connect to two center nodes, while they do not densely connect within themselves. Moreover, the center nodes in both communities indeed have some special contents such as a certain degree in the college or job in the company, which are not shared by other non-center members. (We removed a few isolated noise nodes for clear visualization.)

III. CONE

A. Overall Framework

The analysis in Sec. II clearly demonstrates the deficiency of existing unsupervised community detection algorithms, which is a consequence of their falsifiable assumptions of edge density and node homogeneity. As we motivated in Sec. I, the existence of fuzzy, complex and varying social patterns underlying communities further indicates an urge for developing a novel community detection algorithm that aims at leveraging such patterns.

In this work, rather than enumerating and evaluating all important social patterns, we propose to automatically capture them from labeled communities through weak supervision. Instead of exhaustive graph matching, we model the problem

as representation learning. Specifically, we do not care about the exact shapes of the patterns, but we intuitively require the users within the same patterns to be close in an embedded space. We formulate our framework as follows.

We are given a network modeled by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{C}\}$, where $\mathcal{V}$ is the set of n users, $\mathcal{E}$ is the set of observed links among $\mathcal{V}$. $\mathcal{A}$ is the set of observed user contents on $\mathcal{V}$, where $\mathbf{a_i}$ is the set of contents on user v_i. $\mathcal{C}$ is a set of ground-truth community memberships, where $\mathbf{c}_k$ is the set of users in community c_k. For learning the model, only a small number of ground-truth communities are required as examples, while others are used for evaluation.

To detect more communities, we firstly learn an embedding model that captures social patterns in $\mathcal{G}$ based on $\mathcal{A}$, $\mathcal{E}$ and $\mathcal{C}$. The model should effectively explore various social patterns and transform each user into a p-dimensional vector. In the embedded space, users forming some important social patterns and thus within each example community are close. More importantly, the model should be able to leverage the social patterns and produce embeddings on all users in $\mathcal{V}$ in a similar way. Therefore, new communities can then be detected through generic clustering algorithms such as k-means on the p-dimensional features. Overlapping communities can also be discovered by algorithms like MOC [20].

Figure 5 illustrates the overall architecture of our CONE model. We take the input of $\mathcal{A}$ for all users $\mathcal{V}$ and compute the content embeddings $\mathbf{H}$, which is then regularized by network structures $\mathcal{E}$ to yield the community oriented embeddings $\mathbf{S}$. For representation learning, $\mathbf{S}$ is then used to generate community predictions $\mathbf{Y}$, which is supervised by the ground-truth community labels $\mathbf{L}$ derived from the example communities in $\mathcal{C}$. $\mathbf{L}$ can be either point-wise, where $l_{ik} = 1$ means v_i is a member of community c_k, or pair-wise, where $l_{ij} = 1$ means v_i and v_j are in the same community. To detect more communities, we take $\mathbf{S}$ as the actual output of CONE and apply generic clustering algorithms like k-means on it.

To ensure a desirable representation, ideally we should require users within the same example communities indicated by $\mathbf{L}$, and thus forming some specific social patterns, to be close in the embedded space. That is, we need to compute the pair-wise loss among all users and our overall high-level objective function should look like the following,

$$\mathcal{J} = \Phi(\mathbf{Y}, \mathbf{L}) = \sum_{(v_i, v_j), i \neq j} \phi(\hat{y}_{ij}, l_{ij}). \tag{3}$$

where Φ is a loss function, $\mathbf{L}$ is the pair-wise community labels and $\mathbf{Y}$ is the pair-wise prediction.

In what follows, we further explain the reasons for our model architecture and how it works in details.

B. Deep Architecture

While it is intuitive to automatically learn an embedding model that captures important social patterns, the task is non-trivial and posts some unique challenges:

- Explore high-dimensional noisy user contents.
- Leverage complex links and local structures.
- Exploit limited supervision of example communities.

To deal with all challenges above, we develop a deep architecture of CONE, which inherently combines RNN with random-walks in an end-to-end supervised learning framework.

RNN: deeply explore and understand user contents as signatures of social patterns. Existing network embedding algorithms are insufficient in exploring contents for capturing social patterns. They usually focus on preserving the link structures among users [14], [16], [15], and incorporate user contents as augmented attribute nodes [21], text feature matrices [17] or bag-of-word vectors [18]. Thus, the deep semantics within user contents are not fully explored. In our situation, contents are so important that they often become the signatures of social patterns. For example, in a football fans' club, a popular player identified by the contents of his tweets is likely to be the center, surrounded by groups of fans identified by their semantically different tweet contents.

