Knowledge and Information Systems
https://doi.org/10.1007/s10115-023-02050-x

REGULAR PAPER

Similarity enhancement of heterogeneous networks by weighted incorporation of information

Fatemeh Baharifard[1] · **Vahid Motaghed**[1]

Received: 16 May 2023 / Revised: 29 November 2023 / Accepted: 14 December 2023

Abstract
In many real-world datasets, different aspects of information are combined, so the data is usually represented as heterogeneous graphs whose nodes and edges have different types. Learning representations in heterogeneous networks is one of the most important topics that can be utilized to extract important details from the networks with the embedding methods. In this paper, we introduce a new framework for embedding heterogeneous graphs. Our model relies on weighted heterogeneous networks with star structures that take structural and attributive similarity into account as well as semantic knowledge. The target nodes form the center of the star and the different attributes of the target nodes form the points of the star. The edge weights are calculated based on three aspects, including the natural language processing in texts, the relationship between different attributes of the dataset and the co-occurrence of each attribute pair in target nodes. We strengthen the similarities between the target nodes by examining the latent connections between the attribute nodes. We find these indirect connections by considering the approximate shortest path between the attributes. By applying the side effect of the star components to the central component, the heterogeneous network is reduced to a homogeneous graph with enhanced similarities. Thus, we can embed this homogeneous graph to capture the similar target nodes. We evaluate our framework for the clustering task and show that our method is more accurate than previous unsupervised algorithms for real-world datasets.

Keywords Heterogeneous Graph · Unsupervised learning · Natural language processing · Node clustering

Fatemeh Baharifard and Vahid Motaghed have equally contributed to this work.

✉ Fatemeh Baharifard
f.baharifard@ipm.ir

Vahid Motaghed
vahid.motaghed.2020@gmail.com

[1] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Published online: 27 January 2024

List of symbols

Sets

$\mathcal{N}$	Target set
$\mathcal{A}_i$	Information set
$\mathcal{M}$	Main attribute set
$\mathcal{R}$	Relational attribute set
$\mathcal{T}$	Textual attribute set
C	Clustered set

Text embedding symbols

t_j	Text object
$\vec{t}_j$	Word vector of t_j
$\vec{t}^{\,e}_j$	Embedded vector of t_j
$\overrightarrow{\mathsf{BERT}}(.)$	BERT embedding function
$\mathsf{TF}(.)$	Rank weighted density function
m_j	Number of elements of t_j
t_i^j	i-th word of vector $\vec{t}_j$
x_{ih}^j	h-th element of $\overrightarrow{\mathsf{BERT}}(\mathrm{t}_i^j)$
$\mathcal{B}_h^j$	h-th element of $\vec{t}^{\,e}_j$
$\mathbb{D}$	Feature space size
$\mho(.)$	Term frequency in target set
$\mathbb{H}(.)$	Term frequency in feature space
$\mathbb{L}(.)$	Text length

Graph Symbol

$G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$	Star heterogeneous graph
$G_c = (\mathcal{V}_c, \mathcal{E}_c, \mathcal{W})$	Core graph
$G_s^i = (\mathcal{V}_s^i, \mathcal{E}_s^i, \mathcal{W})$	$\mathcal{M}_i$ shell graph
$\overline{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \overline{\mathcal{W}})$	Homogeneous core graph
$V_{\mathcal{N}}$	Vertex of target set
$V_{\mathcal{M}}$	Vertex of main attribute set
$E_{\mathcal{I}}$	Internal link set
$E_{\mathcal{O}}$	External link set
d_{xy}	Euclidean distance of x, y
$w_{\mathcal{R}}(.,.)$	Relational weight
$w_{\mathcal{T}}(.,.)$	Textual weight
$w_{\mathcal{J}}(.,.)$	Joint presence weight
$w(.,.)$	Total weight
$p(\mathcal{M}_i)$	$\mathcal{M}_i-$path
$\rho\langle ., p(\mathcal{M}_i), .\rangle$	$\mathcal{M}_i$ auxiliary path
$\rho_i\langle ., .\rangle$	$\mathcal{M}_i$ shortest path
$\mathcal{R}(G)$	Remapped graph
$\pi(.)$	Remapped function
$W(.), \overline{W}(.)$	Path weight function
$\mathcal{H}$	Spanner graph

Parameters

$P_i = \{c_i, \sigma_i, \kappa_i\}$	Truncation parameters

$\alpha = \{\alpha_{\mathcal{R}}, \alpha_{\mathcal{T}}, \alpha_{\mathcal{J}}\}$	Weighting coefficients
β_i	Main attribute impact factor
θ_i	Scaling parameter
μ_i	Spanner parameter

Operator

$\mathbb{N}$	Normalization operator

1 Introduction

For most datasets, a graph structure can be used to represent the entities and their relationships as nodes and edges, respectively. For example, datasets about social networks [1], traffic networks [2], citation networks [3] and so on. The graph is homogeneous if it contains only one type of node and one type of edge. But, in many real-world datasets, different aspects of information are combined, and the data tend to be presented as a heterogeneous graph [4], whose nodes and edges have various types. *Heterogeneous information networks* (HINs) are used in many data mining tasks due to the comprehensive information and rich semantics they have. Taking the citation network dataset illustrated in Fig. 1, as an example of HIN, it contains four types of nodes (paper, author, keyword and venue) and four types of edges (write, cite, belongs to and published by). Based on the purpose of the study, this graph can be interpreted as directed or undirected.

The effective analysis of heterogeneous graphs can reveal hidden information contained in them by employing both graph-based methods [5] and graph embedding techniques [6]. The knowledge discovered can be used in useful applications such as node clustering, node classification, link prediction and so forth.

In studying some datasets, in addition to the structural relations of graphs, some auxiliary-rich information such as label, attribute, information propagation and node feature can be added to the graphs to incorporate topological and unstructured information together [6]. One of the challenges is how to integrate these two sources of information, so that the

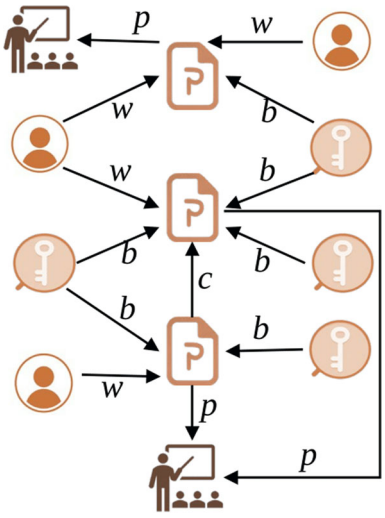

Fig. 1 Citation network as an example of the heterogeneous graph. In this network, we have four types of nodes (paper, author, keyword and venue) and four types of edges (w = an author writes a paper, c = a paper cites another paper, b = a keyword belongs to a paper, and p = a paper published by a venue)

concept of similarity of nodes is preserved. In most previous studies, heterogeneous graphs were considered based solely on the pre-defined structures, and many of the benefits of the other available information in the dataset have been ignored [7–9]. Even if they used this information in their embedding, the semantic knowledge and latent connections which exists between nodes have not been used to reinforce their graphic structure [5, 6, 10, 11].

To overcome the above limitations, we propose a new framework for Similarity Enhancement of Heterogeneous Networks by weighted incorporation of information, which we call SEHN. This framework ensures global consistency between the different types of objects in the datasets by using both structural and semantic knowledge. In this framework, a weighted and undirected star-structure heterogeneous network is constructed with target objects as the center, some natural language processing (NLP) methods are applied to discover the semantic connections of the text attributes of the heterogeneous graph and indirect connections extracted by finding some paths between the attributes improve the structure of the networks. Thus, using this reformed network, we can cluster similar target nodes into coherent and diverse sets, even without training labels.

In summary, we present the following contributions:

(1) We introduce SEHN framework to cluster similar objects in a dataset through an unsupervised method. In SEHN, we construct a star-structure heterogeneous graph based on incorporating structural and attribute similarity as well as semantical knowledge.
(2) For semantic analysis of textual contents, we apply an embedding method that relies on a combination of rank-weighted term frequency density and a pre-trained language model to ignore the noise and obtain appropriate numerical vectors as representations of the texts.
(3) Using auxiliary paths between attribute nodes as indirect connections, we perform edge modification (re-weight, add, delete) in the heterogeneous graph to obtain a reduced and improved homogeneous graph containing only the target points.
(4) We evaluate our method by node clustering task on three datasets, including IMDb, DBLP and a real dataset we collected from the Twitter social network. Experiments on these datasets show that our framework performs better than the previous unsupervised methods.

2 Related works

Researchers have considered unsupervised clustering methods in the past, which has led to several clustering methods [12, 13]. Clustering involves dividing some unlabeled data points into a set of clusters, so that the data points in the same cluster are more similar to each other than those in other clusters. Clustering on texts is very useful for improving retrieval and supporting browsing [14]. To prepare text domains for clustering algorithms, traditional methods such as weighted text representation [15] and word embedding pre-trained models [16] have been used extensively. Recently, Bidirectional Encoder Representations from Transformers (BERT) text embedding model [17] received some attention due to its consideration of the context of a target word. This obtains representations of a word that have a better match to the specific meaning of the word in different sentences. Some researchers have shown that effective representation can improve performance in clustering text [18, 19].

