Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 5943798, 11 pages
https://doi.org/10.1155/2020/5943798

Research Article

Query-Specific Deep Embedding of Content-Rich Network

Yue Li,[1] **Hongqi Wang**,[1] **Liqun Yu**,[1] **Sarah Yvonne Cooper**,[2] **and Jing-Yan Wang**[3]

[1]*School of Economics and Management, Harbin University of Science and Technology, Harbin 150080, China*
[2]*The University of Edinburgh, Edinburgh EH8 9JS, Scotland, UK*
[3]*New York University Abu Dhabi, Abu Dhabi, UAE*

Correspondence should be addressed to Yue Li; liyue47853@outlook.com

Received 3 December 2019; Revised 2 February 2020; Accepted 1 May 2020; Published 25 August 2020

Academic Editor: Sergio Decherchi

In this paper, we propose to embed a content-rich network for the purpose of similarity searching for a query node. In this network, besides the information of the nodes and edges, we also have the content of each node. We use the convolutional neural network (CNN) to represent the content of each node and then use the graph convolutional network (GCN) to further represent the node by merging the representations of its neighboring nodes. The GCN output is further fed to a deep encoder-decoder model to convert each node to a Gaussian distribution and then convert back to its node identity. The dissimilarity between the two nodes is measured by the Wasserstein distance between their Gaussian distributions. We define the nodes of the network to be positives if they are relevant to the query node and negative if they are irrelevant. The labeling of the positives/negatives is based on an upper bound and a lower bound of the Wasserstein distances between the candidate nodes and the query nodes. We learn the parameters of CNN, GCN, encoder-decoder model, Gaussian distributions, and the upper bound and lower bounds jointly. The learning problem is modeled as a minimization problem to minimize the losses of node identification, network structure preservation, positive/negative query-specific relevance-guild distance, and model complexity. An iterative algorithm is developed to solve the minimization problem. We conducted experiments over benchmark networks, especially innovation networks, to verify the effectiveness of the proposed method and showed its advantage over the state-of-the-art methods.

1. Introduction

1.1. Background. Recently, content-rich network analysis has attracted much attention. Different from the traditional network, whose each node is only identified by its ID, the content-rich network has content for each node [1, 2]. For example, in a scientific article citation network, each node is a research paper, and each linkage is a citation between two papers, while each node is enriched by the content of the research paper. In this case, each node is represented by not only the paper ID, but also its content, such as its title, abstract, and text. However, in the past researches, the content of each node is ignored and only the network structure is considered to represent the nodes. For example, a popular network analysis tool is network embedding, where each node is mapped to a low-dimensional vector space, where the network structure is preserved [3–6]. The traditional network embedding methods only consider the network structure by learning from the edges of the network, while the content of the nodes is not encoded to the embedding process [7–10]. However, in many cases, the contents of two nodes have a strong clue of the linkage of two nodes, even though there is no direct edge between such two nodes in the network structure alone. As an example, in the innovation network analysis problem, two research papers recently published may have similar ideas, but they have not cited each other. However, from the content of these two papers, we can conclude that they should be sharing the same idea. Thus the content is a good complementary component for the network embedding besides the network structure itself.

Meanwhile, information retrieval is a major application of network analysis. Given a query node in the graph, the search task is to rank the other nodes according to the similarity between the query and the nodes and return the top-ranked nodes [11–15]. Using both the network structure

and the query information to rank the nodes in the network has been a popular way for information retrieval, while network embedding is another important direction of network analysis. It is natural to combine these two technologies to improve the performance of the retrieval. However, up to now, all the network embedding works have not considered the query information to boost the embedding for the retrieval results. In this paper, we fill this gap of learning network embeddings for a given specific query and the content of the nodes.

1.2. Related Works. In this section, we summarize the related works of network embedding and network-based retrieval. Our work is a query-specific network embedding method that embeds a network for the purpose of searching similar nodes of a given query node. Thus, our work is related to both the network embedding and network-based retrieval works. The related network embedding works are summarized as follows.

He et al. [2] developed the Network-to-Network Network Embedding model to combine the network structure and content of nodes into one embedding vector. To this end, two neural networks are employed, one for the content based on the convolutional neural network (CNN) model [16–18] and another for the network structure based on the graph convolutional network (GCN) model [19–22]. The CNN model embeds the content of each node to a convolutional representation vector and then feed it to the GCN model, where the representation vectors of neighbors of each node are taken as input and converted to a vector of node identity. The learning of the parameters is performed by minimizing the loss of the node identity prediction task.

Zhu et al. [6] proposed embedding a node in a network as a Gaussian distribution and using a Wasserstein distance to measure the dissimilarity between two nodes' Gaussian embedding. Moreover, they developed an encoder-decoder method to map the neighborhood coding vector to the Gaussian embedding parameters and then map it back to the neighborhood coding vector. The embedding parameters are optimized to minimize the decoding error and keep the network structures.

Tu et al. [23] proposed a deep recurrent neural network-based network (RNN) embedding method. It uses a node's neighbors in the network as the input of an RNN model and uses the output of the RNN model to approximate the embedding of the node. The inputs of the neighboring nodes are also their embedding vectors accordingly. Moreover, the RNN outputs are further fed to a multilayer perception (MLP) model to approximate the degree of the node. The learning processes are conducted by minimizing the approximation errors of both the embedding vectors and the degrees.

Wang et al. [24] innovated a novel graph embedding method for a group of networks. This method tries to learn a group of base vectors, and each vector can extend to a base affinity matrix of a base network by self-product. Then each network affinity matrix is approximated by a learning combination of the base matrices. The learning of the base vectors and the combination coefficients are learned jointly by minimizing the approximation error.

The network-based retrieval works are summarized as follows.