However, exploring and understanding user contents in social networks is a non-trivial task, since they can be complex, noisy and high-dimensional. *E.g.*, in text-rich networks like Twitter, contents can be a list of recent tweets; in tag-rich networks like Flickr, they can be a list of frequently used hashtags in a timely order; in networks with explicit attributes like Facebook and LinkedIn, they can be a set of categorical variables like *Birthday, School, Occupation* with lots of noisy and missing data. The situation reminds us of the task in natural language processing (NLP) of understanding noisy text sequences, where the semantics is usually hidden in ordered tokens of variable lengths.

In this work, instead of starting from the links, we start from deeply modeling user contents. To this end, we employ RNN from NLP, which has been proven advantageous over various other methods in understanding text-like sequences, due to its supreme expressiveness within the neural networks to explore, understand and memorize important semantic patterns. The deep learning framework of RNN also provides us with the flexibility to modify the neural network architectures in order to leverage other information like network structures and community supervision. To the best of our knowledge, this is the first work that models user contents as raw texts and successfully applies RNN to network embedding.

To leverage RNN, given a user v_i's textual contents such as a list of recent tweets, we can concatenate them into a single sequence $\mathbf{s}_i$, each element of which is the index of the corresponding word after stemming and stop word removing; given v_i's categorical contents $\mathbf{a}_i$ such as *education* and *work* on Facebook, we transform it into a sequence $\mathbf{s}_i = \{j | a_{ij} = 1\}$. The sequence $\mathbf{s}_i$ is then used as the input of RNN. As mentioned in [22], simple RNN would be difficult to train due to the resulting long term dependencies. Therefore, we use the long-short term memory (LSTM) cells [23] instead to embed $\mathcal{A}$. The architecture and implementation of LSTM can be found in the public website[1].

Upon each input $\mathbf{s}_i$, there will be one output from the LSTM cell as a semantic embedding of user contents. To

[1] http://deeplearning.net/tutorial/lstm.html

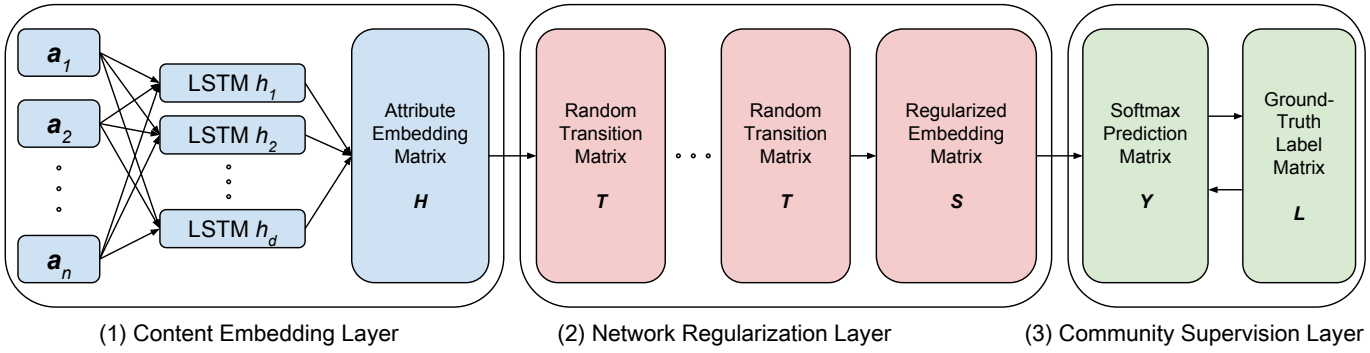


Fig. 5. The end-to-end deep architecture of Community Oriented Network Embedding (CONE).

further improve model expressiveness, we use d LSTM cells to output d embedding vectors $\{\mathbf{h}_i^1, \mathbf{h}_i^2, \ldots, \mathbf{h}_i^d\}$, and apply mean pooling on them, which is commonly used to integrate features and reduce deviation. So we have

$$\mathbf{h}_i = h(\mathbf{s}_i) = \frac{1}{d}\sum_{j=1}^{d} \mathbf{h}_i^j, \tag{4}$$

where $h(\cdot)$ denotes the overall deep content embedding function. Supposing the embedding size of each LSTM cell is p, we get a content embedding matrix $\mathbf{H} \in \mathbb{R}^{p\times n}$ to represent the embeddings of $\mathcal{V}$, where the i-th column of $\mathbf{H}$ equals to $\mathbf{h}_i$.

Note that, while RNN is especially useful for exploring complex text-like sequences, for simple numerical or categorical contents like user attributes, it also makes sense to use simpler models like feedforward neural networks, which can be incorporated into our end-to-end framework in the same way. We will also show the performance of such basic neural networks in our experiments.

