HySAGE: A Hybrid Static and Adaptive Graph Embedding Network for Context-Drifting Recommendations

Sichun Luo
Department of Computer Science
City University of Hong Kong
Hong Kong, China
City University of Hong Kong Shenzhen Research Institute
Shenzhen, China
sichun.luo@my.cityu.edu.hk

Xinyi Zhang
Department of Accounting
Capital University of Economics and Business
Beijing, China
zhangxinyi@cueb.edu.cn

Yuanzhang Xiao
Department of Electrical & Computer Engineering
University of Hawaii at Manoa
Honolulu, HI, USA
yxiao8@hawaii.edu

Linqi Song
Department of Computer Science
City University of Hong Kong
Hong Kong, China
City University of Hong Kong Shenzhen Research Institute
Shenzhen, China
linqi.song@cityu.edu.hk

ABSTRACT
The recent popularity of edge devices and Artificial Intelligent of Things (AIoT) has driven a new wave of contextual recommendations, such as location based Point of Interest (PoI) recommendations and computing resource-aware mobile app recommendations. In many such recommendation scenarios, contexts are drifting over time. For example, in a mobile game recommendation, contextual features like locations, battery, and storage levels of mobile devices are frequently drifting over time. However, most existing graph-based collaborative filtering methods are designed under the assumption of static features. Therefore, they would require frequent retraining and/or yield graphical models burgeoning in sizes, impeding their suitability for context-drifting recommendations.

In this work, we propose a specifically tailor-made Hybrid Static and Adaptive Graph Embedding (HySAGE) network for context-drifting recommendations. Our key idea is to disentangle the relatively static user-item interaction and rapidly drifting contextual features. Specifically, our proposed HySAGE network learns a relatively static graph embedding from user-item interaction and an adaptive embedding from drifting contextual features. These embeddings are incorporated into an interest network to generate the user interest in some certain context. We adopt an interactive attention module to learn the interactions among static graph embeddings, adaptive contextual embeddings, and user interest, helping to achieve a better final representation. Extensive experiments on real-world datasets demonstrate that HySAGE significantly improves the performance of the existing state-of-the-art recommendation algorithms.

CCS CONCEPTS
• Information systems → Recommender systems.

KEYWORDS
Recommender system; Context-aware recommendation; Graph embedding; Attention

ACM Reference Format:
Sichun Luo, Xinyi Zhang, Yuanzhang Xiao, and Linqi Song. 2022. HySAGE: A Hybrid Static and Adaptive Graph Embedding Network for Context-Drifting Recommendations. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM ’22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557354

1 INTRODUCTION
With the proliferation of mobile edge devices, contextual information is explored dramatically to show its powerful impact in recommender systems, especially towards personalized recommendations. Compared to traditional context-aware recommendations with commonly used contextual information of time, location, companion, and environmental situation, these recent contextual recommendations focus more on features from the mobile and edge devices. One example is to recommend Point of Interest (PoI) based on location, weather, and social behavior [2]. As another example, mobile app recommendations can adapt to the mobile device’s resources and usage levels, such as computing power, communication capacity, battery levels, etc [3].

In these emerging contextual recommendations, we observe that contexts are often drifting rapidly, compared to the relatively
stable user-item interactions. For example, in a mobile game recommendation scenario, contextual features like locations, battery, and storage levels of mobile devices are frequently changing over time and there are quite a few versions of the game tailor-made for different mobile resources; while the user’s rating behavior over the mobile game is sparse and relatively static. Recent graph-based deep learning techniques have been widely used in recommendations. However, some of these graph-based methods \cite{6, 7, 25} focused on the user-item interaction and failed to fully exploit the contextual information (or even neglected the contextual information). Other works did not take into account the drifting of contextual information and may cause prohibitively high computation complexity or sparsity problems (e.g., the number of nodes being in the order of user/item-context concatenation pairs) in graph based solutions when the space of drifting contextual information grows large \cite{26, 30}. Therefore, these existing works may not be suitable for computing resource constrained mobile devices and edge devices in the Artificial Intelligent of Things (AIoT) systems.

To tackle these problems, we propose a tailor-made graph based solution, termed a Hybrid Static and Adaptive Graph Embedding (HySAGE) network, for context-drifting recommendations. Intuitively, we adopt a hybrid structure to learn different embeddings for relatively static user-item interaction and rapidly drifting contextual information. By decoupling the adaptive representation of contextual information and the static representation of user-item interaction, our proposed method is especially suitable for context-drifting recommendations. In this way, the drifted contextual attributes would pass through the embedding layer via interactive attention mechanisms and there is no need to re-train the whole graph. Therefore, using such a hybrid structure could potentially save computation resources and retraining time.

Specifically, the proposed HySAGE network first uses user and item relations (e.g., the rating matrix) to construct a bipartite graph and obtain user and item similarity graphs. After that, we adopt a co-occurrence and random walk-based graph embedding algorithm to exploit user and item collaborative embeddings respectively. Meanwhile, multimodal contextual information is also incorporated via various embedding techniques, like specific pre-trained models for texts and images (e.g., Sentence-BERT and ResNet), pre-processing techniques (e.g., normalization and feature crossing) for other categorical and dense contextual features. To reduce the feature dimensionality and learn the feature interaction, the generated feature vectors are fed into the feature crossing layer to learn the higher-order non-linear feature interactions. After obtaining the extracted graph embedding and contextual embedding, we adopt the self-attention mechanism to model the user interests. Instead of the average pooling, the attention mechanism can learn the importance of each component and assign different weights to the components accordingly. As a result, we are able to fuse all the representations from different sources and acquire the final representation for users and items. Finally, these representations are fed into a multi-layer perception to predict the final ratings. Our experiments over four real-world datasets show that our proposed HySAGE outperforms benchmark solutions by up to 20%-30% through effectively processing the drifting contextual information.

