Exploring Expert Cognition for Attributed Network Embedding

Xiao Huang,[†] Qingquan Song,[†] Jundong Li,[‡] Xia Hu[†]
[†]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA 77843
[‡]Computer Science and Engineering, Arizona State University, Tempe, AZ, USA 85281
{xhuang,qqsong,xiahu}@tamu.edu,jundongl@asu.edu

ABSTRACT

Attributed network embedding has been widely used in modeling real-world systems. The obtained low-dimensional vector representations of nodes preserve their proximity in terms of both network topology and node attributes, upon which different analysis algorithms can be applied. Recent advances in explanation-based learning and human-in-the-loop models show that by involving experts, the performance of many learning tasks can be enhanced. It is because the experts have a better cognition in the latent information such as domain knowledge, conventions, and hidden relations. It motivates us to employ experts to transform their meaningful cognition into concrete data to advance network embedding. However, learning and incorporating the expert cognition into the embedding remains a challenging task. Because expert cognition does not have a concrete form, and is difficult to be measured and laborious to obtain. Also, in a real-world network, there are various types of expert cognition such as the comprehension of word meaning and the discernment of similar nodes. It is nontrivial to identify the types that could lead to a significant improvement in the embedding. In this paper, we study a novel problem of exploring expert cognition for attributed network embedding and propose a principled framework NEEC. We formulate the process of learning expert cognition as a task of asking experts a number of concise and general queries. Guided by the exemplar theory and prototype theory in cognitive science, the queries are systematically selected and can be generalized to various real-world networks. The returned answers from the experts contain their valuable cognition. We model them as new edges and directly add into the attributed network, upon which different embedding methods can be applied towards a more informative embedding representation. Experiments on real-world datasets verify the effectiveness of NEEC.

1 INTRODUCTION

Attributed networks are pervasive in real-world systems such as social media and protein-protein interaction networks, where both node dependencies and rich attribute information describing the properties of nodes are available. Attributed network embedding [14, 18] provides an efficient way to model the two information sources jointly. It aims to map each node into a low-dimensional vector representation such that its proximity in terms of both topological structure and node attributes are well preserved. The learned representations could be applied to and advance the performance of many real-world mining tasks, such as node classification [34, 49], community detection [27], and anomaly detection [22].

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159655

The helpfulness of expert cognition motivates us to investigate the potential of utilizing it to learn a more informative embedding representation. We define the *expert cognition* as the intelligence-related information that the experts know beyond the data, such as the understanding of domain knowledge [45], the awareness of conventions [7], and the perception of latent relations. An example would be the comprehension of bias in sampling, i.e., meteorologists know that tornadoes were more likely to be recorded in areas with more population [7]. Existing work such as explanation-based learning [8] and human-in-the-loop models [12, 17] suggest that the involvement of experts could increase the performance of many learning tasks. The expert cognition plays an essential role in advancing these data analysis. Since most existing network embedding algorithms are data-driven, we are motivated to explore how to take advantage of the expert cognition to enhance the performance of embedding. Sentiment analysis is a typical example of the utilization of expert cognition, which has been successfully applied to improve the performance of different recommendations [48]. In product review platforms such as Epinions and Slashdot, items are recommended to users based on their historical reviews. It has been shown that most reviews convey different levels of sentiment polarization [13], and the expert-created sentiment lexicons such as MPQA and SentiWordNet [37] are widely used to advance the review analysis and recommendation [9, 48].

Although some efforts have been devoted to directly learning expert cognition from the existing human-generated data [9], such concrete related data is still often limited and could be in different forms. Motivated by the success of active learning [35, 36] and interactive data mining [1, 12], in this paper, we propose to explore expert cognition via actively querying the experts. We aim to convert their abstract cognition into concrete answers, and incorporate them into attributed network embedding towards a more informative low-dimensional representation.

However, actively exploring expert cognition is a nontrivial task, because of three major challenges as follows. First, expert cognition does not have a concrete format and is not easy to be measured, since it is related to the intelligence. Traditional solutions are to design specific algorithms according to the experts' understanding [8, 17]. It is hard to be generalized as they involve enormous engineering experimentations to transform the domain knowledge to algorithms. Second, in a real-world system, there are usually various types of expert cognition such as the comprehension of word meaning and the discernment of missing edges [12]. It is difficult to model all of them, as well as identify the types that could lead a significant performance gain in the embedding within a limited number of trials. It is also expensive to design specific models for different attributed networks and different embedding algorithms. A general way of modeling the expert cognition is essential. Third, the expert cognition is laborious to obtain as it involves humans in the whole process. Given a limited amount of human effort, we aim to design the queries in an effective way to maximize the total amount of learned expert cognition. Meanwhile, the returned answers from the experts could be heterogeneous with the attributed network. It is desired to have answers that could be easily incorporated into the embedding.

To bridge the gap, we formally define the problem of exploring expert cognition for attributed network embedding and study three research questions. (1) How to design effective, general, and concise format of queries to save the experts' efforts? (2) How to select the most meaningful queries to maximize the total amount of learned expert cognition, given a limited amount of human effort? (3) How to quantify the amount of expert cognition that contained in the returned answers since it is necessary when evaluating the proposed solutions? Following these research objectives, we propose a general expert cognition learning and formulating framework NEEC, and our key contributions can be summarized as follows.

- Formally define the problem of exploring expert cognition for attributed network embedding.
- Present an efficient and concise framework NEEC that could provide the experts a small number of carefully designed general and concise queries, and their answers capture expert cognition systematically and can be directly added into the network to advance the embedding.
- Develop an effective algorithm to tackle the exploration and exploitation balancing in expert cognition learning.
- Empirically evaluate the effectiveness, generalizability, and efficiency of NEEC on three real-world datasets.

2 PROBLEM STATEMENT

Notations: We use a boldface lowercase alphabet to represent a vector (e.g., $\mathbf{u}$), and a boldface uppercase alphabet to denote a matrix (e.g., $\mathbf{U}$). The transpose of a matrix is displayed as $\mathbf{U}^\top$, and the i^{th} row of the matrix $\mathbf{U}$ is written as $\mathbf{u}_i$. We use u_{ij} to represent the $(i,j)^{\text{th}}$ element of the matrix $\mathbf{U}$. We use $\mathbf{I}$ to represent the identity matrix. We use $\mathbf{S}^{(\cdot)}$ to denote the node proximity defined by the rows of a matrix. In this paper, we use cosine similarity as the proximity measure, and it is straightforward to extend to other measures. The important symbols are listed in Table 1.

Let $\mathbf{G}$ be the adjacency matrix of n connected nodes. The edge weights in $\mathbf{G}$ are set to be non-negative to have physical meanings. Let $\mathbf{A}$ denote the node attribute information matrix. Its i^{th} row represents the node attributes of node i. Suppose the raw data of $\mathbf{A}$ is human-readable such as the paper abstract or the product review. For ease of presentation, in this paper, we focus on the undirected networks and propose a general framework. It can also be directly applied to the directed networks.

Definition 1 (Expert Cognition) *For real-world attributed networks, the domain experts often have a better comprehension beyond the data, such as the understanding of domain knowledge, the awareness of conventions, and the perception of latent relations. We refer their knowledge on this type of abstract, meaningful, and intelligence-related information as the expert cognition.*

Table 1: Main symbols and definitions.

