An Inductive Content-Augmented Network Embedding Model for Edge Artificial Intelligence

Bo Yuan, *Member, IEEE*, John Panneerselvam, *Member, IEEE*, Lu Liu, *Member, IEEE*, Nick Antonopoulos, and Yao Lu, *Member, IEEE*

***Abstract*—Real-time data processing applications demand dynamic resource provisioning and efficient service discovery, which is particularly challenging in resource-constraint edge computing environments. Network embedding techniques can potentially aid effective resource discovery services in edge environments, by achieving a proximity-preserving representation of the network resources. Most of the existing techniques of network embedding fail to capture accurate proximity information among the network nodes and further lack exploiting information beyond the second-order neighbourhood. This paper leverages artificial intelligence for network representation and proposes a deep learning model, named inductive content augmented network embedding (ICANE), which integrates the network structure and resource content attributes into a feature vector. Secondly, a hierarchical aggregation approach is introduced to explicitly learn the network representation through sampling the nodes and aggregating features from the higher-order neighbourhood. A semantic proximity search model is then designed to generate the top-k ranking of relevant nodes using the learned network representation. Experiments conducted on real-world datasets demonstrate the superiority of the proposed model over the existing popular methods in terms of resource discovery and the query resolving performance.**

***Index Terms*—Artificial intelligence (AI), deep learning, edge computing, network embedding, resource discovery.**

Manuscript received February 16, 2019; accepted February 24, 2019. Date of publication March 4, 2019; date of current version July 3, 2019. This work was supported in part by the Natural Science Foundation of Jiangsu Province under Grant BK20170069, in part by the UK–Jiangsu 20-20 World-Class University Initiative, and in part by the UK–Jiangsu 20-20 Initiative Pump Priming Grant. Paper no. TII-19-0497. *(Corresponding author: Lu Liu.)*

B. Yuan is with the Key Laboratory of Embedded System and Service Computing, Tongji University, Shanghai 200072, China, and also with the School of Electronics, Computing and Mathematics, University of Derby, DE221GB Derby, U.K. (e-mail: b.yuan@derby.ac.uk).

J. Panneerselvam is with the School of Electronics, Computing and Mathematics, University of Derby, DE221GB Derby, U.K. (e-mail: j.panneerselvam@derby.ac.uk).

L. Liu is with the Department of Informatics, University of Leicester, LE17RH Leicester, U.K. (e-mail: ll297@leicester.ac.uk).

N. Antonopoulos is with the Edinburgh Napier University, EH114BN Edinburgh, U.K. (e-mail: n.antonopoulos@napier.ac.uk).

Y. Lu is with the School of Electronics, Computing and Mathematics, University of Derby, DE221GB Derby, U.K., and also with the School of Computer Science and Telecommunication Engineering and the Jiangsu Key Laboratory of Security Techniques for Industrial Cyberspace, Jiangsu University, Jiangsu 212013, China (e-mail: y.lu2@derby.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2902877

I. Introduction

EDGE computing is an emerging paradigm that brings the computing resources near the edge in a close proximity to the end-users. Industrial Internet of Things (IIoT) applications recently exploit edge computing as a platform to process intensive applications near the edge. With a distributed processing of workloads among the end devices, edge-computing can significantly reduce both the time and computing overheads by shifting the computation towards the periphery of the network [1], [2], rather than relying on centralised servers. However, edge computing is still in its infancy; one notable area requiring immediate attention is efficient resource management among the end devices. Most of the IoT devices are resource-constraint and cannot handle overwhelming traffic and intensive execution beyond their capacity. This issue can be alleviated through distributed processing that requires partial knowledge of the network, such as topologies [3], resource locations [4], and social information [5]. Network embedding techniques [6] can help achieve this knowledge by facilitating efficient representation of the network structure and resources.

Network embedding aims at preserving the network structural properties whilst representing the network resources in a low-dimensional vector space. On one hand, existing embedding techniques have ignored exploiting information beyond the second-order neighbourhood proximity of the nodes, mostly limiting to just the local pair-wise node similarity and node-pair connectedness. Exploiting higher order node proximity information, such as nodes neighbour structures, community level similarity, etc., can help preserve rich and auxiliary node information and their special attributes such as roles, identity, and global ranking. On the other hand, existing strategies have not given enough emphasis to balance the tradeoff between complexity and accuracy of embedding algorithms, thereby generating inefficient and inappropriate representation of large-scale heterogeneous networks. Given such limitations of existing network embedding techniques, developing efficient network embedding methodologies remains an open issue.

To this end, this paper emphasises the incorporation of artificial intelligence (AI) techniques for edge computing whilst learning the network structure and content resources from higher order neighbourhood. AI techniques, especially deep learning models, can facilitate learning the higher order complex proximity and automatic preservation of the key properties of the network. With this in mind, this paper focuses on network

representation learning and resource discovery in order to facilitate a cost-effective, scalable infrastructure for edge artificial intelligence that can support efficient resource discovery in a fully decentralised edge environment. In order to achieve this objective, this paper proposes a general network representation learning model, named inductive content-augmented network embedding (ICANE). The proposed method effectively leverages rich features in higher order proximity networks to generate a low-dimensional vector representation. Furthermore, a deep neural network-based algorithm is proposed to adaptively aggregate information from node neighbourhoods, along with a semantic proximity search method for resource discovery. For each query node, proximity scores are computed for its potential target nodes based on their embedding distance to locate the top-k relevant nodes. We conduct experiments based on real-word network datasets to demonstrate the efficiency of our proposed method in terms of the achieved query correctness and in reducing query latency. The main contributions of this paper are summarised as follows.

1) A network representation learning model ICANE is proposed to learn the content textual features and network structural features, in order to transform heterogeneous network information into a low-dimensional, continuous, and proximity-preserved vector space, which can significantly improve the efficiency of downstream task execution in edge computing environments.
2) A semantic proximity search model based on a deep neural network is then proposed to generate top-k ranking of relevant nodes conditioned on specific queries. This model leverages the similarity of the query embedding and learned network embedding to facilitate efficient resource discovery in decentralised systems.

The rest of the paper is organised as follows. Section II reviews related works. Section III presents the problem definition and preliminaries on network embedding techniques and Section IV details the proposed method. Experiments and results are discussed in Section V and Section VI concludes this paper along with outlining our future research directions.

II. Related Works

Recently, network embedding has been identified as an effective way of learning and representing the network that can facilitate machine learning tasks with improved performance. Most common forms of network embedding techniques [7] fall into reconstruction-oriented and discrimination-oriented based categories. A continuous bag of words (CBOW) [8] model has been built based on a log-linear classifier by training prior and posterior sequence of words as inputs in order to classify the middle sequence of words. This model assigns more weight to the words located closer to the target words, and less weight to farther words, in which the word distance plays a crucial role. Skip-gram model [9] attempts to maximise the word classification based on another word located within the same sentence. Irrelevant words enjoying more weights in the input sequence would considerably affect the classification accuracy in such models.