Li et al. [25] proposed adjusting the affinity matrix of a network according to a given query and a set of positive/negative nodes from the network. This method firstly calculates a ranking vector from the affinity matrix and query node indicator vector and then impose the positive nodes ranking score is larger than the negative nodes. Please note that the positive nodes are the known nodes that are similar to the query, while the negative nodes are the nodes known to be dissimilar to the query node. The learning of the new affinity matrix is conducted by minimizing the loss of the positive/negative nodes constraint and meanwhile keeping the adjusted affinity matrix as similar to the original affinity matrix as possible.

Yang et al. [26] proposed learning an improved affinity matrix of a network by firstly calculating the tensor product of the matrix and then conducting confusion over the tensor product of the affinity matrix. The tensor product of the matrix is an extended network where each node is a pair of nodes of the original network and each edge weight is the product of the weights of the edges between the corresponding two nodes. The confusion of a matrix is calculated as the summation of the different orders of the matrix and the number of orders varies from zero to infinite. The learned affinity matrix is obtained by recovering from the confusion of the tensor product. They proved that the recovered matrix can be obtained by an iterative algorithm, as $Q \longleftarrow AQA^{\top} + I$, where A is the original affinity matrix and Q is the expected recovered matrix.

Bai et al. [27] proposed to learn the ranking scores of nodes in a network to a query node by an iterative label prorogation algorithm. In an iterative algorithm, the ranking score of a node is updated as the weighted average of the neighboring nodes, while, at the beginning of each iteration, the ranking score of the query node itself is updated as one.

1.3. Our Contributions. Our contribution in this paper is of the following folds:

(1) We come up with a novel problem for network analysisthe query-specific content-rich network embedding problem. The setting of this problem is that each node of the network is attached to some

content, such as text and image. Moreover, a node or more nodes are known as the query node(s).The task is to learn effective embedding vectors of the nodes so that, from the embeddings, we can calculate a similarity measure to rank the nodes of the network for the purpose of information retrieval.

(2) We develop a novel solution to this problem. We use a CNN model to extract the content-level features of each node and then use a GCN model to encode the features of the neighboring nodes to represent the node. The new representations of the nodes by GCN are further converted to the Gaussian distribution parameters, including the mean and the covariance for each node, by an encoder. Finally, the Gaussian distribution parameters are decoded to a node identity probability vector. To learn the parameters, we model the learning problem by minimizing the loss of node identity decoding and network structure-preserving. Meanwhile, to utilize the query node, we try to define a set of positives that are supposed to be relevant to the query and returned by the retrieval system and a set of negative nodes, which is supposed to be ignored by the retrieval system. The labels of the positives/negatives are used to learn the distance between nodes.

(3) We design an optimization algorithm to solve the problem of the minimization problem modeled as above. Firstly, the labels of positive and negatives are based on the distance of Gaussian distributions of each pair of nodes measured by the Wasserstein distance, and an upper bound/lower bound of the distance. Secondly, the learning of the parameters of CNN, GCN, Gaussian distribution parameters, and upper bound/lower bound of the labels are learned jointly. Thirdly, learning and labeling are conducted iteratively in an algorithm.

Remark 1. Our work is based on the idea of learning a probabilistic model relying on an autoencoder architecture, which is well known in the literature as the variational autoencoder (VAE) proposed by Kingma and Welling [28]. Our cost function is different from the standard loss function used in VAEs.

1.4. Origination. Our paper is organized as follows. In Section 2, we introduce the novel method for the query-specific network embedding for the content-rich network. In Section 3, we evaluate the proposed method experimentally and compare it against the state-of-the-art. In Section 4, we give the conclusion of this paper.

2. Proposed Method

In this section, we will introduce our query-specific embedding method of a content-rich network. The embedding of nodes of the network is conducted at two layers. The first layer is the representation of the content of each node by using the convolutional representation method. The second layer is the representation of the node neighborhood by using graph convolutional representation of the content of the neighboring nodes, where the embedding of the nodes is in the Wasserstein space. To learn the parameters, of the model, we consider the problem of query-specific search, while keeping the network structure.

2.1. Content-Rich Graph Embedding

2.1.1. Convolutional Representation of Node Content. In this subsection, we will discuss the presentation of the node contents. Given a node of the network, v, we assume its content is a text, which can be denoted as a sequence of words, and each word is represented as an embedding vector:

$$v = \{x_1, \ldots, x_N\}, \tag{1}$$

where $x_i \in R^d$ is the embedding vector of the i-th word, d is the dimension of the word embedding space, and N is the number of words in the text of node v. To represent the text, we employ a CNN model with one convolutional layer and a max-pooling layer.

In the convolutional layer, we have a filter bank, $F = \{f_1, \ldots, f_{|F|}\}$, where $f_k \in R^{(d\times s)}$ is the k-th filter, which filters the word embeddings of a window size of s words. The response of the k-th filter is calculated as follows:

$$q_{i,k} = \mathrm{Re}LU\left(f_k^{\top} x_{i:\, i+s-1} + b_k\right) \text{where } k = 1, \cdots, |F| \text{ and } i = 1, \cdots, N - s + 1, \tag{2}$$

where $x_{i:\, i+s-1} \in R^{(d\times s)}$ is the concatenation word embedding vectors of $x_i, \ldots, x_{i+s-1}$, and $b_k$is the bias parameter of the k-th parameter. $\mathrm{Re}LU(x) = \max(x, 0)$ is the activation function of rectified linear unit ($\mathrm{Re}LU$) [29–32].