Notations	Definitions
n	number of nodes in the network
d	dimension of the embedding representation
B	number of prototype nodes
K	number of queries the oracle can answer
$r_{i,k}$	reward obtained from node i at iteration k
$\mathbf{G} \in \mathbb{R}_+^{n\times n}$	weighted adjacency matrix
$\mathbf{A} \in \mathbb{R}_+^{n\times m}$	node attribute information matrix
$\mathbf{H} \in \mathbb{R}^{n\times d}$	final embedding representation
$\mathbf{U}^{(G)} \in \mathbb{R}^{n\times d}$	embedding representation of the network
$\mathbf{S}^{(\cdot)} \in \mathbb{R}^{n\times n}$	node proximity defined by rows of a matrix

The expert cognition is often valuable but hard to be modeled. For instance, humans' comprehension of word meaning is essential to the analysis of social networks, since texts in social media could be informal and fast-evolving [13]. However, it is challenging to generalize and model this comprehension. Another example could be the experienced doctors' discernment of similar cases of illness, which is valuable to patients but difficult to be learned and performed by computers.

Instead of directly modeling expert cognition, we propose a novel way to learn it, i.e., querying the experts a number of trials to model their abstract cognition as concrete answers. The existing related data is often limited, and it is difficult to design specific models for different systems as the given data could be in various forms. Thus, in this paper, we focus on actively learning from the experts. It is motivated by the success of active learning [36], which has been shown to be helpful in reducing the efforts of labeling. The key idea is, if we are allowed to select the nodes for training, a comparable classification accuracy could be achieved even with fewer training labels. However, existing methods cannot be applied to our problem since it is often expensive to obtain concrete labels, such as the labels in ranking systems or community detection problems. Also, it is challenging to incorporate labels into the embedding since they are heterogeneous to the attributed network.

Therefore, we aim to actively explore the expert cognition in a more general way. We refer the experts as the oracle and formally define the problem of exploring expert cognition for attributed network embedding as follows.

Given an attributed network $\mathbf{G}$ associated with node attributes $\mathbf{A}$ and corresponding raw data, we aim to seek a d-dimensional vector representation $\mathbf{h}_i$ for each node i, such that both the topological structure and node attribute proximity could be well preserved in $\mathbf{H}$. Meanwhile, the system is allowed to learn from the oracle by performing K queries one by one. Given a limited amount of effort the oracle could devote, we aim to design the queries in a general and efficient way to include as much as possible expert cognition in the K returned answers. As a result, we could incorporate the answers into the embedding towards a more informative expert cognition informed representation $\mathbf{H}$.

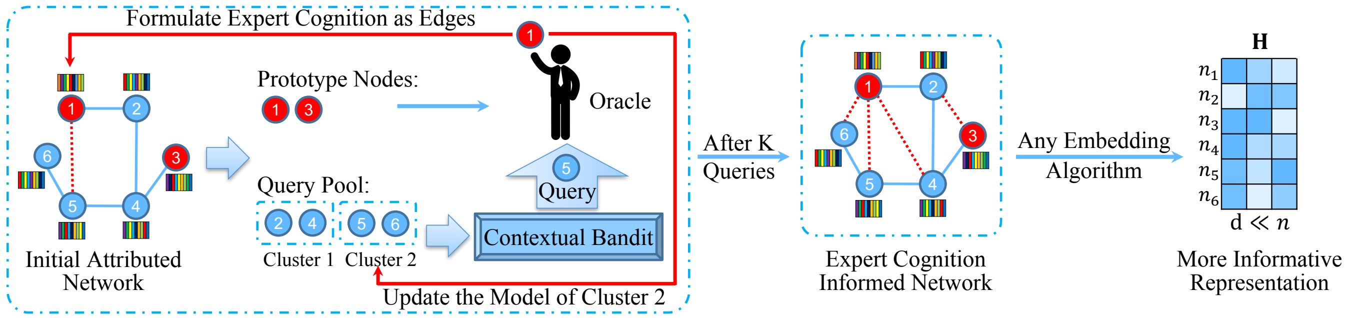
Formulate Expert Cognition as Edges
Prototype Nodes:
Oracle
Query Pool:
Query
Contextual Bandit
Cluster 1
Cluster 2
Update the Model of Cluster 2
Initial Attributed Network
After K Queries
Expert Cognition Informed Network
Any Embedding Algorithm
H
d ≪ n
More Informative Representation

Figure 1: The overall structure of the proposed Network Embedding with Expert Cognition framework - NEEC.

3 EMBEDDING WITH EXPERT COGNITION

To efficiently model and assimilate the expert cognition actively learned from the oracle into attributed network embedding, we propose the Network Embedding with Expert Cognition framework - NEEC. It has three components as illustrated in Figure 1. (1) We design a general and concise form of queries to learn expert cognition from the oracle while greatly save his/her effort. It allows us to model the meaningful but abstract expert cognition as systematical answers. (2) We explore novel algorithms for the two steps to find the top K meaningful queries. The first step aims to find B representative and distinct nodes as prototypes, and the second step iteratively selects K nodes from the remaining nodes with the largest amount of expected learned expert cognition. (3) We formulate the returned K answers from the oracle as cognition edges and add them into the initial network structure. Figure 1 illustrates an example on a network of six nodes. We first select $B = 2$ nodes as prototypes (in red), and set the remaining nodes (in blue) as a query pool. In each query, we select a node i (e.g., $i = 5$) to query. The oracle needs to indicate a node from the prototypes (e.g., $j = 1$) that is the most similar to the queried node i. All these answers will be added into the network structure in the form of weighted edges, named as cognition edges (red dotted lines). In Figure 1, the queried nodes $i = 4, 5, 6$ are more similar to prototype node $j = 1$, so we add weighted edges between them. With these cognition edges, different attributed network embedding methods could be directly applied to the expert cognition informed network towards a more informative low-dimensional representation $\mathbf{H}$.

3.1 Prototype-based Form of Queries

Given a limited amount of effort the oracle could devote, simpler queries would allow us to have a larger number of queries K. Meanwhile, we expect each query to be more effective in exploring the expert cognition from the oracle. These two aspects often contradict each other, since complex and specific queries tend to capture more information, but are both hard to design and answer.

An intuitive solution is to query the latent relations of node pairs to model the expert cognition as identified missing edges. However, it might not be efficient since the total number of queries K is quite limited compared with the total number of node pairs, i.e., $K \ll \binom{n}{2}$. It is also difficult to select the K most meaningful node pairs from all the unconnected pairs. Our proposed solution is to learn a more general type of node relations from the oracle, and in a systematical way to increase the total amount of learned expert cognition.

3.1.1 *Learning Expert Cognition with Prototypes.*

Based on the exemplar theory and prototype theory in cognitive science, we propose the prototype-based form of queries.

Theory 1 (Exemplar Theory) [28] *In psychology, humans cognize new stimuli by comparing with exemplars already have in memory, and classify objects based on their similarity to these exemplars.*

Theory 2 (Prototype Theory) [33] *For a category of stored exemplars in humans' memory, there exists an abstract average of all members called the prototype, and humans perform categorizations based on prototypes when experiencing new objects.*

Motivated by the process of humans cognizing new objects, we first select B nodes and define them as the prototypes of B categories respectively. Then we classify K other selected nodes into these B categories according to the oracle's cognition. Such classification process could capture the oracle's comprehensive comprehension of the K nodes. In the end, we formulate this valuable comprehension as cognition edges defined as follows.