In addition to text clustering, the clustering approach can be applied to various data types as well as various attributes of a dataset. Many of the classic methods that have been studied in the last few decades, including k-means [20] and spectral clustering [21], are

dependent on features of data points. These methods cannot use the relationships between data points. Some other methods, utilizing relationships between data, cluster the nodes by network embedding. Network embedding is a representation framework for projecting a graph into a low-dimensional space. Some random walk embedding methods like node2vec [22] and DeepWalk [23] have gained attention due to their success in natural language processing. DeepWalk, for instance, uses short random walks and skip-grams to represent nodes. Matrix factorization methods such as Laplacian matrix [24] and BoostNE [25] can also be used to embed networks by factorizing node connectivity matrix to low-rank based on matrix approximation strategies. Moreover, graph neural network (GNN) method, as a deep representation learning method, also can embed the networks based on node features and the graph structure. Graph convolutional network (GCN) [26] and graph attention network (GAT) [27] are two graph embedding methods, leverage convolutional operation and attention mechanism, respectively.

Graph-based clustering methods, also known as community detection algorithms, are popular clustering methods that utilize the relationships among entities. These methods usually divide the main graph into a series (overlapping or non-overlapping) subgraphs by defining some measures like modularity [28], motif patterns [29] and ego-nets local structure [30]. The EdMot method [29] builds a motif-based hypergraph and partitions the top largest connected components into modules using a motif-based approach. Enhanced module edges and rewiring strengthen the original network's connectivity. Ego-splitting [30] is a scalable and flexible framework, which uses ego-nets (the subgraph induced by the neighborhood of each node) as guidance to determine overlapping clusters. Recently, a new method has been proposed based on both embedding and community detection simultaneously named GEMSEC [31]. This method tries to take advantage of both techniques. Deep learning-based methods have also been proposed to solve the clustering problem, among them CDDTA [32] and CommunityGAN [33] can be mentioned, which are based on deep transitive autoencoder and generative adversarial nets, respectively. A survey of categorizing the different community detection algorithms in the healthcare applications can be read in [34].

While all the methods mentioned above are designed for homogeneous graphs, they are not efficient when dealing with heterogeneous information networks (HINs). Therefore, the study of HINs has gained increasing attention in recent years due to their ability to model datasets with two types of information, including node attributes and network structure [35, 36]. One of the basic methods for clustering this type of graph is RankClus [37], in which a high-quality net-cluster is built by utilizing links across multi-typed objects and in an iterative method the links are enhanced to achieve effective ranking-based clustering. This method only considers structural properties and not the attributes of the nodes. Some methods are based on graph representation learning. Metapaths are the fundamental concept behind most embedding methods for heterogeneous graphs, such as metapath2vec [7] and HIN2vec [8]. A metapath is a pattern composed of a set of node types and edge types of a heterogeneous network. The hybrid relationship between the different types of nodes in the network is depicted by this pattern. By embedding heterogeneous graphs, it is possible to check some applications on them. For example in [35], node classification and node clustering tasks considered and in [38] a new method for recommendation based on embedding spectral clustering in heterogeneous networks has been discussed.

All of the above methods do not exploit the content features of nodes, but semi-supervised methods [39], such as HAN [35], MAGNN [11] and MeGNN [40] based on neural graph networks, have been recently proposed to improve them. HAN has used the attention mechanism in the construction of its network and MAGNN generates node properties from semantic information and improves community discovery results by simultaneously considering this

information and network topology. Recently, an embedding method for complex heterogeneous networks has also been proposed [10], generalizing higher-order spectral clustering by considering the graphlet concept for heterogeneous networks. However, many of the methods used to study clustering in heterogeneous graphs were semi-supervised or evaluated only structural similarity [9, 41].

Despite clustering in HINs has been studied in the literature, finding a unified distance measure that captures both structural and attributive similarity, which is useful for unsupervised clustering, has not been well studied. We developed our method based on this limitation. We propose a graph-based method to find a general distance measure that takes into account both topological and attributive information while benefiting from the advantage of NLP methods to strengthen this similarity distance.

3 Methodology

We present a new framework for unsupervised clustering of a dataset in this section. Assume that the dataset has a set of target objects $\mathcal{N} = \{N_1, \cdots, N_n\}$, each of which has at most ℓ attributes represented in information sets $\cup_{i=1}^{\ell} \mathcal{A}_i$. Indeed, the information set $\mathcal{A}_i$ contains the value of the i-th attribute of all target objects. In particular, we can divide $\mathcal{A}_i$'s into three attribute type sets:

- *Main attribute set* ($\mathcal{M}$) Includes attributes that are about the main objects of the dataset. In a citation network, for instance, if "paper" is the target object, $\mathcal{M}$ = {"keyword","author","venue"}
- *Relational attribute set* ($\mathcal{R}$) Includes attributes that describe the relations between two main attributes of the same type. For example, "one paper cited by another paper".
- *Textual attribute set* ($\mathcal{T}$) Includes attributes whose content is text and it is related to one main attribute. For example, "abstract of a paper".

The goal of our framework is to use a combination of the given attributes to categorize the target set $\mathcal{N}$ into coherent and diverse sets of $C = \{C_1, \cdots, C_k\}$. More precisely, the proposed framework can be divided into four main parts:

(1) *Processing textual information* We perform some preprocessing operations on the content of the textual attributes which contain some sentences, and embed the processed texts in a quantitative space.
(2) *Construction of a heterogeneous graph* We construct a heterogeneous graph with the star structure, where the target objects act as central nodes and the values of main attributes act as star nodes. Members of the target nodes as well as members of each main attribute type can be connected through internal links. External links are created between target nodes and main attribute nodes.
(3) *Reduction of a heterogeneous graph* We convert the heterogeneous graph into a homogeneous graph as a single layer graph by applying some reduction methods. These methods work in such a way that the side effects of the information of all attributes can be seen in one main layer.
(4) *Graph partitioning* We embed the homogeneous graph and obtain non-overlapping clusters.

For example, taking a Twitter dataset as input to our framework, we can encounter three sub-data, which include the tweets/retweets (T), the hashtags of these posts (H) and the users (U) who tweet or retweet them in a given period of time. Here, we can consider tweet/retweet

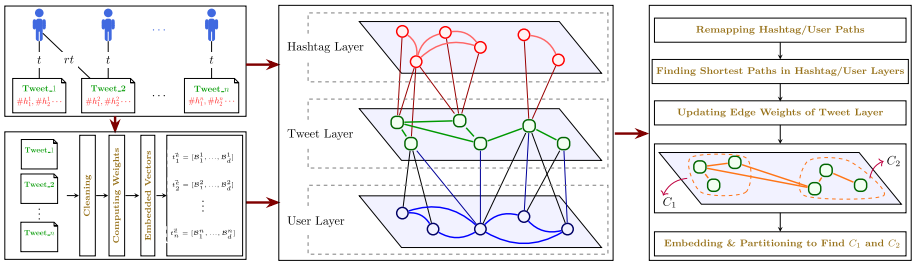

Fig. 2 Overview of our framework for a Twitter dataset. In the first stage, the textual contents of the tweets are converted into quantitative vectors, which are fed to the second stage along with information about hashtags and users. The hashtag–tweet–user (three-layer) heterogeneous graph is constructed in stage 2. Based on the similarity of the texts, the internal links are created in the tweet layer. The internal links of each main attribute are also shown in its layer. The red, blue and black edges between the layers represent tweet–hashtag, tweet–user and retweet–user external links, respectively. Considering the effect of hashtag and user layers, the heterogeneous graph reduces to a homogeneous graph with tweet nodes. Embedding the homogeneous graph and partitioning is the last phase

as the target object, which has two main attributes as "hashtag" and "user" and one textual attribute as "text of tweet/retweet". So, first, we do text processing on the texts and then construct the heterogeneous graph using T as central nodes and sets of H and U as star nodes. According to the dataset, external links between T and two sets of H and U are created. By considering the distances between the embedded vectors of the text of the tweets, some internal links between the nodes of T are added to improve the connectivity between similar tweets. Moreover, by defining a new criterion, some internal links between hashtag nodes as well as user nodes are created. By integrating these three layers, a homogeneous graph is obtained. The graph embedding is applied to the enhanced homogeneous graph, and diverse categories are derived. Figure 2 indicates the overview of the framework for this dataset. If we have more than two main attributes, the three-layer graph of this figure becomes a star-structure heterogeneous graph. In the following, each part of our framework is explained in detail.