Constrained DeepWalk [6], [10] exploits the random walk technique for sampling the word classification based on edge weights. This technique learns the independent latent representation for labels of vertices by relying on the intensity of user interactions. Users characterising more interactions pose better predictability, but users with fewer interactions pose greater learning challenges. Text-associated DeepWalk (TADW) [11] incorporates the network vertex features under a matrix factorisation framework, one of the resulting matrices containing more zero entries is then considered for the vertex representation, despite the data sparsity in real datasets. Although this approach can potentially balance the tradeoff between speed and accuracy, always relying on the zero-entry matrix might risk losing important latent information in the ignored product matrix. Long short-term memory (LSTM) [12] and its related models [13]–[15] have been built based on DeepWalk, with suitable modifications to the random walk phase. Such models have been primarily developed to overcome the drawbacks of the skip-gram model, which can only embed a single node, but LSTM can potentially embed a sequence of nodes [12]. Given the natural computational intensity of the intermediate levels of the memory cells in the LSTM model, its input phase based on random walk adds to its computational complexity. Nonetheless, LSTM has the capability of preserving long sequential relationships between inputs, which makes LSTM an advantage for learning higher order proximity in the networks.

Negative sampling strategy [8] is another deep learning approach used to generate fast approximations of the softmax layer. A nearest negative sampling model [16] discriminates negative nodes located closer to positive nodes, based on pretrained embedding. This pretrained approach may not hold good for evolving networks, since the discrimination boundary is always unique to any network comprising of heterogeneous nodes. Minimising distance-based loss [6] is the strategy of reducing loss by placing the node proximity calculated based on node embedding closer to the proximity that is calculated based on the observed edges. The computation accuracy has a serious impact in this approach, and a subsequent number of iterations may be required whilst minimising the distance loss. Another approach of reducing construction loss is to minimise the margin-based ranking loss [6], which focuses on increasing the similarity of a given node's embedding to relevant nodes than those of irrelevant nodes. With a presumption that social links are not always formed of common interests, a unified embedding approach has been developed [17] by the way of maximising a posterior area under the curve (AUC) ranking criterion. This approach linearly combines the information obtained from the first- and second-order proximities for network representation. The random walk phase of this approach is naturally expensive due to the intensity of learning required.

A supervised approach [18] has been used based on K-nearest neighbouring (KNN) technique to describe the local affinity structure around each node. In general, supervised learning might not be efficient enough to learn the affinities of newly joining nodes with robust characteristics in a dynamic edge network. Convolutional neural network (CNN) has been widely adopted in network embedding for designing and reformatting

TABLE I
COMMON NOTATIONS IN THIS PAPER

Symbols	Meaning
G	The given information network
V	The vertex(node) set in the network
E	The edge set in the network
$\|V\|$	The number of vertices
$\|E\|$	The number of edges
$\|A\|$	The attribute set of vertices
$\|C\|$	The label set of vertices
m	The number of vertex attributes
$\mathbf{x}$	The feature embedding vector
$\mathbf{z}_v$	The embedding vector of node v
q	The resource request query
K	The maximum learning depth
W	The weight matrix
σ	The sigmoid function

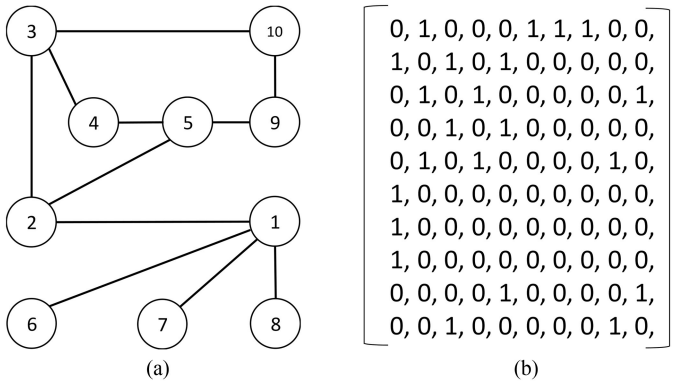


Fig. 1. Illustration of a toy network with ten vertices and its adjacency matrix with 10×10 dimensionality. (a) Toy network. (b) Adjacency matrix.

inputs [19] and to generalise neurons for learning graphs [20]. Deep neural networks are effective in learning the statistical properties of the nodes in the network, but generalising CNN may not be efficient in a dynamic network.

Most of the existing works on network embedding are not exploiting the network information to any degree beyond the node's second-order proximity. Another common drawback is their lack of semantic accuracy in resource-rich networks, since majority of the existing approaches solely focus either on network structure or network content. As a result, they potentially characterise a higher risk of generating poor representations for heterogeneous edge networks, thus not efficient enough to support accurate resource discovery. To address these challenges, this paper proposes a novel network representation learning model ICANE, integrated with a semantic search model with a global objective of facilitating efficient resource discovery in decentralised edge computing environments.

III. PROBLEM DEFINITION

This section introduces the preliminaries and terminologies used in this paper along with a formal definition of the network representation learning problem. Common notations used in this paper are listed in Table I.

A. Preliminaries and Notations

Definition 3.1 (Information networks): An information network can be represented as $G = (V, E, A, C)$, where V is a set of vertices, and $E \subseteq V \times V$ denotes a series of edges connecting those vertices. $A \in \mathbb{R}^{|V| \times m}$ represents the attribute matrix of the vertex, where m is the total number of attributes and $a_i \in A$ is the attribute information for vertex v_i, and finally, C is the label matrix with $c_i \in C$ related to a class label of node v_i. To avoid ambiguity, this paper uses the vertices and nodes in information networks interchangeably, and the edges in the network are considered to be undirected and unweighted, where edge (v_i, v_j) is identical to edge (v_j, v_i).

B. Problem Definitions

Fig. 1 shows an adjacency matrix for a toy graph with ten vertices. The adjacency matrix is the most intuitive way of representing the structure of a given network. However, the adjacency matrix still characterises a few known shortcomings including value sparsity, high dimensionality, high computation complexity, and the lack of semantic representation. Network learning techniques can potentially address such issues of the adjacency matrix by effectively aiding the representation of large-scale networks in a low-dimensional and densely populated vector space for facilitating efficient network analytical tasks. The formal problem definition is described as follows.

Definition 3.2 (Network representation learning, NRL): Suppose an information network $G = (V, E, A, C)$, NRL is the act of learning a mapping function $f(v) : v_i \mapsto h^k_{v_i} \in \mathbb{R}^d$ that transforms the network into a relatively low-dimensional vector space, where $h^k_{v_i}$ is the representation of vertex v_i learned at step k, and d is its dimension. The goal of NRL is to preserve the proximities of the original network in the mapped vector space. Such vector representation is termed as the embeddings of the corresponding network.

Definition 3.3 (Network resource discovery): A resource request node is defined as v_q, which represents a node v with a query q looking for targeting nodes (i.e., resource providers) with resource attributes that can satisfy $q \subseteq a_u$ for any given node u. Let $y_{u|q} = 1$ if $q \subseteq a_u$ and $y_{u|q} = 0$ for otherwise. The task of resource discovery is then simplified to perform a binary classification over the nodes in V for a given query q.

IV. PROPOSED APPROACH FOR NRL

In an edge-computing network, nodes are often heterogeneous in terms of their operating systems, network connections, types of offered resources, size of files, etc. This paper aims at mapping the heterogeneous nodes into a harmonious and accordant vector representation that preserves the original network proximity. Then the learned network representation is used as the meta-data to support semantic-aware resource discovery.