Random-walk: leverage links and capture social patterns with local structures. The objectives of existing network embedding algorithms are not appropriate for leveraging network structures to capture social patterns. They usually model network structures by sampling a set of paths from the networks and applying a Skipgram-based model [24] to uniformly require the embeddings of nodes that share similar graph context to be similar [14], [15], [17], [18]. To capture social patterns, we want the user embedding to be guided by community supervision. Users having similar network context do not necessarily belong to the same communities, and thus should not be required to have similar embeddings without discrimination. Besides, by sampling the networks into paths, structural information beyond paths is not efficiently leveraged. In our situation, the shapes of local network structures are extremely important, since they may well indicate the existence of specific social patterns, like the star and co-star shapes as we mentioned in Sec. I and II.

In this work, we directly leverage local structures to regularize the supervision of example communities. To this end, we armor our RNN-based content embedding with random-walk-based network regularization. It efficiently generalizes the embeddings on single users to their ambient local structures, allowing RNN to explore content embedding under the regularization of network local structures and the guidance of example communities.

Consider the RNN-based content embedding. Due to Eq. 4, if $\mathbf{a}_i$ is similar to $\mathbf{a}_j$, then it is likely that the embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$ are also close. However, this may not be ideal, because similar users can well form different communities. The key question is, are they also on the same local structure?

To account for this, we insert a random-walk-based network regularization layer, which recomputes the embedding of each user *w.r.t.* her neighbors, according to their distances measured by random-walks of k steps, *i.e.*, $\mathbf{s}_i = \sum_{j=1}^{n} t_{ji}^k \mathbf{h}_j$, where t_{ji}^k is the k-step random-walk transition probability from v_j to v_i. In matrix form, a more compact formula is $\mathbf{S} = \mathbf{H}\mathbf{T}^k$. $t_{ij} = w_{ij}/d_i$, where w_{ij} is the binary or real-valued weights on edge e_{ij} and $d_i = \sum_j w_{ij}$.

Intuitively, each v_i 'collects' the embeddings transmitted from its local neighbors. Compared with $\mathbf{h}_i$, $\mathbf{s}_i$ encodes the local structure around v_i, rather than just the semantic information from $\mathbf{a}_i$. As a consequence, only users with similar contents as well as local structures will be embedded as close.

Moreover, consider the loss in Eq. 3 brought by the supervision of example communities. Without network regularization, the loss on content embeddings can be formulated as

$$\mathcal{J}_H = \sum_{(v_i,v_j),i\neq j} \phi(\hat{y}(h(\mathbf{a}_i), h(\mathbf{a}_j)), l_{ij}). \tag{5}$$

Under $\mathcal{J}_H$, *e.g.*, if there are some professors and students in the same example communities, all professors and students will be required to get similar content embeddings, which should not be the case. Instead, we apply supervision on the regularized embeddings as

$$\mathcal{J}_S = \sum_{(v_i,v_j),i\neq j} \phi(\hat{y}(\mathbf{s}_i, \mathbf{s}_j), l_{ij}). \tag{6}$$

As a consequence, only professors and students connected in certain local structures indicated by supervision are embedded as close.

End-to-end learning: exploit community supervision and the mutual reinforcement between contents and links. As we discussed about Eq. 3 before, to directly meet our goal of embedding users in the same example communities to be close, we need to construct a pair-wise loss on each pair of users in the same communities. However, pair-wise loss functions can not be built into our end-to-end embedding

framework and permit efficient training of the neural networks. Moreover, converting the community memberships to pair-wise 0-1 labels will lead to significant information loss, especially for overlapping communities.

To overcome these difficulties, we find the point-wise softmax prediction with cross entropy loss as a suitable substitute to the exact pair-wise loss [25]. While having a different objective, softmax with cross entropy basically ensures that instances predicted with the same label are close in a space that can be viewed as a linear projection of the original embedding space. Therefore, it can be viewed as achieving a similar goal as the direct pair-wise loss. Moreover, softmax is commonly used in end-to-end deep learning frameworks to predict multiple labels, and the cross entropy loss can be efficiently back propagated through neural networks. By allowing the training on multi-label predictions, it is also able to leverage overlapping example communities and distinguish among different communities.

Therefore, we incorporate a softmax supervision layer to form our end-to-end deep learning framework, where community supervision can be exploited as a guidance for exploring social patterns, and the mutual reinforcement between user contents and link structures can be leveraged. Our practical objective function based on cross entropy loss is as follows:

$$\hat{y}_{ik} = \frac{\exp(\mathbf{w}_k^T \mathbf{s}_i)}{\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^T \mathbf{s}_i)}, \quad \mathcal{J} = \sum_{i=1}^{n} \sum_{k=1}^{K} l_{ik} log(\hat{y}_{ik}), \quad (7)$$

where $\mathbf{w}$'s are the model parameters in the softmax layer and K is the total number of example communities. $\mathbf{Y}$ and $\mathbf{L}$ encode the predicted and ground-truth community labels, respectively, where $\hat{y}_{ik} = 1$ means v_i is predicted to be a member of community c_k, and $\mathbf{L}$ is the point-wise community labels.