To summarize, the main contributions of this work are three-fold:

- To effectively learn the fast changing contextual information and relative static user-item interaction, we propose a novel end-to-end HySAGE network for context-drifting recommendations, in which the graph embedding module is combined with contextual feature extraction module and user interest mining module to generate a comprehensive representation from different sources.
- We incorporate recent advanced techniques to better learn the comprehensive representation: a co-occurrence and random walk-based graph embedding technique to extract both global statistical information and local graph information to obtain user and item embeddings accordingly; a multimodal processing technique for jointly exploring multimodal contextual information and other categorical and dense features; a self-attention mechanism to learn the user interest from both the graph embedding and the contextual information embedding; and an interactive attention mechanism to combine different representations into a comprehensive representation.
- We carry out extensive experiments on four real-world datasets to demonstrate the effectiveness of our method (up to 20%-30% gains over benchmarks) and its importance in incorporating contextual information and user interest.

2 RELATED WORK

2.1 Context-Aware Recommendations

Context-aware recommendation is a popular research direction for two decades, since it utilizes contextual information, such as time, location, social relationship data, and environmental data, to facilitate the recommendation and ease the data sparsity and cold-start problems \cite{32}. Contextual recommendations span from early works with feature engineering techniques to extract categorical and dense features, like movies or news recommendations \cite{1, 17}, to recent trends of location and social behavior based Pol recommendations and resource-aware mobile app recommendations \cite{2, 3, 18}.

Feature interaction and user interest modeling are two commonly used methods to incorporate contexts in context-aware recommendations. Factorization machines (FM) and its deep learning versions, such as DeepFM, convolutional FM (CFM) and deep cross network (DCN), are able to capture the interactions between different input features and embed them into a low-dimensional latent space \cite{5, 11, 22, 28, 32}. Interest modeling, e.g., deep interest network (DIN), deep interest evolution network (DIEN), and deep session interest network (DSIN), enables the incorporation of various contextual features and adopts an attention mechanism to model users’ interests based on these features and user-item interactive behaviors \cite{9, 37, 38}.

However, these existing methods did not make specific design to deal with the rapid context-drifting problem. Hence, in this work, we will highlight the importance of disentangling relative static user-item interactions and more dynamic contexts, and show the effectiveness of such a tailor-made solution.

2.2 Graph-Based Recommendations

Recently, graph-based models have attracted more attention in recommendations to extract higher-order relations between users and items due to its powerful ability to capture multi-hop relations in the graph. A neural graph collaborative filtering (NGCF) method
employs a 3-hop graph neural network to learn user and item embeddings from the user-item bipartite graph [29]. LightGCN [13] further improve it by removing the feature transformation and nonlinear activation operation, which contribute little to the model performance. PinSage [34] utilized a 2-hop graph convolutional neural network and random walk to facilitate the representation learning. MMGCN [30] proposes a graph convolutional network to learn representations from the multimodal information in multimedia recommendations. A random walk based graph embedding method [7] is used to extract high-order collaborative signals from the user-item bipartite graph. Graph neural networks (GNNs) are also introduced to model user’s local and global interest for recommendations [31, 33, 35].

Nonetheless, incorporating a large amount of drifting contextual information into the graph based models would lead to an exploding number of nodes (user/item-context concatenation) in the graph, posing difficulties in learning and retraining. In light of this, a specifically designed solution is needed to disentangle user-item interactions and contexts and a more comprehensive design is needed to combine these embeddings learned from different information sources.

3 PRELIMINARIES
We consider a dynamic context-drifting recommender system that consists of N users and M items, and denote the sets of users and items as $\mathcal{U} = \{u_1, ..., u_N\}$ and $\mathcal{I} = \{i_1, ..., i_M\}$, respectively. Different from problem formulation of static recommendations, we consider a time horizon of T time slots.

**Attributes.** At each time slot $t \in \{1, ..., T\}$, the attribute of each user $u_n \in \mathcal{U}$ is denoted by a vector $a_{u_n}(t) \in \mathbb{R}^{d_i}$, and the attribute of each item $i_m \in \mathcal{I}$ by a vector $a_{i_m}(t) \in \mathbb{R}^{d_i}$. In this way, we model the drifting contexts associated with users and/or items: user attributes (e.g., locations, battery levels of the device) and item attributes (e.g., freshness of content, item descriptions).

**User-Item Interactions.** Over the course of T time slots, a user $u_n$ may have interaction with item $i_m$ at time $t$ and generate multimodal information (e.g., texts, images, and videos from reviewing the item). We denote the contextual information by a vector $e_{u_n i_m}(t) \in \mathbb{R}^{d_c}$. At the end of T time slots, we use the matrix $Y \in \{0, 1\}^{N \times M}$ to summarize the user-item interactions, where $Y_{nm} = 1$ if user $u_n$ has interaction with item $i_m$ is observed and $Y_{nm} = 0$ otherwise. Note that this interaction matrix can be relaxed by allowing $Y_{nm} \in \mathbb{R}$ to reflect multiple interactive behaviors (like the frequency of listening to some music).

**User Interests.** Each user $u_n$ has time-varying interests in each item $i_m$, denoted by a vector $\theta_{u_n i_m}(t) \in \mathbb{R}^{d_i}$.

**Ratings.** We wish to learn the users’ ratings of the items at time $t$ based on all the information available up to time $t$. We write the collection of attributes, contextual information, and user interests up to time $t$ by $a_{u_n}[1 : t]$, $a_{i_m}[1 : t]$, $e_{u_n i_m}[1 : t]$, and $\theta_{u_n i_m}[1 : t]$, respectively. Formally, we model user $u_n$’s rating of item $i_m$ at time $t$ as follows

$$R_t(e_{u_n}, e_{i_m}, a_{u_n}(t), a_{i_m}(t), e_{u_n i_m}[1 : t], \theta_{u_n i_m}(t)),$$

where $e_{u_n}$ and $e_{i_m}$ are vectors representing user $u_n$ and item $i_m$. Note that we use vectors $e_{u_n}$ and $e_{i_m}$ instead of indices $n$ and $m$, to represent the user and the item, because more useful information can be encoded into the vectors.

