Chapter 5
Dynamic Heterogeneous Graph Representation

Abstract Graphs are gradually generated with multiple temporal heterogeneous interactions in real-world scenarios, containing both abundant structures and complex dynamics. Compared to static heterogeneous graphs, the dynamics express not only the changing graph topology but also the sequential evolution as well as multiple temporal preferences, indicating the necessity of dynamic heterogeneous graph modeling. This chapter focuses on simultaneously modeling both the evolving dynamics and heterogeneous semantics, and introduces three representative approaches, including DyHNE to handle structure changes via matrix perturbation theory based incremental learning, SHCF to tackle evolving sequences via heterogeneous sequential neural collaborative filtering, and THIGE to model multiple long- and short-term preferences via temporal heterogeneous GNNs.

5.1 Introduction

Heterogeneous graphs in real-world scenarios usually exhibit high dynamics with the evolution of various types of nodes and edges, e.g., the newly added (deleted) nodes or edges, forming the dynamic heterogeneous graph or temporal heterogeneous graph. The complex but valuable temporal heterogeneity introduce essential challenges for representation. Specifically, heterogeneous interactions are accumulated over time, leading to the continuously changes of graphic topologies. Besides, interactions are often sequentially generated, which indicate the corresponding evolution of interests. Furthermore, there are multiple temporal interactions, expressing both heterogeneous sequential evolution of current demands and multi-view historical habits.

However, the current heterogeneous graph embedding methods [37, 42, 16, 6, 3] cannot model the dynamics within structural semantics, which have to be retrained repeatedly at each time step. Focusing on modeling the evolving dynamics, some researchers attempt to integrate with Recurrent Neural Networks (RNNs) [13, 23, 14, 31] and Transformers [18, 36, 46], and some recent approaches [25, 38, 26] pay attention to combine both short-term and long-term interests to generate fine-grained

node representations. Unfortunately, almost all of these sequential methods construct user embeddings based on only their own sequential item interactions while ignoring the abundant heterogeneous information.

In this chapter, we introduce three dynamic heterogeneous graph representation learning methods to solve the challenges in incremental learning, sequential information and temporal interactions. First, we introduce the **Dy**namic **H**eterogeneous **N**etwork **E**mbedding (named **DyHNE**) [26] to handle the incremental learning of temporal semantics by utilizing matrix perturbation theory. Second, we introduce the **S**equence-aware **H**eterogeneous graph neural **C**ollaborative **F**iltering (called **SHCF**) [21] to model the heterogeneous evolution of sequential information for recommendation. Third, we introduce the **T**emporal **H**eterogeneous **I**nteraction **G**raph **E**mbedding (named **THIGE**) [17] to model both long-term habits and short-term demands of temporal interactions via temporal heterogeneous GNNs.

5.2 Incremental Learning

5.2.1 Overview

Heterogeneous graph are often gradually formatted with dynamic edges. The current HG embedding methods can hardly handle such complex evolution effectively in a dynamic heterogeneous graph. Basically, there are two fundamental problems which need to be carefully considered for dynamic heterogeneous graph embedding. One is how to effectively preserve the structure and semantics in a dynamic heterogeneous graph. As the heterogeneous graph evolves with a newly added node, the local structure centered on this node will be changed, and such changes will be gradually propagated across all the nodes via different meta-paths, leading to changes in the global structure. The other problem is how to efficiently update the node embeddings without retraining on the whole heterogeneous graph, when the heterogeneous graph evolves over time. For each time step, retraining a heterogeneous graph embedding method is the most straightforward way to get the optimal embeddings. However, apparently, this strategy is very time consuming, especially when the change of network structure is very slight. In the era of big data, retraining manner becomes unrealistic. These problems motivate us to seek an effective and efficient method to preserve the structure and semantics for dynamic heterogeneous graph embedding.

This section designs the DyHNE with meta-path based proximity to effectively and efficiently learn the node embeddings. Inspired by the perturbation theory [10] widely used for capturing changes of a system, we learn the node embeddings by solving the generalized eigenvalue problem and model the evolution of the heterogeneous graph with the eigenvalue perturbation. We firstly adopt meta-path augmented adjacency matrices to capture the structure and semantics in dynamic heterogeneous graphs. For capturing the evolution of the heterogeneous graph, we then utilize the perturbations of multiple meta-path augmented adjacency matrices to model the

changes of the structure and semantics of the heterogeneous graph in a natural manner. Finally, we employ the eigenvalue perturbation theory to incorporate the changes and derive the node embeddings efficiently.

5.2.2 The DyHNE Model

The core idea of DyHNE is to build an effective and efficient architecture that can capture the changes of structure and semantics in a dynamic heterogeneous graph and derive the node embeddings efficiently.

To achieve this, we first introduce the meta-path based first-order and second-order proximity to preserve structure and semantics in heterogeneous graphs. As shown in Fig. 5.1, three augmented adjacency matrices based on meta-path *APA*, *APCPA* and *APTPA* are defined and fused with weights, which gives rise to the fused matrix $\mathbf{W}^{(t)}$ at time t. Then, we learn node embeddings $\mathbf{U}^{(t)}$ by solving the generalized eigenvalue problem in terms of the fused matrix $\mathbf{W}^{(t)}$. As the heterogeneous graph evolves from time t to $t+1$, new nodes and edges are added into the network (i.e. nodes a_3, p_4 and t_3; edges $(a_3, p_4), (a_1, p_4), (p_4, c_2), (p_4, t_2)$ and (p_4, t_3)), leading to the changes of meta-path augmented adjacency matrices. Since these matrices are actually the realization of structure and semantics in the heterogeneous graph, we naturally capture changes of structure and semantics with the perturbation of the fused matrix (i.e. $\Delta\mathbf{W}$). Further, we tailor the embeddings update formulas for dynamic heterogeneous graph with matrix perturbation theory, so that our DyHNE can efficiently derive the changed embedding $\Delta\mathbf{U}$ and update network embedding from $\mathbf{U}^{(t)}$ to $\mathbf{U}^{(t+1)}$ with $\mathbf{U}^{(t+1)} = \Delta\mathbf{U} + \mathbf{U}^{(t)}$.

In a nutshell, the proposed StHNE is capable of capturing the structures and semantics in a heterogeneous graph with meta-path based first-order and second-order proximity, and DyHNE achieves the efficient update of network embeddings with the perturbation of meta-path augmented adjacency matrices.

5.2.2.1 Static Modeling

Before achieving effective update node embeddings when the heterogeneous graph evolves over time, a proper static heterogeneous graph embedding for capturing structural and semantic information is a must. Hence, we next propose a **St**atic **H**eterogeneous **N**etwork **E**mbedding model (named **StHNE**), which preserves the meta-path based first- and second-order proximity. The meta-path based first-order proximity models the local proximity in heterogeneous graphs, which means that the nodes connected via path instances are similar. Given a node pair (v_i, v_j) connected via path instances following m, we model the meta-path based first-order proximity as:

$$p_1^m(v_i, v_j) = w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2, \tag{5.1}$$

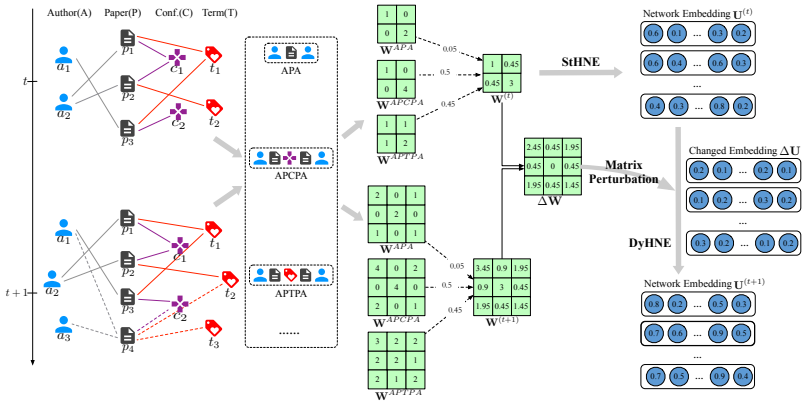


Fig. 5.1 The overall architecture of the proposed StHNE and DyHNE.

where $\mathbf{u}_i \in \mathbb{R}^d$ is the d-dimension representation vector of node v_i. To preserve the meta-path based first-order proximity in heterogeneous graphs, we minimize the following objective function:

$$\mathcal{L}_1^m = \sum_{v_i, v_j \in \mathcal{V}} w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2. \tag{5.2}$$

As larger w_{ij}^m indicates that v_i and v_j have more connections via the meta-path m, which makes nodes v_i and v_j closer in the low-dimensional space.

The meta-path based second-order proximity is determined through the shared neighborhood structure of nodes. Given the neighbors of node v_p under the meta-path m, denoted as $\mathcal{N}(v_p)^m$, we can model the second-order proximity based on meta-path as follows:

$$p_2^m(v_p, \mathcal{N}(v_p)^m) = ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2. \tag{5.3}$$

Here, we normalize w_{pq}^m so that $\sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq} = 1$.

With Eq. (5.3), we keep the node p close to its neighbors under a specific meta-path. Eq. (5.3) guarantees that unconnected nodes are close to each other if they contain the similar neighbors. To preserve the meta-path based second-order proximity in heterogeneous graphs, we minimize the following object function, namely

$$\mathcal{L}_2^m = \sum_{v_p \in \mathcal{V}} ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2. \tag{5.4}$$

Intuitively, minimizing Eq. (5.4) will cause the small distance between node v_p and its neighbors in the low-dimensional space. Thus, nodes that shares the same neighbors with node v_p will also be close to v_p. In this way, the meta-path based

second-order proximity can be preserved. Considering multiple semantic relations in a heterogeneous graph, we define a set of meta-paths $\mathcal{M}$ and assign weights $\{\theta_1, \theta_2, ..., \theta_{|\mathcal{M}|}\}$ to each meta-path, where $\forall \theta_i > 0$ and $\sum_{i=1}^{|\mathcal{M}|} \theta_i = 1$. Thus, our unified model combines multiple meta-paths while preserving both of the meta-path based first- and second-order proximity, namely

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \theta_m (\mathcal{L}_1^m + \gamma \mathcal{L}_2^m), \tag{5.5}$$

where γ is the trade-off factor. Now, the static heterogeneous graph embedding problem is turned to:

$$\arg \min_{\mathbf{U}^\top \mathbf{D} \mathbf{U} = \mathbf{I}} \sum_{m \in \mathcal{M}} \theta_m (\mathcal{L}_1^m + \gamma \mathcal{L}_2^m), \tag{5.6}$$

where $\mathbf{D}$ is the degree matrix that will be described later. The constraint $\mathbf{U}^\top \mathbf{D} \mathbf{U} = \mathbf{I}$ removes an arbitrary scaling factor in the embedding and avoids the degenerate case where all node embeddings are equal.