To optimize Eq. 7, we employ stochastic gradient descent (SGD) with the diagonal variant of AdaGrad from [26]. At the t-th step, the parameters Θ is updated by:

$$\Theta_t \leftarrow \Theta_{t-1} - \frac{\rho}{\sqrt{\sum_{i=1}^{t} g_i^2}} g_t, \quad (8)$$

where ρ is the initial learning rate and g_t is the sub-gradient at time t. Since our network regularization layer is implemented as a matrix multiplication, the gradients can be efficiently back-propagated to the content embedding layer, and the overall embedding framework is end-to-end.

When detecting communities in a network, embeddings of users produced by the CONE model learned on labeled data are fed into generic clustering algorithms like k-means. We apply cross-validation to automatically choose the optimal number of communities as done in [10].

The efficiency of CONE: CONE is an end-to-end deep learning framework implemented using TensorFlow[2]. Only minimum setup is required to run it efficiently on GPU. We will make the code available upon acceptance of the work.

[2]https://www.tensorflow.org/

IV. Experiments

In this section, we evaluate CONE for community detection with extensive experiments on 3 real-world networks.

A. Experimental Settings

Datasets. We use three real-world network datasets. The first is the Facebook dataset we used in Sec. II for data analysis, which consists of 10 ego-networks with community labels explicitly provided by the ego users [8]. The contents in this dataset are well-defined user profiles such as *education, work* and *location*, and links are undirected friendships among users. The second is a Flickr dataset collected by [27]. The community labels are generated from the groups joined by users. The contents are the tags aggregated on users' posted photos and the links are undirected friendships. The third is a Twitter dataset also collected by [8], which consists of 973 ego-networks. The community labels are generated from friend circles (or lists), *i.e.*, friends put into the same list by the ego user are regarded as within one community. The contents are the hashtags and popular account mentions within users' recent tweets and the links are directed followings. Detailed statistics of the three datasets we use are shown in Table 1.

Dataset	#nodes	#comms	#links	#attrs
Facebook	4,039	192	88,234	634
Twitter	81,306	837	1,768,149	12,274
Flickr	35,313	200	3,017,530	77,263

TABLE I
STATISTICS OF 3 REAL-WORLD NETWORK DATASETS.

Compared algorithms. We compare with two groups of algorithms from the state-of-the-art to comprehensively evaluate the performance of CONE.

Community detection algorithms. Some algorithms are based on the edge density assumption alone, while others also assume node homogeneity. We compare with this group of algorithms to show the advantage of abandoning these inaccurate assumptions and leveraging the supervision of example communities.

- **MinCut** [7]: a classic community detection algorithm based on modularity.
- **BigClam** [9]: an advanced community detection algorithm solely based on network structure.
- **Circles** [8]: a generative model of edges *w.r.t.* attribute similarity to detect communities.
- **CESNA** [10]: a generative model of edges and attributes to detect communities.
- **SGM** [28]: a semi-supervised framework incorporating individual labels and pair-wise constraints.

Network embedding algorithms. While we find it intuitive to model the problem as representation learning, we compare with the state-of-the-art network embedding algorithms to show that CONE is advantageous in capturing social patterns. The embeddings learned by all algorithms are fed into the same k-means clustering algorithm as CONE to produce community detection results.

- **DeepWalk** [15]: an embedding algorithm based on truncated random walks that only considers network structures.
- **node2vec** [14]: an embedding algorithm based on $2nd$ order random walks that only considers network structures.
- **TADW** [17]: an embedding algorithm that generalizes DeepWalk to consider both node attributes and network structures by tensor factorization.
- **PTE** [21]: an embedding algorithm that generalizes LINE [16] to consider node attributes, network structures and class labels by graph augmentation.
- **Planetoid** [18]: an embedding algorithm that extends DeepWalk to consider node features, network structures and class labels by jointly predicting labels and contexts.

The number of communities to detect is tuned via standard 5-fold cross validation for all algorithms. The implementations of compared algorithms are all provided by the original authors.

Metrics. Two widely used metrics for evaluating community detection results are used in our experiments. For a detected community c_i^* and a ground-truth community c_i, the *F1 similarity* and *Jaccard similarity* are defined as

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall},$$

$$Jaccard = \frac{|c_i \cap c_i^*|}{|c_i \cup c_i^*|},$$

where $precision = \frac{c_i \cap c_i^*}{|c_i^*|}$, $recall = \frac{c_i \cap c_i^*}{|c_i|}$.