In the max-pooling layer, the maximum response of each filter is selected as the output of the layer:

$$\widehat{q}_k = \max\left(q_{1,k}, \ldots q_{(|v|-s),k}\right), \quad k = 1, \ldots, |F|. \tag{3}$$

The convolutional representation of the content of the node v is the vector of the max-pooling outputs of the $|F|$ filter outputs:

$$u = \left[\widehat{q}_1, \ldots, \widehat{q}_{|F|}\right]^{\top} \in R^{|F|}. \tag{4}$$

2.1.2. Neighborhood Convolutional Encoder-Decoder. In this subsection, we will introduce the neighborhood representation of a node from the content of its neighboring nodes. To this end, we apply an encoding-decoding methodology to code the neighborhood of each node to the Wasserstein space.

(1) Graph Convolutional Encoder. We assume the network is denoted as

$$G = (V, E), \text{ where } V = \{v_1, \ldots, v_n\}, \tag{5}$$

is the set of nodes, v_i is the i-th node, and n is the number of nodes. $E = \{1,0\}^{n\times n}$ is the matrix of edges, where

$$e_{ij} = \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ are connected,} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Thus, we can denote the set of neighbors of a node v_i as $N_i = \left\{v_j \mid e_{ij} = 1 \text{ or } e_{ji} = 1, \quad j = 1,\ldots,n\right\}$. To represent the neighborhood of a node, we normalize the edge weights of its neighbors as

$$\widehat{e}_{ij} = \begin{cases} \dfrac{1+e_{ij}}{1+\sum_{v_{j'}\in N_i} e_{ij'}}, & \text{if } vj \in N_i, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

To utilize the neighborhood to represent a node, we employ the deep graph convolutional network (GCN). The input layer of the GCN is the convolutional representations of nodes, $\{u_1,\ldots,u_n\}$, where u_i is the representation of content of the i-th node,

$$v_i^0 = u_i. \tag{8}$$

For the l-th layer of GCN, the output is calculated as

$$v_i^{l+1} = \tanh\left(W^l\left(\sum_{j\in N_i} \widehat{e}_{ij} v_j^l\right) + b^l\right), \tag{9}$$

where v_j^l is the input of the l-th layer of the j-th node and the neighboring nodes' content representations are linearly combined with normalized edge weights and then pass through a full-connection layer with a tanh activation layer, and W^l and b^l are weight and bias parameters. The number of GCN layers is L, and the output of the last layer of GCN for the i-th node is denoted as v_j^L.

(2) Gaussian-Based Encoder. We further assume that each node is generated from a lower-dimensional Gaussian distribution in the Wasserstein space. The Gaussian distribution is characterized as

$$v_i \sim N(\mu_i, \Sigma_i), \tag{10}$$

where $\mu_i \in R^g$ is the mean of the distribution, $\Sigma_i \in R^{g\times g}$ is the covariance matrix of the distribution, and g is the dimension of the embedding of the lower-dimensional Gaussian distribution. In our work, we assume that the covariance matrix is a diagonal matrix:

$$\Sigma_i = \text{diag}(\sigma_i), \text{ where } \sigma_i \in R^g. \tag{11}$$

To bridge the network structure and the distribution of the node, we assume that the mean and covariance can be reconstructed from the GCN network output, by two full-connection layers:

$$\begin{aligned} \mu_i &= \Theta v_i^L + \theta, \\ \sigma i &= \text{Elu}\left(\Psi v_i^L + \psi\right) + 1, \end{aligned} \tag{12}$$

where Θ, Ψ are the weight matrix, while θ and ψ are the bias vectors.

$$\text{Elu}(x) = \begin{cases} x, & \text{if } x > 0, \\ \alpha \times (e^x - 1), & \text{if } x \le 0, \end{cases} \tag{13}$$

is the Elu (exponential linear unit) activation function [33–36] and $\text{Elu}(x) + 1$ is used to guarantee that σ_i is a positive vector. In this way, each node is encoded to a Gaussian distribution in Wasserstein space.

To measure the dissimilarity between the two nodes, *vi* and *vj*, from their Gaussian distributions, we apply the 2nd Wasserstein distance, as follows:

$$\begin{aligned} \text{dist}(vi, vj) &= W_2\left(N(\mu_i,\Sigma_i), N\left(\mu_j,\Sigma_j\right)\right) \\ &= \left(\left\|\mu_i - \mu_j\right\|_2^2 + \left\|\Sigma_i^{(1/2)} - \Sigma_j^{(1/2)}\right\|_2^2\right). \end{aligned} \tag{14}$$

(3) Node-Identity Decoder. After we have the Gaussian-based encoding result for each node, we want to decode it to its original identity in the graph. Thus, we design a decoder to convert its Gaussian distribution to the probabilities of being the nodes of G. In the decoder, we first sample the data from the Gaussian distribution to obtain a representation of the node, as follows:

$$z_i = \mu_i + \sigma_i \times \psi, \psi \sim N(0, I), \tag{15}$$

where ψ is the sampled weight parameter. Then we calculate the reconstructed node probability function by a full-connection layer and a sigmoid activation layer.

$$\begin{aligned} u_i &= \text{ReLU}(\Omega z_i + \omega), \\ p_i &= \text{sigmoid}(\Upsilon u_i + v) \in [0,1]^n, \end{aligned} \tag{16}$$

where the j-th dimension of p_i, p_{ij} is the probability of the node being the j-th node of the network.

2.2. Problem Modeling and Solving. To learn the parameters of the CNN, GCN, and Gaussian-based encoder-decoder, we consider the following problems.

2.2.1. Decoder Loss of Node Identification. Since the Gaussian-based encoder-decoder is designed to identify the node from the graph, we propose to minimize the loss of the node identification measured the by cross-entropy loss as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} \pi_{ij} \log\left(p_{ij}\right), \tag{17}$$

where $\pi_{ij} = 1$ if v_i is the j-th node, and 0 otherwise.