Inspired by spectral theory [27, 2], we transform the problem of Eq. (5.6) as the generalized eigenvalue problem, so that we can get a closed-form solution and dynamically update embeddings with the eigenvalue perturbation theory [10]. Hence, we reformulate Eq. (5.2) as follows:

$$\mathcal{L}_1^m = \sum_{v_i, v_j \in \mathcal{V}} w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2 = 2tr(\mathbf{U}^\top \mathbf{L}^m \mathbf{U}), \tag{5.7}$$

where $tr(\cdot)$ is the trace of the matrix, $\mathbf{U}$ is the embedding matrix, $\mathbf{L}^m = \mathbf{D}^m - \mathbf{W}^m$ is the Laplacian matrix under the meta-path m, and $\mathbf{D}^m$ is a diagonal matrix with $\mathbf{D}_{ii}^m = \sum_j w_{ij}^m$. Similarly, Eq. (5.4) can be rewritten as follows:

$$\mathcal{L}_2^m = \sum_{v_p \in \mathcal{V}} ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2 = 2tr(\mathbf{U}^\top \mathbf{H}^m \mathbf{U}), \tag{5.8}$$

where $\mathbf{H}^m = (\mathbf{I} - \mathbf{W}^m)^\top (\mathbf{I} - \mathbf{W}^m)$ is symmetric. As discussed earlier, we fuse all meta-paths in $\mathcal{M}$, which gives rise to:

$$\mathbf{W} = \sum_{m \in \mathcal{M}} \theta_m \mathbf{W}^m, \qquad \mathbf{D} = \sum_{m \in \mathcal{M}} \theta_m \mathbf{D}^m. \tag{5.9}$$

Hence, the StHNE can be reformulated as:

$$\mathcal{L} = tr(\mathbf{U}^\top (\mathbf{L} + \gamma \mathbf{H}) \mathbf{U})), \tag{5.10}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{H} = (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})$. Now, the problem of static heterogeneous graph embedding reduces to:

$$\arg \min_{\mathbf{U}^\top \mathbf{D} \mathbf{U} = \mathbf{I}} tr(\mathbf{U}^\top (\mathbf{L} + \gamma \mathbf{H}) \mathbf{U}), \tag{5.11}$$

where $\mathbf{L}+\gamma\mathbf{H}$ is symmetric. The problem of Eq. (5.11) boils down to the generalized eigenvalue problem [40] as follows:

$$(\mathbf{L}+\gamma\mathbf{H})\mathbf{U} = \mathbf{D}\lambda\mathbf{U}, \tag{5.12}$$

where $\lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{N_\mathcal{M}})$ is the eigenvector matrix, $N_\mathcal{M}$ is the number of nodes in the meta-path set $\mathcal{M}$.

Having transformed the StHNE as the generalized eigenvalue problem, the embedding matrix $\mathbf{U}$ is given by the top-d eigenvectors with the smallest non-zero eigenvalues. As the heterogeneous graph evolves from time t to $t+1$, the dynamic heterogeneous graph embedding model focuses on efficiently updating $\mathbf{U}^{(t)}$ to $\mathbf{U}^{(t+1)}$. That is, update the eigenvectors and eigenvalues.

5.2.2.2 Dynamic Modeling

The core idea of a dynamic heterogeneous graph embedding model is to learn node embeddings efficiently in a dynamic manner, thus we next to effectively update the eigenvectors and eigenvalues based on matrix perturbation.

Following the previous works [22, 50], we assume that the network evolves on a common node set of cardinality N. A nonexistent node is treated as an isolated node with zero degree and thereby the evolution of a network can be regarded as the change of edges [1]. Besides, the addition (deletion) of edges may vary by types. It is naturally appealing to capture the evolution of a dynamic heterogeneous graph with the perturbation of meta-path augmented adjacency matrix $\Delta\mathbf{W} = \sum_{m\in\mathcal{M}} \theta_m \Delta\mathbf{W}^m$. Thus, the changes of $\mathbf{L}$ and $\mathbf{H}$ can be calculated as follows:

$$\Delta\mathbf{L} = \Delta\mathbf{D} - \Delta\mathbf{W}, \tag{5.13}$$

$$\Delta\mathbf{H} = \Delta\mathbf{W}^\top\Delta\mathbf{W} - (\mathbf{I}-\mathbf{W})^\top\Delta\mathbf{W} - \Delta\mathbf{W}^\top(\mathbf{I}-\mathbf{W}). \tag{5.14}$$

Since perturbation theory can give approximate solution to a problem by adding a perturbation term [10], we can update eigenvalues and eigenvectors from the eigenvalues and eigenvectors at the previous time with the eigenvalue perturbation. Hence, at new time step, we have the following equation based on Eq. (5.12):

$$(\mathbf{L}+\Delta\mathbf{L}+\gamma\mathbf{H}+\gamma\Delta\mathbf{H})(\mathbf{U}+\Delta\mathbf{U}) = (\mathbf{D}+\Delta\mathbf{D})(\lambda+\Delta\lambda)(\mathbf{U}+\Delta\mathbf{U}), \tag{5.15}$$

where $\Delta\mathbf{U}$ and $\Delta\lambda$ are the changes of the eigenvectors and eigenvalues. Here, we omit the (t) superscript for brevity since the perturbation process for any time step t is the same. Let us focus on a specific eigen-pair $(\mathbf{u}_i, \lambda_i)$, Eq. (5.15) is rewritten as

$$(\mathbf{L}+\Delta\mathbf{L}+\gamma\mathbf{H}+\gamma\Delta\mathbf{H})(\mathbf{u}_i+\Delta\mathbf{u}_i) = (\lambda_i+\Delta\lambda_i)(\mathbf{D}+\Delta\mathbf{D})(\mathbf{u}_i+\Delta\mathbf{u_i}). \tag{5.16}$$

Hence, the dynamic heterogeneous graph embedding problem is how to calculate the changes of the i-th eigen-pair $(\Delta\mathbf{u}_i, \Delta\lambda_i)$, because if we have $\Delta\mathbf{U}$ and $\Delta\lambda$ between t and $t+1$, we can efficiently update the embedding matrix with $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \Delta\mathbf{U}$.

We first introduce how to calculate $\Delta\lambda_i$. By expanding Eq. (5.16) and removing the higher order terms that have limited effects on the accuracy of the solution [10], such as $\Delta\mathbf{L}\Delta\mathbf{u}_i$ and $\Delta\lambda_i\Delta\mathbf{D}\Delta\mathbf{u}_i$, then based on the fact $(\mathbf{L}+\gamma\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{D}\mathbf{u}_i$, we have

$$(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i + (\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{D}\Delta\mathbf{u}_i + \lambda_i\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{D}\mathbf{u}_i. \tag{5.17}$$

Furthermore, left multiplying both sides by $\mathbf{u}_i^\top$, we have

$$\mathbf{u}_i^\top(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i + \mathbf{u}_i^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{u}_i^\top\mathbf{D}\Delta\mathbf{u}_i + \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i. \tag{5.18}$$

As $\mathbf{L}+\gamma\mathbf{H}$ and $\mathbf{D}$ are symmetric, then based on the fact $(\mathbf{L}+\gamma\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{D}\mathbf{u}_i$ and right multiplying both side by $\Delta\mathbf{u}_i$, we have $\mathbf{u}_i^\top(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i = \lambda_i\mathbf{u}_i^\top\mathbf{D}\Delta\mathbf{u}_i$. Thus, we can rewrite Eq. (5.18) as follows:

$$\mathbf{u}_i^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i. \tag{5.19}$$

Based on Eq. (5.19), we get the changes of the eigenvalue λ_i:

$$\Delta\lambda_i = \frac{\mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i + \gamma\mathbf{u}_i^\top\Delta\mathbf{H}\mathbf{u}_i - \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i}{\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i}. \tag{5.20}$$

It is easy to see that $\mathbf{D}$ is a positive-semidefinite matrix, so we have $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i = 1$ and $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_j = 0 (i \neq j)$ [29, 10]. Thus,

$$\Delta\lambda_i = \mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i + \gamma\mathbf{u}_i^\top\Delta\mathbf{H}\mathbf{u}_i - \lambda\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i. \tag{5.21}$$

Having got the change of eigenvalue $\Delta\lambda_i$ between two continuous time steps, our next goal is to calculate the changes of eigenvectors $\Delta\mathbf{u}_i$.

As a heterogeneous graph usually evolves smoothly [1], the network changes based on meta-paths (i.e., $\Delta\mathbf{W}$) are subtle. We assume the perturbation of the eigenvectors $\Delta\mathbf{u}_i$ is linearly weighted by the top-d eigenvectors with the smallest non-zero eigenvalues [10]:

$$\Delta\mathbf{u}_i = \sum_{j=2, j\neq i}^{d+1} \alpha_{ij}\mathbf{u}_j, \tag{5.22}$$

where α_{ij} indicates the weight of $\mathbf{u}_j$ on $\Delta\mathbf{u}_i$. Thus, the problem of calculating $\Delta\mathbf{u}_i$ now is transformed into how to determine these weights. Considering Eq. (5.16), by replacing $\Delta\mathbf{u}_i$ with Eq. (5.22) and removing the higher order terms that have limited effects on the accuracy of the solution [19], we obtain the following:

$$
\begin{aligned}
&(\mathbf{L}+\gamma\mathbf{H})\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j+(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\
=&\lambda_i\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j+\lambda_i\Delta\mathbf{D}\mathbf{u}_i+\Delta\lambda_i\mathbf{D}\mathbf{u}_i.
\end{aligned}
\tag{5.23}
$$

With the fact that $(\mathbf{L}+\gamma\mathbf{H})\sum_{j=2}^{d+1}\alpha_{ij}\mathbf{u}_j=\mathbf{D}\sum_{j=2}^{d+1}\alpha_{ij}\lambda_j\mathbf{u}_j$, and by multiplying $\mathbf{u}_p^\top(2\leq p\leq d+1,p\neq i)$ on both sides of Eq. (5.23), we get:

$$
\begin{aligned}
&\mathbf{u}_p^\top\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\lambda_j\mathbf{u}_j+\mathbf{u}_p^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\
=&\lambda_i\mathbf{u}_p^\top\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j+\lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i+\Delta\lambda_i\mathbf{u}_p^\top\mathbf{D}\mathbf{u}_i.
\end{aligned}
\tag{5.24}
$$

Based on $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i=1$ and $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_j=0(i\neq j)$, we can simplify the above formula and get:

$$
\lambda_p\alpha_{ip}+\mathbf{u}_p^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i=\lambda_i\alpha_{ip}+\lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i. \tag{5.25}
$$

Finally, we obtain the weight α_{ip} as follows:

$$
\alpha_{ip}=\frac{\mathbf{u}_p^\top\Delta\mathbf{L}\mathbf{u}_i+\gamma\mathbf{u}_p^\top\Delta\mathbf{H}\mathbf{u}_i-\lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i}{\lambda_i-\lambda_p},i\neq p. \tag{5.26}
$$

To sum up, we now have the changes of eigenvalues and eigenvectors based on Eq. (5.21), (5.22) and (5.26). The new eigenvalues and eigenvectors at $t+1$ can be updated as follows:

$$
\lambda^{(t+1)}=\lambda^{(t)}+\Delta\lambda,\qquad \mathbf{U}^{(t+1)}=\mathbf{U}^{(t)}+\Delta\mathbf{U}. \tag{5.27}
$$

5.2.2.3 Acceleration

A straightforward idea to update the embeddings is to calculate Eq. (5.21), (5.22) and (5.26) for Eq. (5.27). However, the calculation of Eq. (5.21) is time-consuming due to the definition of $\Delta\mathbf{H}$ (i.e., Eq. (5.14)). Thus, we propose an acceleration solution tailored for dynamic heterogeneous graph embedding.