3.1 Text processing

Given a textual attribute $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$ about n target objects of a dataset. Since textual attributes are usually short and contain slang and misspellings, all texts must first be preprocessed. At this stage, each text is tokenized, the stemming operation is performed, and then punctuation, stop words and links (*url*) are removed. Thus, for text t_j, a word vector $\vec{t_j}$ is obtained as follows

$$\vec{t_j} = [\mathrm{t}_1^j, \ldots, \mathrm{t}_{m_j}^j], \quad j \in \{1, \ldots, n\}, \tag{1}$$

where m_j is the number of terms resulting from the preprocessing of the text t_j and t_i^j is i-th word of preprocessed vector.

Feature vector construction After preprocessing, to calculate the conceptual similarity between different texts, a discriminative feature vector is created for each text. For this purpose, first the contextualized word embedding from BERT [17] is used and thus each term of the texts is mapped to a 768-dimensional space. So, for the i-th term of the text t_j,

the following vector is calculated

$$\overrightarrow{\mathsf{BERT}}(\mathrm{t}_i^j) = [\mathrm{x}_{i1}^j, \ldots, \mathrm{x}_{id}^j], \quad i \in \{1, \ldots, m_j\}, \tag{2}$$

where $d = 768$ and x_h^j is h-th elements of embedding vector of word t_i^j. Moreover, exploiting the TF (term frequency) concept, a rank-weighted density is defined for each term t_i^j. This concept considers the importance of each word based on measures such as its repetition in the entire dataset, its repetition in a particular text and its size, etc. This weighting function can be calculated using the following formula for $i \in \{1, \ldots, m_j\}$ [42]

$$\mathsf{TF}(\mathrm{t}_i^j) = \max\left(0, \frac{\mho(\mathrm{t}_i^j) \times \log\left(\frac{\sqrt{\mathbb{D}}}{\mathbb{H}(\mathrm{t}_i^j)}\right)}{\log\left[\left(\sum_{i=1}^n \mho(\mathrm{t}_i^j)^2\right) \times \left(\frac{\mathbb{L}(j)^2}{\sqrt{\mathbb{D}}}\right)\right]}\right), \tag{3}$$

where $\mho(\mathrm{t}_i^j)$ is the frequency of the term t_i^j in text t_j, $\mathbb{D}$ is the total number of distinct terms in the pre-procedured textual attribute set $\mathcal{T}$ (known as the size of the feature space) and $\mathbb{H}(\mathrm{t}_i^j)$ is the total number of the term t_i^j in the feature space. $\mathbb{L}(j)$ is the length of the texts t_j, measured as the number of distinct terms in $\vec{t_j}$. In $\mathsf{TF}(\mathrm{t}_i^j)$, in addition to the number of repetitions of a term in the text t_j, the ratio of the frequencies of this term and distinct terms in the feature space as well as the ratio of the number of distinct terms in t_j to the number of distinctive terms in set $\mathcal{T}$, are involved to influence the weight of a term based on the number of missing terms in t_j. Text lengths are usually much shorter than the dimensionality of feature spaces, so in order to apply soft normalization, the frequency $\mathcal{T}$ and the length of t_j are squared, while the size of the feature space is square root. [42]. This criterion will be useful for detecting topic noise [1].

Finally, for text t_j, the weighted mean (with nonnegative TF weights) of the d-dimensional vectors of terms that belong to vector $\vec{t_j}$, is taken as its feature vector. That is if we have

$$\mathcal{B}_h^j = \frac{\sum_{i=1}^{m_j}\left(\mathsf{TF}(\mathrm{t}_i^j) \times (\mathrm{x}_{ih}^j)\right)}{\sum_{i=1}^{m_j}\mathsf{TF}(\mathrm{t}_i^j)}, \quad h \in \{1, \ldots, d\}, \tag{4}$$

the feature vector (embedded vector) of t_j is as follows

$$\vec{t_j^e} = [\mathcal{B}_1^j, \ldots, \mathcal{B}_d^j], \quad j \in \{1, \ldots, n\}. \tag{5}$$

Here, $\mathcal{B}_h^j$ is the weighted average on i-th elements of embedded vectors of all the words in text t_j. So, instead of texts, we are dealing with a set of real-valued vectors and we can use the capabilities of vector operations to continue the method. Figure 3 is given to clarify the process of embedding operation.

3.2 Heterogeneous graph construction

To overcome the content bottleneck of target objects and do the right mining, we use a heterogeneous data structure to model the problem. In this structure, the various extraneous communications existing between entities in the dataset can be analyzed to gain more information about the problem. This will enable us to guide and correlate similar target nodes semantically and structurally. If the dataset has $|\mathcal{M}|$ types of main attributes for target objects, we can build a star-structure heterogeneous graph by considering vertex set $V_{\mathcal{N}}$ for target

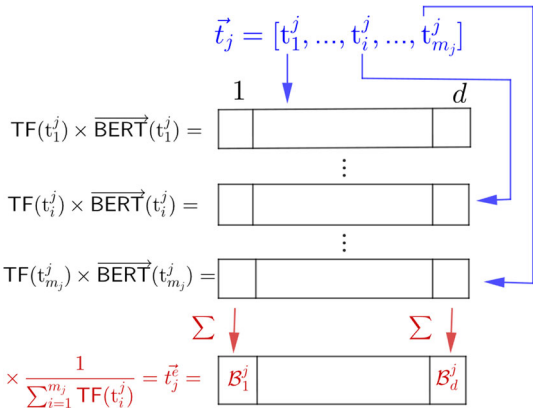

Fig. 3 Embedding processing steps of text t_j

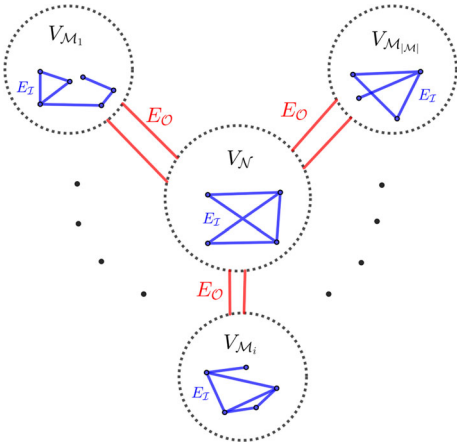

Fig. 4 Star-structure heterogeneous graph

set $\mathcal{N}$, and $|\mathcal{M}|$ star components, each of which is associated with a main attribute. A formal definition of the graph is as follows:

Definition 1 (Star-structure heterogeneous graph) A star-structure heterogeneous graph is defined as a weighted and undirected graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V} = \{V_{\mathcal{N}} \cup V_{\mathcal{M}}\}$ that $V_{\mathcal{N}}$ and $V_{\mathcal{M}} = \cup V_{\mathcal{M}_i} (i = 1, \cdots, |\mathcal{M}|)$, denote the vertex sets of target nodes and main attribute nodes, respectively. The edge set $\mathcal{E} = \{E_{\mathcal{I}} \cup E_{\mathcal{O}}\}$ contains $E_{\mathcal{I}}$ as internal links and $E_{\mathcal{O}}$ as external links. Each internal edge belongs to one of the types of edges $E_{\mathcal{I}} \subseteq V_I \times V_I$, where $I \in \{\mathcal{N}, \mathcal{M}_1, \cdots \mathcal{M}_{|\mathcal{M}|}\}$ and each external edge belongs to one of the types of edges $E_{\mathcal{O}} \subseteq V_{\mathcal{N}} \times V_O$, where $O \in I \backslash \mathcal{N}$. The edges are weighted by a special weight mapping function. $\mathcal{W} : \mathcal{V}^2 \to \mathbb{R}^+$ (Fig. 4).

Weight mapping function We define here the weight of internal and external links separately. – **Internal links** The weighted mean of the following three types of weights is the final weight of internal links. With these weights, we determine the similarity between nodes based on the *direct* connections that exist between them. For simplification, we set $V_{\mathcal{N}} = V_0$ and $V_{\mathcal{M}_i} = V_i$, assume nodes $\{x, y\} \in V_i$ for $i = 0, \cdots, |\mathcal{M}_i|$, and $P_i = \{c_i, \sigma_i, \kappa_i\}$ as positive truncation parameters.

1. *Relational weight* ($w_{\mathcal{R}}$): If a relational attribute exists among x and y, then

$$w_{\mathcal{R}_i}(x, y) = c_i, \tag{6}$$

where $c_i \in [0, 1]$ is a constant. Otherwise $w_{\mathcal{R}_i}(x, y) = 0$. For example, there is no relational attribute in the Twitter dataset, so for $i = 0$, 1 and 2 which are corresponding to tweet, hashtag and user nodes, respectively, we have $w_{\mathcal{R}_i}(x, y) = 0$.