2.2.2. Neighborhood Structure Preservation. With the new coding of each node as a Gaussian distribution in the Wasserstein space, we hope the neighborhood structure can be preserved. To this end, we firstly define a set of triplets of nodes, $T = \left\{(i,j,k) | e_{ij} = 1, e_{ik} = 0, \quad i,j,k = 1,\ldots,n\right\}$,

where v_i and v_j are connected, and v_i and v_k are disconnected in graph G. The energy between two nodes v_i and v_j in the graph is also defined as the Wasserstein distance:

$$\varpi_{ij} = \operatorname{dist}\left(v_i, v_j\right). \tag{18}$$

To keep the structure of the network, we propose minimizing the squared energy between the connected nodes and maximizing the exponential of negative entity between the disconnected nodes:

$$\min \sum_{(i,j,k)\in T} \left(\varpi_{ij}^2 + \exp\left(-\varpi_{ik}\right)\right). \tag{19}$$

With minimizing this objective, we hope the learned Gaussian distributions of the connected nodes are close in the Wasserstein space, while that of the disconnected nodes are far from each other in the Wasserstein space. Thus, the network structure is preserved in the Wasserstein space.

2.2.3. Query-Specific Distance Supervision. In our problem setting, we already have a known query node, and the task is to find similar nodes in the network. We assume the q-th node is the query node, v_q. To use the query to guild the learning process, we define a label for each node to indicate if it is similar to the query. By default, y_q is similar to itself; thus, $y_q = 1$. For the other nodes, it is difficult to define the label accurately. Thus, we develop a heuristic method to learn the labels from the Wasserstein distance between the query node and a given candidate node. For this purpose, we split the distance range to three intervals, divided by an upper bound, u, and a lower bound, l, where $l < u$. The labeling process selects the nodes which have a Wasserstein distance to query larger than u as positives and selects the nodes with a Wasserstein distance to query smaller than l as negative. The nodes whose distance to query is between u and l are left to be ambiguous. Thus the label of a node vi is defined as

$$yi = \begin{cases} 1, & \text{if } i = q, \\ 1, & \text{if } \operatorname{dist}\left(v_i, v_q\right) \le l, \text{ and } i \ne q, \\ -1, & if \text{ dist } (vi, vq) \ge u \text{ and } i \ne q, \\ \text{None}, & \text{otherwise.} \end{cases} \tag{20}$$

We further define an indicator to indicate if v_i is labeled as $\beta_i = 1$ and 0 otherwise. The range of distance between u and l is the range of ambiguous nodes, and we define $u - l$ as the ambiguous range. For the labeled nodes, we minimize a linear loss of $L(v_i, v_j; y_i) = y_i \times \operatorname{dist}(v_i, v_q)$. Meanwhile, we also hope the ambiguous range can be as small as possible so that more nodes can be labeled. Thus, we minimize $u - l$. The overall minimization problem is the combination of both the minimization of the linear losses of the labeled and the ambiguous range:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \beta_i y_i \operatorname{dist}\left(v_i, v_q\right) + \gamma(u - l) \\ \text{s.t.} \quad & 0 \le l \le u, \end{aligned} \tag{21}$$

where γ is a regularization parameter. In this way, for the positive nodes which are labeled to be similar to the query, their distance to the query should be minimized, while the distance to the query for the negatives will be maximized.

The overall optimization problem is the combination of the three subproblems:

$$\begin{aligned} \min{}_{\Phi,u,l}\, o(\Phi, u, l) \quad & = \left\{ \sum_{i=1}^{n}\sum_{i=1}^{n} \pi_{ij} \log\left(p_{ij}\right) + C_1 \sum_{(i,j,k)\in T} \left(\varpi_{ij}^2 + \exp\left(-\varpi_{ik}\right)\right) + C_2 \left(\sum_{i=1}^{n} \beta_i y_i \operatorname{dist}\left(v_i, v_q\right) + \gamma(u - l) \right) + C_3 \|\Phi\|_2^2 \right. \\ \text{s.t.} \quad & 0 \le l \le u, \end{aligned} \tag{22}$$

where Φ represents the set of parameters of CNN, GCN, and Gaussian-based encoder-decoder. Solving this problem directly is difficult, because the label definition, ambiguous range parameters, and the Gaussian-based encoder parameter are coupled. To be specific, the label is defined over the ambiguous range and the distance of the Gaussian-based distributions of the nodes, while the parameters of the Gaussian-based distributions are learned from the labels of the nodes. To solve this problem, we use the fixed point iteration method [37–40] in an iteration algorithm.

We firstly fix the parameters of Φ, and the ambiguous range parameters, u and l, to update the labels according to (20).

Then we fix the labels and ambiguous range parameters to update the parameters of Φ by solving the following problem:

$$\min{}_{\Phi}\, o_1(\Phi) = \left\{ \sum_{i=1}^{n}\sum_{i=1}^{n} \pi_{ij} \log\left(p_{ij}\right) + C_1 \sum_{(i,j,k)\in T} \left(\varpi_{ij}^2 + \exp\left(-\varpi_{ik}\right)\right) + C_2 \left(\sum_{i=1}^{N} \beta_i y_i \operatorname{dist}\left(v_i, v_q\right) \right) + C_3 \|\Phi\|_2^2 \right\}. \tag{23}$$

This problem is solved by the back-propagation algorithm with the ADAM optimizer [41].