Let us focus on $\Delta\lambda_i$ and α_{ij} in a more detailed way. We replace $\Delta\mathbf{H}$ with Eq. (5.14) and remove the higher order terms as earlier, Eq. (5.21) and Eq. (5.26) can be reformulated as follows:

$$
\Delta\lambda_i=\mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i-\lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i+\gamma\{[(\mathbf{W}-\mathbf{I})\mathbf{u}_i]^\top\Delta\mathbf{W}\mathbf{u}_i+(\Delta\mathbf{W}\mathbf{u}_i)^\top(\mathbf{W}-\mathbf{I})\mathbf{u}_i\}, \tag{5.28}
$$

$$\alpha_{ij} = \frac{\mathbf{u}_j^\top \Delta\mathbf{L}\mathbf{u}_i - \lambda_i \mathbf{u}_j^\top \Delta\mathbf{D}\mathbf{u}_i}{\lambda_i - \lambda_j} + \frac{\gamma\{[(\mathbf{W}-\mathbf{I})\mathbf{u}_j]^\top \Delta\mathbf{W}\mathbf{u}_i + (\Delta\mathbf{W}\mathbf{u}_j)^\top (\mathbf{W}-\mathbf{I})\mathbf{u}_i\}}{\lambda_i - \lambda_j}. \tag{5.29}$$

For the sake of convenience, we rewrite $\Delta\lambda_i$ and α_{ij} as follows:

$$\Delta\lambda_i = \mathbf{C}(i,i) + \gamma[\mathbf{A}(:,i)^\top \mathbf{B}(:,i) + \mathbf{B}(:,i)^\top \mathbf{A}(:,i)], \tag{5.30}$$

$$\alpha_{ij} = \frac{\mathbf{C}(j,i) + \gamma[\mathbf{A}(:,j)^\top \mathbf{B}(:,i) + \mathbf{B}(:,j)^\top \mathbf{A}(:,i)]}{\lambda_i - \lambda_j}, \tag{5.31}$$

where $\mathbf{A}(:,i) = (\mathbf{W}-\mathbf{I})\mathbf{u}_i$, $\mathbf{B}(:,i) = \Delta\mathbf{W}\mathbf{u}_i$ and $\mathbf{C}(i,j) = \mathbf{u}_i^\top \Delta\mathbf{L}\mathbf{u}_j - \lambda_i \mathbf{u}_i^\top \Delta\mathbf{D}\mathbf{u}_j$.

Obviously, the calculation of $\mathbf{A}$ is time-consuming. Hence, we define $\mathbf{A}^{(t+1)}(:,i)$ at time step $t+1$ as follows:

$$\mathbf{A}^{(t+1)}(:,i) = (\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})(\mathbf{u}_i + \Delta\mathbf{u}_i). \tag{5.32}$$

Replacing $\Delta\mathbf{u}_i$ with Eq. (5.22), we have

$$\mathbf{A}^{(t+1)}(:,i) = (\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})(\mathbf{u}_i + \sum_{j=2, j\neq i}^{d+1} \alpha_{ij}\mathbf{u}_j) = \sum_{j=2}^{d+1} \beta_{ij}(\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})\mathbf{u}_j, \tag{5.33}$$

where $\beta_{ij} = \alpha_{ij}$ if $i \neq j$, otherwise, $\beta_{ij} = 1$. Furthermore, we can obtain the following:

$$\mathbf{A}^{(t+1)}(:,i) = \sum_{j=2}^{d+1} \beta_{ij}(\mathbf{A}^t(:,j) + \mathbf{B}^t(:,j)). \tag{5.34}$$

Now, we reduce the time complexity of updating $\mathbf{A}^{(t+1)}$ from $O(ed)$ to $O(d^2)$, which guarantees the efficiency of DyHNE.

5.2.3 Experiments

5.2.3.1 Experimental Settings

Datasets. We evaluate models on three datasets, including two academic networks (i.e., DBLP and AMiner) and a social Yelp. Yelp dataset extracts information related to restaurants of three sub-categories: “American (New) Food", “Fast Food" and “Sushi Bars" [24]. The meta-paths that we are interested in are *BRURB* (i.e., the user reviewed on two businesses) and *BSB* (i.e., the same star level businesses). DBLP is an academic network in computer science where the authors are labeled with their research areas. We consider meta-paths including *APA* (i.e., the co-author

relationship), *APCPA* (i.e., authors sharing conferences) and *APTPA* (i.e., authors sharing terms). AMiner is also an academic network, which evolved from 1990 to 2005 in five research domains and the meta-paths are *APA*, *APCPA* and *APTPA*.

Baselines. We compare our proposed StHNE and DyHNE with comprehensive state-of-the-art alternatives, including two homogeneous network embedding methods (i.e., DeepWalk [30] and LINE [39]); two heterogeneous information network embedding methods (i.e., ESim [33] and metapath2vec [8]); and two dynamic homogeneous network embedding methods (i.e., DANE [22], DHPE [50] and DHNE [47]). Additionally, in order to verify the effectiveness of the meta-path based first-order and second-order proximity, we test the performance of StHNE-1st and StHNE-2nd. We use codes of the baseline methods provided by their authors.

5.2.3.2 Effectiveness of StHNE

To evaluate the effectiveness of StHNE, here we learn the node embeddings with the static embedding methods on the whole heterogeneous graph without considering the evolution of the network. In other words, given a dynamic network with 10 time steps $\{\mathcal{G}^1, \cdots, \mathcal{G}^{10}\}$, we conduct all static network embedding methods, including StHNE, on the union network, i.e., $\mathcal{G}^1 \cup \mathcal{G}^1 \cup \cdots \mathcal{G}^{10}$.

Node Classification. Node classification is a common task to evaluate the performance of representation learning on networks. In this task, after learning the node embeddings on the fully evolved network, we train a logistic regression classifier with node embeddings as input features. The ratio of training set is set as 40%, 60%, and 80%. We set the weights of *BSB* and *BRURB* in Yelp to 0.4 and 0.6. In DBLP, we assign weights {0.05, 0.5, 0.45} to {*APA*, *APCPA*, *APTPA*}. In AMiner, we assign weights {0.25, 0.5, 0.25} to {*APA*, *APCPA*, *APTPA*}. We report the results in terms of Macro-F1 and Micro-F1 in Table 5.1.

As we can observe, the StHNE outperforms all baselines on three datasets. It improves classification performance by about 8.7% in terms of Macro-F1 averagely with 80% training ratio, which is due to the weighted integration of meta-paths and the preservation of network structure. Both our model StHNE, ESim and metapath2vec fuse multiple meta-paths with weights, but the performances of ESim and metapath2vec are slight worse on three datasets. This may be caused by the separation of meta-paths fusion and model optimization, which lose some information between multiple relationships for heterogeneous graph embedding. We also notice that StHNE-1st and StHNE-2nd both outperform LINE-1st and LINE-2nd in most cases, which shows the superiority of the meta-path based first- and second-order proximity in heterogeneous graphs. From a vertical comparison, our StHNE continues to perform best against different sizes of training data, which implies the stability and robustness of our model.

Relationship Prediction. For DBLP and AMiner, we are interested in the co-author relationships (*APA*). Hence, we generate training networks by randomly hiding 20%

Table 5.1 Performance evaluation of node classification on static heterogeneous graphs. (Tr.Ratio means the training ratio.)

Datasets	Metric	Tr.Ratio	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE-1st	StHNE-2nd	StHNE
Yelp	Macro-F1	40%	0.6021	0.5389	0.5438	0.6387	0.5872	0.6193	0.5377	**0.6421**
		60%	0.5954	0.5865	0.5558	0.6464	0.6081	0.6639	0.5691	**0.6644**
		80%	0.6101	0.6012	0.6068	0.6793	0.6374	0.6909	0.5783	**0.6922**
	Micro-F1	40%	0.6520	0.6054	0.6105	0.6896	0.6427	0.6838	0.6118	**0.6902**
		60%	0.6472	0.6510	0.6233	0.7011	0.6681	0.7103	0.6309	**0.7017**
		80%	0.6673	0.6615	0.6367	0.7186	0.6875	0.7232	0.6367	**0.7326**
DBLP	Macro-F1	40%	0.9295	0.9271	0.9172	0.9354	0.9213	0.9392	0.9283	**0.9473**
		60%	0.9355	0.9298	0.9252	0.9362	0.9311	0.9436	0.9374	**0.9503**
		80%	0.9368	0.9273	0.9301	0.9451	0.9432	0.9511	0.9443	**0.9611**
	Micro-F1	40%	0.9331	0.9310	0.9219	0.9394	0.9228	0.9421	0.9312	**0.9503**
		60%	0.9383	0.9328	0.9291	0.9406	0.9305	0.9487	0.9389	**0.9519**
		80%	0.9392	0.9323	0.9347	0.9502	0.9484	0.9543	0.9496	**0.9643**
AMiner	Macro-F1	40%	0.8838	0.8929	0.8972	0.9449	**0.9487**	0.9389	0.9309	0.9452
		60%	0.8846	0.8909	0.8967	0.9482	0.9490	0.9401	0.9354	**0.9499**
		80%	0.8853	0.8947	0.8962	0.9491	0.9493	0.9412	0.9381	**0.9521**
	Micro-F1	40%	0.8879	0.8925	0.8958	0.9465	**0.9469**	0.9407	0.9412	0.9467
		60%	0.8881	0.8936	0.8960	0.9482	0.9497	0.9423	0.9431	**0.9509**
		80%	0.8882	0.8960	0.8962	0.9500	0.9511	0.9448	0.9423	**0.9529**

Table 5.2 Performance evaluation of relationship prediction on static heterogeneous graphs.

Datasets	Metric	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE-1st	StHNE-2nd	StHNE
Yelp	AUC	0.7404	0.6553	0.7896	0.6651	0.8187	0.8046	0.8233	**0.8364**
	F1	0.6864	0.6269	0.7370	0.6361	0.7355	0.7348	0.7397	**0.7512**
	ACC	0.6819	0.6115	0.7326	0.6386	0.7436	0.7286	0.7526	**0.7661**
DBLP	AUC	0.9235	0.8368	0.7672	0.9074	0.9291	0.9002	0.9246	**0.9385**
	F1	0.8424	0.7680	0.7054	0.8321	0.8645	0.8359	0.8631	**0.8850**
	ACC	0.8531	0.7680	0.6805	0.8416	0.8596	0.8266	0.8577	**0.8751**
AMiner	AUC	0.7366	0.5163	0.5835	0.8691	0.8783	0.8935	**0.9180**	0.8939
	F1	0.5209	0.5012	0.5276	0.6636	0.6697	0.7037	**0.8021**	0.7085
	ACC	0.6686	0.6475	0.6344	0.7425	0.7506	0.7622	**0.8251**	0.7701

AP in DBLP and 40% *AP* in AMiner as AMiner is much larger. For Yelp, we want to find two businesses that one person has reviewed (*BRURB*), which can be used to recommend businesses for users. Thus, we randomly hide 20% *BR* to generate the training network. We set the weights of *BSB* and *BRURB* in Yelp to 0.4 and 0.6. In DBLP, we assign weights {0.9, 0.05, 0.05} to {*APA*, *APCPA*, *APTPA*}. In AMiner, we assign weights {0.4, 0.3, 0.3} to {*APA*, *APCPA*, *APTPA*}. We evaluate the prediction performance on testing networks with AUC and Accuracy.

Table 5.2 shows the comparison results of different methods. Overall, we can see that StHNE achieves better relation prediction performance than other methods on two metrics. The improvement indicates the effectiveness of our model to preserve structural information in heterogeneous graphs. Benefiting from the second-order proximity preserved based on meta-path, StHNE-2nd outperforms than StHNE-1st

significantly. The reason is that the higher order proximity is more conducive for preserving complex relationships in heterogeneous graphs.