Based on our model (1), we need the following key components:

1. **Static embeddings of user and item identities** $e_{u_n}$ and $e_{i_m}$. This is done by a graph embedding algorithm that captures user-item interactions as well as user and item similarities. Note that the graph embedding produces static embeddings.

2. **Adaptive embeddings of time-varying user and item attributes** $a_{u_n}(t)$ and $a_{i_m}(t)$, and contextual information of user-item interactions $e_{u_n i_m}(t)$. The attributes and contextual information can include multimodal information such as numbers, texts, and images. Therefore, we propose a contextual information extraction module to fusion multimodal information into vectors. Note that the embeddings are time-varying, capturing drifting attributes and contextual information.

3. **Estimation of time-varying user interests** $\theta_{u_n}(t)$. Based on user and item attributes $a_{u_n}[1 : t]$ and $a_{i_m}[1 : t]$, and contextual information of user-item interactions $e_{u_n i_m}[1 : t]$, we estimate the user interests $\theta_{u_n i_m}(t)$.

4. **Estimation of user ratings** $R_t$ in (1). We estimate user ratings based on all available information.

To the best of our knowledge, the proposed framework is the only end-to-end method that combines all four components. The resulting recommendation algorithm is context-aware and interest-aware, which boosts the performance. Moreover, it reduces the computational complexity by decoupling the static graph embeddings and the adaptive embeddings of attributes, context, and user interests. In this way, it avoids repeated training of graph embeddings, while still capturing all the information available.

4 THE PROPOSED FRAMEWORK
We present HySAGE, a Hybrid Static and Adaptive Graph Embedding for dynamic recommendation. The framework consists of four major components: a graph embedding module, a contextual information extraction module, a user interest modeling module, and an interactive attention module.

HySAGE operates in four major steps to perform the recommendation. First, we learn static graph embeddings of users and items by building the user and item bipartite network and mining their high-order collaborative embeddings. Second, we obtain adaptive embeddings of contextual information about user-item interaction through (1) representing drifting user and item attributes, (2) adopting pre-trained neural networks to extract multimodal user-item interaction information (e.g., audio, image, text), and (3) using a feature crossing layer to fusion user and item attributes and user-item interaction information, compress the dimension, reduce redundancy and learn high-level feature interactions. Third, we use the attention mechanism to model the users’ recent interests. Fourth, we use local interactive attention mechanisms to extract bilateral interactions among static graph embeddings, adaptive embeddings of user-item interactions, and adaptive user interests, and use a global interactive attention mechanism to learn the final representations from individual embeddings and their bilateral interactions. Finally, we train a multi-layer perception (MLP) to predict the users’ ratings of the items. Fig. 1 illustrates our proposed framework.
between users and items. The user (item) similarity matrix is a \( Y \) (i.e., the two users have interacted with the same item). We define value between two users (items). A widely-used definition of the similarity graphs, we mine collaborative information using graph the weights of the edges as the corresponding co-interaction values.

Using the co-interaction values \( S \), such as Pearson correlation, cosine distance, and Jaccard similarity. The user similarity matrix defines a user similarity graph \( G \), with each element being a co-interaction value between two users (items). A widely-used definition of the co-interaction value between two users is the number of items they both interacted with, and that between two items is the number of users who interacted with both of them. Mathematically, the user similarity matrix can be calculated as

\[
S^U = Y \cdot Y^T \in \mathbb{R}^{N \times N}, \quad (2)
\]

and the item similarity matrix can be calculated as

\[
S^I = Y^T \cdot Y \in \mathbb{R}^{M \times M}. \quad (3)
\]

Note that our proposed framework can also use other definitions of co-interaction values [24], such as Pearson correlation, cosine distance, and Jaccard similarity.

The user similarity matrix defines a user similarity graph \( G_U \), with the set of the nodes as the user set \( \mathcal{U} \). There exists an edge between user \( u_n \) and user \( u_l \) if their co-interaction value \( s_{nl}^U \) (i.e., the \((n,l)\)-th element of the user similarity matrix \( S^U \)) is non-zero (i.e., the two users have interacted with the same item). We define the weights of the edges as the corresponding co-interaction values. The item similarity graph \( G_I \) can be defined in the same way.

4.1 Graph Embedding

The graph embedding module builds the user and item similarity graphs and learns static graph embeddings \( e_{u_n} \) and \( e_{i_m} \) for each user \( u_n \in \mathcal{U} \) and each item \( i_m \in \mathcal{I} \). We adopt graph embedding method similar to [7]. Fig. 2 illustrates the module.

Building User/Item Similarity Graphs. We first calculate user and item similarity matrices solely based on the interactions \( Y \) between users and items. The user (item) similarity matrix is a \( N \times N \) (\( M \times M \)) matrix, with each element being a co-interaction value between two users (items). We adopt graph embedding techniques. We adopt deep learning based graph embedding techniques [10, 20], which have been widely applied in network and graph analysis for their ability in extracting node and edge features [4].