Finally, we fix Φ and the labels to update the ambiguous range parameters as follows:

$$\begin{aligned} \min_{u,l} \quad & o_2(u,l) = C_2\gamma(u-l) \\ \text{s.t.} \quad & 0 \le l \le u. \end{aligned} \tag{24}$$

We use the gradient descent algorithm to solve this problem:

$$\begin{aligned} u &\longleftarrow u - \rho\frac{\partial o_2(u,l)}{\partial u}, \\ l &\longleftarrow l - \rho\frac{\partial o_2(u,l)}{\partial l}, \end{aligned} \tag{25}$$

where ρ is the descent step size. Since $(\partial o_2(u,l)/\partial u) = 1$ and $(\partial o_2(u,l)/\partial l) = -1$, in each descent step, u is increased by ρ, while l is decreased by ρ, until $u = l$.

3. Experimental Results

In this section, we conduct experiments over benchmark data sets of networks.

3.1. Datasets. In the experiments, we use the following benchmark datasets of the innovation networks.

The first dataset is Cora dataset [42]. This dataset is a network of research articles of machine learning topics. The research articles are treated as nodes, and the edges are the citations of papers. The abstract of each article is treated as the content of each node. This network has only 2,211 nodes and 5,214 edges. The content of articles has around 170 words on average; the total number of unique works of nodes in this network is 12,619.

The second dataset is Citeseer dataset [43]. This dataset is a network of scientific articles of ten different multidisciplinary topics. Each node is also a research article, and each edge is the same citation relation connecting two articles. But in this network, the content of each node is its title, not the abstract. The number of nodes of this network is 4,610, and the number of edges is 5,923. The number of words of content is 10 on average, and the number of unique words overall is 5,523.

The third dataset is the DBLP dataset [44]. This network has the bibliography data of 13,404 articles of computer science. Each node is an article, and each edge is a citation relation. The articles are labeled by four different research topics, including artificial intelligence and computer vision. The content of each node is also the title. The number of edges is 39,861. The average length of the content is 10, and the size of the unique word set is 8,501.

3.2. Experimental Settings. To conduct the experiments, we set up the following protocols by using the leave-one-out validation. For each network, we leave one node out as a query node, and the remaining nodes as the candidate nodes to be retrieved. Since our data is research articles, we define the relevance of two research articles according to their small areas. If an article is in the same small areas as the query article, then it is defined as a positive node. The task is to retrieve as many positives as possible while keeping the negatives out of search results as much as possible. This process is repeated for each node of the network by turns; that is, each node is treated as a query node one by one. Then we apply our algorithm to learn the embeddings of the nodes and use the Wasserstein distances to measure the dissimilarity between the query and a candidate node and rank them according to returning the nodes with the smallest Wasserstein distances. To measure the performance of the retrieval results, we use the mAP (mean average precision) [45, 46].

Remark 2. *mAP* is an effective measure of database retrieval performance. Given a set of queries, Q, the retrieval systems return a list of ranked database objects for each query. For each query, $q \in Q$, we can calculate a precision at each rank k:

$$\text{precision}\ (q)@k = \frac{TP\ (q)@k}{k}, \tag{26}$$

where $TP(q)@k$ is the number of return objects relevant to q at top k ranks. The average precision (AP) of q is calculated as the average overall ranks:

$$AP(q) = \frac{1}{K}\sum_{k=1}^{K}\text{precision}\ (q)@k, \tag{27}$$

while mAP is the mean of AP over the queries at Q.

$$mAP = \frac{1}{|Q|}\sum_{q\in Q}AP(q), \tag{28}$$

where $|Q|$ is the size of Q.

3.3. Experimental Results. We compare the proposed method, named as Query-Specific Deep Embedding of Content-Rich Network (QDECN), against the network-based ranking methods first and then against the other network embedding methods. For the comparison to the network embedding methods, we first embed the nodes of the network and then calculate the dot-product scores of their embedding vectors and the query as the ranking scores for the purpose of retrieval.

3.3.1. Comparison with Network-Based Ranking Methods. We compared the following methods of network-based ranking: graph transduction (GT) [27], tensor product graph diffusion (TPGD) [26], and Query-Specific Optimal Networks (QUINT) [25]. The comparison results are shown in Table 1. From this table, we can see that the proposed method QDECN outperforms the other methods in all cases. This is not surprising at all due to the following reasons.

(1) QDECN is the only method that explores the content of the nodes of a network, while, for all the other

Table 1: Comparison of results of network-based ranking methods.

y	GT	TPGD	QUINT	QDECN
Cora	0.428	0.431	0.486	0.514
Citeseer	0.394	0.406	0.461	0.497
DBLP	0.355	0.361	0.377	0.425

Table 2: Comparison of results of network embedding methods.

Dataset	DRNE	DVNE	Net2Net-NE	QDECN
Cora	0.430	0.433	0.492	0.514
Citeseer	0.403	0.418	0.477	0.497
DBLP	0.397	0.403	0.413	0.425

methods, they only utilize the network structure information, such as edge data. However, in these datasets of innovation networks, two articles may not have the citation relation, but according to their content similarity, they should belong to the same small area. QDECN not only codes the content of the nodes to its representation but also leverage the content features of its neighboring nodes. So it has the capability to learn from both the node content and the edges of the network, while the other network-based ranking methods only learn from the network structure itself.

(2) QDECN is the only method that employs network embedding technology to improve retrieval performance. The remaining methods, such as QUINT and TPGD, aim to learn a better network affinity matrix to guild the learning of the ranking score, but they are still failing to the same schema as GT. The network-based ranking methods use the network affinity matrix to regularize the learning of ranking score, but the network embedding methods map the nodes to a low-dimensional continues vectors, which contains the richer information about the network and instinct information about the relevance of nodes; thus, it is a better choice for the node relevance search tasks.

(3) Only QDECN and QUINT adjust the learning of the network parameters according to the query node. This method can learn a better network representation which is optimal for finding the relevant nodes to the node. This setting does not guarantee the learned network representation is optimal for other tasks, but, for the given query node, it gives better results than other methods. Please also note that the supervision information of QUINT is richer than QDECN, since the positive/negative nodes of QUINT are given as ground truth, but for QDECN the positives and negatives are both learned by the algorithm. However, QDECN still outperforms QUINT in all cases.