A. Network Embedding Generation

The process of network embedding generation is illustrated in Algorithm 1. This algorithm assumes that the parameters of K feature aggregation functions are known with the presumption of a prior knowledge of trained parameters, denoted

Algorithm 1: NETWORK EMBEDDING GENERATION.

Input: Information Network $G = (V, E, A, C)$; a sequence of attributes $\langle x_1, x_2, \ldots, x_n \rangle$ where $x_i \in A$; max learning depth K; embedding dimension d; weight matrices W^k; neighbourhood mappig function: $\mathcal{N} : v \mapsto 2^V$; batch size: T;

Output: The vector representation $\mathbf{z}_v$ in batch $\mathcal{B}$

1 Initialise $V_S = \emptyset$
2 **for** $v \in V$ **do**
3 **for** $i \leftarrow 1$ ***to*** K **do**
4 $V_S \leftarrow V_S \cup SampleNetwork(G, v, i)$
5 $F_i \leftarrow FeatureEncode(V_S)$
6 $\mathcal{B} \leftarrow MiniBatch(V_S, T)$
7 $\mathbf{h}_v^0 \leftarrow Initialization(F_0(\mathbf{x}_v \leftarrow (\mathbf{x}_{cont_v} \frown \mathbf{x}_{struc_v})))$
8 **for** $k \leftarrow 1$ ***to*** K **do**
9 **for** ***all*** $v \in \mathcal{B}$ **do**
10 $AGG_k \leftarrow LSTM_k(d, F_k, \mathcal{B})$
11 $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow AGG_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v) \cap \mathcal{B}\})$
12 $\mathbf{F_k} \leftarrow Dense(\mathbf{h}_v^{k-1} \frown \mathbf{h}_{\mathcal{N}(v)}^k)$
13 $\mathbf{h}_v^k \leftarrow mixPooling\{\sigma(W_{pool}^k \times \mathbf{F_k} + \mathbf{b})\}$
14 **return** $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{B}$

as $\text{AGG}_k, \forall k \in \{1, \ldots, K\}$. Besides, a set of weight matrices $W^k, k \in \{1, \ldots, K\}$ are used as weight factors between the different neural network layers. Algorithm 1 outputs the learned network embeddings for all the nodes in the network.

In the Algorithm 1, K is the maximum learning depth that corresponds to the number of layers in the model. When the learning process is completed, the final layer $\mathbf{h}_v^K$ is outputted as the trained network embedding, which is then used to perform the resource discovery tasks. The main idea behind Algorithm 1 is that at each iteration of **for** loop, the central node in the network aggregates information from its first-order proximity neighbours, and incrementally gathers more information by outreaching the network further. More specifically, at the initial stage, this algorithm samples a few required nodes for enabling the network embedding process. Lines 1–5 correspond to the network sampling phase and feature encoding. The $SampleNetwork$ is a uniform-draw function, which samples a uniform proportion of nodes in each iteration from the K-order neighbourhood. The objective of this neighbourhood sampling is to reduce in-memory computation and to generate a candidate set for the following network embedding learning model. The $FeatureEncode$ is a one-hot encoding function used to generate the feature matrix of the sampled nodes. Furthermore, to improve the computational efficiency and to reduce the memory consumption of the stochastic gradient descent, the training data are split into batches. Each batch contains the nodes required to compute the embeddings of the nodes within that corresponding batch. Next, each node v aggregates the representation of nodes in the first-order neighbourhood $\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}$ into a vector space.

The aggregation function AGG_k is a standard LSTM model with d memory units and T batches of input features, which utilises the representations generated at the previous iteration from $(k-1)$-order neighbourhood, and $\mathbf{h}_u^0$ is the input features of the central node. The output of the AGG_k function is a fixed length node sequence learned from the input features. To be more specific, each hidden LSTM unit comprises three gates, namely an input gate, a forget gate, and an output gate. This unit memorises the values over arbitrary sequences, where the three gates control the sequence flow of input and output arguments. The input gate uses the sigmoid function to select the values to be included into the unit, the forget gate uses the hyperbolic tangent function to control the values to be stored in the units, and the output gate uses a logistic function to determine the values to be outputted by the LSTM unit. Since LSTM can learn both long-term and short-term patterns from input features, it can capture the hop-by-hop node sequence and hidden patterns beyond the co-occurrences in the features.

In this process, the previously iterated representation drives the succeeding aggregation step. When the neighbourhood aggregation is completed, a given node's current representation $\mathbf{h}_v^{k-1}$ and its aggregated neighbourhood vector $\mathbf{h}_{N(v)}^k$ are concatenated, then fed though a fully connected LSTM layer using $dense$ function with a nonlinear activation function $\sigma = (1 + e^{-x})^{-1}$, which is known as the sigmoid function. σ transforms the representations used in the succeeding step for $\mathbf{h}_v^k$. The final representation is denoted as $\mathbf{z}_v \leftarrow \mathbf{h}_v^K$. The mix-pooling strategy [21] is applied to randomly output the mean and max of the hidden LSTM states from each network layer, which can effectively address the over-fitting problem during feature aggregation.

B. Heterogeneous Feature Aggregation

This section describes the feature information that is fed into the multilayer LSTM neural network in Algorithm 1. Our proposed approach incorporates feature aggregation functions, which obtain feature information from the k-order neighbourhood. Each function aggregates information at different learning depths, away from the central node, to generate embeddings for the entire information network. The characteristics of the proposed ICANE model can be summarised as follows: 1) adaptability—ICANE model can learn the network adaptively and incrementally even when the networks are constantly evolving to host new structure and contents; 2) scalability—ICANE model can learn large-scale networks with low latency; 3) low dimensionality—ICANE model can significantly reduce the dimensionality of the original network for achieving quicker convergence of the employed end-to-end machine learning algorithms; 4) community awareness—ICANE model can effectively reflect the semantics between nodes in the network embeddings. Fig. 2 illustrates the feature aggregation process from the sampled nodes (highlighted with green, blue, and grey) within third-order neighbourhood of the central node (orange).

Our proposed ICANE model considers two kinds of features, including the content feature and network structure feature. Each node v has a d-dimensional feature vector $\boldsymbol{x}_v \in \mathbb{R}^d$. By

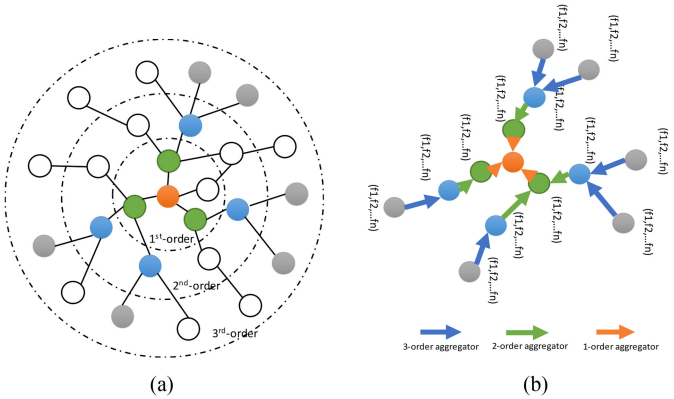


Fig. 2. Example of the neighbourhood aggregation process. (a) Neighbourhood sampling. (b) Feature aggregation.