Definition 2 (Prototype Node) *We select a small number (i.e., B) of representative nodes to serve as prototypes in the oracle's memory, and refer them as prototype nodes.*

Definition 3 (Cognition Edge) *In each query, we add an edge g_{ij} with a predefined weight γ between the newly selected node i and its most similar prototype node j. We refer g_{ij} as a cognition edge. If there is already an edge between nodes i and j in the initial network, we will add the weight γ into this edge.*

Figure 1 illustrates the basic idea of the prototype-based form of queries. Initially, nodes 1 and 3 are selected as prototype nodes. In each query, we select another new node i to query the oracle. He/she needs to determine node i is more similar to which prototype, based on his/her understanding of nodes' human-readable raw data and expertise. A cognition edge would be added to the network according to the returned answer. It models the category of node i in the oracle' cognition. After K queries, we could directly apply an embedding algorithm to the expert cognition informed network towards a more informative unified representation.

3.1.2 *Analysis of Prototype-based Form of Queries.*

We have proposed to learn the expert cognition with prototype nodes and model it as cognition edges. It enjoys several nice properties as follows. First, evaluating similarity takes a much smaller amount of human effort comparing to other technical tasks such as labeling. The prototype-based form of queries are general and simple with no specific similarity value required. Second, the data for some specific nodes might be limited, but domain experts' knowledge and cognition could help. For example, researchers can understand the main idea of papers by reading the abstracts, while these short paragraphs are often too limited to computers. Third, the oracle only needs to check $B + K$ nodes in total, which are selected from a pool with size n. It is much smaller than $\binom{n}{2}$.

3.2 Prototype Node Selection Algorithm

We propose two algorithms for the prototype node selection to handle different scenarios. The key idea is to make prototype nodes representative and avoid being too similar to each other, such that the oracle could easily distinguish them.

The first algorithm is k-medoids, motivated by the work of [29]. This algorithm is designed for the network with rich initial data. It aims to select B prototype nodes that cluster the entire n nodes into B distinct groups. An updating rule that similar to the k-means algorithm is employed. It first selects B nodes randomly, and then alternatively conducts the following two proceeds until no change could be made. (1) Assign nodes to their nearest prototype nodes' clusters based on node attribute similarity. (2) For each node, compute the sum of its similarities to all other nodes within the same cluster. For each cluster, set the node that has the maximum similarity sum as the new prototype node.

The second algorithm is a random strategy for networks with limited initial data or with a large number of nodes. We first select B nodes from the entire n nodes randomly, and then keep removing the prototype nodes that are too similar to each other and replacing them with new random nodes. The similarities are based on the initial attributed network.

3.3 Query Selection Algorithm

We have proposed to learn the expert cognition by linking nodes with their most similar prototype nodes. In such way, we formulate the query selection problem as a node selection task. The goal is to find K effective nodes to maximize the total amount of expert cognition contained in the learned cognition edges.

An intuitive solution is to greedily return the top K important nodes [26], aiming to make the learned cognition edges more meaningful. However, this might not end up with a good result. Because important nodes tend to have more edges included in the initial network [25], and the learned cognition edges would become less influential. It also fails to take other information into consideration, such as the feedback from the oracle and the node attributes.

3.3.1 Exploitation and Exploration Trade-off. To find the K effective nodes, we first formally define the problem. Let $r_{i,k}$ denote the amount of expert cognition contained in the k^{th} returned answer, referred as *reward*. Here i denotes the k^{th} node we have queried. We have no access to the reward $r_{i,k}$ before asking the k^{th} query. The goal is to maximize the total rewards after all K queries, i.e.,

$$\underset{i \in \mathcal{P}_1}{\text{maximize}} \quad \mathcal{J}_K = \sum_{k=1}^{K} r_{i,k}, \tag{1}$$

where $\mathcal{P}_k$ denotes the query pool in the k^{th} iteration. In the first iteration, the initial $\mathcal{P}_1$ has $(n - B)$ nodes since we have employed B nodes as prototype nodes. For the node that is already selected previously, its reward is 0 since no new cognition edge would be added. We remove it from the pool and get the $\mathcal{P}_{k+1}$.

In the beginning, we have limited information about the relation between the node i and $r_{i,k}$. Therefore, when performing the querying, we aim to not only maximize the learned expert cognition in the current trial, but also explore the system to optimize the incoming iterations. To tackle this exploration and exploitation trade-off and maximize the total rewards $\mathcal{J}_K$, we propose a novel contextual bandit [21, 47] algorithm to perform the node selection.

3.3.2 Contextual Bandit Algorithm. Our main idea is to assume that nodes within the same cluster share the same model that measures their relations to the rewards. For nodes $i \in \mathcal{P}_1$, we cluster them into d models, and define an information vector $\mathbf{x}_i$ to describe the property of each node i. We employ a linear function to model the correlation between $\mathbf{x}_i$ and $r_{i,k}$, i.e.,

$$\mathrm{E}[r_{i,k}|\mathbf{x}_i] = \mathbf{x}_i\boldsymbol{\theta}_a + \eta_a, \tag{2}$$

where $a \in \{1, 2, \ldots, d\}$ denotes the cluster that node i belongs to, and η_a is a random noise with Gaussian distribution $\mathcal{N}(0, \sigma_a^2)$.

We aim to maximize $\mathcal{J}_K$ by conducting two processes for K iterations as follows. In iteration k: (1) We choose a node $i \in \mathcal{P}_k$ with the maximum expectation to get higher $\mathcal{J}_K$. It is based on all the learned $\boldsymbol{\theta}_a$, for $a = 1, 2, \ldots, d$. (2) After querying the selected node i, we would get a reward $r_{i,k}$ from the oracle. Based on $r_{i,k}$ and $\mathbf{x}_i$, we update the $\boldsymbol{\theta}_a$, aiming to improve the node selection strategy, where a is the cluster that node i belongs to.

We now introduce the details. There are three types of information dominate node i, i.e., network structure, node attributes, and correlations to the B prototype nodes. We define $\mathbf{x}_i \in \mathbb{R}^{1\times(d+2)}$ as,

$$\mathbf{x}_i = [\mathbf{h}_i, \mathrm{maxSimi}(i), \mathrm{Corr}(i)], \tag{3}$$

where $\mathbf{h}_i$ is the attributed network embedding representation of node i, and $\mathrm{maxSimi}(i)$ denotes the maximum similarity from node i to all B prototype nodes. $\mathrm{Corr}(i)$ denotes the average of the correlations between node i and all prototype nodes. To make the algorithm efficient, in this paper, $\mathbf{x}_i$ is fixed from the initialization. It is flexible to explore dynamic frameworks that update $\mathbf{x}_i$ in every iteration. Both the distance and correlation are based on the cosine similarities in the initial embedding representation space $\mathbf{H}_0$.