2. *Textual weight* ($w_{\mathcal{T}}$): If x and y have textual attributes, then

$$w_{\mathcal{T}_i}(x, y) = \mathbb{N}(\gamma_i - d_{xy}), \tag{7}$$

where $d_{xy} = ||\vec{t}^{\,e}_x - \vec{t}^{\,e}_y||$ is the Euclidean norm and $\gamma_i = \max_{\{x,y\}\in V_i}(d_{xy})$ and assume notation $\mathbb{N}$ as normalization operator. Note that $\vec{t}^{\,e}_x$ and $\vec{t}^{\,e}_y$ obtained by Eq. (5). We assume that if $w_{\mathcal{T}_i}(x, y) \leq \sigma_i$, then we set it equal to zero. For example, in the Twitter dataset, the similarity between the text of the tweets can be used to determine the textual weights. In the case of two texts that are more similar, the edge between them gets more weight, and in the case of two texts that do not have similar meanings, there is no edge added between them, so the edge's weight is zero.

3. *Joint presence weight* ($w_{\mathcal{J}}$): If $\{x, y\} \in V_{\mathcal{M}_i}$, we define a similarity criterion that is related to the degree of joint presence of x and y in the features of the target nodes as bellow

$$w_{\mathcal{J}_i}(x, y) = \frac{|f_i(x, y)|}{|f_i(x, x) \cup f_i(y, y)|}, \tag{8}$$

where $f_i(x, y) = \{v \in V_{\mathcal{N}} : \{x, y\} \subseteq S_i(v)\}$ and $S_i(v)$ represents the values of the $\mathcal{M}_i$ attribute for target node v. If $w_{\mathcal{J}_i}(x, y) \leq \kappa_i$, then we set it equal to zero. For example, in the Twitter dataset for two hashtags x and y, $f_1(x, y)$ represents the number of times that both hashtags were seen simultaneously in the same tweet. When $f_1(x, y)$ is divided by the total number of x and y presences in the feature space, $w_{\mathcal{J}_1}(x, y)$ is obtained. Additionally, if two hashtags appear in very few tweets simultaneously, their weight is considered zero by κ_1 truncation parameter.

Based on the above weights, we can define the weight of the internal edges as follows. Here, the weighting coefficients are $\alpha = \{\alpha_{\mathcal{R}}, \alpha_{\mathcal{T}}, \alpha_{\mathcal{J}}\}$, which are valued based on the importance of each of the above weights such that $\sum_{r\in\{\mathcal{R},\mathcal{T},\mathcal{J}\}} \alpha_r = 1$.

- (**target**−**target**): An edge is established between two target nodes x and y by weight

$$w(x, y) = \sum_{r\in\{\mathcal{R},\mathcal{T}\}} \alpha_r w_{r_i}(x, y). \tag{9}$$

- (**main attribute**−**main attribute**): An edge is established between two $\mathcal{M}_i$ attribute nodes x and y, by weight

$$w(x, y) = \sum_{r\in\{\mathcal{R},\mathcal{T},\mathcal{J}\}} \alpha_r w_{r_i}(x, y). \tag{10}$$

− **External links** With $w(x, y)$, the latent information contained in each attribute set and target set were used to enhance the network. But in addition to the similarities that the nodes of each category have to each other, the *indirect* connection between the nodes of different sets can also help in strengthening the network as much as possible. For this reason, we can use the external links. These links are related to the relationship between the target nodes and the main attribute nodes. For example, in the network related to the Twitter dataset, external

edges are added between the tweet and the hashtags that exist in it, just as external edges are added between the tweet and the users who tweeted or retweeted it. We set the same weights (usually the maximum value of the weight of the internal edges) to these links. Therefore, the following links can be defined.

- (**target−main attribute**): An edge is established between target node x and $\mathcal{M}_i$ attribute node y, where $y \in S_i(x)$.

In this graph, using σ_i's and κ_i's parameters, the edges with low weight and therefore low impact are not considered. By doing this, we do not encounter a dense graph for subsequent calculations. Moreover, the upper bound defined in the following lemma can be applied to $w_{\mathcal{J}}$ weights category. Utilizing this upper bound, there is no need to calculate the weight of many edges whose upper bound is less than κ_i's. This significantly reduces computation costs by shrinking the search space.

Lemma 1 *If* $\{x, y\} \in V_{\mathcal{M}_i}$, $\frac{\min_D^{\mathcal{O}}(x,y)}{\max_D^{\mathcal{O}}(x,y)}$ *is an upper bound for* $w_{\mathcal{J}_i}(x, y)$, *where* $\min_D^{\mathcal{O}}(x, y)$ *and* $\max_D^{\mathcal{O}}(x, y)$ *indicated the minimum and maximum values between the external degrees of nodes* x *and* y *in star-structure heterogeneous graph, respectively.*

Proof for $\{x, y\} \in V_{\mathcal{M}_i}$, $|f_i(x, y)| \leq \min_D^{\mathcal{O}}(x, y)$ and $|f_i(x, x) \cup f_i(y, y)|) \geq \max_D^{\mathcal{O}}(x, y)$ and so $w_{\mathcal{J}_i}(x, y) \leq \frac{\min_D^{\mathcal{O}}(x,y)}{\max_D^{\mathcal{O}}(x,y)}$. □

After creating the proposed heterogeneous graph, we use the connections between main attribute nodes to strengthen the relationship between target nodes by applying their side effects in a graph that only contains target nodes. So, we define a homogeneous graph with one type of node as a reduced graph of the heterogeneous graph.

3.3 Homogeneous graph construction

To introduce the homogeneous graph, first, we define two types of subgraphs of our heterogeneous graph that are related to target nodes and main attribute nodes, respectively.

Definition 2 (Core Graph) Given a star-structure heterogeneous graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$. The core graph $G_c = (\mathcal{V}_c, \mathcal{E}_c, \mathcal{W})$ is a subgraph of G that is obtained by considering only target nodes and their internal weighted edges, e.i. $\mathcal{V}_c = \{V_{\mathcal{N}}\}$ and $\mathcal{E}_c \in V_{\mathcal{N}} \times V_{\mathcal{N}}$ (Fig. 5a).

Definition 3 ($\mathcal{M}_i$ Shell Graph) Given a star-structure heterogeneous graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{W})$. The $\mathcal{M}_i$ shell graph $G_s^i = (\mathcal{V}_s^i, \mathcal{E}_s^i, \mathcal{W})$ is a subgraph of G on target nodes and $\mathcal{M}_i$ attribute nodes while deleting the internal edges of target nodes, e.i. $\mathcal{V}_s^i = \{V_{\mathcal{N}} \cup V_{\mathcal{M}_i}\}$ and $\mathcal{E}_s^i \subseteq \mathcal{E} \backslash \mathcal{E}_c$, where $\mathcal{E}_c$ is the edge set of the core graph (Fig. 5b).

We use the shell graphs to identify indirect paths between two target nodes, which only contain main attribute nodes as the middle nodes. Here, we define these path models formally that are later used to update the edge weights of the core graph.

Definition 4 ($\mathcal{M}_i$ Auxiliary Path) The $\mathcal{M}_i$ auxiliary path, denoted by $\rho\langle u, p(\mathcal{M}_i), v\rangle$, is a path in the $\mathcal{M}_i$ shell graph, connecting two target nodes u and v, where $p(\mathcal{M}_i)$ is any $\mathcal{M}_i$−path. $\mathcal{M}_i$−path means that only $V_{\mathcal{M}_i}$ are used along the path (Fig. 5c).

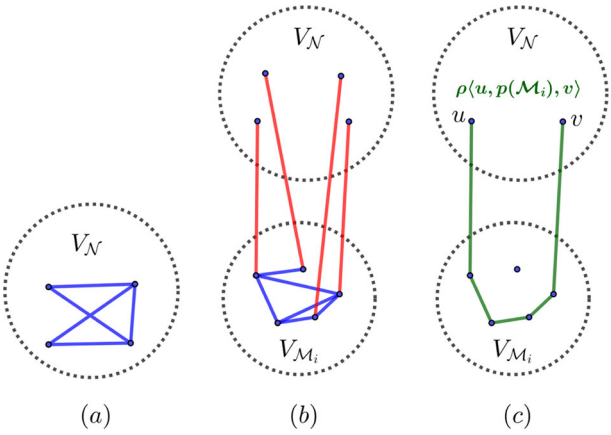

Fig. 5 **a** Core graph, **b** $\mathcal{M}_i$ Shell graph, **c** $\mathcal{M}_i$ Auxiliary Path

Remapping shell graphs Due to the weights assigned to the edges, edges with a higher weight show stronger correlations. So if there is a high weight path for two target nodes in $\mathcal{M}_i$ shell graph, it indicates a strong connection via $V_{\mathcal{M}_i}$ between them. Here, for finding these longest paths (since we have positive circles and this becomes the problem NP-hard [43]), we solve instead the shortest path problem in a *remapped graph* by applying some simple changes to the edge weight of $\mathcal{M}_i$ shell graph by the following similarity-to-distance remapping function to obtain $\mathcal{R}(G_s^i)$ graph

$$\pi(w(x,y)) = \begin{cases} 1 - \mathbb{N}(w(x,y)) & x, y \in V_{\mathcal{M}_i}, \\ |V_{\mathcal{M}_i}| & x \in V_{\mathcal{N}}, \quad y \in V_{\mathcal{M}_i}, \end{cases} \tag{11}$$

where $\mathbb{N}(w(x,y))$ indicates the normalized value of $w(x,y)$ calculated from Eq. (10) and is definitely between zero and one.