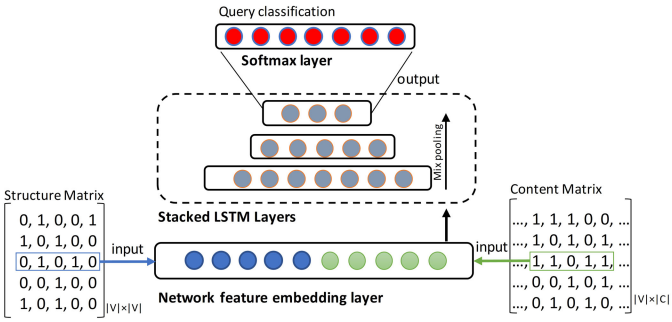


Fig. 3. Architecture of feature embedding method.

vertically stacking up all the feature vectors, a feature matrix can be constructed, which can be denoted as $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}]^\top \in R^{n \times d}$. In a decentralised edge environment, the features are considered the node's local attributes and nodes can proactively gather features from their directly connected neighbours. Feature aggregation is an offline learning process that only incurs an insignificant portion of network overhead for feature aggregation at the network embedding training phase. Once this training is completed, the learned network embeddings are stored as local knowledge of the global network resources, in order to direct newly generated queries (unseen data) to the most appropriate nodes. Herein, the proposed feature aggregation process is very cost effective for resource discovery tasks.

Fig. 3 illustrates the feature embedding method of the proposed model. In this paper, the original network structure information and content attributes are presented in a matrix using one-hot encoding to indicate the presence of the observations. Observations with the value '1' in the structure matrix indicate the connection among nodes, while observations with the value '1' elements in the content matrix represent the presence of the given attributes. To generate the embedding for a given node v, the row vectors of v in its structure matrix and content matrix are selected as inputs to form a feature vector. For a given sampled node i at the learning depth k, the feature vector is denoted as $\mathbf{x}_i^{(k)} = (\mathbf{x}_{\text{cont}_i}^{(k)} \frown \mathbf{x}_{\text{struc}_i}^{(k)})$, where $\frown$ denotes the concatenation of the two given vectors. Then, aggregation function AGG_k incorporates the feature vectors into the LSTM model to generate the representations. The LSTM network includes

Algorithm 2: SEMANTIC PROXIMITY RESOURCE DISCOVERY (SPRD).

Input: Informaiton Network $G = (V, E, A, C)$; node embeddings $\mathbf{z}_u$; query $q_v = \langle w_1, w_2, \ldots, w_n \rangle$; query node features $\mathbf{x}_v$; max learning depth K

Output: The distance score between query q and node $u, u \in V$ in embedding space

for $v \in V_q$ **do**
  $\widetilde{v} \leftarrow CreateShadowNode(v, E_v, A_v, C_v)$
  $\mathbf{x}_{cont_{\widetilde{v}}} \leftarrow GenerateOneHot(q)$
  $\widetilde{\mathbf{h}_v^0} \leftarrow Initialization(\widetilde{\mathbf{x}}_v \leftarrow (\mathbf{x}_{cont_{\widetilde{v}}} \frown \mathbf{x}_{struc_v}))$
  $V \leftarrow V + \widetilde{v}$
for $u \in V$ **do**
  $\mathbf{z}_u = LSTM_{mixpool}\{\mathbf{h}_u^K, \mathbf{h}_u^{K-1}, \ldots, \mathbf{h}_u^0\}$
  $\mathcal{L}_\mathcal{G}(\mathbf{z}_u) = \sum_{u \in V} \sum_{v \in N_R(u)} -log(P(v|\mathbf{z}_u))$
$\mathcal{D}_{u|q} \leftarrow \mathcal{D}(\mathbf{z}_u, \mathbf{z}_{q_v}) \leftarrow cos(\mathbf{z}_u, \mathbf{z}_{q_v})$
return $\mathcal{D}_{u|q}, \forall u \in V$

K stacked hidden layers, where each layer contains fixed number of fully connected LSTM units. The core function of the multiple hidden layers is to capture the weight factors at each learning depth $k, k \in [1, K]$. Then the mix-pooling strategy is used to randomly choose the mean and maximum elements in the prior hidden layer. The output of the final layer is a softmax layer with a dimensionality d. The softmax layer contains the normalised outputs obtained from the prior hidden layer within a bounded interval [0, 1] that represents the probability distribution over the proximities. The vector generated by the softmax layer is treated as the embedding $\mathbf{z}_v$ of the given node v. The dimensionality for all the embedding vectors is equal in size and is decided by the output sequence shape of the LSTM model. In this paper, we found that the dimensionality $d = 128$ tends to achieve the best performance in the resource discovery task.

C. Query Representation and Similarity Calculation

This section focuses on the resource discovery tasks using network embeddings generated from Algorithm 1. Our proposed model incorporates predicting the likely nodes comprising the requested resources during the resource discovery task. Such a prediction can be particularly challenging in a decentralised edge computing environment, due to the lack of global knowledge of nodes requesting resources. We extended the node embeddings to unify the query embedding space and the node embedding space in order to compute the comparable similarity for enabling faster node classification and ranking process. First, we embed queries containing the information of the query node in the same vector space that can be seamlessly compared with the node representation. Then a semantic proximity-based resource discovery (SPRD) strategy is presented to predict the top-k relevant nodes using cosine distance. The workflow of the SPRD is summarised in Algorithm 2.

The objective of Algorithm 2 is to embed the queries with node embeddings in the same vector space. The main idea here is to characterise the resource requester's intention by

combining the node's attributes with the query attributes, then to train a loss function that forces the query embeddings to be similar to the embeddings of nodes that can satisfy a given query and are a close proximity with the resource requester. In this way, queries can be directed to resource providers (i.e., nodes satisfy the query) with a shorter routing path in the decentralised edge environments. Algorithm 2 first generates the resource requester's intention using the $GenerateShowNode$ function, which creates a structurally identical node to that of the resource requester. Given a query word sequence $\{w_1, w_2, \ldots, w_n\}$, we use the $GenerateOneHot$ function to convert the original sequence into a query feature matrix $\mathbf{q} = \{q_w^1, q_w^2, \ldots, q_w^m\}$, where n denotes the number of words in a given query, and m is vocabulary size of words. Then the query features are concatenated with the query node's structure attributes and further fed into a fully connected LSTM to generate the final representations for the query. Given a node that is embedded to a vector space $\mathbf{z}_v, v \in V$, the semantic similarity score between the query and a given node v is given by

$$\mathrm{Sim}(\mathbf{q}, \mathbf{z}_v) = \cos(\mathbf{q}, \mathbf{z}_v) = \frac{\mathbf{q} \cdot \mathbf{z}_v}{||\mathbf{q}|| \cdot ||\mathbf{z}_v||} \tag{1}$$

where $\mathbf{q}$ is the query embedding vector and $\mathbf{z}_v$ is the node embedding vector. Equation (1) measures the similarity given by the cosine distance between two embedding vectors. In particular, let $V_{\{q\}}$ denote the targeted nodes which can satisfy the given query q, it is favourable to generate query embeddings $\mathbf{q}$ to obtain $\mathrm{Sim}(\mathbf{q}, \mathbf{z}_v) \approx 1, \forall v \in V_q$ and $\mathrm{Sim}(\mathbf{q}, \mathbf{z}_v) \approx 0, \forall v \notin V_{\{q\}}$.