We now focus on each cluster a. Let $\mathbf{Y}_a \in \mathbb{R}^{y\times(d+2)}$ be a matrix that collects the information vectors of the y queried nodes in cluster a. Let $\mathbf{y}_a \in \mathbb{R}^{y\times1}$ denote the corresponding y rewards. By leveraging ridge regression to estimate $\boldsymbol{\theta}_a$ and adding a penalty term λ to improve the estimation performance, we have,

$$\hat{\boldsymbol{\theta}}_a = (\mathbf{Y}_a^\top\mathbf{Y}_a + \lambda\mathbf{I})^{-1}\mathbf{Y}_a^\top\mathbf{y}_a. \tag{4}$$

Then for any $\delta > 0$, with probability at lease $1 - \delta$, the expectation of the reward of selecting a node i in cluster a is bounded by [43],

$$\mathbf{x}_i\hat{\boldsymbol{\theta}}_a - \alpha f_a(i) \leq \mathrm{E}[r_{i,k}|\mathbf{x}_i] \leq \mathbf{x}_i\hat{\boldsymbol{\theta}}_a + \alpha f_a(i), \tag{5}$$

where $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$ is a constant, and

$$f_a(i) \triangleq \sqrt{\mathbf{x}_i(\mathbf{Y}_a^\top\mathbf{Y}_a + \lambda\mathbf{I})^{-1}\mathbf{x}_i^\top}. \tag{6}$$

The work of [2, 3] have shown that choosing the node maximizing the upper confidence bound $\mathbf{x}_i\hat{\boldsymbol{\theta}}_a + \alpha f_a(i)$ in each iteration achieves the best performance in maximizing $\mathcal{J}_K$. The key idea is, when a node with large $\mathbf{x}_i\hat{\boldsymbol{\theta}}_a$ is chosen, a high reward is expected, and such trial is an exploitation process. When a node with large $\alpha f_a(i)$ is chosen, this high variance means that we have limited knowledge on the model for cluster a, thus, we explore it more in the current trial. Jointly maximizing $\mathbf{x}_i\hat{\theta}_a + \alpha f_a(i)$ makes a trade-off between exploitation and exploration. As a conclusion, in iteration k, the best node i for querying could be computed as follows.

$$\underset{i \in \mathcal{P}_k}{\text{maximize}} \quad g(i) = \mathbf{x}_i\hat{\theta}_a + \alpha f_a(i). \tag{7}$$

3.4 Expert Cognition Quantization

We have defined the amount of expert cognition included in the k^{th} returned answer as $r_{i,k}$. We now introduce how to quantify it. The key idea of our proposed metric is that incorporating more expert cognition could reduce the disagreements between the network proximity and node attribute proximity. Given a new node, experts usually jointly consider different types of information to get a comprehensive understanding, including its relation network and node attributes. It matches the observation that node attributes often highly tie in with the topological structure [24, 41]. However, in the data, there always exist some disagreements between the two sources, and we quantify them as follows.

$$\mathbf{P} = |\mathbf{S}^{(\mathbf{U}^{(G)})} - \mathbf{S}^{(\mathbf{U}^{(A)})}|, \tag{8}$$

where $|\cdot|$ calculates each absolute value. $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(A)}$ are the embedding representations of the initial network and node attributes. We embed them to alleviate the effect of noise and missing data. Let j be the most similar prototype node of i based on the answer. Since adding a cognition edge (i, j) could alleviate the disagreement, we quantify the learned expert cognition in the k^{th} iteration as,

$$r_{i,k} = p_{ij}. \tag{9}$$

3.5 Network Embedding with Expert Cognition

We introduce a simple attributed network embedding algorithm based on spectral embedding [25, 42]. It should be noted that NEEC can be generalized to advance different embedding algorithms.

3.5.1 Network embedding. Network embedding [10, 30, 38] aims to use a low-dimensional vector representation $\mathbf{u}_i$ to map each node i, such that all the network proximity could be preserved in $\mathbf{U}$. The main idea of the spectrum technique is to enforce nodes i and j with larger proximity s_{ij} to be closer to each other in the embedding space. It can be achieved by minimizing the loss function $\mathcal{J}_{\text{spec}}$, i.e.,

$$\mathcal{J}_{\text{spec}} = \frac{1}{2}\sum\nolimits_{i,j=1}^{n} s_{ij} \| \frac{\mathbf{u}_i}{\sqrt{d_i}} - \frac{\mathbf{u}_j}{\sqrt{d_j}} \|_2^2, \tag{10}$$

where s_{ij} is the proximity in $\mathbf{S}^{(G)}$. d_i is the sum of the i^{th} row of $\mathbf{S}^{(G)}$, and it is employed for normalization. $\mathbf{u}_i$ needs to be orthonormal to avoid being arbitrary. We could further rewrite it as a maximization problem by defining $\mathcal{J}_G = 1 - \mathcal{J}_{\text{spec}}$, i.e.,

$$\begin{aligned} &\underset{\mathbf{U}^{(G)}}{\text{maximize}} \quad \mathcal{J}_G = \text{Tr}(\mathbf{U}^{(G)\top}\mathcal{L}^{(G)}\mathbf{U}^{(G)}) \\ &\text{subject to} \quad \mathbf{U}^{(G)\top}\mathbf{U}^{(G)} = \mathbf{I}, \end{aligned} \tag{11}$$

where $\mathcal{L}^{(G)} = \mathbf{D}^{(G)-\frac{1}{2}}\mathbf{S}^{(G)}\mathbf{D}^{(G)-\frac{1}{2}}$ is the graph Laplacian [42]. $\mathbf{D}^{(G)}$ is the degree matrix of $\mathbf{S}^{(G)}$, with d_i in the diagonal.

Similarly, we apply the same technique to the node attribute information $\mathbf{A}$, and calculate its embedding representation $\mathbf{U}^{(A)}$ by maximizing $\mathcal{J}_A = \text{Tr}(\mathbf{U}^{(A)\top}\mathcal{L}^{(A)}\mathbf{U}^{(A)})$, with $\mathbf{U}^{(A)\top}\mathbf{U}^{(A)} = \mathbf{I}$.

Algorithm 1: Network Embedding with Expert Cognition

```
Input: $\mathbf{G}$, $\mathbf{A}$, $d$, $B$, $K$, $\beta$, the oracle.
Output: Cognition informed representation $\mathbf{H}$.
1  Select $B$ prototype nodes via k-medoids or random strategy;
2  Set the remaining nodes as $\mathcal{P}_1$ and split them into $d$ clusters;
3  Collect the information vectors of nodes in $\mathcal{P}_1$ based on Eq. (3);
4  Calculate $\mathbf{P}$ based on Eqs. (8) and (11);
5  Calculate initial representation $\mathbf{H}_0$ based on Eq. (13);
6  Set all the $\mathbf{Q}_a \leftarrow \lambda\mathbf{I}$, $\mathbf{q}_a \leftarrow \mathbf{0}$, and calculate all the $g(i)$;
7  for $k = 1 : K$ do
8      Choose node the $i$ with maximum $g(i)$;
9      Query the oracle node $i$ and get the answer;
10     Incorporate node $i$'s cognition edge to $\mathbf{G}$;
11     Calculate $r_{i,k}$ based on Eq. (9);
12     Update $\mathbf{Q}_a \leftarrow \mathbf{Q}_a + \mathbf{x}_i^\top\mathbf{x}_i$ and $\mathbf{q}_a \leftarrow \mathbf{q}_a + r_{i,k}\mathbf{x}_i^\top$;
13     Remove node $i$ from the set $\mathcal{P}_k$ and get $\mathcal{P}_{k+1}$;
14     For nodes in cluster $a$, set $g(i) \leftarrow \mathbf{x}_i\mathbf{Q}_a^{-1}\mathbf{q}_a + \alpha\sqrt{\mathbf{x}_i\mathbf{Q}_a^{-1}\mathbf{x}_i^\top}$;
15 Calculate cognition informed $\mathbf{H}$ based on current $\mathbf{G}$ and $\mathbf{A}$.
```