5.2.3.3 Effectiveness of DyHNE

In this section, our goal is to verify the effectiveness of DyHNE compared with these baselines designed for dynamic networks (i.e., DANE and DHPE). Since some baselines (e.g., DeepWalk, LINE and StHNE) cannot handle dynamic networks and we have reported the performance of these methods in Section 5.2, here we only apply these methods to initial networks as in [22, 50]. Specifically, given a dynamic network with 10 time steps $\{\mathcal{G}^1, \cdots, \mathcal{G}^{10}\}$, for the static network embedding methods, including StHNE, we only conduct them on $\mathcal{G}^1$ and report the results, while for the dynamic network embedding methods, i.e., DANE, DHPE and DyHNE, we conduct them from $\mathcal{G}^1$ to $\mathcal{G}^{10}$ to update the embedding incrementally, and report the final results to evaluate their performance in a dynamic environment.

Node Classification. For each dataset, we generate the initial and growing heterogeneous graph from the original network. Each growing heterogeneous graph contains ten time steps. In Yelp, reviews are time-stamped, we randomly add 0.1% new *UR* and *BR* to the initial network at each time step. For DBLP, we randomly add 0.1% new *PA*, *PC* and *PT* to the initial network at each time step. Since AMiner itself contains the published year of each paper, we divide the edges appearing in 2005 into 10 time steps uniformly.

We vary the size of the training set from 40% to 80% with the step size of 20% and the remaining nodes as testing. We repeat each classification experiment for ten times and report the average performance in terms of both Macro-F1 and Micro-F1 scores, as shown in Table 5.3. We can see that DyHNE consistently performs better than other baselines on all datasets with all varying sizes of training data, which demonstrates the effectiveness and robustness of our learned node embeddings when served as features for node classification. (a) Especially, our DyHNE significantly outperforms the two dynamic homogeneous network embedding methods, DANE and DHPE. The reason is that our model considers the different types of nodes and relations and can capture the structure and semantic information in heterogeneous graphs. (b) We also notice that our DyHNE achieves better performance than DHNE which is also designed for dynamic heterogeneous graphs. We believe that the improvement is due to the preserved meta-path based first-order and second-order proximity in node embeddings learned by our DyHNE. (c) Compared with the baselines designed for static heterogeneous graphs (i.e., DeepWalk, LINE, ESim and metapath2vec), our method also achieves the best performance, which proves the effectiveness of the update algorithm without losing important structure and semantic information in heterogeneous graphs.

Relationship Prediction. For each dataset, we generate the initial, growing and testing heterogeneous graph from the original heterogeneous graph. For Yelp, we first build the testing network containing 20% *BR*. The remaining constitutes the

Table 5.3 Performance evaluation of node classification on dynamic heterogeneous graphs. (Tr.Ratio means the training ratio.)

Datasets	Metric	Tr.Ratio	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE	DANE	DHPE	DHNE	DyHNE
Yelp	Macro-F1	40%	0.5840	0.5623	0.5248	0.6463	0.5765	0.6118	0.6102	0.5412	0.6293	**0.6459**
		60%	0.5962	0.5863	0.5392	0.6642	0.6192	0.6644	0.6342	0.5546	0.6342	**0.6641**
		80%	0.6044	0.6001	0.6030	0.6744	0.6285	0.6882	0.6471	0.5616	0.6529	**0.6893**
	Micro-F1	40%	0.6443	0.6214	0.5901	0.6932	0.6457	0.6826	0.6894	0.5823	0.6689	**0.6933**
		60%	0.6558	0.6338	0.5435	0.6941	0.6656	0.7074	0.6921	0.5981	0.6794	**0.6998**
		80%	0.6634	0.6424	0.6297	0.7104	0.6722	0.7281	0.6959	0.6034	0.6931	**0.7298**
DBLP	Macro-F1	40%	0.9269	0.9266	0.9147	0.9372	0.9162	0.9395	0.8862	0.8893	0.9302	**0.9434**
		60%	0.9297	0.9283	0.9141	0.9369	0.9253	0.9461	0.8956	0.8946	0.9351	**0.9476**
		80%	0.9322	0.9291	0.9217	0.9376	0.9302	0.9502	0.9051	0.9087	0.9423	**0.9581**
	Micro-F1	40%	0.9375	0.9310	0.9198	0.9383	0.9254	0.9438	0.8883	0.8847	0.9352	**0.9467**
		60%	0.9346	0.9245	0.9192	0.9404	0.9281	0.9496	0.8879	0.8931	0.9404	**0.9505**
		80%	0.9371	0.9297	0.9261	0.9415	0.9354	0.9543	0.9071	0.9041	0.9489	**0.9617**
AMiner	Macro-F1	40%	0.8197	0.8219	0.8282	0.8797	0.8673	0.8628	0.7642	0.7694	0.8903	**0.9014**
		60%	0.8221	0.8218	0.8323	0.8807	0.8734	0.8651	0.7704	0.7735	0.9011	**0.9131**
		80%	0.8235	0.8238	0.8351	0.8821	0.8754	0.8778	0.7793	0.7851	0.9183	**0.9212**
	Micro-F1	40%	0.8157	0.8189	0.8323	0.8729	0.8652	0.8563	0.7698	0.7633	0.8992	**0.9117**
		60%	0.8175	0.8182	0.8361	0.8734	0.8693	0.8574	0.7723	0.7698	0.9045	**0.9178**
		80%	0.8191	0.8201	0.8298	0.8751	0.8725	0.8728	0.7857	0.7704	0.9132	**0.9203**

Table 5.4 Performance evaluation of relationship prediction on dynamic heterogeneous graphs.

Datasets	Metric	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE	DANE	DHPE	DHNE	DyHNE
Yelp	AUC	0.7316	0.6549	0.7895	0.6521	0.8164	0.8341	0.7928	0.7629	0.8023	**0.8346**
	F1	0.6771	0.6125	0.7350	0.6168	0.7293	0.7506	0.7221	0.6809	0.7194	**0.7504**
	ACC	0.6751	0.6059	0.7300	0.6185	0.7395	0.7616	0.7211	0.7023	0.7024	**0.7639**
DBLP	AUC	0.9125	0.8261	0.7432	0.9053	0.9196	0.9216	0.5413	0.6411	0.8945	**0.9278**
	F1	0.8421	0.7840	0.7014	0.8215	0.8497	0.8621	0.7141	0.6223	0.8348	**0.8744**
	ACC	0.8221	0.7227	0.6754	0.8306	0.8405	0.8436	0.5511	0.5734	0.8195	**0.8635**
AMiner	AUC	0.8660	0.6271	0.5648	0.8459	0.8694	0.8659	0.8405	0.8412	0.8289	**0.8823**
	F1	0.7658	0.5651	0.6071	0.7172	0.7761	0.7567	0.7167	0.7158	0.7386	**0.7792**
	ACC	0.7856	0.5328	0.5828	0.7594	0.7793	0.7733	0.7527	0.7545	0.7498	**0.7889**

initial and growing network, where the growing network is divided into 10 time steps, and 0.1% new *UR* and *BR* are added to the initial network at each time step. For DBLP, we use the similar approach as described above. For AMiner, we take the data involved in 1990-2003 as the initial network, 2004 as the growing network and 2005 as the testing network.

We report the prediction performance in Table 5.4 and have some findings: (a) Our method consistently improves the relationships prediction accuracy on the three datasets, which is attributed to the structural information preserved by the meta-path based first-order and second-order proximity. (b) DANE and DHPE obtain poor performance due to the neglect of multiple types of nodes and relations in heterogeneous graphs. (c) Compared to DHNE, our DyHNE consistently performances better on three datasets, which is benefit from the effectiveness of update algorithm. Additionally, the meta-path based second-order proximity ensures that our DyHNE captures the high order structures of heterogeneous graph, which is also preserved with the updated node embeddings.

The more detailed method description and experiment validation can be seen in [45]. While DyHNE mainly focuses on incremental learning of dynamic semantics, the sequential interactions of entities indeed express the evolution of heterogeneous interests, and we introduce the following model SHCF to extract such valuable information for recommender systems.

5.3 Sequence information

5.3.1 Overview

Heterogeneous graph modeling, as an effective information fusion method containing different types of nodes and links, can be used to integrate multiple types of objects and their complex interactions in the recommendation system that may produce more accurate recommendation results [35]. These methods mainly model a user's static preference [34], while ignoring the interactions' sequential pattern that a small set of the most recent interactions can better reflect a user's dynamic interests over time. Considering the sequential pattern to model a user's latest interests, there is another line of work called sequential recommendation. The sequential recommendation system is to predict which item a user most probably would like to interact with next time given her sequential interaction data as context. However, almost all of the sequential recommendation methods [13, 23, 14, 31] model user embeddings based on only their own sequential item interactions while ignoring the heterogeneous information widely existing in recommendation system, such like item attributes. When the data is sparse and there are few user interaction behaviors, these methods also suffer from cold-start problem.

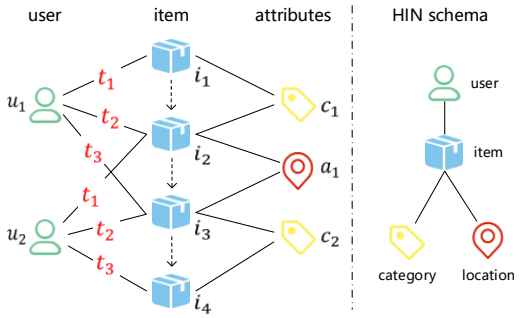


Fig. 5.2 A toy example of user-item interactions with heterogeneous information.

In this section, we construct a heterogeneous graph with user-item interactions and item attributes in Fig. 5.2 and propose the SHCF to fully consider both the sequential patterns and the high-order heterogeneous collaborative signals. For user

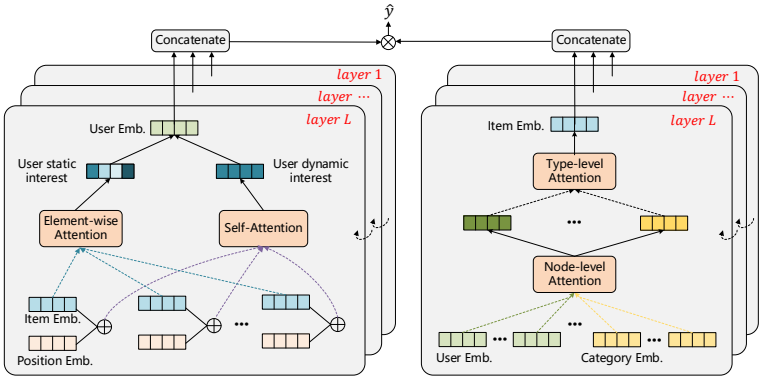


Fig. 5.3 The framework of SHCF.

embedding, we aggregate the static representation with a novel element-wise attention mechanism, and the dynamic interests by aggregating her interacted item sequence with a sequence-aware self-attention mechanism. For item embedding, we aggregate the heterogeneous information of its neighboring nodes including users and item attributes with dual-level attention. We can not only learn the importance of different nodes but also pay attention to important types of nodes. By stacking multiple message passing layers, we can enforce the embeddings to capture the high-order collaborative relationships.

5.3.2 The SHCF Model

Fig. 5.3 shows the framework of our SHCF, which contains three major steps. First, we construct a heterogeneous graph with user-item interactions and item attributes as shown in Fig. 5.2. Notice that here we only consider item attributes to focus on clearly illustrating how to handle sequence patterns and heterogeneous information. In fact, user attributes and other heterogeneous information can be easily added into our SHCF through concatenating embedding learned from these heterogeneous attributes as item embedding does. Then we apply an embedding layer to initialize the representations of users, items, and item attributes (e.g., item categories). Second, we design multiple message passing layers over the heterogeneous graph to learn the user and item embeddings. For user embedding, we capture a user's fine-grained static interests on different aspects of an item with an element-wise attention mechanism. We also consider a user's dynamic interests by aggregating her interacted item sequence with a sequence-aware self-attention mechanism. For item embedding, we aggregate the heterogeneous information of its neighborhoods including users and item attributes with dual-level attention which considers the importance of different neighboring nodes with different types. Finally, the prediction layer aggregates the

learned embeddings from different message passing layers for both user and item representations, and outputs the prediction of target user-item pairs.