The semantic similarity is computed based on the likelihood between the enhanced query embeddings and node embeddings. Then, cosine distance is applied to calculate the similarity between a given query and the nodes. A cosine distance of 1 implies an exact match and 0 indicates decorrelation. The cosine similarity measure is widely used in vector distance measurement, which is adopted in this paper for the following reasons. The cosine similarity between two embedding vectors is bound to the interval $[-1, 1]$, which indicates the closeness of the two vectors in the same direction with origin as the reference point in the vector space. In addition, it can capture meaningful semantics for ranking relevant resources conditioned on a given query, since it is sensitive at capturing spatial differences of the elements in the compared vectors, meanwhile disregarding the amplitudes of the elements. Fig. 4 illustrates the semantic process tailored specifically to a given query node's features by incorporating information about the node beyond the given query. The main purpose here is to achieve a personalised ranking of the predicted nodes that can satisfy the given query. The query is augmented with the query node's content and structure information to represent its query embedding, where the assumption is based on the following experimental studies: 1) similar nodes tend to interact with nodes that share common characteristics [22], [23] and 2) node's interests and local attributes can enhance the query's efficiency in fully decentralised environments [24], [25].

D. Training the Model

This paper employs the supervised learning technique to learn the network embeddings. The nodes are formed as a pair and a

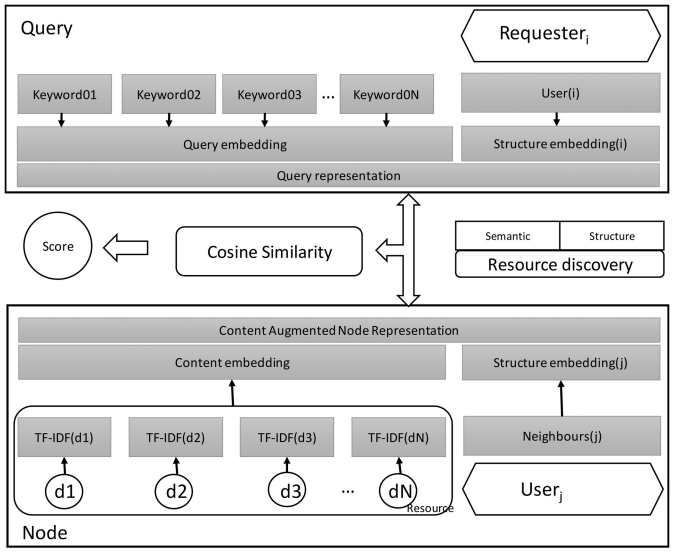


Fig. 4. Query match method for resource discovery.

label $(u, v, label)$. The label is a boolean number which indicates whether two given nodes are located in a closer proximity in the network or not. We define the $label = 1$ if node v and node u share common structure attributes or content attributes, and $label = 0$ otherwise. The objective of the training process is to optimise the probability of correct label classifications by minimising the predefined loss function. The loss function encourages the embedding of a given node to be similar to the embeddings of nodes in its closer proximity (i.e., $label = 1$) and different from the embeddings of nodes which are not in its closer proximity (i.e., $label = 0$). In order to learn effective embeddings, the random walk-based loss function is applied to the output embeddings $\mathbf{z}_u$ of the final layer of LSTM, and to train the weight matrices W^k. The training process proposed in this paper adopts the widely used stochastic gradient descent (SGD) method [26]–[28] to minimise the loss function in the aggregation process. The loss function preserves the node proximity in the original network, which encourages nearby nodes with similar content to characterise a small vector distance in the output representations, while disparate nodes characterise highly distinct output representations. The objective of the loss function is to maximise the likelihood of random walk co-occurrences. The likelihood function is defined as

$$\mathcal{L}_{\mathcal{G}}(\mathbf{z}_u) = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u)) \tag{2}$$

where $\mathbf{z}_u$ is the embeddings of node u. Each node u collects the co-occurrence from the neighbourhood $N_R(u)$ which represents the node's sets on random walks starting from the node u. $P(v|\mathbf{z}_u)$ is the probability of a node v to identify a seen node u on a random walk over the network. Equation (2) summarises all nodes in the network and all the co-occurring nodes v in the random walk starting form u. Further to calculate the probability, we use

$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{v \in N_R(u)} \exp(\mathbf{z}_u^\top \mathbf{z}_v)} \tag{3}$$

where $\mathbf{z}_u^\top \mathbf{z}_v$ is the probability of a given node v to co-occur near u during fixed-length random walks over the network. Equation (3) utilises softmax [29] to parameterize $P(v|\mathbf{z}_u)$. The random work process is further optimised to find embeddings $\mathbf{z}_u$ which minimises $\mathcal{L}_\mathcal{G}$. However, when V is extremely large, calculating the denominator by going through all the words for every single sample is computationally impractical. Due to the complexity of normalisation of $\sum_{v \in N_R(u)} \exp(\mathbf{z}_u^\top \mathbf{z}_v)$, negative sampling method is used to approximate the conditional probability as

$$P(v|\mathbf{z}_u) \approx \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{c} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i})) \qquad (4)$$

where σ is the sigmoid function, $n_i \sim P_n$ which is a negative sampling distribution [9], and c is the number of negative samples. The objective of negative sampling is to approximate the probability of $P(v|\mathbf{z}_u)$. The idea of negative sampling is to maximise the similarity between the node vectors for nodes characterising close association in both structure and contents, and to minimise the similarity of nodes that do not show closer proximity. The samples selected by the negative sampling method are called negative samples, which are usually the randomly selected nodes with neither common structure attributes nor content attributes to a given node. It is worth noting that Equation (4) normalises the feature range against c random negative samples instead of normalising against all nodes. The benefit of this strategy is the significantly enhanced training performance, since only a small percentage of neuron weights in the neural network needs to be modified, rather than updating all the weights.

V. Experiments

This section presents the performance evaluation of our proposed model in a simulated edge environment. The performance of our proposed model ICANE is tested by feeding in resources discovery tasks. The objectives of the experiments are threefold: 1) to compare the performance of network embeddings of ICANE against the state-of-the-art methods; 2) to investigate the impacts of features in the learning phase whilst optimising the network embedding to support semantic-proximity search; 3) to visualise the learned network representations in two-dimensional graphs. Programming environment: The network embeddings of compared models and evaluations are implemented using TensorFlow 1.12.0 and scikit-learn 0.20.1, and the visualisation of embeddings uses t-SNE [30].

A. Experiment Settings

Datasets: Three citation networks benchmarking datasets[1] Cora, Citeseer, and Pubmed are used to evaluate the performance of the proposed model. In the citation networks of Cora, Citeseer, and PubMed, a node is a publication article and an edge is the citation link between publications. First, we establish the network topology by connecting the nodes with undirected edges. Then, the node structure attributes and content attributes are assigned as properties of the nodes, where the structure attributes are the first-order neighbours and content attributes are the terms' occurrence in a publication for the corresponding node. Finally, each node is assigned with the predefined class label as the ground truth for classification tasks. The processed datasets are detailed as follows.