To capture the structures of a graph, most graph embedding techniques simulate a random walk process to create node sequences. We describe the process to generate user sequences based on the user similarity graph; the item sequences are generated in the same way. Denote the user sequence of a fixed length \( K \) by \( \{u_{n_1}, \ldots, u_{n_{K+1}}\} \). Given the \( k \)-th node \( u_{n_k} \), the next node \( u_{n_{k+1}} \) is randomly selected from node \( u_{n_k} \)'s neighbors according to the following probabilities [10]:

\[
P(u_{n_{k+1}} | u_{n_k}) \propto \begin{cases} 
\frac{S_{n_k,n_{k+1}}^U}{p,} & \text{if } u_{n_{k+1}} = u_{n_{k-1}}, \\
\frac{S_{n_k,n_{k+1}}^I}{q,} & \text{if } u_{n_{k+1}} \text{ is a neighbor of } u_{n_{k-1}}, \text{ and } \quad (4)
\end{cases}
\]

with \( p, q > 0 \). In other words, the next node is chosen with a probability proportional to the similarity to the current node (measured by the co-interaction value \( S_{nl}^I \)), moderated by parameters \( p, q \). For example, we can choose \( p > 1 \) to discourage the selection of the previous node \( u_{n_{k-1}} \) and \( q < 1 \) to encourage the selection of nodes that are not neighbors of the previous node \( u_{n_{k-1}} \). As we can see from (4), a user more similar to the current node, as measured by a higher co-interaction value, is more likely to be selected as the next node in the sequence. Hence, this sampling process helps us to better capture collaborative information.

Given all the \( K \)-node sequences, we use the global co-occurrence based method in [19] to learn the user embeddings \( e_{u_n} \). The co-occurrence between two users \( u_n \) and \( u_l \), denoted \( O_{nl} \), is defined as the empirical frequency of these two users appearing in the

Figure 1: The overall architecture of the proposed model. We first learn the graph embeddings for users and items by co-occurrence and random walk-based techniques. Based on the learned graph embeddings, we introduce an attention mechanism to learn the users' interests. Besides, we extract features from contextual information by pretrained models and use a feature crossing network and an attention layer to learn the hidden representation of contextual information. All of these are concatenated, go through a global interactive attention layer, and are fed to a MLP to get the final recommendation.
The contextual information extraction module learns the adaptive feature crossing layer and the attention mechanism for better representation learning of the feature interaction. Fig. 3 illustrates user-item contextual embedding.

**4.2 Contextual Information Extraction**

The contextual information extraction module learns the adaptive embeddings of drifting user and item attributes $a_{u,i}(t)$, $a_{i,m}(t)$ and user-item interactions $c_{u,i,m}(t)$. These adaptive embeddings are obtained from multimodal information (e.g., categorical, numerical, textual, image). Based on the embeddings, the module uses a feature crossing layer and the attention mechanism for better representation learning of the feature interaction. Fig. 3 illustrates the contextual information extraction module.

**Category Features.** Some of the user and item attributes are categorical data (e.g., occupations of the users, product categories of the items). We use one-hot embedding of the categorical data initially, and feed the sparse one-hot vectors to the feature crossing layer to obtain dense vectors later.

**Dense Features.** Dense features (e.g., user ages) are normalized using min-max scalers and then fed into the feature crossing layer.

**Text Data.** Text data (e.g., titles and description of items, review by the users) contains critical information. In this work, we use a pretrained model called Sentence-BERT [21] to process the raw data and get fixed-length vectors as output.

**Image Data.** Image data (e.g., display images of items, pictures uploaded by the users) contains crucial hidden information. Due to the high dimensionality of raw images, here we adopt a pretrained convolutional neural network (CNN), namely ResNet[12], to process the images and get fixed-length vectors as output.

After one-hot embedding of categorial features, preprocessing of dense features, and embedding of text and image data, we obtain embeddings of user and item attributes $a_{u,i}(t)$, $a_{i,m}(t)$ and user-item interactions $c_{u,i,m}(t)$. We write the embedding vector of all the features as

$$x_0 = \text{Concat} \left( a_{u,i}(t), a_{i,m}(t), c_{u,i,m}(t) \right) \in \mathbb{R}^{d_u + d_i + d_c} \quad (7)$$

where Concat() is the concatenation operation. Instead of using the embedding vector $x_0(t)$ directly, we pass it through a feature crossing network to learn high-order interaction across features [28] and a fusion layer to reduce the dimensionality.

**The Feature Crossing Network.** The feature crossing network consists of $L$ feature crossing layers. The output $x_{l+1} \in \mathbb{R}^{d_u + d_i + d_c}$ of layer $l + 1$ is obtained by [28]

$$x_{l+1} = x_0 x_l^T w_l + b_l + x_l, \quad (8)$$

where $x_l \in \mathbb{R}^{d_u + d_i + d_c}$ is the output of layer $l$, $w_l, b_l \in \mathbb{R}^{d_u + d_i + d_c}$ are weight and bias parameters. The feature crossing is achieved by the product $x_0 x_l^T$. A deeper feature crossing network captures higher-order interaction across features.

**The Fusion Layer.** Given the embedding vector $x_0$ and the high-order cross feature embedding $x_L$, the fusion layer uses the self-attention mechanism to obtain the final embedding of the contextual information. As opposed to simply concatenating or adding $x_0$ and $x_L$, the self-attention mechanism assigns different weights to features and hence can focus on more important features.
self-attention mechanism can be mathematically expressed as:

$$e_c = \sum_{i \in \{0, L\}} \exp(\tanh(w_i^T \cdot x_i + b_i) / \sum_{i'=0}^{L} \exp(\tanh(w_i^T \cdot x_i + b_i'))) \cdot x_i,$$

(9)

where $w_i$ and $b_i$ are the weights and the bias of the attention layer, and $e_c$ is the final output for contextual information extraction module.

4.3 User Interest Modeling

Previous works show user interests are diverse and dynamic, and may have a considerable impact on the recommendation performance [37, 38]. Therefore, we utilize the user-items interactions to learn the user interests.