3.5.2 Jointly Embed Network and Node Attributes. In many real-world networks such as social media, the topological structure and node attributes influence each other and are inherently correlated. The homophily hypothesis [24] and social influence theories [41] demonstrate that similar nodes tend to bond and connect with each other. Therefore, jointly learning from the two information sources could draw towards a better embedding representation $\mathbf{H}$. We apply a simple strategy to assimilate the two heterogeneous information towards a joint node proximity as follows.

$$\mathbf{S}^{(H)} = \beta\mathbf{S}^{(G)} + (1 - \beta)\mathbf{S}^{(A)}, \tag{12}$$

where β balances the contributions of network and node attributes. Similarly, we could define the Laplacian $\mathcal{L}^{(H)} = \mathbf{D}^{(H)-\frac{1}{2}}\mathbf{S}^{(H)}\mathbf{D}^{(H)-\frac{1}{2}}$, where $\mathbf{D}^{(H)}$ is the degree matrix of $\mathbf{S}^{(H)}$, and apply the normalized spectral embedding technique to learn $\mathbf{H}$ as follows,

$$\begin{aligned} &\underset{\mathbf{H}}{\text{maximize}} \quad \mathcal{J} = \text{Tr}(\mathbf{H}^\top\mathcal{L}^{(H)}\mathbf{H}) \\ &\text{subject to} \quad \mathbf{H}^\top\mathbf{H} = \mathbf{I}. \end{aligned} \tag{13}$$

3.5.3 Embed the Learned Expert Cognition. We summary the processes of NEEC in Algorithm 1. Given an attributed network with $\mathbf{G}$ and $\mathbf{A}$, we first select B prototype nodes and set the remaining nodes as the initial query pool $\mathcal{P}_1$. To choose the optimal K nodes from $\mathcal{P}_1$, we perform a contextual bandit algorithm based on Eq. (7). It has two items that are not necessary to be computed from scratch. We denote them as follows.

$$\mathbf{Q}_a = \mathbf{Y}_a^\top\mathbf{Y}_a + \lambda\mathbf{I}, \quad \text{and} \quad \mathbf{q}_a = \mathbf{Y}_a^\top\mathbf{y}_a. \tag{14}$$

Thus, in each iteration k, we update them according to the rules in line 12. Then we select the node with the maximum explanation upper confidence bound as defined in Eq. (7). After querying the oracle, we incorporate the answer into $\mathbf{G}$ as a cognition edge. When all the budget is used, we apply an attributed network embedding method to the current network $\mathbf{G}$ and $\mathbf{A}$, and jointly embed them towards a more informative cognition informed representation $\mathbf{H}$.

4 EXPERIMENTS

We apply NEEC on three real-world attributed networks to demonstrate its effectiveness and generalizability. In the experiments, we target to investigate three questions. (1) How effective are the proposed prototype-based form of queries and the query selection algorithm? (2) Could the learned expert cognition be used to improve different embedding methods? (3) What are the impacts of the parameters B, K, and d, on the expert cognition learning?

4.1 Datasets

We now introduce the three datasets. They are all publicly available, and their statistical information is listed in Table 2.

BlogCatalog [20] is a social networking service that helps people share blogs. The bloggers can interact with each other and form a network. They could also have a list of keywords to describe their blogs, which are employed as node attributes. We leverage the predefined groups that each blogger subscribed as the labels.

Flickr [20] is a photo and video sharing service. Users follower each other and form a big community. We sampled 7,575 users to construct a network. Node attributes of each user is defined as the list of tags of interest specified by him/her. Users could also join some predefined groups, thus we employ them as labels.

ACM [39] dataset collects a citation network for papers published before 2016. We select papers in nine areas as follows. Artificial Intelligence (AAAI, IJCAI, etc.), Computer Vision (TPAMI, CVPR, etc.), Computational Linguistics (ACL, EMNLP, etc.), Data Mining (KDD, WSDM, etc.), Databases (VLDB, TKDE, etc.), Human Computer Interaction (CHI, UIST, etc.), Information Retrieval (CIKM, SIGIR, etc.), Machine Learning (ICML, COLT, etc.), Robotics (ICRA, IROS, etc.). We make the network undirected since it is too sparse for spectral embedding. We apply the bag-of-words model to represent the abstracts as node attributes and employ areas as labels.

4.2 Experimental Settings

To create the oracle, for each dataset, we randomly sample a certain percentage of the entire edges and attributes as the initial attributed network. The remaining data is considered as the cognition of the oracle, so he/she answers the queries based on the entire original dataset. The oracle determines the most similar prototype nodes based on the similarity of the original node attributes. Then the expert cognition learning methods could be employed to query the oracle and enhance the initial networks. A network embedding or attributed network learning method would be applied to both the initial network and the learned network. In such way, we are able to evaluate the amount of learned expert cognition by measuring the performance improvement of the embedding representation.

To evaluate the performance of embedding representations, we follow the traditional way [30, 38] to apply it to the node classification task [34, 49]. The goal is to predict the labels of new nodes based on the training nodes that have labels. We use 5-fold cross-validation. The nodes are randomly separated into two groups, training and test groups. The labels of nodes in test group are set as ground truth. We apply the embedding method to the entire attributed network and learn an embedding representation $\mathbf{H}$. It contains the vector representations of nodes in both training and test groups, and we split it into $\mathbf{H}_{\text{train}}$ and $\mathbf{H}_{\text{test}}$. For each of the label category, we learn a binary Support Vector Machine (SVM) classifier based on $\mathbf{H}_{\text{train}}$ and labels of nodes in the training group. By using these trained classifiers and $\mathbf{H}_{\text{test}}$, we predict the labels of nodes in the test group.

Table 2: Statistical information of the datasets.

	Nodes	Edges	Density	Attributes	Label
BlogCatalog	5,196	171,743	$1.27e{-}002$	8,189	6
Flickr	7,575	239,738	$8.36e{-}003$	12,047	9
ACM	16,484	71,980	$5.30e{-}004$	8,337	9

We use two widely adopted evaluation criteria to measure the performance of a representation in the node classification task, i.e., macro-average and micro-average [16]. The former one is the arithmetic average of F-measure of all label categories. F-measure is a commonly used metric in binary classification. The latter one is the harmonic mean of precision and recall. If it is not specified, we set $d = 100$ and $B = 9$. All results are the means of ten test runs.