5.3.2.1 Embedding Layer

The users, items and item attributes in real datasets are usually identified by some unique IDs, whereas these original IDs have a very limited representation capacity. Therefore, we create a user embedding matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{U}| \times d}$, where d is dimensionality of the latent embedding spaces. The j^{th} row of the embedding matrix $\mathbf{U}$ encodes the user u_j to the real-valued embedding $\mathbf{u}_j$, which is more informative. In the same way, we respectively create an item embedding matrix $\mathbf{I} \in \mathbb{R}^{|\mathcal{I}| \times d}$ and item attribute embedding matrices, e.g. item category embedding matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{C}| \times d}$.

Positional Embedding. Motivated by the recent works of transformer [41, 7], as for the interacted item sequence $S_u = \{i_1, \cdots, i_{t-1}, i_t\}$ sorted by the time t, we correlate each item with a learnable position embedding $\mathbf{p} \in \mathbb{R}^d$ to capture the sequential pattern of the items.

$$\hat{\mathbf{I}}_u = \begin{bmatrix} \mathbf{i}_t + \mathbf{p}_1 \\ \mathbf{i}_{t-1} + \mathbf{p}_2 \\ \cdots \\ \mathbf{i}_1 + \mathbf{p}_t \end{bmatrix}. \tag{5.35}$$

Notice that we add the position embedding in a reverse order to capture the relevant distance to the target item.

5.3.2.2 Sequence-Aware Heterogeneous Message Passing

To capture high-order heterogeneous collaborative information and the sequential information, we first construct a heterogeneous graph enriching user-item interactions with added item attributes as shown in Fig. 5.2. We also model a user's interacted item sequence with a sequence-aware attention mechanism, in order to capture the user's dynamic interests. In this way, we can not only better model user preferences but also alleviate the sparsity of the interactions. In the following, we will first present a single graph convolution layer to model item embedding and user embedding with considering heterogeneous information and sequential information, and then generalize it to multiple layers.

Item Modeling with Heterogeneous Information. To alleviate the sparsity problem, we add item attribute information to the user-item bipartite graph and create a heterogeneous graph that includes different types of nodes. Inspired by HGAT [15], in this part, we present our proposed message passing layers which consider the heterogeneous information. Taking the item node as an example, it has different types of neighboring nodes such as users, categories and cities. On one hand, different types of neighboring nodes may have different impacts on it. For example, the category of one item may be more informative than the user who interacted with it. On the

other hand, different neighboring nodes of the same type could also have different importance. For example, different users may have different preference on one item. To capture both the different importance at both node level and type level, we design a dual-level attention mechanism when aggregating embeddings from neighboring nodes.

(a) Node-Level Attention. We design the node-level attention to learn the different importance of neighboring nodes with the same type and aggregate the representation of these neighbors to form a specific type embedding. Formally, given a specific item v and its neighboring node $v' \in \mathcal{N}_v^{\tau'}$ with type τ', the weight coefficient $\alpha_{vv'}$ of the specific node pair (v, v') can be formulated as follows:

$$\alpha_{vv'} = \frac{\exp(\sigma(\mathbf{a}_{\tau'}^{\top} \cdot [\mathbf{i}_v || \mathbf{h}_{v'}]))}{\sum_{k \in \mathcal{N}_v^{\tau'}} \exp(\sigma(\mathbf{a}_{\tau'}^{\top} \cdot [\mathbf{i}_v || \mathbf{h}_k]))}, \tag{5.36}$$

where $\sigma(\cdot)$ is the activation function, such like LeakReLU, $\mathbf{a}_{\tau'}$ is the attention vector for the type τ', $\mathbf{h}$ is the embedding of neighboring node and $||$ denotes the concatenate operation.

Then, for the item v, we can get the specific type embedding $\mathbf{g}_v^{\tau'}$ by aggregating neighboring nodes of the same type with corresponding coefficients as follows:

$$\mathbf{g}_v^{\tau'} = \sigma(\sum_{v' \in \mathcal{N}_v^{\tau'}} \alpha_{vv'} \cdot \mathbf{h}_{v'}). \tag{5.37}$$

(b) Type-Level Attention. For any type τ' belonging to the item v's neighboring node type set $\mathcal{T}$, we can get a type specific embedding $\mathbf{h}_v^{\tau'}$ following the Eq. (5.37). To capture different importance of different node types, we design a type-level attention defined as follows:

$$m_v^{\tau'} = V \cdot \tanh(\mathbf{w} \cdot \mathbf{g}_v^{\tau'} + b), \tag{5.38}$$

$$\beta_v^{\tau'} = \frac{\exp(m_v^{\tau'})}{\sum_{\tau \in \mathcal{T}} \exp(m_v^{\tau})}. \tag{5.39}$$

With the learned weights as coefficients, we can fuse these type embeddings $\mathbf{g}_v^{\tau'}$ to obtain the final item embedding $\tilde{\mathbf{i}}_v$ as follows:

$$\tilde{\mathbf{i}}_v = \sigma(\sum_{\tau' \in \mathcal{T}} \beta_v^{\tau'} \cdot \mathbf{g}_v^{\tau'}). \tag{5.40}$$

Notice that the above is an example of how to obtain item embedding with heterogeneous information, other typed nodes in heterogeneous graph such as attribute nodes can be modeled in the same way.

User Modeling with Fine-grained Static and Dynamic Interest. A great challenge for recommendation is how to accurately model user preferences. For traditional collaborative filtering or heterogeneous graph based recommendation methods, on one hand, they usually view an item as an entirety, which ignore the fact that

users may have different preferences on different aspects of an item; on the other hand, they always neglect the sequential information of a user's interaction history, thus failing to capture the user's dynamic interests. Therefore, for user nodes, we present a carefully designed message passing layer to capture a user's fine-grained static interests and dynamic interests. More specifically, we propose an element-wise attention mechanism that assumes each dimension of the item embedding reflects a distinct aspect of the item. In addition, to capture a user's dynamic interests, we adopt a sequence-aware self-attention mechanism where each item embedding is correlated with a position embedding, and self-attention is applied to pay attention to important items.

(a) Element-Wise Attention. Here we present the details of element-wise attention to capture user's fine-grained static preference. For a specific item i_j in user u's interaction sequence $\mathcal{S}_u$, we can calculate a weight vector $\boldsymbol{\gamma}_j$ for different aspects of item i_j as follows:

$$\boldsymbol{\gamma}_j = \tanh(\mathbf{W}_u \cdot \mathbf{i}_j + b), \tag{5.41}$$

where $\mathbf{W}_u \in \mathbb{R}^{d \times d}$, $\boldsymbol{\gamma}_j$ is the attention coefficients of different aspects, and a large γ_j^k means that the k^{th} aspect of item embedding $\mathbf{i}_j$ is strongly relevant to the user's preference.

Then we aggregate with element-wise product between the weight coefficients $\boldsymbol{\gamma}_j$ and the user integrated item $\mathbf{i}_j$ to learn the embedding $\mathbf{u}_s$ reflects user's fine-grained static interests:

$$\mathbf{u}_s = \sum_{j \in \mathcal{S}_u} \boldsymbol{\gamma}_j \odot \mathbf{i}_j. \tag{5.42}$$

(b) Sequence-Aware Self-attention. Motivated by the self-attention mechanism widely used in NLP tasks such like machine translation[41, 7], we adopt a sequence-aware self-attention mechanism. Specifically, each item $\hat{\mathbf{I}}_u$ is integrated with its position embedding and self-attention is used to pay attention to critical items, which to learn the embedding $\mathbf{u}_d$ reflects user's dynamic interests over time:

$$\textsc{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}) \cdot \mathbf{V}, \tag{5.43}$$

$$\mathbf{u}_d = \Big\|_{h=1}^{H} \textsc{Attention}(\hat{\mathbf{I}}_u\mathbf{W}^Q, \hat{\mathbf{I}}_u\mathbf{W}^K, \hat{\mathbf{I}}_u\mathbf{W}^V), \tag{5.44}$$

Eq. (5.43) is the paradigm of self-attention, where $\textsc{Attention}()$ calculates a weighted sum of all values $\mathbf{V}$ by queries $\mathbf{Q}$ and keys $\mathbf{K}$, and the scale factor $\sqrt{d}$ is to avoid overly large values of the inner product result. In Eq. (5.44), $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$ is the projection matrices and we extend the self-attention to multi-head attention by repeating it for H times and concatenate the learned embeddings to get the final user dynamic interest representation.

After getting the static interest embedding $\mathbf{u}_s$ and dynamic interest embedding $\mathbf{u}_d$, we combine them with a balance weight to get the final user embedding $\tilde{\mathbf{u}}$:

$$\tilde{\mathbf{u}} = \lambda \mathbf{u}_d + (1 - \lambda)\mathbf{u}_s \tag{5.45}$$

(c) High-order Propagation. The above shows a single messaging passing layer with heterogeneous information and sequential information, which aggregates information from the first-order neighbors. To capture the high-order collaborative information, we can stack it to multiple layers in which each layer takes the last layer's output representation as its input [1]. After L-layer embedding propagation, we can get output embeddings of L different layers.

5.3.2.3 Optimization Objective

The embeddings of different layers may have different contributions in reflecting user preferences, following [44], we concatenate the representation of each layer to constitute the final embedding for both users and items:

$$\mathbf{u} = \tilde{\mathbf{u}}^1 || \tilde{\mathbf{u}}^2 || \cdots || \tilde{\mathbf{u}}^L, \quad \mathbf{i} = \tilde{\mathbf{i}}^1 || \tilde{\mathbf{i}}^2 || \cdots || \tilde{\mathbf{i}}^L. \tag{5.46}$$

Finally, we use the simple dot product to estimate the user's preference towards the target item:

$$\hat{y}(u, i) = \mathbf{u}^\top \mathbf{i}. \tag{5.47}$$

To optimize our model, we use the Bayesian Personalized Ranking (BPR) loss [32] as our loss function:

$$\mathcal{L} = \sum_{i \in S_u, j \notin S_u} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \eta ||\Theta||, \tag{5.48}$$

where $\sigma(\cdot)$ is the sigmid function, Θ denotes all the trainable parameters and η is the regularization coefficient, S_u is the interaction sequence of the user u, and for each positive sample (u, i), we sample a negative sample j for training.

5.3.3 Experiments

5.3.3.1 Experimental Settings

Datasets. To verify the effectiveness of our method, we conduct extensive experiments on three real-world datasets. Movielens is a widely used benchmark dataset for recommendation task.In our experiments, we adopt a small version of 100K interactions ML100K and a larger version of 1M interactions ML1M. Yelp is a local business recommendation dataset which records the user ratings on local businesses.

Baselines. We compare our method with three groups of recommendation baseline methods, namely collaborative filtering methods (BPR-MF [32], NeuMF [12],

[1] According to our experiments, fine-grained and dynamic user interest modeling layer is taken as the L-th layer for user modeling.

Table 5.5 Recommendation performance of different models. The best result in each row is bold and the second best result is underlined. The improvements of our method over the second best models are shown in the last column.