For a set of ground-truth communities $\{c_i^*\}_{i=1}^M$ and a set of detected communities $\{c_i\}_{i=1}^N$, we compute the score as

$$\frac{1}{2|C|} \cdot \sum_{c_i \in C} \max_{c_i^* \in C^*} eval(c_i, c_i^*) + \frac{1}{2|C^*|} \cdot \sum_{c_i^* \in C^*} \max_{c_i \in C} eval(c_i, c_i^*),$$

where $eval(c_i, c_i^*)$ can be either replaced by $F1$ or $Jaccard$.

B. Performance Comparison with Baselines

We quantitively evaluate CONE against all baselines on community detection. We randomly split the labeled communities into training and testing sets. We use 10% of the labeled communities as examples to learn the models for CONE, SGM, PTE and Planetoid. All compared algorithms are then evaluated on the rest 90% labeled communities. To observe significant difference in performance, we split the training and testing sets 10 times and conduct statistical t-tests with p-value 0.01.

Table II shows the average F1 and Jaccard scores evaluated on all compared algorithms over the same 10 random splits of training and testing sets. The results all passed our t-tests. The parameters of baselines are all set to the default values as suggested in the original works. Some numbers are a bit different with those in the original work, because we directly use all node contents as input and only evaluate on the testing set. For CONE, we intuitively use two random transitions in the regularization layer to stress network locality and leverage the link structures in close neighborhoods.

CONE reaches around 30% relative improvements on the Facebook dataset and more than 10% improvements on the other two datasets, compared with the second-runner among the 10 baselines on both F1 and Jaccard scores, while they have varying performance. This indicates the robustness and general advantages of our proposed approach.

Algorithm	F1 Score			Jaccard Score		
	FB	Flickr	Twitter	FB	Flickr	Twitter
MinCut	0.265	0.059	0.114	0.181	0.031	0.081
BigClam	0.284	0.076	0.121	0.198	0.041	0.084
Circles	0.267	0.042	0.102	0.185	0.029	0.069
CESNA	*0.298*	0.076	0.126	*0.242*	0.042	0.084
SGM	0.281	0.077	0.177	0.202	0.043	0.116
DeepWalk	0.211	0.071	0.136	0.135	0.039	0.094
node2vec	0.245	0.074	0.119	0.145	0.041	0.084
TADW	0.269	0.073	0.146	0.193	0.036	0.105
PTE	0.221	0.069	0.147	0.142	0.038	0.110
Planetoid	0.272	*0.079*	*0.188*	0.231	*0.045*	*0.121*
CONE	**0.397**	**0.096**	**0.206**	**0.313**	**0.052**	**0.149**

TABLE II
PERFORMANCE COMPARISON ON 3 DATASETS.

Taking a closer look, we observe that the scores on the Facebook dataset look much better than those on the other two. This is due to the high quality of both contents and links in the dataset. Since the Facebook dataset is the only one that has explicit human labels of communities, the large performance improvement on it indicates the advantage of CONE in leveraging user provided community examples to understand specific social patterns.

The scores of network embedding algorithms are lower than traditional community detection algorithms on the Facebook dataset, probably because traditional models work better on few and clean contents. However, as the contents become high-dimensional and noisy like in the Flickr and Twitter datasets, network embedding algorithms excel. This indicates the advantage of leveraging embeddings for traditional social network tasks and further proves the efficacy of CONE in dealing with complex user contents.

All experiments are done on a local PC with two 2.5 GHz Intel i7 processors and 8GB memory. The runtime of CONE is comparable to all baselines, while it is trivial to run more efficiently on GPU.

C. Model Selection

We comprehensively evaluate the performance of CONE with different amounts of supervision and varying neural architecture.

Impact of supervision. We study the impact of supervision on the performance of CONE by varying the training and testing set portions. Figure 7 shows the results. CONE's advantage of leveraging community examples is significant on all three datasets. As can be observed, CONE efficiently leverages supervision by learning from very small amounts of labeled communities, and the performance converges rapidly as the training set portion reaches around 11% on the Facabook and Flickr datasets. On datasets with large amounts of labeled communities such as Twitter, around 6% of them are enough to learn a good CONE model.

Note that the other three supervised algorithms do not leverage community labels effectively, basically because they are not designed to capture social patterns and leverage them for out-of-sample community detection.

Impact of architecture. We study the impact of parameters inside CONE by varying the number of random-walk transitions and the size of embeddings. We also substitute the RNN in the content embedding layer with basic fully connected feed forward neural networks to demonstrate the effectiveness of using RNN to deeply explore the high-dimensional noisy contents. Among the basic neural networks we experimented on with varying layers (from 1 to 4) and embedding sizes (the last layer with sizes of 4, 8, 16 and 32), we show the performance of a three-layer basic neural network (embedding sizes are $64 \rightarrow 32 \rightarrow 16$ from bottom up as a common practice [29]) with ReLU as the activation function, which generally yields the best performance.