3.3.2. Comparison to Network Embedding Methods. We compare QDECN to the following network embedding methods: Network to-Network Network Embedding (Net2Net-NE) [2], Deep Variational Network Embedding in Wasserstein Space (DVNE) [6], and Deep Recursive Network Embedding (DRNE) [23]. The results are shown in Table 2. We have the following observations from this table.

(1) Again, QDECN obtains better results than the other methods. Compared to DRNE and DVNE, the proposed method and Net2Net-NE can use the content of the nodes to enrich the embedding results. Compared to the Net2Net-NE itself, our method can further sense which node is the query node and take this advantage to guild the embedding process, which Net2Net-NE cannot. Due to the above reasons, the overall results of QDECN are better than the others.

(2) DRNE and DVNE are common network embedding methods, which have no supervisor of node content and query node. Meanwhile, Net2Net-NE has the supervision from the content of the node but cannot access the query. Thus, this is not a fair competition. However, our method is the very first algorithm that can use both the node content and query node of the networks. The fact that QDECN gives the best results is a piece of strong evidence that it is necessary to develop an effective method to take both node content and query node into account during the process of network embedding, especially for the purpose of information retrieval.

Remark 3. To obtain the results of Tables 1 and 2, we set the values of the parameters as follows: for Cora dataset, $C_1 = 100$, $C_2 = 100$, $C_3 = 100$; for Citeseer dataset, $C_1 = 10$, $C_2 = 10$, $C_3 = 100$; and for DBLP, $C_1 = 100$, $C_2 = 100$, $C_3 = 100$. There are two dimensionalities of the latent spaces, d and g, and we set their values to 300 and 500 for three datasets. The learning rate of the optimizer is set to 0.01 for all three datasets.

3.4. Parameter Analysis. In our model, there are three tradeoff parameters, C_1, C_2, and C_3. We conduct experiments to analyse them one by one.

3.4.1. Analysis of C_1. C_1 is the weight of the network structure preservation loss term. We vary the value of C_1 and measure the changes of *mAP* over three different datasets and plot the curves in Figure 1. We can see that the performance of our algorithm keeps improving when the value of C_1 is increased. Since this parameter is the weight of the network neighborhood structure preservation term, this phenomenon indicates that the neighborhood structure plays an important role in a good quality network embedding process and also is critical for the node-level information relevance search problem. Moreover, we also observe that performance improvement becomes minor after a certain value. For example, for the DBLP network, this value is 1, while, for the Citeseer network, this value is 10.

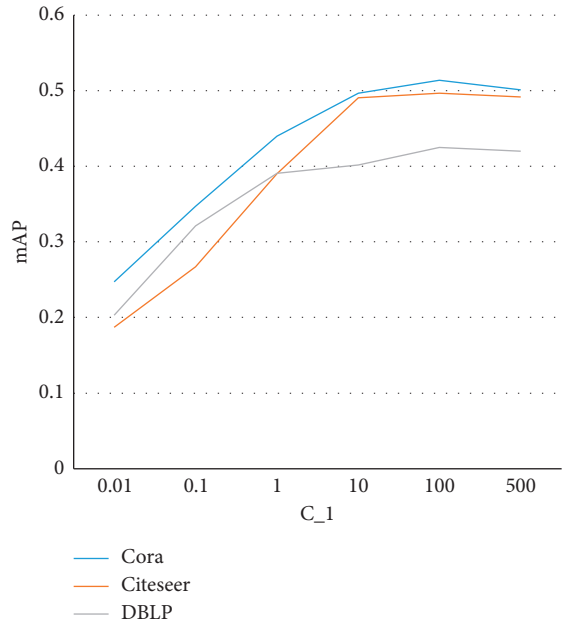

Figure 1: Sensitivity analysis of C_1.

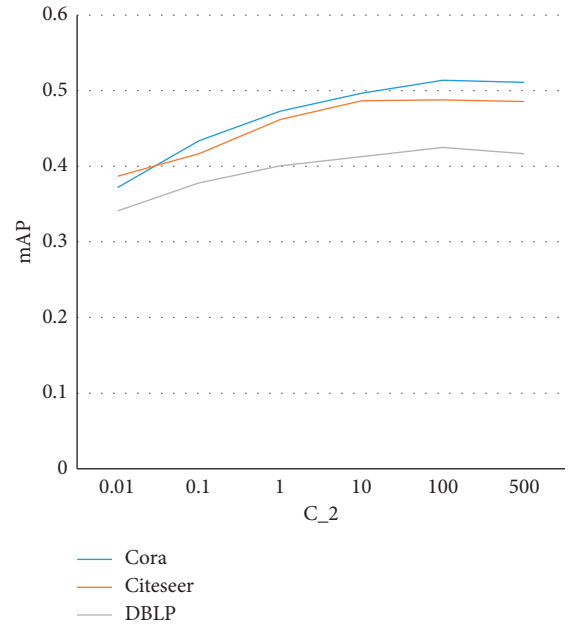

Figure 2: Sensitivity analysis of C_2.

3.4.2. Analysis of C_2. C_2 is the weight of the loss term of the supervision of positive/negatives nodes regarding the Wasserstein distance. The performance curves of different C_2 values are shown in Figure 2. From this figure, we can also conclude that overall a larger value of C_2 can give a better performance in terms of mAP. But the improvement is limited, and the performance is not sensitive to the change of the values of C_2. A possible reason is that the positive and negative labeling is not at the level of ground truth but is estimated by the upper/lower bound. However, the upper/lower bound itself is learned as variable. Thus, in nature, the learning process is an unsupervised learning process. So the improvement is not comparable to supervised learning. Even though it is unsupervised, we still can see the improvements with increasing C_2, which is the benefit from the sensing of the query node.

