An Inductive Heterogeneous Graph Recommendation Model for High-Scoring Items Applied to Business Intelligence

Songlin Tian 1, Ying Yang 1,∗ and Lei Yang 2,∗

1 School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China; tiansonglin96@126.com
2 Guangxi Academy of Sciences, Nanning 530007, China
∗ Correspondence: yingy2004@126.com (Y.Y.); leiy2004@126.com (L.Y.)

Abstract: Business intelligence (BI), as a system for business data integration, processing, and analysis, is receiving increasing attention from enterprises. Data visualization is an important feature of BI, which allows users to visually observe the distribution and direction of data and assists them in making correct decisions. The core of this feature is visual analysis charts, which need to be pre-created and integrated into the dashboard by the chart makers, so there are situations where user needs cannot be accurately grasped. At the same time, there may be omissions in the work of users, and a method is needed to remind them. Introducing recommendation models into data visualization is a good solution; therefore, this paper proposes a recommendation model suitable for this type of scenario, which recommends high-scoring items (charts) to users. This model consists of an inductive heterogeneous graph recommendation algorithm with user preferences and a slow-acting collaborative filtering method. The experimental results in two datasets showed an improvement of 0.020/0.045, 0.083/0.019 and 0.076/0.023 in Hit, F-score, and NDCG compared to baselines, which proves that it is more suitable for data visualization requirements and other similar scenarios that require inductive recommendations based on user preferences.

Keywords: business intelligence; data visualization; recommendation models; heterogeneous graphs; user preferences; collaborative filtering

1. Introduction

Business intelligence (BI) is a support system that enhances and improves business decisions and is a collection of enterprise decision support technologies [1]. It can integrate and manage existing data and knowledge of enterprises, providing decision-making support for business and management personnel. Data visualization is an important feature of BI, and there are two main ways to use this feature. One is for analysts to interactively query and generate data visualization charts for subsequent analysis work. The other is for chart makers to pre-create analysis charts and integrate them into a dashboard, which is a page containing several or dozens of analysis charts which are then provided to analysts for analysis work. Due to the convenience of the second method for analysts and the ease of access control for administrators, it has become a commonly used method.

However, this usage method also has some shortcomings. When analysts browse analysis pages, if the required charts are far apart or not on the same page, there may be situations wherein they need to repeatedly drag up and down the page or constantly switch pages, increasing the workload and interrupting the analysis process. In addition, analysts may sometimes overlook certain key information and charts due to negligence during their analysis work, so a reminder mechanism is needed to reduce the probability of such situations occurring. Therefore, introducing recommendation models into the data visualization of BI can reduce the operational complexity of analysts, alleviate the situation of interrupted thinking, and inspire analysts based on the behavior of other analysts, reducing omissions in the analysis process and improving the analysis effect.
In terms of personalized recommendation, early personalized recommendation systems predicted user interests by modeling the frequently visited items, keywords of interest, or identities of users [2,3]. The continuous advancement of technology has made the design of recommendation models increasingly complex and effective. With the development of computer equipment and technology, deep learning models have become popular, which has also led to the development of recommendations in this direction. Deep learning methods are playing an increasingly important role in recommendation systems for generating low-dimensional embeddings of items such as images, text, and user personal information [4]. The irregularity of data, such as personalized user data or user access records, which have variable length, makes Convolutional Neural Networks (CNNs) and other neural networks unable to directly perform routine operations such as convolution and pooling. However, utilizing the structure of graphs has natural advantages in processing such data. Graph Neural Networks (GNNs) extend traditional neural networks to graphs and are a series of neural networks that learn feature patterns from graphs. Gori et al. [5] first used a Recurrent Neural Network (RNN) to compress the information of data nodes and proposed the concept of GNN. Due to their strong representational power and high interpretability, GNN-based methods have been widely welcomed in the field of recommendation, and many models have been proposed based on this foundation [6,7]. The most noteworthy among them is that of J Bruna et al., who applied the idea of a CNN to a graph and proposed the Graph Convolutional Network (GCN) [8], which become one of the most successful deep learning architectures for graph structure learning.