Dataset	Metrics	BPR-MF	NeuMF	NGCF	NeuACF	HeRec	NARM	SR-GNN	SHCF	Improve
ML100K	HR@5	0.4030	0.4057	0.4274	0.4337	0.4255	0.5228	0.5010	**0.5414**	3.56%
	NDCG@5	0.2747	0.2676	0.2889	0.2874	0.2798	0.3659	0.3510	**0.3859**	5.47%
	HR@10	0.5801	0.5689	0.5864	0.6034	0.6012	0.6723	0.6660	**0.7108**	5.73%
	NDCG@10	0.3312	0.3127	0.3402	0.3420	0.3325	0.4142	0.4048	**0.4401**	6.25%
	HR@15	0.6787	0.6706	0.6649	0.7084	0.6981	0.7529	0.7598	**0.7817**	2.88%
	NDCG@15	0.3573	0.3462	0.3611	0.3697	0.3521	0.4354	0.4298	**0.4592**	5.47%
	HR@20	0.7455	0.7595	0.7434	0.7720	0.7524	0.8038	0.8048	**0.8324**	3.43%
	NDCG@20	0.3731	0.3672	0.3796	0.3847	0.3721	0.4475	0.4396	**0.4693**	4.87%
ML1M	HR@5	0.4921	0.5092	0.5017	0.5050	0.4923	0.6713	0.6634	**0.6927**	3.18%
	NDCG@5	0.3376	0.3511	0.3437	0.3508	0.3455	0.5201	0.5233	**0.5299**	1.26%
	HR@10	0.6577	0.6803	0.6688	0.6684	0.6601	0.7603	0.7699	**0.7964**	3.19%
	NDCG@10	0.3910	0.4066	0.3977	0.4038	0.3982	0.5565	0.5580	**0.5639**	1.06%
	HR@15	0.7551	0.7761	0.7587	0.7593	0.7403	0.8230	0.8184	**0.8503**	3.32%
	NDCG@15	0.4168	0.4320	0.4216	0.4279	0.4194	0.5687	0.5709	**0.5785**	1.33%
	HR@20	0.8159	0.8369	0.8167	0.8232	0.8105	0.8602	0.8584	**0.8844**	2.81%
	NDCG@20	0.4311	0.4463	0.4353	0.4430	0.4328	0.5723	0.5803	**0.5968**	2.84%
Yelp	HR@5	0.3077	0.3571	0.4097	0.4094	0.3982	0.3490	0.3754	**0.4421**	7.91%
	NDCG@5	0.2086	0.2419	0.2855	0.2844	0.2765	0.2373	0.2565	**0.3100**	8.58%
	HR@10	0.4325	0.5018	0.5584	0.5553	0.5505	0.4900	0.5208	**0.5878**	5.27%
	NDCG@10	0.2488	0.2885	0.3335	0.3314	0.3311	0.2828	0.3035	**0.3572**	7.11%
	HR@15	0.5084	0.6006	0.6434	0.6504	0.6423	0.5851	0.6054	**0.6725**	3.40%
	NDCG@15	0.2689	0.3146	0.3544	0.3565	0.3499	0.3080	0.3259	**0.3796**	6.48%
	HR@20	0.5643	0.6717	0.7005	0.7138	0.6923	0.6496	0.6684	**0.7260**	1.71%
	NDCG@20	0.2821	0.3315	0.3686	0.3715	0.3603	0.3232	0.3408	**0.3922**	5.57%

NGCF [43]), heterogeneous graph recommendation methods (NeuACF [11], HeRec [34]) , and sequential recommendation methods (NARM [23], SR-GNN [13]).

Implementation Details. For all the methods, we apply a grid search for hyper parameters. For NGCF and SR-GNN, the layer of GNN is searched from 1 to 4. We implement our proposed model based on Tensorflow. The dimension of embeddings d is set as 64. For the self-attention network, the attention head number H is set as 8. The hyper parameter λ to balance the weights of a user's dynamic interests and static interests is set as 0.5 and 0.2 for MovieLens and Yelp, respectively. In addition, the learning rate is 0.0005 for MovieLens and 0.00005 for Yelp. The coefficient of L2 normalization η for all the datasets is set to 10^{-5}. We set the depth of our proposed SHCF L as 4. We randomly initialize the model parameters with Xavier initializer, then use the Adam as the optimizer. To avoid over-fitting, we apply early stopping strategy and apply dropout (dropout rate is 0.1) in every layer of our proposed SHCF.

5.3.3.2 Performance Comparison

We first compare the recommendation performance of all the methods. For a fair comparison, the embedding dimension of all the methods is set as 64. Table 5.5

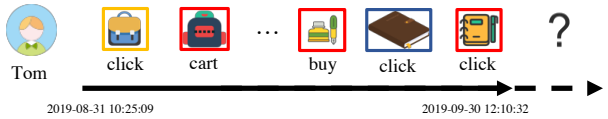
shows the experiment results of different methods. We have the following observations: 1) heterogeneous graph based recommendation methods generally perform better than traditional collaborative filtering methods. They especially have great improvements on the sparse dataset (i.e., Yelp), which illustrates that applying heterogeneous graph to incorporate side information for recommendation can alleviate the data sparsity problem and improve recommendation performance. 2) For the dense datasets ML100K and ML1M that users have adequate interaction behaviors (the average number of interactions per user is 103.9 and 165.3 respectively), sequential recommendation methods perform better than collaborative filtering methods and heterogeneous graph based recommendation methods. But for the sparse dataset Yelp where the average number of interactions per user decreases to 10.0, the performance of sequential recommendation methods significantly declines. It illustrates the limitation of sequential recommendation methods in modeling user embeddings based on only the user's own sequential item interactions without considering the collaborative information of similar users or items when the user does not have sufficient interaction records. 3) Our proposed model SHCF consistently outperforms all the baselines on all the datasets including the two dense datasets (i.e., ML100K and ML1M) and one sparse dataset (i.e., Yelp). These results verify the effectiveness of SHCF in modeling users and items in both sparse and dense datasets by fully considering the high-order heterogeneous collaborative information and sequential information.

The more detailed methods and experiments description can be obtained in [21]. While the SHCF focuses on dealing with single-typed sequences, there are multiple interactions in real-world systems indeed, and the time span of recorded interactions becomes larger and larger, indicating not only short-term but also long-term heterogeneous preferences. The following section is to study the temporal heterogeneous interaction modeling.

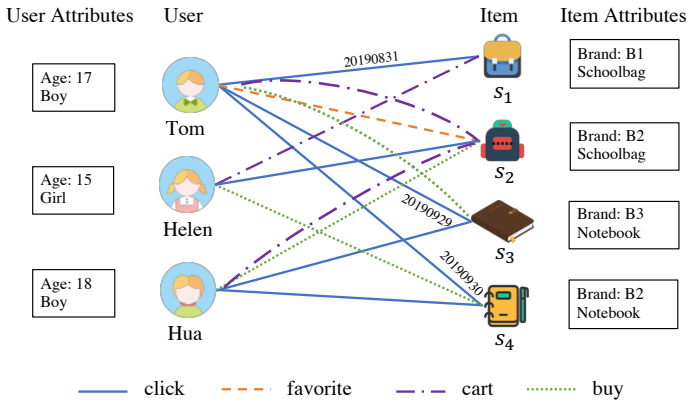
5.4 Temporal Interaction

5.4.1 Overview

By modeling historical user-item interactions, recommender systems play a fundamental role in e-commerce [25, 38]. Existing methods [49, 20] are mainly capable of modelling short-term preferences from relatively recent interactions, capturing long-term preferences (e.g., preferred brands) from historical habits is also an important element of temporal dynamics [38]. However, these methods usually model short- and long-term preferences independently, ignoring the role of habits in driving the current, evolving demands. Taking Fig. 5.4a as an example, when browsing similar items (e.g., two schoolbags), users prefer to click those with attributes they habitually care (e.g., the brands). This presents the first research challenge: How to effectively model the complex temporal dynamics, coupling both historical habits and evolv-



(a) The temporal evolution of interactions.



(b) An example of THIG.

Fig. 5.4 Toy example of next-item recommendation, from (a) a temporal sequence of interactions, to (b) a Temporal Heterogeneous Interaction Graph (THIG).

ing demands? Another dimension overlooked by existing sequential models is the abundant heterogeneous structural information, as shown in Fig. 5.4b. This leads to the second research challenge: How to make full use of the temporal heterogeneous interactions to model the preferences of different types?

In this section, we propose THIGE to effectively learn user and item embeddings on temporal heterogeneous interaction graphs for next-item recommendation. THIGE first encodes heterogeneous interactions with temporal information. Building upon the temporal encoding, we take into account the influence of long-term habits on short-term demands, and design a habit-guided attention mechanism to couple short- and long-term preferences. To fully exploit the rich heterogeneous interactions to enhance multifaceted preferences, we further capture the latent relevance of varying types of interaction via heterogeneous self-attention mechanisms.

5.4.2 The THIGE Model

The overall framework is shown in Fig. 5.5. Specifically, we divide the historical interactions of a user into long and short term based on their timestamps. For short-term preferences, we model users' sequences of recent interactions with gated

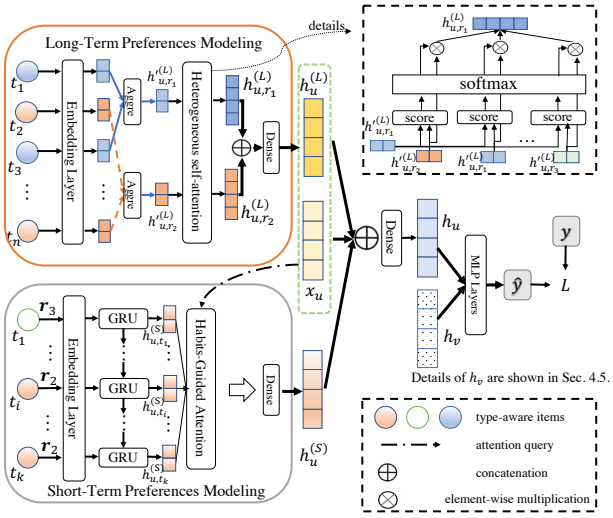


Fig. 5.5 Overall framework of user embedding in THIGE for next-item recommendation.

recurrent units (GRU), to embed users' current demands $\boldsymbol{h}_u^{(S)}$. For long-term preference, we model users' long-term interactions with a heterogeneous self-attention mechanism, to embed users' historical habits $\boldsymbol{h}_u^{(L)}$. Different from the decoupled combination (e.g., simple concatenation) of long- and short-term embeddings in previous methods [20, 48], we propose to exploit the long-term historical habits to guide the learning of short-term demands using the habit-guided attention, which effectively captures the impact of habits on recent behaviors. Notice that Fig. 5.5 only shows the learning of user representations. For items, we do not distinguish their long- and short-term interactions, and only adopt a long-term model similar to that of users. The reason lies in the fact that there may be numerous users interacting with an item around a short period of time, and these users have no significant short-term sequential dependency.

5.4.2.1 Embedding Layer with Temporal Information

Each interacted item of a user is associated with not only attributes but also a timestamp. As shown in Fig. 5.5, the timestamps is in the form of $[t_1, t_2, \cdots, t_n]$.