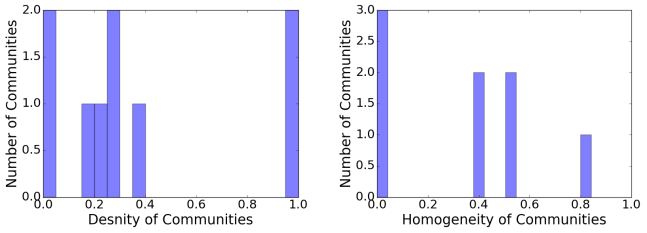
Figure 8 shows the results. The number of random-walk transitions has a large impact on the performance of CONE, especially when the number is small. This demonstrates the utility of our novel network reconfiguration layer. As the number of transitions grows larger and the stationary distribution of random walk is approximated, the performance becomes stable. Note that large numbers of transitions do not necessarily lead to optimal performance, because community structures are often well described within small neighborhoods.

The number of embedding dimensions does not have a significant impact on the performance with RNN as the content embedding layer, indicating the robustness of CONE inherited from RNN. However, the performance is significantly worse without RNN as the content embedding layer, which justifies our motivation of leveraging RNN to effectively explore the noisy contents in high-dimensional spaces. The runtime of RNN is also significantly shorter than basic neural networks, especially with more layers and larger embedding sizes.

D. Case Study

To understand the advantage of CONE, we conduct the same density and homogeneity analysis on our detected communities and present the results on the same ego-networks we showed in Sec. II. Comparing Figure 6(a) and 6(b) with Figure 2(a) and 3(a), we observe that CONE indeed finds communities that are similar in distributions of density and homogeneity as the ground-truth, which indicates its effectiveness in capturing the social patterns that essentially define user-described communities.

We also looked into the structures of communities found by different algorithms. Still taking the example of the clear co-star pattern, when we took the ground-truth communities in ego-network 1684 as supervision, CONE indeed found a total of 6 co-star communities on other parts of the dataset, specifically, in ego-networks 0, 107, 348 and 3437. We did not observe communities of such a co-star pattern detected by any other algorithms, from all communities they detected through a single execution on the same training and testing sets.



(a) Density in ego-net 686 (b) Homogeneity in ego-net 698

Fig. 6. **Communities detected by CONE.**

V. Related Work

A. Community detection

Traditional community detection algorithms are mostly unsupervised, based on either of the edge density or node homogeneity assumptions, or both. Therefore, the communities they detect are only constructed by densely connected or homogeneous nodes.

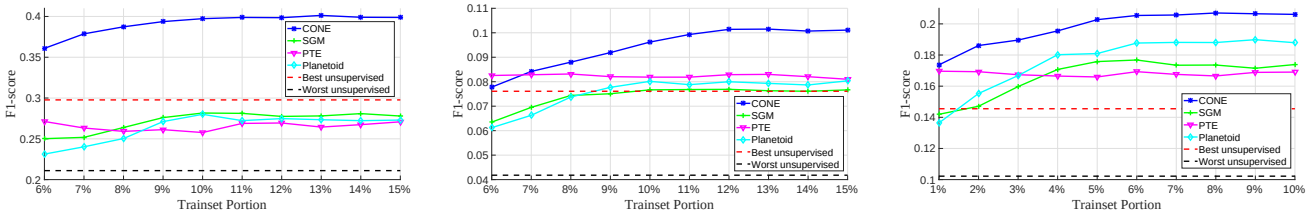
Among them, many are based on probabilistic generative models. For instance, COCOMP [11] extends LDA [30] by taking community assignments as latent variables. It leverages node homogeneity by assuming that each community is a group of people that use a specific distribution of words. BigClam [9] does not make use of node content at all. It leverages edge density by modeling the assignments of communities solely based on existing links. A variation of BigClam called CESNA [10] is later proposed, with additional consideration of binary-valued node attributes. Another method within the state-of-the-art, Circles [8], also leverages the same assumptions with a slightly different model.

Many non-generative algorithms have also been proposed. Among them, some are based on graph metrics such as modularity [7] and maximal clique [31]. They are fundamentally leveraging the edge density assumption. As for node homogeneity, some algorithms firstly augment the graph with content links and then do clustering on the augmented graph [32], [33], while some others attempt to find a partition that yields the minimum encoding cost based on information theory [5], [19]. They do not scale trivially to networks with high-dimensional noisy contents.