Due to the expected advantages of using recommendation models in BI’s visual analysis and the advantages of using graph-based models in solving these problems, this paper proposes an inductive heterogeneous graph recommendation model, PreferSAGE-SCF, for high-scoring items, which recommends analysis charts that analysts may be interested in based on their preferences. More specifically, the data displayed in the chart may change at any time, but the attributes of the chart (such as title, style, content, etc.) are fixed, so that it is feasible to use them to determine and describe the chart. In summary, the chart can be seen as a special item, and the purpose of this paper is to recommend these special items to users. The method based on GCN can predict the relationships between different items, and the inductive method and mini-batch meet requirements for running on computers with lower computing power. Therefore, this paper uses them as the basis for the model. It should be noted that the research in this paper focuses on the chart recommendation problem in the data visualization scenario of BI, as well as the problem of recommending high-scoring items for users in other similar situations. Therefore, further exploration is needed for specific integration and application methods. The PreferSAGE section of the model in this paper refers to PinSAGE [9] and makes improvements to it. Firstly, the sampling method of PinSAGE was extended, expanding its importance sampling based on node visits to consider both spatial and logical importance simultaneously. Secondly, in order to alleviate the problem of information loss caused by multiple convolution operations in the information aggregation process, skip connections [10,11] directly connecting to the outermost node features and corresponding pooling layers were added to the innermost aggregation process. The Slow-acting Collaborative Filtering (SCF) part of this model is an integrated method for the recommendation results of conventional collaborative filtering algorithms, which has a slow effect. The experimental results demonstrate that the PreferSAGE-SCF model proposed in this paper can achieve better performance compared to baselines in solving the tasks of analyzing charts’ recommendations for data visualization and recommending high-scoring items for users in similar situations.

The contributions are as follows:

1. An analysis was conducted on the requirements for recommendation models that can be applied to the data visualization of BI.
2. We proposed an inductive heterogeneous graph recommendation model for high-scoring items, PreferSAGE.
3. A collaborative filtering algorithm integration method, SCF, was designed with limited influence.

4. Experiments were conducted on a collected dataset and a public dataset, and the results demonstrated the effectiveness of the model in performing high-preference item recommendation tasks.

The remainder of the paper is organized as follows. In Section 2, related works are introduced. In Section 3, the characteristics and requirements of data visualization are explained, and the PreferSAGE-SCF model is constructed. In Section 4, experiments and evaluations are conducted. Finally, the conclusion, current relevant issues, and future research directions are presented in Section 5.

2. Related Work

BI and recommendation models are currently undergoing rapid development. As mentioned in the Introduction, they play an important role in an increasing number of applications. Since the purpose of the model proposed in this paper is to perform graph-based high-scoring item recommendation applied to BI and other similar scenarios, this section will focus on the relevant situations of BI and GNN as well as on popular recommendation algorithms related to the purpose of this paper or graph structure.

2.1. Business Intelligence

In recent years, with the rapid development of data storage and processing technologies such as data warehousing and data mining and the urgent requirements of enterprises in terms of performance management and enterprise diagnosis, BI has gradually become popular in various enterprises [12]. As BI gradually becomes popular, its role in the development of enterprises has also become increasingly significant. Therefore, research on BI has gradually received attention in recent years. So far, most research on BI has focused on business or implementation aspects, including how to build specific domain-specific BI architectures, or how to choose service providers. Here are several examples. L Wu et al. [13] designed a service-oriented and seamlessly integrated BI architecture to achieve simplified data delivery and low latency analysis; Wang C H [14] proposed a method for evaluating BI suppliers, providing a reference for enterprises in deciding how to choose BI suppliers. However, there is currently relatively little research on the specific usage process of BI, though it still arouses people’s interest. Drushku K et al. [15] established a collaborative recommendation system based on Markov models to reflect user interests in the search interaction process between users and BI data warehouses, suitable for creating visual analysis charts for interactive searches; Kretzer et al. [16] conducted detailed experiments and surveys on the impact of social interaction on users’ use of BI and confirmed its importance in BI. Based on the above research status, with the development of economy and technology, the importance of BI for enterprises is constantly increasing, and the attention that enterprises pay to BI is also growing. It is believed that in the future, BI may play a greater role in production and operation.