To learn user $u_t$’s interest in a target item $i_m$, we randomly select $K$ items that this user has interacted with. We select $K$ random items, instead of the most recent $K$ items, because we do not assume the availability of time and sequence information. Denote this set of $K$ items as $I_{u_t} = \{i_{u_t,1}, i_{u_t,2}, \ldots, i_{u_t,K}\}$, where $K$ is a hyperparameter. In Section 4.1, we have obtained the embedding $e_{i_{m,k}}$ of the target item $i_m$ and the embeddings $e_{i_{u,t,k}}$ of the selected items $i_{u,t,k}$ for $k = 1, \ldots, K$. We learn user $u_t$’s “valuation” (relative to the target item $i_m$) of the $k$ selected item $i_{u,t,k}$ through a MLP:

$$v_k = MLP\left(e_{i_{u,t,k}}, e_{i_m}, e_{i_{u,t,k}} - e_{i_m}\right) \in \mathbb{R}. \quad (10)$$

Then we calculate user $u_t$’s interest in the target item $i_m$ by passing the embeddings of the selected items through a self-attention layer:

$$\theta_{u_t,i_m} = \sum_{k=1}^{K} \exp(v_k) / \sum_{j=1}^{K} \exp(v_j) \cdot e_{i_{u,t,k}}. \quad (11)$$

The user interest module is illustrated in Fig. 4.

4.4 Multi-Interactive Learning

At this point, we have obtained a set of latent vectors for the users and items in interaction-based feature learning, denoted as $V_u, V_i, V_c$, representing user interest vector, contextual vector, and collaborative vector respectively. Previous works indicate low-order and high-order interactions could contribute to model performance. Here we highlight the importance of learning high-order interactive information from different feature representations. The intuition is that different representations describe different aspects of the users and items. Therefore, directly concatenate them is not enough for well representation learning. Therefore, we propose a Multi interactive learning module. More specifically, we learn the importance of each component by an attention mechanism. Besides, we explicitly learn the representation interaction by a feature crossing structure.

Local Interaction Learning. We regard the contextual representation of the user and item as their inherent attributes, thus the latent contextual vector learned above describes the characteristics of users and items. Besides, the collaborative feature provides rich information about the user’s general preferences. Moreover, the user interest model the personalized preference with respect to the specific items. Hence, each latent vector expresses the user/item in a different way and a single latent vector does not represent the users and items to their fullness. Therefore, we derive a learning process for latent representation of the interaction between interest features, collaborative features, and contextual features. Specifically, we learn the interaction between every two components. Similar to the user interest part, we introduce an attention mechanism to quantify the importance of representation interaction and calculate the scaled attention representation as follows. Let $V$ represent a set of feature vectors from collaborative vector, contextual vector, and interest vector respectively, $V$ is a subset of $V$. The $R_{ui}$ represents the feature interaction between the candidate user interest and contextual information. Firstly, we simply concatenate the contextual information with the candidate user interest representation to get the interactive feature representation, formulated by:

$$V_{ui}^k = Concat(V_u^k, V_c). \quad (12)$$

Then we adopt the attention mechanism to emphasize different parts of user interest by assigning different weights. The formula for the final representation $R_{ui}$ is

$$R_{ui} = \sum_{k=1}^{K} \frac{\exp(\tanh(V_{ui}^k \cdot W_{ui}^k + b_{ui}^k))}{\sum_{k'=1}^{K} \exp(\tanh(V_{ui}^k \cdot W_{ui}^k + b_{ui}^k))} \cdot V_{ui}^k. \quad (13)$$

Moreover, let the $R_{uc}$ represents the feature interaction between the candidate user interest and collaborative information. Similarly, we get the final representation formula for $R_{uc}$:

$$V_{uc}^k = Concat(V_u^k, V_c), \quad \quad \quad \quad R_{uc}^k = \frac{\exp(\tanh(V_{uc}^k \cdot W_{uc}^k + b_{uc}^k))}{\sum_{k'=1}^{K} \exp(\tanh(V_{uc}^k \cdot W_{uc}^k + b_{uc}^k))} \cdot V_{uc}^k. \quad (14)$$

Global Interaction Learning. In the global interaction learning part, we still adopt attention, which is a technique that is often used in natural language processing applications [27]. Specifically, we concatenate all the embedding, denoted as $R_{ui} = [R_1, R_2, \ldots, R_n]$ where $R_{ui}$ in $[R_1, R_2, \ldots, R_n]$ is a kind of embedding extracted in the previous module. Then we use the attention mechanism to get the final output for the global interaction learning module $R_{ao}$, and the formula is denoted as:

$$R_{ao} = \sum_{i=1}^{n} \frac{\exp(\tanh(R_i \cdot W_i + b_i))}{\sum_{j=1}^{L} \exp(\tanh(R_j \cdot W_j + b_j))} \cdot R_i. \quad (15)$$
4.5 DNN MLP Structure

A tower-shaped MLP structure is often used for predicting ratings in recommender systems [8, 14]. By utilizing neural structures, the prediction functions have strong ability in handing non-linear relations and can capture user-item interactions far better than simple inner product function. However, these existing works did not consider using graph embedding and user interest to obtain initial features.

In our framework, we adopt a tower-shaped MLP as the prediction network, as shown at the top of Fig. 1. The input of the MLP includes the global feature representation generated by the global interaction module. To make accurate predictions, we also sample 4 un-interacted items for a user to generate negative samples. The MLP has fully connected layers, uses softmax as the activation function of the output layer, and uses rectified linear unit (ReLU) as the activation function of the other layers. We use a mini-batch gradient descent and Adaptive Moment Estimation (Adam) optimization algorithm to train the neural network.

5 EXPERIMENTS

In this section, we perform experiments on real-world datasets from different domains for performance evaluation. We also analyze the components for our proposed framework. We aim to answer these following research questions:

RQ1: How does our model performs in comparison to baseline methods? Will mining and encoding collaborative information help? If the method can outperform others, why using graph-enhanced method can help?