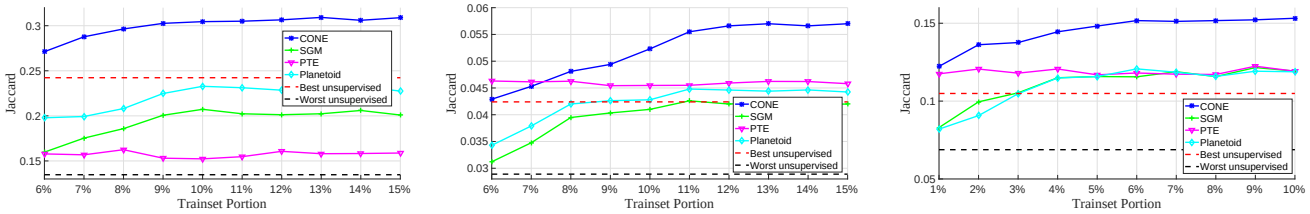
As we leverage whole labeled communities as examples to learn the underlying social patterns, CONE is also different from the few semi-supervised community detection algorithms that leverage individual labels and pair-wise constraints as part of some communities [28].

B. Network embedding

We aim to find node embeddings that are suitable for the task of community detection, under the social pattern assumption. Unlike the techniques that compute embeddings based on node proximities in the network [34], [35], [36], we frame the objective as learning a representation that captures social patterns. Moreover, we aim to learn it under the guidance of communities, instead of the unsupervised ways based on relational data in the networks [37], [38] or structural relationships between concepts [39], [40].

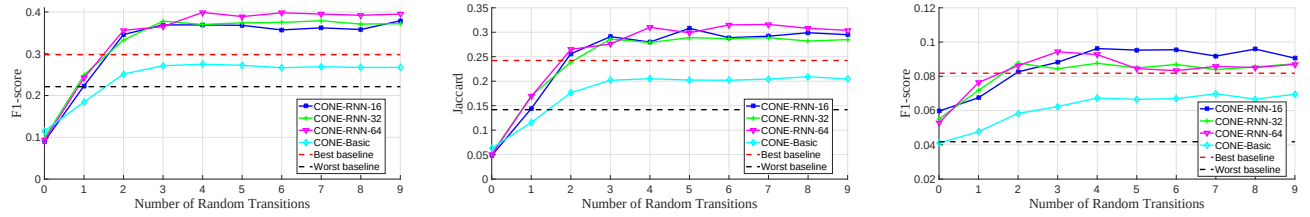


(a) Facebook (b) Flickr (c) Twitter

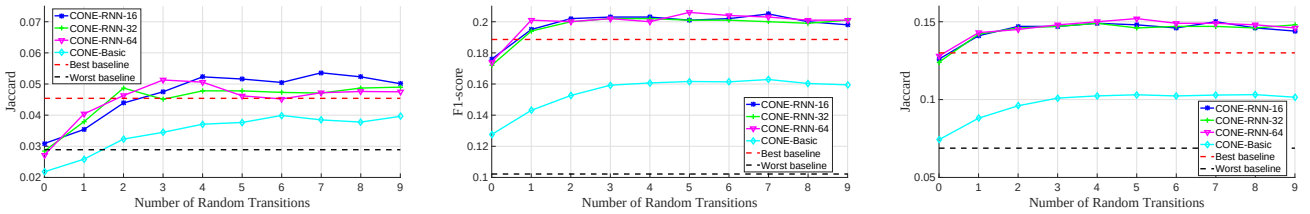


(d) Facebook (e) Flickr (f) Twitter

Fig. 7. **Model selection results with varying training set portion.**



(a) Facebook (b) Flickr (c) Twitter



(d) Facebook (e) Flickr (f) Twitter

Fig. 8. **Model selection results with varying neural architecture.**

As discussed in Sec. III, the most related techniques to ours are the Skipgram-based network embeddings [14], [15], [16]. Recent works [21], [18], [17] have extended such techniques to incorporate node attributes and class labels. Their models using augmented nodes, text feature matrices and bag-of-word vectors are ineffective for deeply exploring high-dimensional noisy contents. Moreover, they all use network context as supervision to uniformly require nodes with similar context to have similar embeddings. Our objective is different, where we only require users within the same communities to be embedded as close. To this end, our supervision is directly guided by community examples, and we leverage network structures only as a regularization. Finally, since they sample networks into paths and predict context based on paths, the techniques are inefficient in capturing local network structures beyond paths. Our regularization efficiently generalizes the embedding of single users to their ambient local structures, which are deterministic rather than approximated by paths.

C. *Node classification*

Traditional supervised learning on networks is essentially node classification [41], [42], [35], [43], which uses node attributes as labels and leverages network structures for tasks like content prediction. Our problem is quite different from node classification. Although we utilize supervision to learn the community oriented features, our final goal is to perform unsupervised detection of unlabeled communities. To be more specific, our predictions are new clusters of nodes, for which no predefined categories or labeled data are available at all. As a consequence, node classification algorithms are not applicable to our problem.