2.2. Graph Neural Networks

Graph Neural Networks (GNNs) are neural network algorithms designed for graph structured data. Gori et al. [5] first proposed neural networks applied to graphs. Subsequently, Scarselli et al. [17] further elaborated on GNNs. After learning from graphs, GNNs can be used to solve various machine learning problems related to graphs, including node classification (determining the category or label of a node based on its characteristics and neighboring features), graph classification (using the structural information of the graph and the information of each node to classify the entire graph, rather than classifying certain nodes or edges within it), link prediction (learning information about nodes and edges to predict whether there are potential connections or edges between nodes), community detection (dividing nodes into different clusters based on the closeness of their connections), and other more advanced usages. Although GNNs can effectively model and learn graph
structured data, there are still some limitations. Firstly, GNNs have lower computational efficiency and require more resources and longer training time. Secondly, GNNs mainly learn the features of nodes, while neglecting the learning of edge features. Thirdly, GNNs are greatly affected by the depth of convolution, and after multiple layers of convolution, node features may exhibit issues of over-smoothing and over-compression. Therefore, these limitations should be addressed in practical use to avoid significant impact on the results. In summary, with the passage of time, GNNs will become more powerful and diverse, and can be expected to be widely applied in more fields.

2.3. Recommendation Algorithms

2.3.1. Collaborative Filtering

In the field of recommendation, Collaborative Filtering (CF) [18] is a kind of classic algorithm. Its basic principle is to determine the degree of similarity between the recommended objects based on various pieces of similarity information, in order to recommend content that the recommended objects may like. If Collaborative Filtering is classified according to neighborhood-based data, it can be divided into User-Based Collaborative Filtering (UserCF) and Item-Based Collaborative Filtering (ItemCF). UserCF determines the user’s neighbors by calculating the similarity of access items between users, and recommends users based on their items’ access records. ItemCF determines the relevance of items by calculating the similarity of user visits between them, and recommends items based on the similarity between items that the users has visited and other items that have not been visited. There is a certain difference in the scope of application between these two methods. UserCF is mainly used to recommend items to users who share common interests and hobbies. Therefore, it has group and social characteristics. From a technical perspective, it is necessary to maintain a similarity matrix of users, which is suitable for situations where users are stable but items are updated quickly. ItemCF recommends items to users that are similar to their previous likes, thus better reflecting their interests. From a technical perspective, it is necessary to maintain a similarity matrix of items, which is suitable for situations wherein items are updated slowly. Overall, the effectiveness of using collaborative filtering alone is limited. If it is mixed with other types of algorithms, better results can be achieved.

2.3.2. Pattern Mining

Pattern mining is used to discover relationships and patterns in specified data. Overall, it is suitable for extracting effective content from a large amount of existing data, thereby achieving noise filtering, data cleaning, prediction, and recommendation. Pattern mining has the advantage of being easy to understand and implement, and has achieved a number of excellent results in the field of recommendation. For example, Cauteruccio et al. [19] proposed an extended and efficient framework for pattern mining called e-HUPM. The framework extends the classical High-Utility Pattern Mining (HUPM) [20] and provides modular ASP encoding of it. The pattern masks used in the data processing of this paper have a filtering function, which can extract valid patterns and make effective predictions. However, pattern masking is relatively fixed, so this requires more in-depth discussion. Similar to CF, if it is mixed with other types of algorithms, better results can be achieved.