RQ2: How does the incorporation of user interest and contextual information will affect the recommendation performance?

RQ3: How do hyperparameters (i.e., the embedding size, the random walk length) affect the performance?

5.1 Experiment Setup

5.1.1 Datasets. We perform experiments on four real-world datasets to evaluate our proposed HySAGE framework. Among the four datasets, Kaggle-Movie and MovieLens-100K are movie datasets, and Yelp and Amazon-Kindle are review datasets. Table 1 summarizes the statistics of the datasets. Detailed description of the datasets are as follows.

- **Yelp**: This is from Yelp-2018 challenge. The task is to recommend business webpages to users. To ensure the quality of dataset, we remove the users who give less than five reviews and the business who receive less than five reviews.

- **Amazon-Kindle**: This is from the Amazon review data. The task is to recommend e-books to users. We use the 10-core setting to ensure the quality of dataset. Compared to the Yelp dataset, it has similar numbers of users and items, but much fewer interactions. Therefore, it allows us to test our method when user-item interactions are sparse.

- **MovieLens-100K**: This is a canonical movie recommendation dataset widely used for evaluating recommendation algorithms.

- **Kaggle-Movie**: This is an extended MovieLens dataset released on Kaggle. We remove the movies with less than two records.

5.1.2 Baselines. We compare our proposed HySAGE algorithm with the following canonical and state-of-the-art algorithms.

- **Item popularity**: We simply rank items by its popularity and utilize no collaborative information.

- **Bayesian Personalized Ranking (BPR)** [23]: Instead of optimizing over point-wise user-item ratings, this model is trained to rank interacted items higher than those with no interaction.

- **MF** [16] + **NS**: a traditional matrix factorization method with negative sampling to enhance the data.

- **Neural Collaborative Filtering (NCF)** [14]: a deep learning-based collaborative filtering method using deep tower-shaped MLPs.

- **JRL** [36]: a multi-layer neural network incorporating element-wise products of the user and item embedding.

- **NeuMF** [14]: a deep learning based collaborative filtering model combining generalized matrix factorization [14] and NCF.

- **GEM-RS** [7]: GEM-RS uses graph embedding to learn collaborative information from user-item interaction matrix.

- **PinSage** [34]: a large scale graph convolutional network that aggregates random walks and graph convolution operations.

- **Neural Graph Collaborative Filtering (NGCF)** [29]: a state-of-the-art neural graph collaborative filter method that mines collaborative signals from the graph structure.

5.2 Experiments Settings

For movie datasets, we use the pre-trained ResNet50 model [12] to extract the textual features from the movie posters, and use the Sentence-BERT [21] to encode the textual features from the movie descriptions. For all deep-learning-based methods, the activation functions are ReLU, the learning rates are 0.001 to 0.01, the L2 regulations are $10^{-7}$, and the batch sizes are 256. For all four datasets, we set the embedding size to 64 for fair comparison. In NCF, NeuMF, and GEM-RS, the number of nodes is halved by each layer and the last layer has 8 nodes. In JRL, there are three layers with the same number of nodes. In HySAGE, the walk length is 20 and the walk number is 100. We perform a grid search on the hyperparameters (e.g., learning rate, dropout rate) and select the best for each method according to the performance on the validation set.

We adopt the commonly used leave-one-out evaluation strategy, which reserves the latest interacted item of each user as the test item. Then we generate a list of items and rank them using the predicted scores. We use two metrics to evaluate the performance: the Hit Ratio (HR), which measures the chance that our recommendation list contains users’ interested items, and a weighted version of HR, termed Normalized Discounted Cumulative Gain (NDCG), which puts more weights on items that are ranked higher in the recommendation list. All performance metrics are computed on the test set and averaged over five runs.

| Table 1: Dataset description |
|-------------------------------|
| Dataset          | # of user | # of item | # of rating | sparsity  |
| Yelp             | 21,284    | 16,771    | 984,000     | 99.72%    |
| Amazon-Kindle    | 14,535    | 15,884    | 367,478     | 99.84%    |
| MovieLens-100K   | 943       | 1,682     | 100,000     | 94.12%    |
| Kaggle-Movie     | 670       | 5,977     | 96,761      | 98.55%    |

5.3 Experiments Results

For Amazon-Kindle dataset, we perform experiments on Amazon-Kindle dataset which is a large-scale review data. We use the pre-trained ResNet50 model to extract the textual features from the product descriptions. For all deep-learning-based methods, the activation functions are ReLU, the learning rates are 0.001 to 0.01, the L2 regulations are $10^{-7}$, and the batch sizes are 256. For all four datasets, we set the embedding size to 64 for fair comparison. In NCF, NeuMF, and GEM-RS, the number of nodes is halved by each layer and the last layer has 8 nodes. In JRL, there are three layers with the same number of nodes. In HySAGE, the walk length is 20 and the walk number is 100. We perform a grid search on the hyperparameters (e.g., learning rate, dropout rate) and select the best for each method according to the performance on the validation set.

We adopt the commonly used leave-one-out evaluation strategy, which reserves the latest interacted item of each user as the test item. Then we generate a list of items and rank them using the predicted scores. We use two metrics to evaluate the performance: the Hit Ratio (HR), which measures the chance that our recommendation list contains users’ interested items, and a weighted version of HR, termed Normalized Discounted Cumulative Gain (NDCG), which puts more weights on items that are ranked higher in the recommendation list. All performance metrics are computed on the test set and averaged over five runs.

5.3.1 Baselines. We compare our proposed HySAGE algorithm with the following canonical and state-of-the-art algorithms.

- **Bayesian Personalized Ranking (BPR)** [23]: Instead of optimizing over point-wise user-item ratings, this model is trained to rank interacted items higher than those with no interaction.

- **MF** [16] + **NS**: a traditional matrix factorization method with negative sampling to enhance the data.