Lemma 2 *The shortest path between two target nodes in $\mathcal{R}(G_s^i)$ is a $\mathcal{M}_i$ auxiliary path.*

Proof According to Eq. (11), the weights of the edges between target nodes and $\mathcal{M}_i$ attribute nodes in $\mathcal{R}(G_s^i)$ are greater than the weight of the Hamiltonian path passing through all the $V_{\mathcal{M}_i}$ nodes. So the shortest path cannot have more than two target-main attribute edges and all middle nodes are from $V_{\mathcal{M}_i}$, which is a $\mathcal{M}_i$ auxiliary path. □

Finding (Approximate) shortest path For $\{x, y\} \in V_{\mathcal{M}_i}$, we find the shortest paths through $\mathcal{R}(G_s^i)$ graph for $i = 1, \cdots, |\mathcal{M}|$. If the number of edges in a shell graph is not in the order of the number of its vertices (even after applying truncation parameters κ_i's and σ_i's), we can compute the weight of the approximate shortest paths instead of the exact shortest paths. For this purpose, we use the concept of *spanner* [44]. We apply a greedy method [45] with parameter $\mu_i \geq 1$ on $\mathcal{R}(G_s^i)$ graph and create a subgraph with the number of edges of linear order for it. So, the μ_i-approximate shortest paths can be found in the subgraph which is a spanner of $\mathcal{R}(G_s^i)$. In this method, the edges of graph $\mathcal{R}(G_s^i)$ are first sorted by non-decreasing weight order in set $\mathcal{L}$ and a graph $\mathcal{H} = (\mathcal{V}_s^i, \{\})$ is considered. Then, for every edge $e = [u, v]$ in $\mathcal{L}$, it is added to $\mathcal{H}$ if $\mu_i \times \text{weight}(e) < \text{weight}(P_{\mathcal{H}}(u, v))$, where $P_{\mathcal{H}}(u, v)$ is the shortest path from u to v in graph $\mathcal{H}$. So, a modified Dijkstra's algorithm [46] can be used to find the shortest path in graph $\mathcal{H}$ with $O(|\mathcal{V}_s^i|)$ edges. The shortest path between nodes u and v in graph $\mathcal{H}$ is at most μ_i times the weight of the shortest path in graph $\mathcal{R}(G_s^i)$.

Applying the side effects For two target nodes x and y, which are in the same connected component of $\mathcal{R}(G_s^i)$, assume that $W(\rho_i\langle x, y\rangle)$ is the weight of the $\mathcal{M}_i-$shortest path between x and y in this graph. As mentioned before in the definition of $\mathcal{M}_i$ auxiliary path, here $\mathcal{M}_i-$shortest path is a path, which is obtained by removing the target nodes from the initial and final edges of the shortest path. We apply the positive effect of the weight of the $\mathcal{M}_i-$shortest path on the weight of the edge $e = [x, y]$ in the core graph. For this purpose, the weight of this edge is updated through the following formula and graph $\overline{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \overline{\mathcal{W}})$ is obtained

$$\overline{\mathcal{W}}_{xy} = \beta_0 \mathbb{N}(w(x, y)) + \sum_{i\in\{1,\cdots,|\mathcal{M}|\}} \beta_i \overline{W}(\rho_i\langle x, y\rangle), \tag{12}$$

where $w(x, y)$ obtained from Eq. (9), and the impact factors of β_i's are chosen according to the importance of $\mathcal{M}_i$ main attributes, while $\sum_i \beta_i = 1$ and

$$\overline{W}(\rho_i\langle x, y\rangle) = 1 - \theta_i W(\rho_i\langle x, y\rangle), \tag{13}$$

when $\theta_i = 1/\max_{\{x,y\}\in\mathcal{Q}}\{W(\rho_i\langle x, y\rangle)\}$ applied for scaling the path weights. Here, $\mathcal{Q}$ is a set of pair nodes that are in the same connected component. If nodes x and y are located in two disconnected components, then $\overline{W}(\rho_i\langle x, y\rangle) = 0$. This section is summarized by the following theorem.

Theorem 1 *Graph $\overline{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \overline{\mathcal{W}})$ is a homogeneous core graph in which the positive side effect of shell graphs is applied to its edges.*

3.4 Graph partitioning

We apply the weighted node2vec [22] model, which can take a weighted graph as input, for embedding graph $\overline{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \overline{\mathcal{W}})$. Then evaluate our proposed framework by feeding the embedded nodes to the k-means algorithm. Algorithms 1 and 2 demonstrate the pseudocode of the proposed framework. The input parameters of the algorithms can be divided into three general categories, including the pruning parameters (P_i), the impact factor of attributes parameters (α) and the impact factor of layers (star nodes) parameters (β_i). So, we can adjust the parameters according to the importance of the available information.

Algorithm 1: TARGET_NODE_CLUSTERING

Input: k, $V_{\mathcal{N}}$, α, and $V_{\mathcal{M}_i}$, P_i, β_i for $i = 1$ to $|\mathcal{M}_i|$
Output: k similar clusters
$\mathcal{V}_c \leftarrow V_{\mathcal{N}}$, $\mathcal{E}_c \leftarrow \{\}$, $G_c \leftarrow G(\mathcal{V}_c, \mathcal{E}_c)$
for *each* $\{x, y\} \in V_{\mathcal{N}}$ **do**
  calculate $w(x, y)$ /* use equation (9) */
  $\mathbb{N}(w(x, y)) \leftarrow$ normalized value of $w(x, y)$
end
for $i = 1$ *to* $|\mathcal{M}_i|$ **do**
  $\mathcal{R}(G_s^i)$, $\overline{W}(\rho_i) \leftarrow \mathcal{M}_i$_SHELL_GRAPH($V_{\mathcal{N}}$, $V_{\mathcal{M}_i}$, P_i, α)
end
for *each* $\{x, y\} \in \mathcal{V}_c$ **do**
  $\overline{\mathcal{W}}_{xy} \leftarrow \beta_0\mathbb{N}(w(x, y)) + \sum_{i\in\{1,\cdots,|\mathcal{M}|\}} \beta_i \overline{W}(\rho_i\langle x, y\rangle)$
  add edge $e = [x, y]$ to $\mathcal{E}_c$ by weight $\overline{\mathcal{W}}_{xy}$
end
embed G_c by weighted node2vec method and apply k-means algorithm

Algorithm 2: $\mathcal{M}_i$_SHELL_GRAPH

Input: node sets $V_{\mathcal{N}}$, $V_{\mathcal{M}_i}$ and parameter sets P_i, α
Output: $\mathcal{R}(G_s^i)$, matrix $\overline{W}(\rho_i)$
$\mathcal{V}_s^i \leftarrow \{V_{\mathcal{N}} \cup V_{\mathcal{M}_i}\}$, $\mathcal{E}_s^i \leftarrow \{\}$, $G_s^i \leftarrow G(\mathcal{V}_s^i, \mathcal{E}_s^i)$
for *each* $x \in V_{\mathcal{N}}$ *and* $y \in V_{\mathcal{M}_i}$ **do**
  if y *is an* $\mathcal{M}_i$ *attribute of* x **then**
    add edge $e = [x, y]$ to $\mathcal{E}_s^i$ /* external link */
  end
end
for *each* $\{x, y\} \in V_{\mathcal{M}_i}$ **do**
  add edge $e = [x, y]$ to $\mathcal{E}_s^{\mathcal{M}_i}$ by weight $w(x, y)$ /* internal links, use equation (10) */
end
$\mathcal{R}(G_s^i) \leftarrow$ remapping graph $G_s^{\mathcal{M}_i}$
$\mathcal{Q} \leftarrow \{\}$
for *each* $\{x, y\} \in V_{\mathcal{N}}$ **do**
  if x *and* y *are in the same connected component* **then**
    /* If $\mathcal{E}_s^i = O\left((\mathcal{V}_s^i)^2\right)$ consider approximate shortest path */
    $W(\rho_i\langle x, y\rangle) \leftarrow$ weight of the $\mathcal{M}_i-$shortest path from x to y in $\mathcal{R}(G_s^i)$
    $\mathcal{Q} \leftarrow \{\mathcal{Q} \cup (x, y)\}$
  end
end
$\theta_i \leftarrow 1/\max_{\{x,y\}\in\mathcal{Q}} \{W(\rho_i\langle x, y\rangle)\}$
for *each* $\{x, y\} \in V_{\mathcal{N}}$ **do**
  if $\{x, y\} \in \mathcal{Q}$ **then**
    $\overline{W}(\rho_i\langle x, y\rangle) \leftarrow 1 - \theta_i\,(W(\rho_i\langle x, y\rangle))$
  else
    $\overline{W}(\rho_i\langle x, y\rangle) \leftarrow 0$
  end
end

Remark 1 (Importance of indirect connections) When it comes to clustering, the indirect connection we get from the shortest paths of our proposed framework can be more helpful for many datasets. Consider the Twitter dataset as an example. In this dataset, the words used in hashtags may contain misspellings or even have no specific meaning, so their direct embedding may not find the correct links between them. However, since our model considers the shortest path between hashtags, it can identify related hashtags and thus related tweets. For example, in Fig. 6, although the corresponding hashtags in tweets t_i and t_j are written in two models $\#Kyiv$ and $\#Kiev$, respectively, the existing hashtag auxiliary path (black path) creates an indirect link between these tweets that can strengthen the connection between them.