2.3.3. Random Walk

Random walk algorithms are mainly used in graph structures, and the results of the walk can generate graph embeddings. Random walk algorithms can learn the interaction relationships between users or items nodes, and using vectors to represent nodes can effectively solve the problem of graph data sparsity. Especially when a graph involves billions of nodes/items, generating the nodes’ embeddings become more difficult, and random walk algorithms can effectively address this problem. The foundation of random walk algorithms is Word2Vec [21], which can convert words into vectors to represent and explore the relationships between words. DeepWalk [22] is an algorithm that combines
random walk and Word2Vec to apply to the graph. It uses the random walk to extract vertex sequences from the graph, and uses the same training approach as Word2Vec to ultimately obtain embeddings for each node. Node2Vec \cite{23} is an improved algorithm for DeepWalk. Compared to DeepWalk using unbiased random walks, Node2Vec uses biased random walks, similar to BFS and DFS. LINE \cite{24} takes into account the similarity between first-order and second-order nodes, trains them separately, and combines the two to form the final node embedding. Random walk algorithms are suitable for describing the spatial relationship between nodes, so they cannot effectively classify or recommend newly added nodes. The advantage of this algorithm compared to GCN is its faster training speed.

2.3.4. Recommendation Algorithm Based on GCN

On the basis of GNNs, the convolution operation was extended to graphs, resulting in GCN. Bruna et al. designed a graph convolutional model based on spectral theory, which is the origin of the GCN method. In addition to GCN, there are many algorithms for node embedding learning on graphs, such as matrix factorization or random walks (e.g., Deepwalk, LINE, Node2vec). However, GCN-based algorithms often outperform other algorithms in actual results, which has sparked interest in in-depth research on GCNs. Research on recommendation models based on GCNs is also becoming increasingly widespread and in-depth \cite{25}.

In the past, graph learning was generally based on transductive learning, which directly learned the embedding of each node on a fixed graph, and each only considered the current data. This form of graph learning requires all nodes to be determined from the beginning, and once the nodes or structure of the graph have changed, it is necessary to relearn the entire graph again, through, for example, Neural Graph Collaborative Filtering (NGCF) \cite{26} or a Light Graph Convolution Network (LightGCN) \cite{27}. Hamilton W et al. designed Graph SAmple and aggreGatE (GraphSAGE) \cite{28}, which changed this situation. GraphSAGE aggregates the features of neighboring nodes after sampling, and it has good generalization ability. It can directly provide representation learning for newly added nodes based on their neighbor aggregation, without the need to iterate the entire network. GraphSAGE is divided into three parts: node sampling, aggregation of node neighbors, and learning from the aggregated information. GCNs and GraphSAGE treat neighboring nodes equally. When aggregating, they usually sum the features of all neighboring nodes (i.e., the output of the previous layer of the network) without considering the importance of information from different neighboring nodes to the current target node. Therefore, incorporating attention mechanisms into GCN-like algorithms is a natural idea. Graph Attention Networks (GAT) \cite{29} propose to incorporate attention mechanisms into the neighbor information aggregation stage to achieve weighted aggregation of neighbor information based on its importance. Therefore, it can be regarded as an improved algorithm of GraphSAGE, and performs better than GraphSAGE in multiple tasks such as semi-supervised node classification and link prediction. In addition to using attention mechanisms, there are also many algorithms based on GCNs and their structures; alternatively, methods have been improved. For example, ImprovedSAGEGraph \cite{30} is an improved algorithm based on GraphSAGE. This algorithm uses two layers of convolution to aggregate features, with the first layer having an activation function of Tanh and the second layer having an activation function of RELU6. The paper which proposed this algorithm simultaneously proposes a hybrid two-step with a clustering-based recommendation model. Due to the fact that traditional GCN methods require operation on the entire graph during the training process, the size of graph has a significant constraint on the efficiency of a traditional GCN. In order to use the GCN method in recommendation systems involving billions of nodes/items, the PinSAGE algorithm was designed. PinSAGE is a highly successful improved algorithm of a GCN, which has been successfully applied in the industrial field. The PinSAGE algorithm is closely related to the GraphSAGE algorithm, and the two have significant similarities. The difference lies in the fact that the PinSAGE algorithm uses importance sampling instead of uniform sampling, and the aggregation stage takes into account the weight of edges. The
upper layer features are aggregated after passing through a fully connected layer instead of being directly aggregated.