3.5. Computational Intensity. In this section, we study how computational intensive the proposed method is. The average running time (in seconds) of the learning process for a query over the datasets of Cora, Citeseer, and DBLP is 43.66, 93.34, and 437.04. The running time is rather long because the model needs to be retrained for each query node, and the neighborhood preserving term in (19) scale as n_3 as we are considering all possible triples within the network. This problem is even more serious with the network being large. Thus, we proposed to reduce the size of training triplet set size. To this end, for each node, instead of using all the disconnected nodes to construct the training triplets, we only sample a few disconnected node for this purpose. After this change, the running time is reduced to 21.12, 55.07, and 94.34, respectively.

3.6. Contribution of Different Objective Terms during the Training Phase. In this section, we analyse the contributions of the different terms of the objective for the database of Cora. To measure the contribution of a term, we firstly remove this term from the objective and learn the model to retrieve the nodes for queries and calculate the mAP of the retrieval results. Then we add the term back to the objective and measure the retrieval results by mAP again. The contribution of this term is measured by the improvement of the mAP after the term is added to the objective. The term-wise mAP improvement is reported in Figure 3. From this figure, we can see that the query-specific distance supervision term gives the largest contribution, while the regularization term has the least contribution. The second and third significant contributions are from the node identification term and the neighborhood structure preservation term.

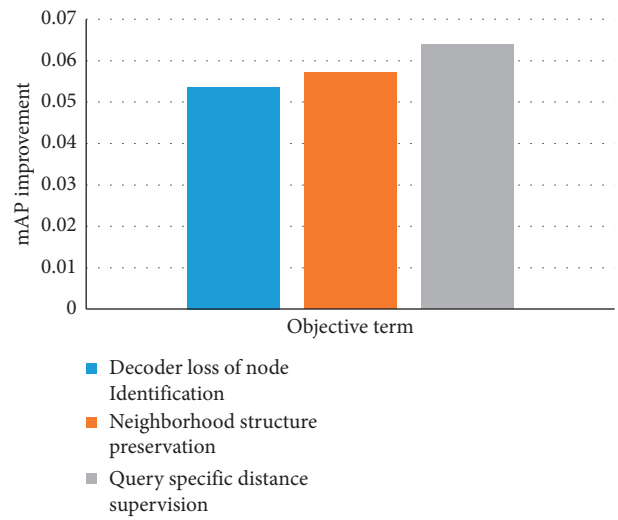

Figure 3: Term-wise mAP improvement.

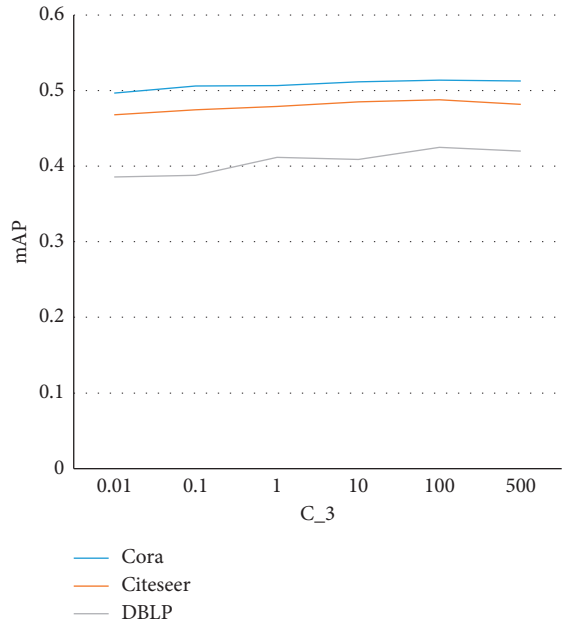

Figure 4: Sensitivity analysis of C_3.

4. Conclusions

In this paper, we develop a novel method of network embedding, for the content-rich network, for the purpose of node-level information retrieval. We firstly use a CNN to extract features from the content and then use a GCN to code the features of the neighboring nodes, and finally use a deep encoder-decoder to map these features to a Gaussian distribution and convert it to the node's identity. The learning of the parameters is performed by minimizing a loss function. In the loss function, except for the node identification loss, the neighborhood preservation loss, and the complexity of models, we also consider the query node regularization problem. For this purpose, we define positive/negative nodes according to the Wasserstein distance between the query and the candidate nodes. Experimental results show the advantages of the proposed method which embeds the content-rich network guided by the query node.

Appendix

Sensitivity to Parameter C_3

The tradeoff parameter C_3 is the weight of the squared ℓ_2 norms of the parameters of models to control the complexity of the models. The changes of the performances of the algorithm with different values of C_3 are shown in Figure 4. It is shown that the performance remains stable with the changing C_3. The algorithm is insensitive to this parameter.

Data Availability

All the data sources used in this work to produce the experimental results are available online.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was funded by the National Natural Science Foundation of China (Project nos 71704036 and 71473062) and Social Science Foundation of Ministry of Education of China (Project nos. 16YJC630061 and 19YJA790087).

References

[1] H. Chen and B. M. Sharp, "Content-rich biological network constructed by mining pubmed abstracts," *BMC Bioinformatics*, vol. 5, no. 1, p. 147, 2004.

[2] Z. He, J. Liu, N. Li, and Y. Huang, "Learning network-to-network model for contentrich network embedding," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1037–1045, ACM, Anchorage, AK, USA, July 2019.

[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, Geneva; Switzerland, May 2015.

[4] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, ACM, San Francisco, CA, USA, August 2016.

[5] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9931-9932, 2017.