On the other hand, according to social psychology, most people do not have as much control over their thoughts and behavior as they think [47]. People always learn from their environment, especially from other people. The concept of group polarization states that like-minded people in a group reinforce each other's views [48]. So, users who follow the activities of each other on social networks, can tweet or retweet almost similar topics. Thus, user paths can also strengthen the connection between two related tweets.

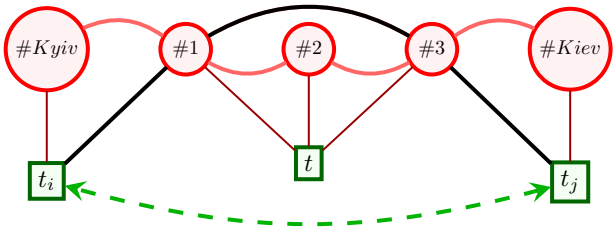

Fig. 6 The shortest path (black path) between two far-away tweets is a hashtag auxiliary path that strengthens the connection between tweets t_i and t_j

4 Experiments

4.1 Comparative models

The purpose of this section is to demonstrate the effectiveness of the proposed unsupervised framework (SEHN) in clustering. Most of the methods presented for clustering on heterogeneous graphs are semi-supervised and are not comparable to our method. However, we compare SEHN with the traditional approach of k-means and some state-of-the-art baselines introduced for the unsupervised clustering (which codes are publicly available). We chose these methods based on considering different types of clustering methods, including the embedding model, the matrix factorization model and the community detection approach. These methods are based on graph structures and do not use semantic information. So, in addition to considering the methods themselves, we also integrated them with the BERT text embedding method (as far as the method allows) to have a fair comparison with our method. The following is a list of baseline models.

- ***k*-means** [20] is a traditional method of vector quantization that aims to partition observations into clusters based on the nearest mean for each observation. Here, we feed the BERT embeddings of text attributes (if available) of the dataset to the k-means algorithm.
- **node2vec** [22] is a generalized version of DeepWalk [23], that uses embedding for graph representation. By optimizing a neighborhood preserving objective, the node2vec framework learns low-dimensional representations for nodes in a graph. We apply the method to the heterogeneous graphs by neglecting the heterogeneity of the graph structure. For evaluation, once the embedded vectors of the method and once the concatenation of embedded vectors and BERT embedding vectors are fed to k-means method.
- **BoostNE** [25] is a multi-level network embedding method based on the technique of matrix factorization. Using a gradient boosting framework, it learns multiple network embedding representations of varying granularity without imposing any global low-rank assumption. The method is also applied to the heterogeneous graphs in the same way as node2vec.
- **GEMSEC** [31] is an embedding algorithm that addresses both embedding and community detection simultaneously while leveraging the benefits of each. The embedding converges through iterations in such a way that nodes become close to their neighbors, while clusters are separated in the embedding space. We evaluate this method as the previous two methods.

4.2 Datasets

We evaluated our framework by considering three datasets, including IMDb, DBLP and our own Twitter data. The information of each dataset can be converted to a star-structure heterogeneous graph, which statistics are tabulated in Table 1. More details of these datasets are also given below.

- **IMDb**[1] is a movie dataset. Here, we analyze a subset of IMDb, which has 4,800 movies (M), 5,851 actors (A) and 2,277 directors (D) after data preprocessing. In terms of genre, movies can be divided into three classes: *action*, *comedy* and *drama*. A bag-of-words description of the plot keywords (K) is also displayed for each movie.
- **DBLP**[2] is a citation network dataset. We analyze a subset of DBLP, which has 29,909 papers (P), 43,784 authors (A) and 20 venue (V) after data preprocessing. There are four areas of papers: *machine learning, information retrieval, data mining* and *database*. Each paper has an abstract. The main keyword of the title of a paper is also considered as its keyword set (K). The citations between papers in this dataset are considered to represent the relationship between them. Here, we consider two versions of DBLP: once with venue attributes (DBLP_1) and once without venue attributes (DBLP_2).
- **Twitter** is a social networking service on which users post tweets or retweet the messages of other users. Each tweet may or may not have hashtags. For the data gathering from this platform, we use the free Twitter API.[3] Using Twitter's streams, we filtered the tweet and retweet posts which contain keywords "Russia" and "Ukraine". These keywords are widely used these days on Twitter because of the 2022 Russia-Ukraine war. By collecting both tweets and retweets, we did not limit ourselves to a specific time and considered the propagation of the information. The data collection period was from 1 April to 15 April 2022. By removing non-English posts, we are left with 301,249 tweets and retweets. Since ground truth topics are not available for this dataset, we used set of keywords with high usage counts as ground truth labels. These keywords are obtained by the human experts and are shown in Table 2.

4.3 Performance evaluation

For assessing the accuracy of the proposed method (SEHN), two standard metrics are used, which are the *normalized mutual information* (NMI) and the *adjusted rand index* (ARI). SEHN and baseline models are compared using the IMDb, DBLP and Twitter datasets. In the following, we explain the setting of parameters for each dataset.

- **IMDb**: In this dataset, the target nodes are "movies" and we have a heterogeneous 3-pointed star-structure graph with center M (movie) and points A (actor), D (director) and K (keyword). Due to only having three main attributes and no relational or contextual attributes, $\alpha_{\mathcal{R}} = \alpha_{\mathcal{T}} = 0$ and $\alpha_{\mathcal{J}} = 1$, respectively, in Eqs. (9) and (10). Given the different values for the κ_i's, we found that the value of 0.3 for these parameters could be the most appropriate. With these values, the number of edges is not very large, and relatively similar nodes have edges in common. Because the effect of all three main attributes for categorizing the movies is the same, we set β_i's equal in Eq. (12), means $\beta_1 = \beta_2 = \beta_3 = 1/3$.

[1] https://www.imdb.com/.

[2] https://dblp.uni-trier.de/.

[3] developer.twitter.com.

Table 1 The statistics of the star-structure heterogeneous graph in our experiments

Dataset	Target node	Main attribute	Textual attribute	Relational attribute	External link	Internal link
IMDb	# movie (M)	# director (D): 12,277			M–D	D–D
	4,800	# actor (A): 5,851	–	–	M–A	A–A
		# keyword (K): 7978			M–K	K–K
DBLP_1	# paper (P)	# author (A): 43,784	abstract	citation	P–K	P–P
	29,909	# keyword (K): 12,370	(P–P)	(P–P)	P–A	A–A
		# venue (V): 20			P–V	K–K
DBLP_2	# paper (P)	# author (A): 43,784	abstract	citation	P–K	P–P
	29,909	# keyword (K): 12,370	(P–P)	(P–P)	P–A	A–A
						K–K
Twitter	# tweet/retweet (T)	# hashtag (H): 11,258	text of tweet		T–H	T–T
	301,249	# user (U): 182,497	(T–T)	–	T–U	H–H
						U–U

Table 2 The ground truth labels of Twitter dataset

Topic	Keyword
1	leak, anonymous, hack, cyber
2	biden, chemical, weapon, biological
3	katerina, maria, daughter, sanction
4	france, election, presidential, macron
5	moskva, black sea, neptune, flagship
6	nft, wallet, nftcommunity, cryptoartist
7	oil, trade, ban, gas, senate, sanction
8	paratrooper, refuse, fight, elite
9	railway, station, kramatorsk, strike
10	nuclear, germany, gas, plant