2.3.5. Hybrid Recommendation Systems

Hybrid recommendation systems [31] are a research hotspot in recommendation systems; the term refers to mixing multiple recommendation algorithms to compensate for each other’s shortcomings in order to achieve better recommendation results and finally provide more accurate and personalized recommendation content to users. Combining collaborative filtering with other technologies is a common approach in hybrid recommendation systems, which can alleviate cold-start issues and improve various indicators of the final result.

3. Materials and Methods

3.1. The Characteristics and Requirements of Data Visualization

3.1.1. Data Characteristics

Generally speaking, visual analysis charts are created by personnel with certain industry experience, and their attribute information is relatively clear. The attributes that can be used to describe visual analysis charts include title, style, amount of data involved, interactivity, and others. It is common for multiple charts to use the same title, as their other properties may differ. The relationship between analysts and visual analysis charts can be seen as a binary relationship, known as “analysts–charts”. Using them can create a heterogeneous graph with two types of node that describes their relationship. Analysts usually do not have direct connections, and there is usually no direct connection between analysis charts (except for some charts placed on the same page, where these charts have a certain logical relationship and usually increase the probability of being accessed simultaneously). Analysts need to access specific charts due to job requirements, and compared to conventional movie or book recommendations, they have much less room for selection. Therefore, the selection results will also show a more concentrated trend, and the similarity between users will be higher. At the same time, visual analysis charts often do not have functions such as commenting or labeling that allow analysts to express their preferences in a rich way. In other words, there is usually no more auxiliary information to indicate the analyst’s attitude. Therefore, it is necessary to use clear information, such as the browsing time of charts or preferences expressed by analysts in advance, as a basis for determining analyst preferences. Moreover, analysts need to interact with many charts while working, inevitably including many invalid interactions. High-user-preference items can be described by high item ratings, and this is one of the reasons why the model proposed in this paper is aimed at recommending high-scoring items.

3.1.2. Requirements for the Recommendation Model

The recommendation model suitable for BI’s data visualization module has some special requirements. Firstly, the computer running the BI server is usually not a professional computer server, which restricts the model in terms of computational scale. That is, the training and prediction process of the model cannot use too many computer resources, and overly large and cumbersome models are not suitable. Secondly, as the work of chart makers continues, the analysis charts will often undergo changes, so the model must have inductive ability, and transductive models do not meet the conditions. Meanwhile, the recommendation process should have a fast speed, as excessive computation time can reduce the user experience. In addition, in order to adapt to constantly changing needs, the design of the model should have a certain degree of scalability; for example, the addition of new attributes to charts should not cause significant adjustments to the model structure.

3.1.3. The Focus of the Results

In practical use, the final result page presented to analysts should include both regular charts (such as those determined during dashboard creation) and recommended charts.
Due to the limited display size of computer screens, excessive recommendation results have little significance. That is, the more accurate and satisfying the recommendation results concentrated at the top of the overall recommendation results are, the better the recommendation effect of the model. This is the another reason that the model proposed in this paper aims to recommend high-scoring items.

3.1.4. Practical Computational Viability

Dashboards and charts are not blindly created. Each chart has its own clear meaning, so charts can be accurately defined and described, and chart makers can easily carry out this work. Due to the fact that the tasks of employees are determined, the data that need to be used during work are basically determined by the work tasks, and there is a logical relationship between charts in terms of tasks; thus, the recommendation model based on these data is determined to converge and can achieve effective results. When using graph-based models to learn these data, the training duration mainly depends on the number of item nodes. Early stopping is also acceptable, as it will only weaken the partial effect of the results, and the final model will still be valid and useful.