Thus, the temporal embedding of an item v consists of both a static and a temporal component. The static component $\boldsymbol{x}_v = \boldsymbol{W}_{\phi(v)}\boldsymbol{a}_v$, where the input vector $\boldsymbol{a}_v \in \mathbb{R}^{d_\phi(v)}$ encodes the attributes of v, $\boldsymbol{W}_{\phi(v)} \in \mathbb{R}^{d \times d_\phi(v)}$ denotes the latent projection, $d_\phi(v)$ and d are the dimension of attributes and latent representation of v. Moreover, at time t, denoting Δt as the time span before the current time T and dividing the overall time span into B buckets, the temporal component of v is defined as $\boldsymbol{W}\xi(\Delta t)$, where $\xi(\Delta t) \in \mathbb{R}^B$ denotes the one-hot bucket representation of Δt, and $\boldsymbol{W} \in \mathbb{R}^{d_\mathcal{T} \times B}$

denotes the projection matrix and $d_{\mathcal{T}}$ is the output dimension. Thus, the temporal embedding of an item v at time t is

$$\boldsymbol{x}_{v,t} = [\boldsymbol{W}\xi(\Delta t) \oplus \boldsymbol{x}_v], \tag{5.49}$$

where $\oplus$ denotes concatenation. Similarly, we generate the static representation of a user u as $\boldsymbol{x}_u \in \mathbb{R}^d$, and temporal representation of u at time t as $\boldsymbol{x}_{u,t}$.

To further consider the sequential evolution of heterogeneous interactions, we generate the i^{th} interacted item embedding $\boldsymbol{x}_{v_i,t_i,r_i} = [\boldsymbol{x}_{v_i,t_i} \oplus \boldsymbol{r}_i]$ as the combination of the temporal embedding $\boldsymbol{x}_{v_i,t_i}$ and the corresponding type embedding $\boldsymbol{r}_i = \boldsymbol{W}_{\mathcal{R}}\boldsymbol{I}(r_i)$ where $\boldsymbol{I}(r_i)$ denotes the one-hot vector of r_i with dimension $|\mathcal{R}|$, $\boldsymbol{W}_{\mathcal{R}} \in \mathbb{R}^{d_{\mathcal{R}} \times |\mathcal{R}|}$ is the projection matrix and $d_{\mathcal{R}}$ is the latent dimension. For long-term preference modeling, we input the temporal embedding into type-aware aggregators to distinguish preferences of different types.

5.4.2.2 Short-Term Preference Modeling

Recent interactions of users usually indicate the evolving current demands. For instance, as shown in Fig. 5.4a, Tom's current demand has been evolved from bags to notebooks. In order to model the short-term and evolving preferences, we adopt gated recurrent units (GRU) [5], which can capture the dependency of recent interactions. Given a user u here, let his/her k recent interactions be $\{(v_i, t_i, r_i) \mid 1 \leq i \leq k\}$, where t_k is the most recent timestamp before the current time T. Subsequently, we encode the user preference at time t_i as $\boldsymbol{h}^{(S)}_{u,t_i}$, using a GRU based on the embedding of interaction (v_i, t_i, r_i) (i.e., $\boldsymbol{x}_{v_i,t_i,r_i}$) and his/her preference at t_{i-1}, as follows.

$$\boldsymbol{h}^{(S)}_{u,t_i} = \text{GRU}(\boldsymbol{x}_{v_i,t_i,r_i}, \boldsymbol{h}^{(S)}_{u,t_{i-1}}), \quad \forall 1 < i \leq k, \tag{5.50}$$

where $\boldsymbol{h}^{(S)}_{u,t_i} \in \mathbb{R}^d$. The time-dependent user embeddings $\{\boldsymbol{h}^{(S)}_{u,t_i} \mid 1 \leq i \leq k\}$ can be further aggregated to encode the current demand of user u.

However, the current and evolving demands of user are not only influenced by their recent transactions. Their long-term preferences, i.e., historical habits such as brands and lifestyle inclinations, often play a subtle but important role. Thus, we enhance the encoding of short-term preferences under the guidance of historical habits, in order to discover more fine-grained and personalized preferences. Specially, we propose a habit-guided attention mechanism to aggregate short-term user preferences, as follows.

$$\boldsymbol{h}^{(S)}_u = \sigma\left(W^{(S)} \cdot \sum_i a_{u,i}\boldsymbol{h}^{(S)}_{u,t_i} + b_s\right), \quad \forall 1 \leq i \leq k, \tag{5.51}$$

where $\boldsymbol{h}^{(S)}_u \in \mathbb{R}^d$ denotes the overall short-term preference of u, $W^{(S)} \in \mathbb{R}^{d \times d}$ denotes the projection matrix, σ is the activation function and we adopt RELU here to ensure the non-linearity , b_s is the bias, and $a_{u,i}$ is the habit-guided weight:

$$a_{u,i} = \frac{\exp\left([\boldsymbol{h}_u^{(L)} \oplus \boldsymbol{x}_u]^T \boldsymbol{W}_a \boldsymbol{h}_{u,t_i}^{(S)}\right)}{\sum_{j=1}^{k} \exp\left([\boldsymbol{h}_u^{(L)} \oplus \boldsymbol{x}_u]^T \boldsymbol{W}_a \boldsymbol{h}_{u,t_j}^{(S)}\right)}, \tag{5.52}$$

where $\boldsymbol{h}_u^{(L)} \in \mathbb{R}^d$ is the long-term preference of u which would have encoded the habits of u, and $\boldsymbol{W}_a \in \mathbb{R}^{2d \times d}$ is a mapping to quantify the fine-grained relevance between the short-term preference of u and the long-term habits of u at different times. Therefore, how to encode the long-term habits $\boldsymbol{h}_u^{(L)}$ in the context of heterogeneous interactions is the second key thesis of this work, as we will introduce next.

5.4.2.3 Long-Term Preference Modeling

Besides short-term preferences to encode current and evolving demands, users also exhibit long-term preferences to express personal and historical habits. In particular, there exist multiple types of heterogeneous interactions which have different relevance w.r.t. each other. For example, a “click" is more relevant to a “cart" or “buy" on the same item or similar items; “favorite" could be less relevant to “cart" or “buy", but is closely tied to the user’s brand or lifestyle choices in the long run. Thus, different types of interactions entail both latent relevance and multifaceted preferences. Thereby, our goal is to fully encode the latent, fine-grained relevance of multifaceted long-term preferences.

Consider a user u, and his/her long-term interactions $\{(v_i, t_i, r_i) \mid 1 \le i \le n\}$ where $n \gg k$ (k is the count of recent interactions in short-term modeling). To differentiate the explicit interaction types, we first aggregate the embeddings of items which the user have interacted with under a specific type r:

$$\boldsymbol{h}_{u,r}^{\prime(L)} = \sigma\left(\boldsymbol{W}_r \cdot \text{aggre}(\{\boldsymbol{x}_{v_i,t_i} \mid 1 \le i \le n, r_i = r\})\right), \tag{5.53}$$

where $\boldsymbol{h}_{u,r}^{\prime(L)} \in \mathbb{R}^d$ is the type-r long-term preferences of user u, $\boldsymbol{W}_r \in \mathbb{R}^{d \times (d_\mathcal{T} + d)}$ is the type-r learnable mapping, aggre$(\cdot)$ is an aggregator, and we utilize mean-pooling here.

While we can simply sum or concatenate the type-specific long-term preferences into an overall representation, there exists latent relevance among the types (e.g., “click" and “buy") and latent multifaceted preferences (e.g., brands and lifestyles). In this section, we design a heterogeneous self-attention mechanism to express the latent relevance of different-typed interactions and long-term multifaceted preferences. By concatenating all long-term preferences of different types as $\boldsymbol{H}_u^{(L)} = \oplus_{r \in \mathcal{R}} \boldsymbol{h}_{u,r}^{\prime(L)}$ with size d-by-$|\mathcal{R}|$, we first formulate the self-attention to capture the latent relevant of heterogeneous types in $\mathcal{R}$ w.r.t. each other:

$$\boldsymbol{h}_{u,r}^{(L)} = \sum_{r' \in \mathcal{R}} \left(\frac{\exp\left(\boldsymbol{Q}_{u,r}^T \boldsymbol{K}_{u,r'} / \sqrt{d_a}\right)}{\sum_{r'' \in \mathcal{R}} \exp\left(\boldsymbol{Q}_{u,r}^T \boldsymbol{K}_{u,r''} / \sqrt{d_a}\right)} \boldsymbol{V}_{u,r'} \right), \tag{5.54}$$

where $\boldsymbol{Q}_u = \boldsymbol{W}_Q \boldsymbol{H}_u^{(L)}, \boldsymbol{K}_u = \boldsymbol{W}_K \boldsymbol{H}_u^{(L)}, \boldsymbol{V}_u = \boldsymbol{W}_V \boldsymbol{H}_u^{(L)}, \boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d_a \times d}$ and $\boldsymbol{W}_V \in \mathbb{R}^{d \times d}$ are the projection matrices, and d_a is the dimension of keys and queries.

Next, to express multifaceted preferences, we adopt a multi-head approach to model latent fine-grained facets. Specifically, the original embeddings of preferences are split into multi-heads and we adopt the self-attention for each head. The type-r long-term preference is concatenated from the h heads:

$$\boldsymbol{h}_{u,r}^{(L)} = \oplus_{m=1}^{m=h} \boldsymbol{h}_{u,r,m}^{(L)}, \tag{5.55}$$

where $\boldsymbol{h}_{u,r,m}^{(L)}$ denotes the m^th head based preference and there are h heads. The overall long-term preference can also be derived by fusing different types in $\mathcal{R}$:

$$\boldsymbol{h}_u^{(L)} = \sigma\left(\boldsymbol{W}^{(L)}(\oplus_{r \in \mathcal{R}} \boldsymbol{h}_{u,r}^{(L)}) + b_l\right), \tag{5.56}$$

where $\boldsymbol{W}^{(L)} \in \mathbb{R}^{d \times |\mathcal{R}|d}$ and b_l are the projection parameters. By now, both short- and long-term preferences haven been modeled. Taking the inherent attributes of users into consideration, the final representation of user u is calculated by

$$\boldsymbol{h}_u = \sigma(\boldsymbol{W}_{\boldsymbol{u}}[\boldsymbol{x}_u \oplus \boldsymbol{h}_u^{(S)} \oplus \boldsymbol{h}_u^{(L)}] + b_u), \tag{5.57}$$

where $\boldsymbol{h}_u \in \mathbb{R}^d$ will be used for next-item prediction, and $\boldsymbol{W}_{\boldsymbol{u}} \in \mathbb{R}^{d \times 3d}$ and b_u are learnable parameters.

5.4.2.4 Preference Modeling of Items

The temporal interactions of an item is significantly different from those of a user. In practice, on a mass e-commerce platform, it is typical that many users interact with the same item around the same time constantly, without a meaningful sequential effect among different users. In other words, it is more reasonable to only model the general, long-term popularity of items. Thus, we model item representation $\boldsymbol{h}_v^{(L)}$ similar to the long-term preference modeling of users in Eq. (5.56) with heterogeneous multi-head self-attention, and then encode the item representation as follows:

$$\boldsymbol{h}_v = \sigma(\boldsymbol{W}_v[\boldsymbol{x}_v \oplus \boldsymbol{h}_v^{(L)}] + b_v), \tag{5.58}$$

where $\boldsymbol{h}_v \in \mathbb{R}^d$ is the final representation of item v for next-item prediction, and $\boldsymbol{W}_v$ and b_v are learnable parameters and $\boldsymbol{x}_v$ is the attribute vector of item v.

5.4.2.5 Optimization Objective

To deal with next-item recommendation, we predict $\hat{y}_{u,v}$ between user u and item v, indicating whether u will interact with v (under a given type) at the next time. Here we utilize a Multi-Layer Perception (MLP) [28]:

$$\hat{y}_{u,v} = \text{sigmoid}(\text{MLP}(\boldsymbol{h}_u \oplus \boldsymbol{h}_v)), \tag{5.59}$$

where $\boldsymbol{h}_u$ and $\boldsymbol{h}_v$ are the final representation of user u and item v, respectively. Model parameters can be optimized with the following cross-entropy loss:

$$L = -\sum_{\langle u,v \rangle} (1 - y_{u,v}) \log(1 - \hat{y}_{u,v}) + y_{u,v} \log(\hat{y}_{u,v}), \tag{5.60}$$

where $\langle u, v \rangle$ is a sample of user u and item v, and $y_{u,v} \in \{0, 1\}$ is the ground truth of the sample. We also optimize the L2 regularization of latent parameters to ensure the robustness.