4.3 Effectiveness of NEEC

To study the first question proposed at the beginning of this section, we compare NEEC with three types of baselines. First, to investigate the effectiveness of the proposed prototype-based form of queries, we include RandomPair, LinkPredict, and AddKEdges. Second, to demonstrate the effectiveness of the proposed way of formulating expert cognition, we include HiddenEdge. Third, to study the effectiveness of the proposed query selection algorithm, we include a variation of NEEC, i.e., w/o_Bandit. For NEEC, we include both k-medodis and random prototype node selection algorithms. The details of them are described as follows.

- *RandomPair*: No prototype node is defined. It randomly selects $B + K$ unconnected node pairs for querying. The oracle will indicate an edge, if the two nodes are linked in the original dataset or have an original node attribute similarity that is greater than a threshold.
- *LinkPredict*: Similar to RandomPair, but the $B + K$ pairs are selected from the unconnected pairs based on the probabilities of having edges, which are based on the weighted-ℓ_1 edge features [10] in spectral embedding space.
- *HiddenEdge*: It queries the oracle based on prototype nodes, but without using the cognition edges. It adds edges between the selected node i and any prototype nodes that have latent relationships with i or are highly similar to i.
- *w/o_Bandit*: NEEC without using the proposed query selection algorithm. It chooses the top K important nodes with the maximum eigenvector centrality [26].
- *AddKEdges*: It randomly queries the oracle a large number of node pairs until K edges were added. This baseline unfairly takes much more human effort than all others.

4.3.1 Form of Queries Investigation. To study the effectiveness of querying with prototype nodes, we compare NEEC with RandomPair, LinkPredict, and AddKEdges. All these three baselines query the oracle without a prototype. For RandomPair and LinkPredict, the total number of queries is set as $B+K$, such that the oracle needs to check at least $B+K$ nodes and pay more effort than in NEEC. AddKEdges needs to make an extremely large number of queries. The

Table 3: The classification performance of different methods on the three datasets in terms of micro-average when $d = 100$.

	BlogCatalog				Flickr				ACM			
Training	10%	25%	50%	100%	10%	25%	50%	100%	10%	25%	50%	100%
Node Num	1,455	2,078	3,118	5,196	2,121	3,030	4,545	7,575	4,616	6,594	9,890	1,6484
K	242	495	1,114	3,092	338	691	1,556	4,316	348	709	1,598	4,435
Initial	0.287	0.359	0.430	0.520	0.261	0.304	0.358	0.432	0.232	0.270	0.319	0.400
RandomPair	0.286	0.358	0.431	0.520	0.261	0.306	0.365	0.435	0.231	0.267	0.317	0.412
LinkPredict	0.287	0.360	0.430	0.523	0.261	0.303	0.359	0.433	0.230	0.278	0.310	0.392
AddKEdges	0.312	0.379	0.462	0.547	0.255	0.312	0.363	0.457	**0.258**	**0.300**	**0.380**	**0.508**
HiddenEdge	0.298	0.370	0.437	0.529	0.265	0.301	0.363	0.436	0.232	0.275	0.322	0.418
w/o_Bandit	0.295	0.366	0.447	0.556	0.259	0.326	0.368	0.439	0.238	0.265	0.306	0.391
NEEC_Rand	**0.317**	**0.393**	0.463	0.558	0.277	**0.367**	0.433	0.508	0.235	0.288	0.355	0.447
NEEC_Kmed	0.313	0.389	**0.469**	**0.571**	**0.281**	0.364	**0.443**	**0.515**	0.242	0.284	0.363	0.445
RandomPair	−0.47%	−0.27%	0.07%	−0.07%	0.04%	0.87%	1.87%	0.64%	−0.59%	−1.35%	−0.63%	3.11%
LinkPredict	−0.03%	0.35%	−0.07%	0.41%	0.00%	−0.36%	0.15%	0.15%	−0.80%	2.90%	−2.92%	−2.09%
AddKEdges	8.43%	5.55%	7.36%	5.14%	−2.28%	2.79%	1.20%	5.83%	**10.85%**	**10.96%**	**19.23%**	**27.00%**
HiddenEdge	3.61%	3.06%	1.52%	1.68%	1.60%	−0.80%	1.41%	0.81%	−0.26%	1.81%	0.93%	4.56%
w/o_Bandit	2.64%	2.06%	3.82%	6.75%	−0.80%	7.43%	2.61%	1.58%	2.30%	−2.07%	−4.13%	−2.17%
NEEC_Rand	**10.41%**	**9.63%**	7.69%	7.27%	6.24%	**20.87%**	20.78%	17.43%	1.07%	6.74%	11.25%	11.66%
NEEC_Kmed	9.03%	8.34%	**9.01%**	**9.67%**	**7.64%**	19.78%	**23.60%**	**19.22%**	4.26%	5.16%	13.71%	11.30%

Table 4: The computation time of each iteration in NEEC.

Dataset	BlogCatalog	Flickr	ACM
Average Time (s)	$2.97e{-}003$	$8.02e{-}003$	$2.28e{-}002$

classification performance w.r.t. micro-average is shown in Table 3. All of them use the same embedding algorithm in Section 3.5. From the results, we observe that NEEC outperforms all the baselines on all the three datasets. For instance, on Flickr, both RandomPair and LinkPredict have almost no improvement, while NEEC_Kmed achieves 19.22% improvement comparing with the embedding of the initial attributed network. Even though AddKEdges performs a large number of queries, NEEC improves more on BlogCatalog and Flickr. On ACM, NEEC achieves less improvement. The reason is that AddKEdges needs to perform up to 9.4×10^6 queries.

To further investigate the performance of NEEC under different training percentages, i.e., the percentage of data in training group that has been used, we vary it from 10% to 100%. The results in Table 3 show that NEEC consistently achieves higher performance than all baselines except AddKEdges. For instance, when the training percentage is 25% on Flickr, NEEC_Rand achieves 20.87% improvement. It also demonstrates that the prototype-based form of queries is effective in learning expert cognition.

4.3.2 Effectiveness of Cognition Edge Investigation. To study the effectiveness of the proposed way of transforming expert cognition into concrete data, we compare NEEC with HiddenEdge, which models the expert cognition as latent edges. Their performance in items of different training percentage is summarized in Table 3. As we can see, HiddenEdge achieves limited improvement. For instance, NEEC_Kmed achieves 9.67% while HiddenEdge achieves 1.68%, when the training percentage is 100% on BlogCatalog. It demonstrates the effectiveness of the proposed cognition edges.

4.3.3 Prototype Node Selection and Query Selection Algorithms Investigation. We provide two algorithms to select the prototype nodes, i.e., NEEC_Rand and NEEC_Kmed. As shown in Table 3, NEEC_Rand might achieve slightly better performance when the training percentage is 10% or 25%. It is because when the number of nodes is small, the initial information would be too limited to perform the clustering well in NEEC_Kmed.

To study the effectiveness of our contextual bandit algorithm in query selection, we compare NEEC with w/o_Bandit. The performance with respect to different training percentage is shown in Table 3. From the results, we find that NEEC achieves more improvement. For example, on Flickr, NEEC_Rand achieves 17.43% improvement while w/o_Bandit achieves 1.58% improvement.