- **DBLP**: In this dataset, the target nodes are "papers" and we have a heterogeneous 3-pointed star-structure graph with center P (paper) and A (author), K (keyword) and V (venue) as points of it. According to the abstract, the dataset has a textual attribute and because of the citation, there is a relational attribute between the target nodes. So, the weight of the edges between the target nodes calculated based on Eq. (9). Since the citation of an article is completely based on the topic of it, we set c_0 to the maximum state, i.e., 1. By calculating the embedding of the abstracts, we add an edge between papers that have a textual weight greater than $\sigma_0 = 1/4 \sum d_{xy}$, where $\{x, y\}$ belong to a paper set (upper quartile). Based on the co-occurrence of two authors in an article and the co-occurrence of two keywords in an article, we have internal edges that we set the amount of κ_i's related to them to 0.3 by trial and error. To adjust α parameters, since the similarity of abstracts can greatly help the classification, we divide half of the weight to the textual concept and the rest of the weight equally between relational and joint presence concepts. So, we set $\alpha_{\mathcal{T}} = 0.5$ and $\alpha_{\mathcal{J}} = \alpha_{\mathcal{R}} = 0.25$. As with dataset IMDb, the impact of all three main attributes in the classification of papers is the same and so we consider the value of β_i's in Eq. (12) to be the same.
- **Twitter**: In this dataset, as shown in Fig. 2, the target nodes are "tweets" and two layers U (user) and H (hashtag) are considered as the main attributes. Based on the text of the tweets, we have edges between the target nodes that show their similarity. The degree of similarity calculated based on embedding is a textual attribute. Due to the colloquial nature of the text of tweets and the possibility of errors in the embedding, we consider the lower bound of $\sigma_0 = 1/3 \sum d_{xy}$ for these edges, where $\{x, y\}$ belong to tweet set. Also, based on the presence of two hashtags in one tweet and the tweeting/retweeting of one tweet by two people, joint presence edges are created according to Eq. (8). The same as the previous two datasets, we consider the truncation parameters κ_i's to be 0.3. The embedding of the tweets is very helpful in categorizing tweets as well as the indirect connections (according to Remark 1). So, we set their importance equal and we have $\alpha_{\mathcal{T}} = \alpha_{\mathcal{J}} = 0.5$. Because in this dataset the relation between target nodes can be more influential in the clustering, we consider a higher impact factor for it and set $\beta_0 = 0.6$ and $\beta_1 = \beta_2 = 0.2$.

In node2vec model, we assign walk length and walks per node to 80 and 10, respectively. Also $p = 2, q = 0.5$ and dimension is 200. The node2vec embeddings of the core graph is given as input to the k-means algorithm. The value of k is corresponding to the number of

Table 3 Comparison of node clustering results (%) in datasets

Dataset *target node*	IMDb *Movie*		DBLP_1 *Paper*		DBLP_2 *Paper*		Twitter *Tweet*	
Metric	*NMI*	*ARI*	*NMI*	*ARI*	*NMI*	*ARI*	*NMI*	*ARI*
node2vec	0.21	0.19	84.58	88.66	0.10	0.30	23.42	23.16
BoostNE	0.87	0.67	51.79	31.25	7.01	1.91	46.08	35.37
GEMSEC	0.64	0.63	37.88	19.47	15.42	2.33	21.44	16.92
k-means (+BERT)	–	–	19.20	18.89	19.20	18.89	46.14	45.79
node2vec (+BERT)	–	–	84.59	88.67	0.09	0.28	24.58	36.06
BoostNE (+BERT)	–	–	19.49	19.17	19.21	18.90	49.10	38.05
GEMSEC (+BERT)	–	–	35.38	17.05	9.80	1.55	24.71	20.05
SEHN (our method)	**1.02**	**1.06**	**90.67**	**91.53**	**52.61**	**57.14**	**61.52**	**58.49**

classes in each dataset. IMDb, DBLP and Twitter have 3, 4 and 10 classes, respectively. To help minimize the effect of the selection of the centers on the clustering result, we repeat the k-means 10 times and report the best of the obtained results. All the algorithms are implemented in Python with PyTorch and PyTorch Geometric. The experiments are performed on a Linux machine with Intel i7-12700K 2.7 GHz CPU, 12 GB Nvidia 3080-Ti GPU and 128 GB RAM.

Table 3 indicates the results of the experiments comparing our model to other baselines. As can be seen from the table, the methods for the IMDb dataset have not worked well. This is because the movies in this dataset are not properly labeled and each movie has several genres. Our evaluation assumed that the first genre mentioned in each movie's information was its correct label. However, our method has better performance than others.

For the DBLP dataset, we consider two models. Once, we used all available information to create a heterogeneous star graph (DBLP_1), and once we removed the venue nodes from the graph (DBLP_2). Since conferences seem to be a great help in determining the correct cluster for each paper, we investigated the impact of their elimination in the results.

In DBLP_1, our answer is more accurate than the rest. The node2vec method has also obtained the answer with appropriate accuracy. The reason is the random walk-based approach, which is beneficial for node clustering. In this method, nodes are embedded such that nearby nodes in the graph have close embedding vectors [49]. Thus, the location information of the nodes is contained in the embedded vectors and thus k-means works better [11]. Here, due to the presence of venue nodes, the papers of each category are close together in the graph. So it has been expected that node2vec would yield accurate results. It is also observed that adding BERT vectors of abstract at the end of the embedding vectors of the baseline methods does not have much effect on the results. This could be due to the dimensionality curse of k-means for high-dimensional clustering [50] or insufficient information of the abstract.

We also illustrate the result of our method and the comparison methods (the most accurate answer in each case) for DBLP_1 in Fig. 7. For this purpose, we used t-SNE to visualize the embedded points in 2-dimensional space. This figure shows that our method identifies coherent and well-separated categories for this dataset. The nodes in these diagrams each correspond to a paper, and the colors denote the domain in which the papers fall, including machine learning, information retrieval, data mining and database.

As can be seen from the results of DBLP_2 in Table 3, our method not only performed better but also differs greatly from the results of others. It seems that this significant difference

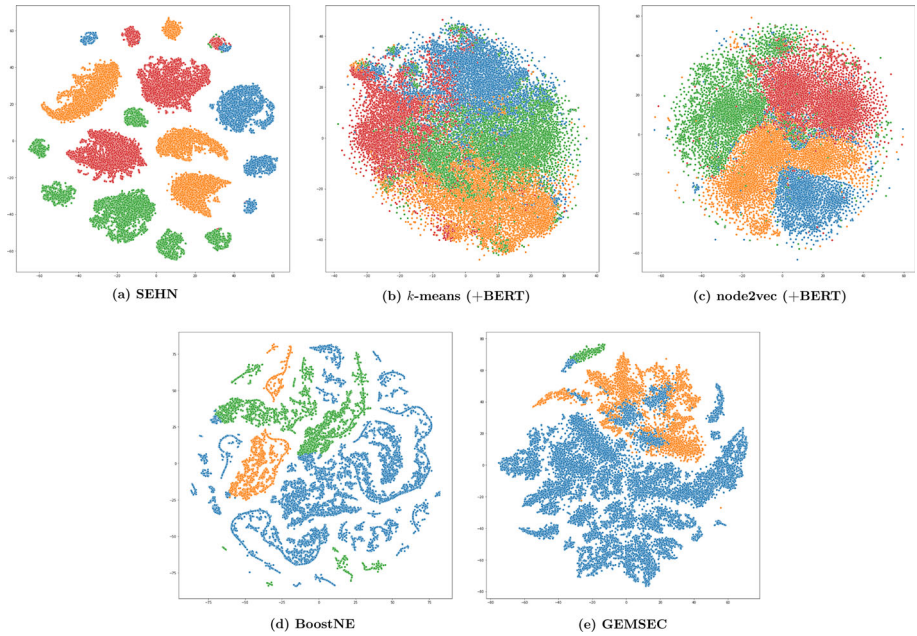

Fig. 7 A visual representation of node clustering in DBLP_1. A point indicates a paper, and its color indicates the research area

between our accuracy and that of other methods is due to the use of indirect paths in the heterogeneous graph, as well as to the advantages of effectively using the structure and attribute information simultaneously.

For the Twitter dataset, our method performed good and was able to separate the 10 topic categories listed in Table 2 with almost high accuracy. Indirect links between hashtags and users enhanced the network compared to using only the text embedding of the tweet to identify the topic. For the members of the set {#rudern, #rowing, #aviron}, for example, our framework could find the shortest paths between them, although there is no direct edge between their members. These three words refer to the same subject but in three different languages. Another example would be the set {#russiancruiser, #shipsinks}.

In summary, based on the results obtained, the advantages of the presented method can be stated as follows.

- Using different types of information in the database to improve the similarity of heterogeneous networks.
- Observing the positive effect of indirect connections in the result, especially when we do not have enough direct information.
- Converting a heterogeneous graph into a homogeneous graph and utilizing the useful and accurate methods for embedding homogeneous graphs.
- Based on the structure of the shell graphs, they can be processed in parallel so that the computational load does not increase.
- As the number of edges of the network increases, the computation can be made applicable by considering truncation parameters and approximating shortest paths. This makes it possible to check larger graphs compared to deep learning methods.

Also, the weaknesses of the method can be stated as follows, and our next goal is to improve the method to eliminate these weaknesses.

- Since only the shortest path is used for indirect communication and other paths are ignored, complex network relationships are not taken into account. We therefore have a local view instead of a global view.
- Only the connection between target nodes and other main attributes is taken into account and the connection between two main attribute sets is ignored.
- The deep learning mechanism was not used in the process of the algorithm.
- The attention mechanism was not used for the influence coefficients of the different attributes. For one target node, the importance of an attribute may differ from the importance of the same attribute for another target node.