1) *Cora:* It comprises 2708 nodes from seven classes and 5429 links and 1433 content attributes.
2) *Citeseer:* It contains 3312 nodes from six classes and 4732 links and 3703 content attributes.
3) *Pubmed:* It consists of 19 717 nodes from three classes with 44 338 links and 500 content attributes.

Baseline methods: The performance of our proposed method is evaluated against the following popular network embedding approaches.

1) *LINE [10]:* This method learns the first $d/2$-dimension embeddings from the first-order neighbours and learns the second $d/2$-dimension embeddings from the second-order neighbours. Then the embedding is presented as the concatenation of the two learned parts to form a d-dimensional embedding.
2) *Node2vec [31]:* This method preserves higher order proximity through fix-length biased random walk to generate a d-dimensional embedding. The window size is set to ten, the walk length is 80, and the number of walks is ten.
3) *HOPE [32]:* This method utilises the singular value decomposition (SVD) [33] to factorise the higher order similarity matrix between nodes to form a d-dimensional embedding. HOPE is configured to learn the third-order proximity in the experiments.
4) *TADW [11]:* This method learns a d-dimensional embedding from matrix decomposition, considering both network structure and content.

To conduct fair comparisons, the dimensionality d of embeddings for the information network is set to 128 for all the compared baseline methods. TADW includes the network structure and node content information, while others only adopt the network structure information to generate network embeddings. The resource discovery process is equivalent to a binary classification for multiple queries. Our proposed ICANE model is trained for the node classification task, where the loss function is minimised at each step of the LSTM model. During the training process, the negative sampling method randomly selects five negative samples. All models are implemented using TensorFlow with a learning rate of 0.01. ICANE model sets the max learning depth as $K = 3$, which means the model samples nodes in the third-order neighbourhood.

B. Evaluation Metrics

The requirements of resource discovery in a decentralised edge environment are two-fold: 1) correctness—the model should find correct nodes which can satisfy a given query; 2) low response latency—the located nodes should be close to the query initiator in the network structure. In the experiments, network embeddings of different methods are evaluated for their efficiencies of the resource discovery process. The network

[1]The datasets can be downloaded here: https://linqs.soe.ucsc.edu/data

simulated ten queries for each node, then we observe the average performance for the queries. To evaluate the correctness, the commonly used Micro-F1 and Macro-F1 metrics are chosen to calculate the overall accuracy. Furthermore, the average query length is observed to evaluate query response latency. In the evaluation, we adopt the classic confusion matrix (a.k.a, the error matrix) to compute the metrics [34]. In detail, we denote TP_q, FP_q, and FN_q as the number of true positives, false positives, and false negatives for query q, respectively. The evaluation metrics used in the confusion matrix are listed as follows.

Precision is the fraction of true instances among positive instances, which is defined as $P = \frac{\sum_q TP_q}{\sum_q TP_q + \sum_q FP_q}$.

Recall is the fraction of true instances identified over the total amount of true instance, which is defined as $R = \frac{\sum_q TP_q}{\sum_q TP_q + \sum_q FN_q}$. By balancing the effectiveness of recall and precision, Micro-F1 and Macro-F1 are further defined as:

Micro-F1 is the harmonic average of the precision and recall, which is defined as Micro $- F_1 = \frac{2 \cdot P \cdot R}{P+R}$;

Macro-F1 is the unweighed mean F1-measure of all queries, which is defined as $\frac{\sum_{q \in Q} \text{Micro} - F1(q)}{|Q|}$, where Micro $- F1(q)$ is the F1 score for query q and $|Q|$ is the size of evaluated queries.

Average query path length (AQPL) measures the global average distance from query nodes to predicted target nodes, which is defined as AQPL $= \frac{\sum_{q \in Q} L_q}{|Q|}$, where L_q is the average distance $\frac{\sum_{j \in V_q} d(v_q, v_j)}{|V_q|}$ from query node v_q to the target nodes $v_j \in V_q$ for query q, $|V_q|$ is the size of the target nodes and $|Q|$ is the size of evaluated queries.

TABLE II
RESOURCE DISCOVERY RESULTS ON CORA DATASET

	10% training		30% training		50% training	
Model	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
LINE	0.2278	0.2189	0.3387	0.2898	0.3278	0.2918
Node2Vec	0.4324	0.4217	0.4831	0.4691	0.4824	0.4084
HOPE	0.5431	0.5123	0.5622	0.5187	0.5431	0.5902
TADW	0.6323	0.5929	0.6767	0.6356	0.7123	0.6924
ICANE	**0.7615**	**0.7421**	**0.7974**	**0.7863**	**0.8615**	**0.8393**

The bold entities are the experimental results of our proposed model, which is used to differentiate our results from other compared research works.

TABLE III
RESOURCE DISCOVERY RESULTS ON CITESEER DATASET

	10% training		30% training		50% training	
Model	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
LINE	0.1861	0.1716	0.2589	0.2484	0.2946	0.2657
Node2Vec	0.2012	0.1912	0.3186	0.2975	0.3296	0.2939
HOPE	0.3144	0.2962	0.4766	0.4385	0.5186	0.4851
TADW	0.4912	0.4723	0.5509	0.5393	0.6529	0.6443
ICANE	**0.5561**	**0.5292**	**0.6197**	**0.5815**	**0.7517**	**0.7197**

The bold entities are the experimental results of our proposed model, which is used to differentiate our results from other compared research works.

TABLE IV
RESOURCE DISCOVERY RESULTS ON PUBMED DATASET

	10% training		30% training		50% training	
Model	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
LINE	0.3131	0.2933	0.4127	0.3695	0.4689	0.4358
Node2Vec	0.3287	0.3191	0.4288	0.3902	0.4974	0.4605
HOPE	0.4122	0.3941	0.5164	0.4983	0.5693	0.5587
TADW	0.4974	0.4726	0.6412	0.6045	0.7384	0.7014
ICANE	**0.6667**	**0.6491**	**0.7887**	**0.7345**	**0.8852**	**0.8673**

The bold entities are the experimental results of our proposed model, which is used to differentiate our results from other compared research works.

C. Results and Analysis

This section discusses the experiment results of our proposed model for resource discovery tasks in decentralised edge environments, and provides a performance comparison against the baseline methods.

1) Resource Discovery Evaluation: In the experiments, we verify the correctness of node semantic proximity embedding and the query efficiency of the studied embedding models using real-world citation datasets.