- **Neural Collaborative Filtering (NCF)** [14]: a deep learning-based collaborative filtering method using deep tower-shaped MLPs.

- **JRL** [36]: a multi-layer neural network incorporating element-wise products of the user and item embedding.

- **NeuMF** [14]: a deep learning based collaborative filtering model combining generalized matrix factorization [14] and NCF.

- **GEM-RS** [7]: GEM-RS uses graph embedding to learn collaborative information from user-item interaction matrix.

- **PinSage** [34]: a large scale graph convolutional network that aggregates random walks and graph convolution operations.

- **Neural Graph Collaborative Filtering (NGCF)** [29]: a state-of-the-art neural graph collaborative filter method that mines collaborative signals from the graph structure.

5.3.2 Experiments Settings

For movie datasets, we use the pre-trained ResNet50 model [12] to extract the textual features from the movie posters, and use the Sentence-BERT [21] to encode the textual features from the movie descriptions. For all deep-learning-based methods, the activation functions are ReLU, the learning rates are 0.001 to 0.01, the L2 regulations are $10^{-7}$, and the batch sizes are 256. For all four datasets, we set the embedding size to 64 for fair comparison. In NCF, NeuMF, and GEM-RS, the number of nodes is halved by each layer and the last layer has 8 nodes. In JRL, there are three layers with the same number of nodes. In HySAGE, the walk length is 20 and the walk number is 100. We perform a grid search on the hyperparameters (e.g., learning rate, dropout rate) and select the best for each method according to the performance on the validation set.

We adopt the commonly used leave-one-out evaluation strategy, which reserves the latest interacted item of each user as the test item. Then we generate a list of items and rank them using the predicted scores. We use two metrics to evaluate the performance: the Hit Ratio (HR), which measures the chance that our recommendation list contains users’ interested items, and a weighted version of HR, termed Normalized Discounted Cumulative Gain (NDCG), which puts more weights on items that are ranked higher in the recommendation list. All performance metrics are computed on the test set and averaged over five runs.
worse than the other methods, which stresses the importance of
we repeat the experiment by removing one module from the pro-
To prove the effectiveness of each critical module in our proposed
Table 2 reports the overall performance of all methods. The best
validates that user interest mining and context-aware user-item
the significant performance gap between HySAGE and GEM-RS
proposed HySAGE model and test the performance of these incomplete
5.4 Ablation Study (RQ2)
To prove the effectiveness of each critical module in our proposed
5.3 Performance Comparison (RQ1)
Table 2 reports the overall performance of all methods. The best
Results and the improvements over the second best are highlighted
HySAGE-Variation 1 (w/o multimodal information): The proposed
HySAGE-Variation 2 (w/o side information): The proposed
HySAGE-Variation 3 (w/o user interest): The proposed
HySAGE-Variation 4 (w/o multi-interactive): The proposed
From Table 3, we observe that the HySAGE achieves the best
possibly to introduce neural networks and graph structures into the
Third, our proposed HySAGE approach consistently achieves
can see that the performance of the HySAGE improves with the
impact of the Hyperparameter $K$. In the user interest mining
module, we randomly sample $K$ items and derive user potential
Impact of the Hyperparameter $K$. In the user interest mining
can see that the performance of the HySAGE improves with the
Impact of the Embedding Size. We study how the embedding
to increase the embedding size, there is an degrade in performance. This
representation or multimodal information extraction, the performance
in the contextual information extraction module.
user interest mining drop sharply, which proves that the lack of
attentively learning the user interest could significantly decrease
the learning ability of the framework. Without the side information
extraction or multimodal information extraction, the performance
of the framework still decreases considerably. This reflects that
the contextual information contains abundant auxiliary informa-
tion, which improves the model performance. However, compared
with the other modules, the model of contextual information is
relatively small. The contextual information module is like a resid-
ual network to supplement the extra content information to the
HySAGE framework. Throughout these three ablation experiments,
it turns out that each module improves the model performance
from different aspects and is meaningful.
5.5 Analysis of HySAGE (RQ3)
We study the impact of different model settings of HySAGE.
Impact of the Hyperparameter $K$. In the user interest mining
module, we randomly sample $K$ items and derive user potential
interest using the attention mechanism. We study the impact of the
hyperparameter $K$, which determines the model capacity. Figure 5
lists the model performance of HySAGE under different $K$. We
can see that the performance of the HySAGE improves with the
increase of $K$. In essence, higher $K$ allows HySAGE to learn user
interest information from more data, and therefore achieve better
representation learning of user interest.
Impact of the Embedding Size. We study how the embedding
size (the dimension of the embedding/latent features) affects the
performance of HySAGE. Table 4 shows the performance when the
embedding size varies from 32 to 128. For the two datasets tested,
the embedding size of 32 achieves the best performance. As we
increase the embedding size, there is an degrade in performance. This