4.3.4 Effectiveness and Efficiency of NEEC Investigation. Up till now, we have demonstrated the effectiveness of each component of NEEC. The number of queries we performed also demonstrates the effectiveness of NEEC. We summarize the number of queries K and performance improvement in Table 3. As we can see, NEEC achieves significant improvement with a small number of queries. For example, NEEC_Kmed achieves 19.78% improvement by conducting 691 queries in a network with 3,030 nodes on Flickr.

NEEC is also an efficient interactive system that could respond the oracle immediately. The average computation time of each iteration in NEEC is shown in Table 4. For example, on Flickr, NEEC needs 8.02 milliseconds to update the system and compute a new query. It verifies the efficiency of NEEC.

4.4 Generalizability Evaluation

We now answer the second question that how general is the expert cognition learning framework NEEC. We include two network embedding methods, i.e., Spectrum and DeepWalk, and two attributed network embedding methods, i.e., AANE and LANE, as baselines.

- *Spectrum* [42]: It performs spectral embedding on the pure topological structure to evaluate the amount of information that we have learned in the network space.
- *DeepWalk* [30]: It makes an analogy between random walks on a graph and sentences in a document, and embed the paths via language modeling techniques.

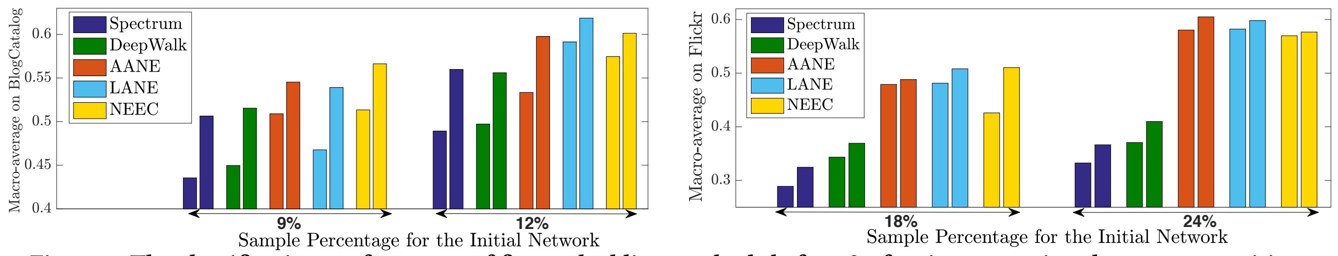

Figure 2: The classification performance of five embedding methods before & after incorporating the expert cognition.

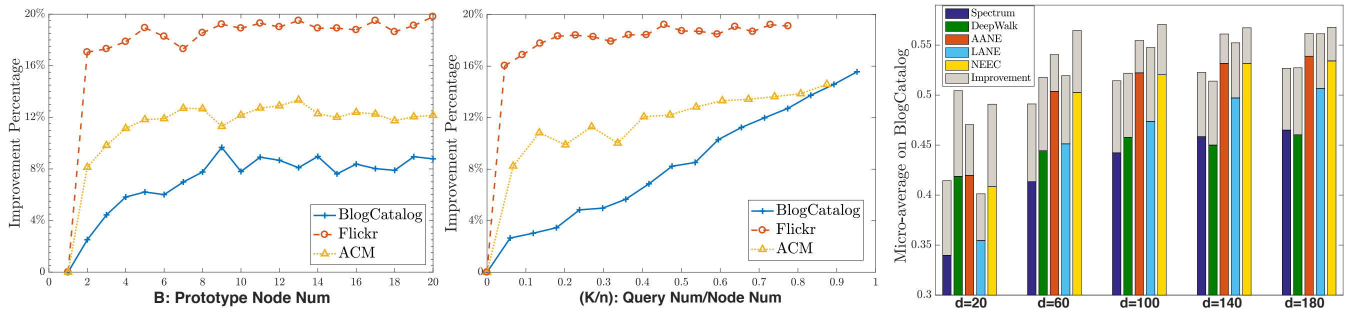

Figure 3: Performance improvement of embedding methods after incorporating the expert cognition with parameters varying.

- *AANE* [14]: It performs embedding based on the factorization of node attribute proximity and penalty of difference in the vector representations of highly correlated nodes.
- *LANE* [15]: It is used for incorporating labels into network embedding. We set the labels as zeros. It jointly embeds the network and node attributes by maximizing their correlations based on the spectral embedding.

The performance of the five embedding methods before and after incorporating the expert cognition is shown in Figure 2. We omit the result on ACM since we obtain similar observations. From the figure, we find that all the five embedding methods have significant performance improvements after incorporating the expert cognition. For example, DeepWalk achieves 14.60% improvement and LANE achieves 15.29% improvement on BlogCatalog. We further increase one-third of the sample percentage for the initial network to study its impact. From the results in Figure 2, the same observations are observed, i.e., all the five methods have significant performance improvements after incorporating the expert cognition. We also find that the amount of improvement decreased. This can be explained by the fact that, as the sample percentage increases, the amount of knowledge that the oracle obtained decreases.

4.5 Parameter Analysis

We now study the impacts of the number of prototype nodes B, the number of queries K, and the embedding representation dimension d. The performance improvement w.r.t. B and $\frac{K}{n}$ after incorporating the expert cognition is shown in Figure 3. As we can see, the performance improvement increases as B increases on all the three datasets. It is because more prototypes tend to create more accurate cognition of nodes. But it also takes more human effort. The performance improvement increases rapidly at the beginning and then increases smoothly thereafter. Similar observations are made for the parameter K. Larger query number means more expert cognition, but their correlation is not always linear. We also vary d to see the performance improvement of the five embedding methods and show the results in Figure 3. The gray bars denote the corresponding performance improvement. We see that all methods perform better after incorporating the cognition edges as d increases from 20 to 180. This also verifies the generalizability of NEEC.

5 RELATED WORK

Attributed network embedding [14, 15, 49] attracts lots of attention in recent years due to its effectiveness in modeling various information systems. Qi et al. [31] advanced the latent semantic indexing algorithm to jointly model the content similarity and context link into a low-dimensional semantic space. Le and Lauw [18] proposed to learn a joint low-dimensional representation for networked documents by incorporating a topic model into the network embedding. Chang et al. [6] considered the content as a network and embedded it along with the original network jointly via the deep learning. Several coupled matrix factorization based models [46, 49] also have been developed to assimilate these two sources of information.

Our work is also related to pool-based active learning [35, 36], which learns from the oracle by inquiring labels. It aims to select a number of unlabeled instances from a pool, such that the classification performance could be maximized by using the labels of selected instances. The typical methods are based on uncertainty sampling [19, 40], query by committee [23], expected error and variance reduction [11], or expected model change [36]. Several efforts [4, 5] also have been devoted to balancing exploration and exploitation based on the multi-armed bandit algorithms.

Multi-armed bandit algorithms [2, 3] are widely used to perform online human behavior or knowledge learning. Radlinski et al. [32] exploited a bandit algorithm to conduct an online document ranking that can learn from users behavior. Several contextual bandit based personalized recommendation frameworks [21, 44, 47] have been developed to incorporate dynamic content and user information.