VI. Conclusion

In this paper, we doubt the generality of the two widely trusted assumptions about community characteristics, *i.e.*, edge density and node homogeneity, and we show the deficiency of mainstream algorithms adopting those assumptions through real data analysis. To deal with this deficiency, we propose

to leverage the underlying social patterns that define and detect network communities. We design CONE that effectively explores and captures important social patterns under the guidance of example communities through network embedding. Generic clustering algorithms performed on the embeddings can yield reliable community detection results.

While CONE was originally designed for leveraging supervision for reliable community detection free from falsifiable assumptions, it can be easily applied to many network learning tasks to coherently leverage node contents, link structures and various kinds of supervision or constraints. As we observe from our experiments, CONE is especially advantageous in dealing with high-dimensional noisy contents, such as sequences of hashtags and even raw texts.

REFERENCES

[1] A. P. Streich, M. Frank, D. Basin, and J. M. Buhmann, "Multi-assignment clustering for boolean data," in *ICML*. ACM, 2009, pp. 969–976.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JMLR*, vol. 9, no. Sep, pp. 1981–2014, 2008.

[3] R. Li, C. Wang, and K. C.-C. Chang, "User profiling in an ego network: co-profiling attributes and relationships," in *WWW*. ACM, 2014, pp. 819–830.

[4] J. Chang and D. M. Blei, "Relational topic models for document networks," in *AIStats*, vol. 9, 2009, pp. 81–88.

[5] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos, "Pics: Parameter-free identification of cohesive subgroups in large attributed graphs," in *ICDM*. SIAM, 2012, pp. 439–450.

[6] T. Chakraborty, S. Patranabis, P. Goyal, and A. Mukherjee, "On the formation of circles in co-authorship networks," in *KDD*. ACM, 2015, pp. 109–118.

[7] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[8] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *NIPS*, 2012, pp. 539–547.

[9] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *WSDM*. ACM, 2013, pp. 587–596.

[10] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*. IEEE, 2013, pp. 1151–1156.

[11] W. Zhou, H. Jin, and Y. Liu, "Community discovery and profiling with social messages," in *KDD*. ACM, 2012, pp. 388–396.

[12] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *KIS*, vol. 42, no. 1, pp. 181–213, 2015.

[13] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C.-C. Chang, and X.-L. Li, "Semantic proximity search on graphs with metagraph-based learning," in *ICDE*. IEEE, 2016, pp. 277–288.

[14] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*. ACM, 2016, pp. 855–864.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*. ACM, 2015, pp. 1067–1077.

[17] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *IJCAI*, 2015, pp. 2111–2117.

[18] Z. Yang, W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016.

[19] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[20] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney, "Model-based overlapping clustering," in *KDD*. ACM, 2005, pp. 532–537.

[21] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *KDD*. ACM, 2015, pp. 1165–1174.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[25] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[26] X. Qiu and X. Huang, "Convolutional neural tensor network architecture for community-based question answering." in *IJCAI*, 2015, pp. 1305–1311.

[27] X. Wang, L. Tang, H. Liu, and L. Wang, "Learning with multi-resolution overlapping communities," *KAIS*, 2012.

[28] E. Eaton and R. Mansbach, "A spin-glass model for semi-supervised community detection," in *AAAI*, 2012, pp. 900–906.

[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, no. Jan, pp. 993–1022, 2003.

[31] N. Du, B. Wu, X. Pei, B. Wang, and L. Xu, "Community detection in large-scale social networks," in *WebKDD*. ACM, 2007, pp. 16–25.

[32] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *WWW*. ACM, 2013, pp. 1089–1098.

[33] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: a discriminative approach," in *KDD*. ACM, 2009, pp. 927–936.

[34] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in *KDD*. ACM, 2011, pp. 663–671.

[35] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *KDD*. ACM, 2009, pp. 817–826.

[36] ——, "Leveraging social media networks for classification," *DMKD*, vol. 23, no. 3, pp. 447–478, 2011.

[37] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 37–48.

[38] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: a general framework for dimensionality reduction," *TPAMI*, vol. 29, no. 1, pp. 40–51, 2007.

[39] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *TASLP*, vol. 20, no. 1, pp. 30–42, 2012.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[41] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*. Springer, 2011, pp. 115–148.

[42] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, "Manifold-ranking based image retrieval," in *ICM*. ACM, 2004, pp. 9–16.

[43] X. Zhu, Z. Ghahramani, J. Lafferty *et al.*, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, vol. 3, 2003, pp. 912–919.