Correctness: The performance of our model is evaluated on a query-node matching process, which is equivalent to a multi-label classification task. The task is to predict whether a node can satisfy a given query using pretrained embeddings. The node embeddings and query embeddings are learned by the model as the input to conduct binary classifications for each query. For a comprehensive comparison against the baseline methods, the training ratio for each dataset is varied from 10%, 30%, to 50% and the Micro-F1 and Macro-F1 scores in the test datasets are reported in Tables II–IV. It can be observed from the results that our proposed ICANE model outperforms the other models in the three network datasets. To be more specific, structure-based embedding models, such as the LINE, Node2Vec, and HOPE, generally performed worse than the hybrid embedding models, such as TADW and ICANE. This is due to the fact that the structural embedding models ignore the content and semantic proximity, inevitably leading to the lower F1 scores during resource discovery tasks. TADW and ICANE are both hybrid embedding methods. ICANE achieved higher F1 scores in comparison with the TADW model. This is due to the fact that our proposed ICANE model uses the feature aggregation functions to significantly enhance the higher order proximity discovery for distant nodes. Moreover, it can be observed that the proposed ICANE model achieved better F1 scores with a small training ratio. This can be noted as a favourable characteristic in supervised machine learning, as ICANE is able to learn the overarching network features with a small portion of training data.

To further evaluate the correctness against the cost, the ROC curves are presented in Fig. 5. ROC is commonly used in information retrieval and classification, which is defined by FPR (false positive rate) and TPR (true positive rate) to evaluate relative tradeoffs between benefits (i.e., the TPR) and costs (i.e., the FPR). This experiment combines the FPR and TPR from the ICANE model for the three datasets. It can be observed that all the three ROC curves are located above the diagonal dashed line (black), reflecting that the classification results are better than random. Our proposed ICANE model exhibits better performance in Cora and Pubmed datasets with promising early retrieval of true positives, while its performance in Citeseer dataset is mediocre due to the fact that the size of its query embedding 128 is much smaller than its content dimension 3703.

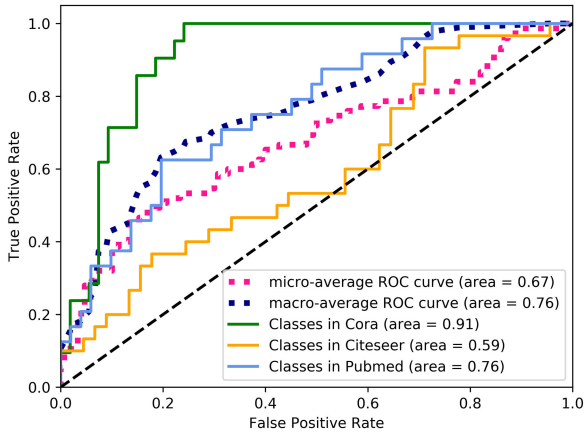


Fig. 5. ROC curves and area under curves for three datasets.

TABLE V
AVERAGE QUERY PATH LENGTH ON THREE DATASETS

	Cora	Citeseer	Pubmed
LINE	13.94	11.67	30.14
Node2Vec	12.32	13.46	25.41
HOPE	14.57	12.84	18.23
TADW	8.49	9.24	12.42
ICANE	**5.89**	**7.27**	**9.35**

The bold entities are the experimental results of our proposed model, which is used to differentiate our results from other compared research works.

The area under ROC is using a normalised unit that equals the probability that ICANE can rank the randomly chosen positives higher than the randomly chosen negatives. It is clear that the area under ROC for three datasets are above 0.5, therefore, it is safe to conclude that our proposed ICANE model can yield better and reliable classification results, in other words, achieving considerable benefits for the incurred costs. The costs (i.e., the FPR) will increase the overheads of query routing by building up query latency, which is discussed as follows.

Query latency: It is vital to examine latency of query in decentralised edge environments. This experiment simplifies the latency measurement by using average query path length as an alternative. The assumption behind this is that query match is conducted in a hop-by-hop process on the physical network layer. Therefore, a lower number of hops reflects a lower query response latency.

Table V shows the AQPL measurement for the embedding models based on the three datasets. A given query is first measured in the embedding space and then the most similar nodes are selected using Equation (1). Furthermore, the query path length is measured on the original network from the query node to the predicted nodes. It is evident that our proposed ICANE model achieves shorter query path lengths than the baseline methods. This also highlights the dependability of our proposed model in dynamic realistic edge environments with less query latency. The following characteristics can be postulated as the reasons for the better performance of our proposed model: 1) ICANE can persevere the higher order semantic proximity of

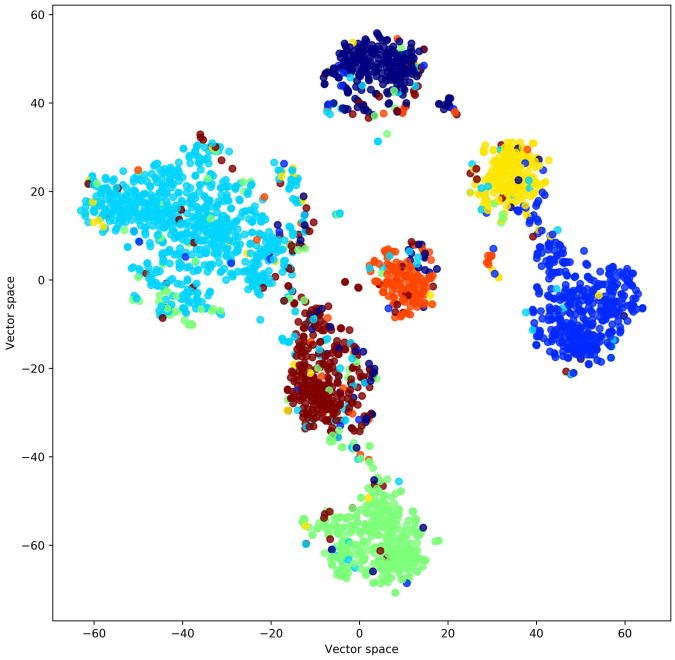
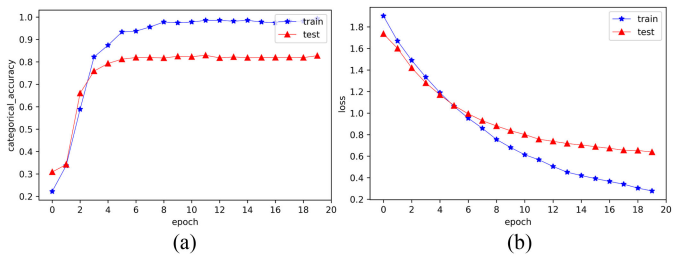


Fig. 6. Visualisation of ICANE embedding on Cora dataset.



(a) (b)

Fig. 7. Accuracy and loss of ICANE embedding on Cora dataset. (a) Categorical accuracy curve. (b) Loss curve.

the network nodes and 2) the query search is enhanced when node structure and content attributes are considered in the same embedding space. Thus, the query matching process in embedding space can aid more efficient resource discovery in dynamic edge networks.

2) Embedding Visualisation—A Demo on Cora: Due to space limitation, this section only presents the experiments conducted based on the Cora network as a case study to visualise the high-dimensional network representation using t-SNE [30]. The visualisation of network embedding is shown in Fig. 6. The colors in the graph represent the class label of each node. There are seven class labels in the Cora dataset. It can be observed that our proposed ICANE model achieves compact and separated clusters in the embedding space. Fig. 7(a) shows the accuracy of node classification in both training and test datasets. Besides, Fig. 7(b) depicts the loss in Equation (2) in the training and test datasets. The two curves are close together until the fourth epoch and separate after the sixth epoch, where an epoch represents one forward pass of all the training parameters to the next layer in the neural network. ICANE utilises a neighbourhood sampling method to achieve good training parameters for the

unseen data in the test dataset, which significantly avoids the over-fitting issues [35] in the neural networks.