6 CONCLUSIONS AND FUTURE WORK

Expert cognition is an essential type of knowledge that could advance various real-world data analysis tasks such as attributed network embedding. Learning and modeling expert cognition is promising but challenging as it is often abstract, diverse, and laborious to obtain. To fill the gap, we study a novel problem of exploring expert cognition for attributed network embedding and develop a general framework NEEC. Instead of directly modeling expert cognition, we learn it from the oracle by performing a number of concise but effective queries. The queries are carefully designed based on the exemplar theory and prototype theory, and systematically selected via a contextual bandit algorithm. Based on the returned answers from the oracle, the meaningful but abstract expect cognition is modeled as new cognition edges, which could be directly added into the network. Thus, the learned expert cognition could be incorporated into the latent representation by any embedding algorithm. Experiments on real-world datasets demonstrate the effectiveness and generalizability of NEEC. As the future work, we plan to study some open questions. (1) NEEC models the relation between queries and rewards via linear functions. How can we extend it to more general ones to improve the performance? (2) The returned answers change the network dynamically. How can we take it into consideration to advance the query selection?

ACKNOWLEDGMENTS

This work is, in part, supported by DARPA (#N66001-17-2-4031) and NSF (#IIS-1657196 and #IIS-1718840). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] Elke Achtert, Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. 2013. Interactive Data Mining with 3D-parallel-coordinate-trees. In *SIGMOD*. 1009–1012.
[2] Rajeev Agrawal. 1995. Sample Mean Based Index Policies by O(log n) Regret for the Multi-armed Bandit Problem. *Advances in Applied Probability* 27, 4 (1995), 1054–1078.
[3] Peter Auer. 2002. Using Confidence Bounds for Exploitation-exploration Trade-offs. *JMLR* 3 (2002), 397–422.
[4] Yoram Baram, Ran El-Yaniv, and Kobi Luz. 2004. Online Choice of Active Learning Algorithms. *JMLR* 5 (2004), 255–291.
[5] Djallel Bouneffouf, Romain Laroche, Tanguy Urvoy, Raphael Féraud, and Robin Allesiardo. 2014. Contextual Bandit for Active Learning: Active Thompson Sampling. In *ICONIP*. 405–412.
[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. 2015. Heterogeneous Network Embedding via Deep Architectures. In *KDD*. 119–128.
[7] Robert Davies-Jones. 2015. A review of supercell and tornado dynamics. *Atmospheric Research* 158–159 (2015), 274–291.
[8] Gerald DeJong and Shiau Hong Lim. 2011. Explanation-based Learning. *Encyclopedia of Machine Learning* (2011), 388–392.
[9] Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A Holistic Lexicon-based Approach to Opinion Mining. In *WSDM*. 231–240.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. 855–864.
[11] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale Text Categorization by Batch Mode Active Learning. In *WWW*. 633–642.
[12] Andreas Holzinger. 2016. Interactive Machine Learning for Health Informatics: When do We Need the Human-in-the-loop? *Brain Informatics* 3, 2 (2016), 119–131.
[13] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised Sentiment Analysis with Emotional Signals. In *WWW*. 607–618.
[14] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated Attributed Network Embedding. In *SDM*. 633–641.
[15] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label Informed Attributed Network Embedding. In *WSDM*. 731–739.
[16] Ling Jian, Jundong Li, Kai Shu, and Huan Liu. 2016. Multi-label Informed Feature Selection. In *IJCAI*. 1627–1633.
[17] Waldemar Karwowski. 2001. *International Encyclopedia of Ergonomics and Human Factors*. CRC Press.
[18] Tuan M. V. Le and Hady W. Lauw. 2014. Probabilistic Latent Document Network Embedding. In *ICDM*. 270–279.
[19] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *SIGIR*. 3–12.
[20] Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. 2015. Unsupervised Streaming Feature Selection in Social Media. In *CIKM*. 1041–1050.
[21] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *WWW*. 661–670.
[22] Ninghao Liu, Xiao Huang, and Xia Hu. 2017. Accelerated Local Anomaly Detection via Resolving Attributed Networks. In *IJCAI*. 2337–2343.
[23] Andrew McCallum and Kamal Nigam. 1998. Employing EM and Pool-based Active Learning for Text Classification. In *ICML*. 350–358.
[24] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* 27, 1 (2001), 415–444.
[25] Hariharan Narayanan, Mikhail Belkin, and Partha Niyogi. 2006. On the Relation Between Low Density Separation, Spectral Clustering and Graph Cuts. In *NIPS*. 1025–1032.
[26] M. E. J. Newman. 2003. The Structure and Function of Complex Networks. *SIAM Rev.* 45, 2 (2003), 167–256.
[27] M. E. J. Newman. 2006. Modularity and Community Structure in Networks. In *PNAS*, Vol. 103. 8577–8582.
[28] Robert M. Nosofsky. 1986. Attention, Similarity, and the Identification-categorization Relationship. *Journal of Experimental Psychology: General* 115, 1 (1986), 39–57.
[29] Hae-Sang Park and Chi-Hyuck Jun. 2009. A Simple and Fast Algorithm for K-medoids Clustering. *Expert Systems with Applications* 36, 2 (2009), 3336–3341.
[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*. 701–710.
[31] Guo-Jun Qi, Charu Aggarwal, Qi Tian, Heng Ji, and Thomas S. Huang. 2012. Exploring Context and Content Links in Social Media: A Latent Space Method. *TPAMI* 34, 5 (2012), 850–862.
[32] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *ICML*. 784–791.
[33] Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M. Johnson, and Penny Boyes-Braem. 1976. Basic Objects in Natural Categories. *Cognitive Psychology* 8, 3 (1976), 382–439.
[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI magazine* 29, 3 (2008), 93–106.
[35] Burr Settles. 2009. *Active Learning Literature Survey*. CS Technical Reports. University of Wisconsin–Madison.
[36] Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *EMNLP*. 1070–1079.
[37] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based Methods for Sentiment Analysis. *Computational Linguistics* 37, 2 (2011), 267–307.
[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.
[39] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and Mining of Academic Social Networks. In *KDD*. 990–998.
[40] Simon Tong and Daphne Koller. 2001. Support Vector Machine Active Learning with Applications to Text Classification. *JMLR* 2 (2001), 45–66.
[41] Oren Tsur and Ari Rappoport. 2012. What's in a Hashtag? Content Based Prediction of the Spread of Ideas in Microblogging Communities. In *WSDM*. 643–652.
[42] Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing* 17, 4 (2007), 395–416.
[43] Thomas J. Walsh, István Szita, Carlos Diuk, and Michael L. Littman. 2009. Exploring Compact Reinforcement-learning Representations with Linear Regression. In *UAI*. 591–598.
[44] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2016. Learning Hidden Features for Contextual Bandits. In *CIKM*. 1633–1642.
[45] Adam B Wilcox and George Hripcsak. 2003. The Role of Domain Knowledge in Automating Medical Text Report Classification. *JAMIA* 10, 4 (2003), 330–338.
[46] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *IJCAI*. 2111–2117.
[47] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online Context-aware Recommendation with Time Varying Multi-armed Bandit. In *KDD*. 2025–2034.
[48] Yongfeng Zhang. 2015. Incorporating Phrase-level Sentiment Analysis on Textual Reviews for Personalized Recommendation. In *WSDM*. 435–440.
[49] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. 2007. Combining Content and Link for Classification Using Matrix Factorization. In *SIGIR*. 487–494.