3.2. Overview of PinSAGE

PinSAGE [9] is a highly successful improved algorithm of a GCN which has been successfully applied in the industrial field. Traditional GCNs perform better when dealing with small graphs, but they are somewhat inadequate when dealing with large graphs because they require computation of the entire graph. In order to use the GCN method in recommendation systems involving billions of nodes/items, the PinSAGE algorithm was designed. PinSAGE is closely related to GraphSAGE. In terms of implementation details, the two algorithms are quite similar, but there are also some differences. In the sampling stage, GraphSAGE uses uniform and unbiased sampling, while PinSAGE uses importance sampling and records the number of visits to the sampled nodes as edges’ weight during sampling. In the aggregation stage, the upper-level features are aggregated after passing through the fully connected layer, taking into account the weight of the edges during the aggregation process. That is, edges with higher weights propagate more information. In addition, the paper [9] also proposes some new training techniques to improve performance. These features enable PinSAGE to have faster training speed, lower resource consumption, and better performance on large graphs compared to prior GCN methods.

3.3. PreferSAGE-SCF

PreferSAGE-SCF is an inductive, heterogeneous graph-based recommendation model that recommends potentially high-scoring content to users. The PreferSAGE-SCF model consists of two parts: PreferSAGE and SCF. PreferSAGE is a model that improves the sampling and aggregation process of the PinSAGE model. In addition to the original spatial importance, logical importance is added during the importance sampling process of neighboring nodes, and the aggregation process of PinSAGE is optimized. SCF is a collaborative filtering algorithm with limited influence, which improves the performance of the model by combining it with PreferSAGE recommendation results in a slow-acting manner. SCF’s main innovation lies not in collaborative filtering algorithms, but in the process of integrating results. The model proposed in this paper has lower hardware resource requirements and can adapt to partial changes in data without the need for retraining. At the same time, it has a certain degree of scalability. When the attribute structure of charts or items is adjusted, there is no need to adjust the model content, only retraining is needed. In order to provide a more unified description of the model, analysts will be referred to as “users” and charts will be referred to as “items” in the following text; user preference for an item will be determined based on their rating of the items. The architecture of the PreferSAGE-SCF model for high-scoring item recommendation applied to BI is depicted in Figure 1 and described in the this figure’s title.
Figure 1. The architecture of PreferSAGE-SCF. Firstly, we created a heterogeneous graph and a user similarity matrix using the data in the dataset, and then submitted them to PreferSAGE and SCF for calculation. After that, PreferSAGE used the graph structure to calculate the embeddings of nodes’ attributes, and calculated the top n items with the highest inner product based on the target user’s historical visited items; UserCF calculates the top n nearest neighbors based on user similarity and organizes the items that they have visited before. Finally, the results of the two parts are merged using a misaligned method SCF and obtain the final results.

3.3.1. Data Preprocessing

Data preprocessing is divided into two parts.

The first part of data preprocessing is to generate the structure of the graph. Due to the binary relationship between users and items, it is necessary to first read the relationship between nodes from the dataset and construct a heterogeneous graph structure. The constructed heterogeneous graph includes two types of nodes, user and item, and one type of edge, that is, watched. After creating a heterogeneous graph structure, it is necessary to read the data (title, style, etc.) of items from the dataset and associate them with nodes, then read the relationship data (item scores) between the two type of nodes and associate it with the connecting edges of users and items.

The second part of data preprocessing is to generate a similarity matrix for collaborative filtering. This paper uses similarity between users. We calculated the similarity between users based on their items access and ratings, recorded the top n neighbors with the highest similarity to each user, and stored them in a matrix for future use.