5 Conclusion

In this paper, we presented a framework for unsupervised clustering of real-world datasets based on the star-shaped heterogeneous structure. In this structure, we placed the nodes that are the main target of the desired task as the target nodes in the center of the star-shaped structure and considered the other information (main attributes) in the dataset as points of the star. Depending on the existing connections between the nodes, edges with appropriate weights were added to the network based on three criteria, including relational weight, textual weight and joint presence weight. Thus, by properly weighting the edges, we were able to create effective connections between entities that are similar to each other. The presence of these edges created different paths between the target nodes, which allowed us to strengthen the edges between the target nodes. So by considering weights instead of these paths, we achieved a homogeneous graph on the target nodes and embedded this graph with the classical and widely used node2vec method. We then applied the clustering task to the embedded nodes.

The presented method has three categories of parameters, including the parameters for pruning, the parameters for the influencing factors of the attributes and the parameters for the influencing factors of the star components. Thus, we can adjust the parameters according to the importance of the available information. We applied the presented method in clustering three datasets (IMDb, DBLP and Twitter) and obtained better results than the previous unsupervised methods. The Twitter dataset collected by the authors was so large that we were able to process it using appropriate truncation techniques and the use of path approximations.

As mentioned earlier, the proposed method uses the shortest paths between data features to find indirect links. In the future, we plan to consider the shortest paths with the least variance instead. With this assumption, we could find paths where each pair of features is close to each other along the path. To better utilize the information of the dataset, in addition to connecting the target nodes to other main attributes, the connections between pairs of main attributes can also be used. Another goal is to use deep learning methods and attention mechanisms in the algorithm process to consider the influence factor of each feature. Another plan is to introduce a new method for embedding our final homogeneous graph, even with larger datasets.

References

1. Churchill R, Singh L (2021) Topic-noise models: modeling topic and noise distributions in social media post collections. In: 2021 IEEE international conference on data mining (ICDM), pp. 71–80

2. Li Y, Yu R, Shahabi C, Liu Y (2017) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv preprint arXiv:1707.01926
3. Atwood J, Towsley D (2016) Diffusion-convolutional neural networks. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds) Advances in neural information processing systems. 30th Conference on neural information processing systems (NIPS 2016), Barcelona, Spain), vol 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/390e982518a50e280d8e2b535462ec1f-Paper.pdf
4. Shi C, Li Y, Zhang J, Sun Y, Philip SY (2016) A survey of heterogeneous information network analysis. IEEE Trans Knowl Data Eng 29(1):17–37
5. Moscato V, Sperli G (2021) A survey about community detection over on-line social and heterogeneous information networks. Knowl-Based Syst 224:107112
6. Wang X, Bo D, Shi C, Fan S, Ye Y, Philip SY (2022) A survey on heterogeneous graph embedding: methods, techniques, applications and sources. IEEE Tran Big Data 9(2):415–436
7. Dong Y, Chawla NV, Swami A (2017) metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 135–144
8. Fu T-y, Lee W-C, Lei Z (2017) Hin2vec: explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp. 1797–1806
9. Li X, Wu Y, Ester M, Kao B, Wang X, Zheng Y (2017) Semi-supervised clustering in attributed heterogeneous information networks. In: Proceedings of the 26th international conference on World Wide Web, pp. 1621–1629
10. Carranza AG, Rossi RA, Rao A, Koh E (2020) Higher-order clustering in complex heterogeneous networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 25–35
11. Fu X, Zhang J, Meng Z, King I (2020) MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of The Web Conference 2020, pp. 2331–2341
12. Malliaros FD, Vazirgiannis M (2013) Clustering and community detection in directed networks: a survey. Phys Rep 533(4):95–142
13. Rokach L, Maimon O (2005) Clustering methods. Springer, Berlin
14. Aggarwal CC, Zhai C (2012) A survey of text classification algorithms. In: Mining text data. Springer, Boston, MA, pp 163–222. https://doi.org/10.1007/978-1-4614-3223-4_6
15. Leskovec J, Rajaraman A, Ullman JD (2020) Mining of massive data sets. Cambridge University Press, Cambridge
16. Pennington J, Socher R, Manning CD (2014) GloVe: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543
17. Devlin J, Chang M-W, Lee K, Toutanova K (2018) BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
18. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H (2017) Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the 26th international joint conference on artificial intelligence, pp. 1965–1972
19. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International conference on machine learning, pp. 478–487
20. Wagstaff K, Cardie C, Rogers S, Schrödl S (2001) Constrained *k*-means clustering with background knowledge. In: ICML, vol. 1, pp. 577–584
21. Ng A, Jordan M, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: Dietterich T, Becker S, Ghahramani Z (eds) Advances in neural information processing systems, vol 14. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf
22. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 855–864
23. Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 701–710
24. Belkin M, Niyogi P (2001) Laplacian Eigenmaps and spectral techniques for embedding and clustering. In: Dietterich T, Becker S, Ghahramani Z (eds) Advances in neural information processing systems, vol 14. https://proceedings.neurips.cc/paper_files/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf
25. Li J, Wu L, Guo R, Liu C, Liu H (2019) Multi-level network embedding with boosted low-rank matrix approximation. In: Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining, pp. 49–56

26. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
27. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint arXiv:1710.10903
28. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
29. Li P-Z, Huang L, Wang C-D, Lai J-H (2019) EdMot: an edge enhancement approach for motif-aware community detection. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data Mining, pp. 479–487
30. Epasto A, Lattanzi S, Paes Leme R (2017) Ego-splitting framework: from non-overlapping to overlapping clusters. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 145–154
31. Rozemberczki B, Davies R, Sarkar R, Sutton C (2019) GEMSEC: graph embedding with self clustering. In: Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining, pp. 65–72
32. Xie Y, Wang X, Jiang D, Xu R (2019) High-performance community detection in social networks using a deep transitive autoencoder. Inf Sci 493:75–90
33. Jia Y, Zhang Q, Zhang W, Wang X (2019) Communitygan: community detection with generative adversarial nets. In: The World Wide Web Conference, pp. 784–794
34. Rostami M, Oussalah M, Berahmand K, Farrahi V (2023) Community detection algorithms in healthcare applications: a systematic review. IEEE Access 11:30247
35. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: The World Wide Web Conference, pp. 2022–2032
36. Zhang C, Song D, Huang C, Swami A, Chawla NV (2019) Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 793–803
37. Sun Y, Yu Y, Han J (2009) Ranking-based clustering of heterogeneous information networks with star network schema. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 797–806
38. Forouzandeh S, Berahmand K, Sheikhpour R, Li Y (2023) A new method for recommendation based on embedding spectral clustering in heterogeneous networks (reschet). Expert Syst Appl 231:120699
39. Sheikhpour R, Berahmand K, Forouzandeh S (2023) Hessian-based semi-supervised feature selection using generalized uncorrelated constraint. Knowl-Based Syst 269:110521
40. Chang Y, Chen C, Hu W, Zheng Z, Zhou X, Chen S (2022) MEGNN: meta-path extracted graph neural network for heterogeneous graph representation learning. Knowl-Based Syst 235:107611
41. Cheng H, Zhou Y, Yu JX (2011) Clustering large attributed graphs: a balance between structural and attribute similarities. ACM Trans Knowl Discov Data 5(2):1–33
42. Sabbah T, Selamat A, Selamat MH, Al-Anzi FS, Viedma EH, Krejcar O, Fujita H (2017) Modified frequency-based term weighting schemes for text classification. Appl Soft Comput 58:193–206
43. Schrijver A (2003) Combinatorial optimization: polyhedra and efficiency. Springer, Berlin
44. Narasimhan G, Smid M (2007) Geometric spanner networks. Cambridge University Press, Cambridge
45. Althöfer I, Das G, Dobkin D, Joseph D, Soares J (1993) On sparse spanners of weighted graphs. Discret Comput Geom 9(1):81–100
46. Thorup M, Zwick U (2005) Approximate distance oracles. J ACM 52(1):1–24
47. Deutsch M, Krauss RM (1965) Social psychology. Basic Books, New York
48. Isenberg DJ (1986) Group polarization: a critical review and meta-analysis. J Pers Soc Psychol 50(6):1141
49. You J, Ying R, Leskovec J (2019) Position-aware graph neural networks. In: International conference on machine learning, pp. 7134–7143
50. Ding C, Li T (2007) Adaptive dimension reduction using discriminant analysis and k-means clustering. In: Proceedings of the 24th international conference on machine learning, pp. 521–528

Fatemeh Baharifard is a senior postdoc researcher at Institute for Research in Fundamental Sciences (IPM). She received her PhD in Computer Science from the School of Computer Science, IPM in 2018. Her research interests focus on algorithmic graph theory, approximation algorithms and machine learning.

Vahid Motaghed received his M.Sc. degree in artificial intelligence from the Shiraz University, Shiraz, Iran, in 2012. He is currently a researcher at the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), and his research interests are deep learning, natural language processing and machine learning.