| Model                        | Yelp HR@10 | Yelp NDCG@10 | Amazon-Kindle HR@10 | Amazon-Kindle NDCG@10 | MovieLens-100K HR@10 | MovieLens-100K NDCG@10 | Kaggle-Movie HR@10 | Kaggle-Movie NDCG@10 |
|------------------------------|------------|--------------|---------------------|------------------------|---------------------|------------------------|---------------------|---------------------|
| Item popularity              | 0.2940     | 0.1611       | 0.2755              | 0.1466                 | 0.4189              | 0.2337                 | 0.4829              | 0.2713              |
| MF+NS                        | 0.6712     | 0.4011       | 0.5925              | 0.3691                 | 0.6351              | 0.3549                 | 0.5956              | 0.3691              |
| BPR                          | 0.7239     | 0.4212       | 0.6925              | 0.4176                 | 0.5762              | 0.3021                 | 0.5887              | 0.3753              |
| JRL                          | 0.6889     | 0.4217       | 0.6834              | 0.4331                 | 0.6473              | 0.3657                 | 0.6349              | 0.3881              |
| NCF                          | 0.7245     | 0.4316       | 0.7134              | 0.4372                 | 0.6525              | 0.3789                 | 0.6617              | 0.3973              |
| NeuMF                        | 0.7688     | 0.4734       | 0.7194              | 0.4469                 | 0.6522              | 0.3918                 | 0.6647              | 0.4036              |
| GEM-RS                       | 0.7886     | 0.4920       | 0.7637              | 0.4911                 | 0.6681              | 0.3950                 | 0.6885              | 0.4280              |
| PinSage                      | 0.7625     | 0.4371       | 0.7184              | 0.4424                 | 0.6660              | 0.4010                 | 0.6706              | 0.3986              |
| NGCF                         | 0.7833     | 0.4616       | 0.7207              | 0.4544                 | 0.6706              | 0.4229                 | 0.6734              | 0.4031              |
| HySAGE                       | 0.9527     | 0.8571       | 0.9329              | 0.8095                 | 0.8348              | 0.6832                 | 0.8841              | 0.7443              |
| Improvement                  | 20.81%     | 74.21%       | 24.77%              | 64.83%                 | 24.95%              | 72.96%                 | 28.41%              | 73.90%              |
Table 3: Ablation study of recommendation performance in two datasets

| Model                      | Movielens   | Kaggle     |
|----------------------------|-------------|------------|
|                            | HR@5  | NDCG@5 | HR@10 | NDCG@10 | HR@20 | NDCG@20 | HR@5  | NDCG@5 | HR@10 | NDCG@10 | HR@20 | NDCG@20 |
| HySAGE                    | 0.8147 | 0.7348  | 0.8336  | 0.7572   | 0.9417  | 0.7718  | 0.8349  | 0.7471  | 0.9038  | 0.7694  | 0.9319  | 0.7817  |
| HySAGE-Variation1        | 0.7335  | 0.6301  | 0.8332  | 0.6619   | 0.9103  | 0.7086  | 0.7574  | 0.7517  | 0.8991  | 0.7607  | 0.9488  | 0.7826  |
| HySAGE-Variation2        | 0.7489  | 0.6553  | 0.8348  | 0.6832   | 0.9104  | 0.7023  | 0.8096  | 0.7210  | 0.8841  | 0.7443  | 0.9411  | 0.7584  |
| HySAGE-Variation3        | 0.5047  | 0.3549  | 0.6702  | 0.4076   | 0.8213  | 0.4445  | 0.5406  | 0.3831  | 0.6792  | 0.4256  | 0.8133  | 0.4590  |
| HySAGE-Variation4        | 0.7385  | 0.6329  | 0.8255  | 0.6611   | 0.9077  | 0.6846  | 0.7914  | 0.7056  | 0.8763  | 0.7328  | 0.9359  | 0.7472  |

Table 4: Performance under different embedding sizes

| Embedding size | Movielens | Kaggle |
|----------------|-----------|--------|
|                | HR@5     | NDCG@5 | HR@5     | NDCG@5 |
| 32             | 0.8354   | 0.7418 | 0.8031   | 0.7748 |
| 64             | 0.8147   | 0.7341 | 0.7765   | 0.7471 |
| 96             | 0.7635   | 0.6601 | 0.7487   | 0.6958 |
| 128            | 0.7532   | 0.6597 | 0.7348   | 0.6781 |

Table 5: Performance under different random walk lengths

| Random Walk Length | Movielens | Kaggle |
|-------------------|-----------|--------|
| 5                 | HR@5     | NDCG@5 | HR@5     | NDCG@5 |
|                   | 0.6556   | 0.6258 | 0.7190   | 0.6556 |
| 10                | 0.5337   | 0.6258 | 0.8148   | 0.6566 |
| 20                | 0.5718   | 0.6258 | 0.8838   | 0.7572 |
|                   | 0.5979   | 0.6258 | 0.7914   | 0.7718 |

Impact of the Random Walk Length. We also study how the random walk length affects the performance. The random walk length is the number of steps taken in the random walk. The larger the walk length is, the more likely a walk is to visit further nodes and discover more complex graph structures. We test different walk lengths ranging from 5 to 20. The results are displayed in Table 5. We can see that the performances of models improve continuously as the walk length increases. Under large walk lengths, similar users and items are more likely to appear in the same walk sequence. Hence, longer walkers produce more expressive and representative embeddings.

5.6 Further Discussions

For context drifting settings, the traditional GNN model would need to train the graph frequently due to the fast drifting context. In contrast, our proposed model could save the computational overhead since the graph does not need to be trained continually. The contextual information extraction module would adaptively extract useful information from the drifting context, and the model could still work well.

6 CONCLUSION

In this paper, we have proposed HySAGE, a Context and Interest Enhanced Graph Embedding technique to boost the performance of multimedia recommendations. We build a bipartite graph from user-item interactions and use random walk-based graph embedding techniques to extract user and item embeddings. The graph embedding is incorporated with attention mechanism to mine the user potential interest, then joint with contextual embeddings that are extracted from multimedia and side information to make multimedia recommendations. Experiments on four real datasets demonstrate the effectiveness of our proposed framework and show significant benefits over existing state-of-the-art algorithms.

ACKNOWLEDGMENTS

This work was supported in part by the Changsha Science and Technology Program International and Regional Science and Technology Cooperation Project under Grants kh2201026, the Hong Kong RGC grant ECS 21212419, the Technological Breakthrough Project of Science, Technology and Innovation Commission of Shenzhen Municipality under Grants JSGG2020110216200001, InnoHK initiative, the Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, the Hong Kong UGC Special Virtual Teaching and Learning (VTL) Grant 6430300, and the Tencent AI Lab Rhino-Bird Gift Fund.