3.3.2. PreferSAGE Sampling

The neighbor node sampling method used in this paper is importance sampling, which is achieved through weighted random walks between nodes. Different from the true neighbors on the graph, the importance sampling defines the top n nodes that have the greatest influence on the starting nodes as neighboring nodes. Algorithms such as GCN and GraphSAGE are based on the sampling of real neighbors, only sampling and aggregating the real neighbors on the graph structure. However, using importance sampling can identify relatively far but more important nodes and treat them as neighboring nodes for sampling and aggregating. This method effectively improves the propagation efficiency of node features. The PinSAGE algorithm uses the normalized number of hits of nodes during random walks as the basis for importance sampling. In this paper, the idea proposed by PinSAGE is defined as the spatial importance of nodes (not considering user features), and the authors believe that the importance of nodes is considered to be determined by both spatial and logical importance. Spatial importance refers to whether a node is spatially important. If some nodes belong to key nodes on the walking path or have connecting edges with many other nodes, they are more easily accessed during random walks, indicating that these nodes are spatially important. Logical importance is a characteristic of locality and relativity between nodes. Simply put, when determining the logical importance of node B to node A, it depends on whether the preference weight from node A to another node B is relatively high compared to the other connecting nodes of node A. If it is relatively high, then node B has a higher logical importance to node A. The paper [32] mentions that
a biased random walk relies on user features, but it has the problem of dynamically setting the walk probability for each user during the walk, resulting in performance issues and a lack of suitability for this paper. The random walk in this paper is suitable for situations wherein there is no clear user preference. It calculates and sets the walk probability for an item node’s relationship at one time, which can improve the final result without reducing learning efficiency. In the process of sampling neighboring nodes using random walks, logical importance is reflected when the current node determines the direction of the next walk, while spatial importance is reflected when an organized walk results. For each node (including item nodes and user nodes), the calculation formula for the probability of selecting the next node is shown in Formula (1):

\[
P^{(k)}_{\text{next}} = \frac{\omega^{(k)}_{\text{item}}}{\sum_{n\in N} \omega^{(n)}_{\text{item}}}
\]

(1)

Among them, \(P^{(k)}_{\text{next}}\) is the probability of the current node accessing neighboring node \(k\), \(\omega^{(k)}_{\text{item}}\) is the preference weight between the current node and node \(k\), and \(\sum_{n\in N} \omega^{(n)}_{\text{item}}\) is the sum of preference weights from the current node to each adjacent node. The formula for calculating preference weights in this paper’s model is shown in Formula (2):

\[
\omega_{\text{item}} = \begin{cases} 
\left( \frac{s_{\text{item}}}{\max(s_{\text{item}})} \right)^2, & s_{\text{item}} \geq B_s \\
0, & s_{\text{item}} < B_s 
\end{cases}
\]

(2)

Among them, \(\omega_{\text{item}}\) is the current user’s preference weight for a certain item. \(s_{\text{item}}\) is the user’s rating on the item, \(\max(s_{\text{item}})\) is the maximum historical rating of the user, and \(B_s\) is the boundary value of score validity. If it is less than the boundary value, the user’s rating on the item is considered to be 0. The formula is in square form, used to increase the preference weight difference of the items and make the difference in random walk probability more significant. The schematic diagram of nodes sampling is shown in Figure 2.

![Figure 2](image)

Figure 2. Sampling process of nodes, assuming that the number of nodes sampled in each layer is 2 and the preference weight range is from 0 to 1. White nodes represent users, green nodes represent positive sampling nodes, red nodes represent negative sampling nodes, and gray nodes represent low sampling probability nodes.

Using the above method for neighbor sampling can enable the starting node to acquire more knowledge from other nodes with higher importance, and the amount of knowledge obtained is related to the importance of the node. When calculating recommended items,
using high-scoring items as benchmark items can sequentially recommend other high-scoring items from high to low.

In summary, the specific sampling process is described below. In order to speed up training efficiency while reducing resource consumption, this paper uses mini-batch sampling. For positive sampling, we first randomly selected the number of item nodes that match the batch number of query nodes. Then, we used a random walk with preference probability and a walk length of 2 to select the next item node with a connecting edge to the query nodes. Finally, the nodes obtained from random walks one-to-one with the query nodes were used to form positive sample node groups. For negative sampling, the model in this paper uses the Random Negative Sampling (RNS) method, which randomly selects nodes from all nodes and forms negative sample groups corresponding to one-to-one query nodes. Random negative sampling has a high sampling efficiency, and the sampling process does not introduce any other biases. After generating positive and negative sample groups, all nodes in these two groups are sampled according to the method described above, and the final sampling result is generated.