VI. Conclusions and Future Research Directions

This paper proposed ICANE, an inductive content augmented network embedding model, to preserve higher order structural and semantic content proximity in large-scale decentralised networks leveraged by artificial intelligence techniques. ICANE can effectively learn the embedding function to generate a low-dimensional vector representation of complex networks in an inductive way. Furthermore, a semantic proximity search method has been proposed to locate the top-k relevant nodes using the learned network representation. Experiments conducted based on three network datasets demonstrate that our proposed method significantly outperforms the popular baseline models for resource discovery tasks in terms of the achieved query correctness and in reducing the query latency.

As a part of future work, we plan to explore the following: 1) extend the implementation of the proposed ICANE model with dynamic network analysis; 2) perform better model optimisation and regulation for heterogeneous resource representations; and 3) deploy and evaluate our model in industrial edge computing environments.

References

[1] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, pp. 37–42, Sep. 2015.

[2] D. Miao, L. Liu, R. Xu, J. Panneerselvam, Y. Wu, and W. Xu, "An efficient indexing model for the fog layer of industrial internet of things," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4487–4496, Oct. 2018.

[3] S. K. Datta, R. P. F. Da Costa, and C. Bonnet, "Resource discovery in Internet of things: Current trends and future standardization aspects," in *Proc. IEEE 2nd World Forum Internet Things*, 2015, pp. 542–547.

[4] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Large-scale indexing, discovery, and ranking for the Internet of things (IoT)," *ACM Comput. Surv.*, vol. 51, no. 2, 2018, Art. no. 29.

[5] L. Liu, N. Antonopoulos, M. Zheng, Y. Zhan, and Z. Ding, "A socioecological model for advanced service discovery in machine-to-machine communication networks," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 2, 2016, Art. no. 38.

[6] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Trans. Knowl. Data Eng.*, to be published.

[7] Y. Wang, Y. Yao, H. Tong, F. Xu, and J. Lu, "A brief review of network embedding," *Big Data Mining Analytics*, vol. 2, no. 1, pp. 35–47, 2019.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, arXiv:1301.3781.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Neural Inf. Process. Syst. Conf.*, 2013, pp. 3111–3119.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[11] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.

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

[13] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Neural Inf. Process. Syst. Conf.*, 2017, pp. 1024–1034.

[14] K. Xu, L. Wu, Z. Wang, and V. Sheinin, "Graph2seq: Graph to sequence learning with attention-based neural networks," 2018, arXiv:1804.00823.

[15] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," 2018, arXiv:1808.07624.

[16] B. Kotnis and V. Nastase, "Analysis of the impact of negative sampling on link prediction in knowledge graphs," 2017, arXiv:1708.06816.

[17] Q. Zhang and H. Wang, "Not all links are created equal: An adaptive embedding approach for social personalized ranking," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 917–920.

[18] Y. Han and Y. Shen, "Partially supervised graph embedding for positive unlabelled feature selection," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1548–1554.

[19] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.

[20] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[21] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Proc. Int. Conf. Rough Sets Knowl. Technol.*, 2014, pp. 364–375.

[22] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annu. Rev. Sociol.*, vol. 27, no. 1, pp. 415–444, 2001.

[23] L. Jiang, L. Shi, L. Liu, J. Yao, B. Yuan, and Y. Zheng, "An efficient evolutionary user interest community discovery model in dynamic social networks for internet of people," *IEEE Internet Things J.*, to be published.

[24] M. Bertier, R. Guerraoui, V. Leroy, and A.-M. Kermarrec, "Toward personalized query expansion," in *Proc. 2nd ACM EuroSys Workshop Social Netw. Syst.*, 2009, pp. 7–12.

[25] B. Yuan, L. Liu, and N. Antonopoulos, "Efficient service discovery in decentralized online social networks," *Future Gener. Comput. Syst.*, vol. 86, pp. 775–791, 2018.

[26] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, 2010, pp. 177–186.

[27] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[28] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. Neural Inf. Process. Syst. Conf.*, 2010, pp. 2595–2603.

[29] Y. Goldberg and O. Levy, "word2vec explained: Deriving Mikolov *et al.*'s negative-sampling word-embedding method," 2014, arXiv:1402.3722.

[30] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov., pp. 2579–2605, 2008.

[31] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[32] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.

[33] C. C. Paige and M. A. Saunders, "Towards a generalized singular value decomposition," *SIAM J. Numer. Anal.*, vol. 18, no. 3, pp. 398–405, 1981.

[34] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, 2011, pp. 37–63 2011.

[35] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

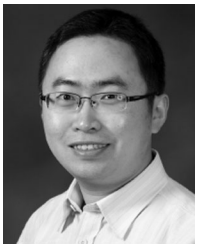
Bo Yuan (M'15) received the B.Sc. degree in computer science and technology and the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2011 and 2018, respectively.

He is currently a Postdoctoral Researcher with the School of Electronics, Computing and Mathematics, University of Derby, Derby, U.K. His research interests include Internet of things, service discovery, and data science, and his current research focuses on industrial big data analytics using cloud computing and deep learning techniques.

John Panneerselvam (M'14) received the M.Sc. degree in advanced computer networks in 2013 and the Ph.D. degree in computing from the University of Derby, Derby, U.K., in 2018.

He is a Lecturer of Computing with the University of Derby. His current research is focused on energy efficient cloud systems and data analytics for high performance computing.

Dr. Panneerselvam is a Member of British Computer Society and a Fellow of HEA.



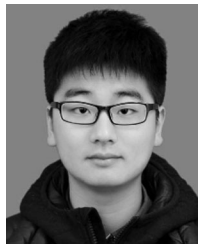
Lu Liu (M'07) received M.S. degree from Brunel University, London, U.K. and the Ph.D. degree from the University of Surrey, Surrey, U.K.

He is the Professor of Informatics and Head of the Department of Informatics, University of Leicester, Leicester, U.K. He has over 180 scientific publications in reputable journals, academic books, and international conferences. His research interests are in the areas of data analytics, cloud computing, service computing, and social computing.

Prof. Liu is the Fellow of British Computer Society (BCS).

Nick Antonopoulos received the Ph.D. degree in computer science from the University of Surrey, Guildford, U.K., in 2000.

He is the Vice Principal Research and Innovation with Edinburgh Napier University, Edinburgh, U.K. He is an active research Professor of Distributed Systems and Cloud Computing with over 150 peer refereed publications and about 2000 citations. His research interests include cloud computing, P2P computing, software agent architectures, and security.



Yao Lu (M'18) received the master's degree from Jiangsu University, Jiangsu, China. He is currently working toward the Ph.D. degree in computer science at the University of Derby, and he is a Visiting Ph.D. Student at Jiangsu University.

His current research focuses on energy efficient cloud systems and his research interests include cloud computing, big data analytics, and high-performance computing.

Mr. Lu is the recipient of the Best Paper Award at the IEEE International Conference on Data Science and Systems, Exeter, U.K., 2018.