[6] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in wasserstein space," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2827–2836, ACM, London, UK, July 2018.

[7] H. Gao, J. Pei, and H. Huang, "Progan: network embedding via proximity generative adversarial network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 1308–1316, ACM, New York, NY, USA, July 2019.

[8] D. Jin, R. A. Rossi, E. Koh, S. Kim, A. Rao, and D. Koutra, "Latent network summarization: bridging network embedding and summarization," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 987–997, ACM, Anchorage AK USA, August 2019.

[9] N. Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu, "Is a single vector enough?: exploring node polysemy for network embedding," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 932–940, ACM, New York, NY, USA, July 2019.

[10] D. Yang, P. Rosso, B. Li, and P. Cudre-Mauroux, "Nodesketch: highly-efficient graph embeddings via recursive sketching," in *: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 1162–1172, ACM, New York, NY, USA, 2019.

[11] S. n. Dominich and A. Skrop, "Pagerank and interaction information retrieval," *Journal of the American Society for*

Information Science and Technology, vol. 56, no. 1, pp. 63–69, 2005.

[12] S.C. Geyik, S. Ambler, and K. Kenthapadi, "Fairness-aware ranking in search & recommendation systems with application to linkedin talent search," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 2221–2231, ACM, New York, NY, USA, July 2019.

[13] J. W. Hughes, K. H. Chang, and R. Zhang, "Generating better search engine text advertisements with deep reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 2269–2277, ACM, New York, NY, USA, 2019.

[14] T. Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends R ◯ in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[15] Z. Wang, C. Long, G. Cong, and C. Ju, "Effective and efficient sports play retrieval with deep representation learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 499–509, ACM, New York, NY, USA, July 2019.

[16] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao, "Event extraction via dynamic multi-pooling convolutional neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, pp. 167–176, Beijing, China, July 2015.

[17] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multicolumn convolutional neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, pp. 260–269, Beijing, China, July 2015.

[18] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan, "Transferable Representation Learning with Deep Adaptation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 3071–3085, 2018.

[19] W. L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. J. Hsieh, "Cluster-gcn: an efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 257–266, ACM, New York, NY, USA, August 2019.

[20] H. Gao, J. Pei, and H. Huang, "Conditional random field enhanced graph convolutional neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 276–284, ACM, New York, NY, USA, August 2019.

[21] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 723–731, ACM, New York, NY, USA, July 2019.

[22] D. Zügner and S. Günnemann, "Certifiable robustness and robust training for graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'19*, pp. 246–256, ACM, New York, NY, USA, July 2019.

[23] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2357–2366, ACM, London, UK, August 2018.

[24] S. Wang, J. Arroyo, J. T. Vogelstein, and C. E. Priebe, "Joint Embedding of Graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, 2019.

[25] L. Li, Y. Yao, J. Tang, W. Fan, and H. Tong, "Quint: on query-specific optimal networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 985–994, ACM, San Francisco, CA, USA, August 2016.

[26] X. Yang, L. Prasad, and L. J. Latecki, "Affinity learning with diffusion on tensor product graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 28–38, 2013.

[27] X. Bai, X. Yang, L. J. Latecki, W. Liu, and Z. Tu, "Learning context-sensitive shape similarity by graph transduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 861–874, 2010.

[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, https://arxiv.org/abs/1312.6114.

[29] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018, https://arxiv.org/abs/1803.08375.

[30] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8609–8613, IEEE, Vancouver, Canada, October 2013.

[31] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Killarney, Ireland, July 2015.

[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, Haifa, Israel, June 2010.

[33] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," 2015, https://arxiv.org/abs/1511.07289.

[34] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming, "Improving deep neural network with multiple parametric exponential linear units," *Neurocomputing*, vol. 301, pp. 11–24, 2018.

[35] A. Shah, E. Kadam, H. Shah, S. Shinde, and S. Shingade, "Deep residual networks with exponential linear unit," in *Proceedings of the Third International Symposium on Computer Vision and the Internet*, pp. 59–65, Jaipur, India, September 2016.

[36] L. Trottier, P. Gigu, B. Chaib-Draa et al., "Parametric exponential linear unit for deep convolutional neural networks," in *Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 207–214, IEEE, Cancun, Mexico, January 2017.

[37] S. Ishikawa, "Fixed points by a new iteration method," *Proceedings of the American Mathematical Society*, vol. 44, no. 1, p. 147, 1974.

[38] C. Martinez-Yanes and H.-K. Xu, "Strong convergence of the cq method for fixed point iteration processes," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 64, no. 11, pp. 2400–2411, 2006.

[39] E. D. Popova, "Generalization of a parametric fixed-point iteration," in *Proceedings of the PAMM: Proceedings in Applied Mathematics and Mechanics*, vol. 4, Wiley Online Library, Hoboken, NJ, USA, pp. 680-681, 2004.

[40] B. E. Rhoades, "Comments on two fixed point iteration methods," *Journal of Mathematical Analysis and Applications*, vol. 56, no. 3, pp. 741–750, 1976.

[41] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[42] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[43] K. W. Lim and W. Buntine, "Bibliographic analysis with the citation network topic model," 2016, https://arxiv.org/abs/1609.06826.

[44] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 990–998, ACM, Las Vegas, NV, USA, August 2008.

[45] P. Henderson and V. Ferrari, "End-to-end training of object class detectors for mean average precision," in *Proceedings of the Asian Conference on Computer Vision*, Springer, Berlin, Germany, pp. 198–213, 2016.

[46] K. Li, Z. Huang, Y. C. Cheng, and C. H. Lee, "A maximal figure-of-merit learning approach to maximizing mean average precision with deep neural network based classifiers," in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4503–4507, IEEE, Florence, Italy, 2014.