5.4.3 Experiments

5.4.3.1 Experimental Settings

Datasets. We evaluate the empirical performance of THIGE for next-item recommendation on three real-world datasets including Yelp, CloudTheme and UserBehavior. For all datasets, we utilize the actual feedback of users as labels—among the candidates displayed to users.

Baselines. We compare THIGE with six representative models, including sequential models (**DIEN** [49], **STAMP** [25]), long-term interest learning models (**SHAN** [48], **M3R** [38]) and heterogeneous GNNs (**MEIRec** [9], **GATNE** [4]).

Implementation Details. For all baselines and our method, we set embedding size $d = 128$, $d_a = 128$, $d_\mathcal{T} = 16$, heads $h = 8$, the maximum iterations as 100, batch size as 128, learning rate as 0.001 and weight of regularization as 0.001 on all three datasets. The number of temporal buckets B is set as 60, 14, and 7 on the three datasets, respectively. For DIEN, MEIRec and our THIGE, we set three-layers MLP with dimensions 64, 32 and 1. For our THIGE and all baselines learning long- or short-term preferences, we consider the last 10, 10 and 50 interactions as the short term, and sample up to 50, 50 and 200 historical interactions as the long term for Yelp, CloudTheme and UserBehavior respectively. We evaluate the performance of next-item recommendation with the metrics of F1, PR-AUC and ROC-AUC.

5.4.3.2 Performance Comparison

We report the results of different methods for next-item recommendation in Table 5.6. In general, THIGE achieves the best performance on the three datasets, outperform the second best method by 4.04% on Yelp, 5.84% on CloudTheme and 0.51% on UserBehavior.

Table 5.6 Performance of next-item recommendation (with standard deviation). The best result is in bold while the second best is underlined. PR. denotes PR-AUC and ROC. denotes ROC-AUC.

Dataset	Metric	DIEN	STAMP	SHAN	M3R	MEIRec	GATNE	THIGE
Yelp	F1	39.52 (1.31)	40.37 (0.94)	40.17 (1.10)	33.49 (1.04)	42.86 (0.44)	42.21 (0.96)	**43.77** (0.66)
	PR.	30.04 (0.37)	31.36 (1.23)	32.35 (1.10)	26.40 (0.92)	32.69 (0.54)	33.39 (1.42)	**36.45** (1.66)
	ROC.	74.69 (0.57)	73.74 (1.15)	70.91 (1.14)	72.03 (1.33)	74.65 (0.23)	76.15 (0.64)	**79.23** (0.80)
CT.	F1	25.70 (1.25)	21.42 (0.91)	26.25 (1.09)	33.54 (1.67)	25.02 (0.98)	27.33 (0.50)	**37.17** (1.36)
	PR.	41.16 (0.22)	25.65 (0.44)	40.92 (1.09)	34.23 (0.95)	43.86 (0.42)	44.74 (0.20)	**51.94** (0.43)
	ROC.	68.41 (0.34)	52.97 (0.52)	67.48 (1.06)	62.92 (0.89)	69.98 (0.35)	71.22 (0.11)	**75.38** (0.33)
UB.	F1	67.32 (3.45)	63.06 (1.51)	58.84 (7.83)	61.37 (2.20)	66.48 (1.16)	**67.81** (1.14)	67.19 (0.98)
	PR.	63.38 (0.19)	59.09 (0.22)	63.86 (4.76)	57.68 (0.03)	64.94 (0.15)	65.42 (0.05)	**65.71** (0.09)
	ROC.	62.90 (0.23)	58.29 (0.40)	55.45 (3.98)	57.82 (0.09)	64.82 (0.16)	65.06 (0.08)	**65.39** (0.06)

Compared with sequential models (DIEN, STAMP, SHAN and M3R), the reason that THIGE is superior is twofold. First, THIGE designs a more effective way to integrate long- and short-term preferences, such that the current demands are explicitly guided by historical habits. Second, it also considers different types of interactions between users and items, leading to better performance.

Compared with GNN-based models (MEIRec and GATNE), the main improvement of THIGE comes from jointly modeling historical habits and evolving demands. Moreover, MEIRec models heterogeneous interactions in an entirely decoupled manner, whereas GATNE and THIGE achieves better performance by modeling their latent relevance. It is also not surprising that heterogeneous GNN-based methods typically outperform sequential models, as the former accounts for multi-typed interactions whereas the latter only models single-typed interactions.

The whole details and experimental analysis of THIGE are introduced in [17].

5.5 Conclusion

Focusing on the preserving dynamics on real-world heterogeneous graphs, this chapter introduces three representative models including DyHNE, SHCF and THIGE to model the incremental heterogeneous structures, the type-aware sequential evolution of entities as well as the multiple long-term habits and short-term demands. Specifically, this chapter designs a matrix perturbation based DyHNE to model the changes between different semantic-level snapshots and learns the dynamic embedding of heterogeneous nodes in Section 5.2. And then, as the dynamical interactions are in the form of type-aware sequences, we present the heterogeneous graph neural collaborative filtering model to make full use of high-order heterogeneous collaborative singles and sequential information for sequential recommendation in Section 5.3. Furthermore, with the consideration that heterogeneous dynamics express the multiple historical habits and current demands of entities, this chapter proposes a unified model which integrates both heterogeneous long-/short-term preferences to handle the problem of next-item recommendation in Section 5.4.

References

1. Aggarwal, C., Subbian, K.: Evolutionary network analysis: A survey. ACM Computing Surveys **47**(1), 10 (2014)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: NeurIPS, pp. 585–591 (2002)
3. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering **30**(9), 1616–1637 (2018)
4. Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., Tang, J.: Representation learning for attributed multiplex heterogeneous network. arXiv preprint arXiv:1905.01669 (2019)
5. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. IEEE Transactions on Knowledge and Data Engineering **31**(5), 833–852 (2018)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
8. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD, pp. 135–144 (2017)
9. Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., Li, Y.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: SIGKDD, pp. 2478–2486 (2019)
10. Golub, G.H., Van Loan, C.F.: Matrix computations. JHU Press (2012)
11. Han, X., Shi, C., Wang, S., Philip, S.Y., Song, L.: Aspect-level deep collaborative filtering via heterogeneous information networks. In: IJCAI, pp. 3393–3399 (2018)
12. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW, pp. 173–182 (2017)
13. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
14. Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D.: Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: RecSys, pp. 241–248 (2016)
15. Hu, L., Yang, T., Shi, C., Ji, H., Li, X.: Heterogeneous graph attention networks for semi-supervised short text classification. In: EMNLP, pp. 4823–4832 (2019)
16. Jacob, Y., Denoyer, L., Gallinari, P.: Learning latent representations of nodes for classifying in heterogeneous social networks. In: WSDM, pp. 373–382 (2014)
17. Ji, Y., Yin, M., Fang, Y., Yang, H., Wang, X., Jia, T., Shi, C.: Temporal heterogeneous interaction graph embedding for next-item recommendation. In: ECML-PKDD, pp. 314–329 (2020)
18. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: ICDM, pp. 197–206 (2018)
19. Kato, T.: Perturbation theory for linear operators, vol. 132. Springer Science & Business Media (2013)
20. LE, D.T., LAUW, H.W., Fang, Y.: Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In: IJCAI (2018)
21. Li, C., Hu, L., Shi, C., Song, G., Yuanfu, L.: Sequence-aware heterogeneous graph neural collaborative filtering. In: SDM (2021)
22. Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., Liu, H.: Attributed network embedding for learning in a dynamic environment. In: CIKM, pp. 387–396 (2017)
23. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: CIKM, pp. 1419–1428 (2017)
24. Li, X., Wu, Y., Ester, M., Kao, B., Wang, X., Zheng, Y.: Semi-supervised clustering in attributed heterogeneous information networks. In: WWW, pp. 1621–1629 (2017)
25. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: Stamp: short-term attention/memory priority model for session-based recommendation. In: SIGKDD, pp. 1831–1839 (2018)

26. Lu, Y., Wang, X., Shi, C., Yu, P.S., Ye, Y.: Temporal network embedding with micro-and macro-dynamics. In: CIKM, pp. 469–478 (2019)
27. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NeurIPS, pp. 849–856 (2002)
28. Pal, S.K., Mitra, S.: Multilayer perceptron, fuzzy sets, and classification. IEEE Transactions on Neural Networks **3**(5), 683–697 (1992)
29. Parlett, B.N.: The symmetric eigenvalue problem, vol. 20. SIAM (1998)
30. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD, pp. 701–710 (2014)
31. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: RecSys, pp. 130–137 (2017)
32. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
33. Shang, J., Qu, M., Liu, J., Kaplan, L.M., Han, J., Peng, J.: Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. arXiv preprint arXiv:1610.09769 (2016)
34. Shi, C., Hu, B., Zhao, W.X., Yu, P.S.: Heterogeneous information network embedding for recommendation. IEEE Transactions on Knowledge and Data Engineering **31**(2), 357–370 (2019)
35. Shi, C., Li, Y., Zhang, J., Sun, Y., Philip, S.Y.: A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering **29**(1), 17–37 (2016)
36. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: CIKM, pp. 1441–1450 (2019)
37. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author relationship prediction in heterogeneous bibliographic networks. In: ASONAM, pp. 121–128 (2011)
38. Tang, J., Belletti, F., Jain, S., Chen, M., Beutel, A., Xu, C., H Chi, E.: Towards neural mixture recommender for long range dependent user sequences. In: WWW, pp. 1782–1793 (2019)
39. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
40. Trefethen, L.N., Bau III, D.: Numerical linear algebra, vol. 50. SIAM (1997)
41. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS, pp. 5998–6008 (2017)
42. Wang, H., Zhang, F., Hou, M., Xie, X., Guo, M., Liu, Q.: Shine: signed heterogeneous information network embedding for sentiment link prediction. In: WSDM, pp. 592–600 (2018)
43. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
44. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW, pp. 2022–2032 (2019)
45. Wang, X., Lu, Y., Shi, C., Wang, R., Cui, P., Mou, S.: Dynamic heterogeneous information network embedding with meta-path based proximity. IEEE Transactions on Knowledge and Data Engineering pp. 1–1 (2020)
46. Xu, C., Zhao, P., Liu, Y., Sheng, V.S., Xu, J., Zhuang, F., Fang, J., Zhou, X.: Graph contextualized self-attention network for session-based recommendation. In: IJCAI, vol. 19, pp. 3940–3946 (2019)
47. Yin, Y., Ji, L.X., Zhang, J.P., Pei, Y.L.: Dhne: Network representation learning method for dynamic heterogeneous networks. IEEE Access **7**, 134,782–134,792 (2019)
48. Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., Wu, J.: Sequential recommender system based on hierarchical attention networks. In: IJCAI (2018)
49. Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., Gai, K.: Deep interest evolution network for click-through rate prediction. In: AAAI, vol. 33, pp. 5941–5948 (2019)
50. Zhu, D., Cui, P., Zhang, Z., Pei, J., Zhu, W.: High-order proximity preserved embedding for dynamic networks. IEEE Transactions on Knowledge and Data Engineering **30**(11), 2134–2144 (2018)