3.3.3. PreferSAGE Aggregation

After sampling positive and negative samples, as well as neighboring nodes, the next step is to aggregate neighbors’ information. PreferSAGE uses local feature aggregation, which aggregates subgraphs generated by sampled nodes. This paper designs PreferSAGE as a three-layer nodes features aggregation structure, and the aggregation process refers to the PinSAGE algorithm, with some improvements made. In order to alleviate the problem of information loss caused by multiple convolution operations in the information aggregation process, a skip-connection connecting the outermost nodes features was added before the final activation function, retaining the detailed information in the original input data and making it easier for information to propagate backwards. The aggregation process of the PreferSAGE model is shown in Figure 3, and the algorithm flow is shown in Algorithm 1.

**Algorithm 1** PreferSAGE aggregation

**Input:** Nodes embedding $z_u$, neighbors weight $\alpha$, aggregate function $\gamma$

**Output:** New nodes embedding $z_u^{(1)}$

1: $n_u^{(3)} \leftarrow \gamma(ReLU(Q_2 \cdot dropout(z_u^{(3)}) + q_2), \alpha)$
2: $z_u^{(2)} \leftarrow ReLU(W_2 \cdot dropout(concat(z_u^{(2)}, n_u^{(3)})) + w_2)$
3: $z_u^{(2)} \leftarrow z_u^{(2)} / \| z_u^{(2)} \|_2$
4: $n_u^{(3)} \leftarrow \gamma(ReLU(Q_1 \cdot dropout(z_u^{(2)}) + q_1), \alpha)$
5: $z_u^{(1)} \leftarrow ReLU(W_1 \cdot dropout(concat(z_u^{(1)}, n_u^{(2)})) + w_1 + pooling(z_u^{(3)}))$
6: $z_u^{(1)} \leftarrow z_u^{(1)} / \| z_u^{(1)} \|_2$

**Figure 3.** The aggregation process of the PreferSAGE model.
Node embedding are obtained by summing the node features embedding. The nodes embedding of the third layer go through a dropout layer and are then transformed using a fully connected layer and activated. At this point, the calculation of the third layer nodes is completed (line 1). Subsequently, the results are aggregated using an aggregation function with weights normalized by the number of hits, which is called importance pooling [9]. This method can enhance the influence of important node information during propagation. Afterwards, the aggregated result \( n_u^{(3)} \) is concatenated with the nodes embedding of the second layer through a concatenate operation [33]. The operation result is transformed and activated using another fully connected layer after passing through a dropout layer (line 2), and the activated features are normalized (line 3) to obtain the nodes embedding of the second layer. The information propagation process from the second-layer nodes to the first-layer nodes is similar to before, but the difference is that it needs to sum the original nodes embedding of the third layer before the final activation function, and then the activation operation is performed (line 5). Finally, the activated results are normalized (line 6), and the entire aggregation process is completed.

Due to the inconsistency between the number of nodes in the third layer and the number of nodes in the first layer, direct operations cannot be performed. Therefore, before aggregating the features of the first layer and the third layer, a pooling layer needs to be added to perform pooling operations on the features of the third-layer nodes. The optional forms of pooling layers include average pooling, maximum pooling, etc., and this paper uses average pooling in experiments.

### 3.3.4. PreferSAGE Loss Function

The model in this paper uses the Max-margin-based loss function. It does so to maximize the inner product of the embedding between the positive sample nodes in the positive sample group. At the same time, we try to ensure that the inner product between the nodes in the negative sample group is as small as possible compared to the inner product of the positive samples. The specific formula is shown in Formula (3). Among them, \( P_n(q) \) represents the negative example distribution of item \( q \), and \( \Delta \) represents the margin hyperparameter.

\[
J_g(z_q z_i) = E_{w_k \sim P_n(q)} \max \left\{ 0, z_q \cdot z_{w_k} - z_q \cdot z_i + \Delta \right\} \tag{3}
\]

### 3.3.5. SCF and Final Recommendation Results

Slow-acting Collaborative Filtering (SCF) is a collaborative filtering integration method with limited influence designed in this paper. Its main body is a conventional collaborative filtering algorithm, with the main difference being the integration method, which combines misaligned results to give it a slow-acting effect. For the PreferSAGE model in this paper, SCF plays a role in optimizing the results. Compared to the rigid proportional merging of results, this method does help further improve various indicators. This slow-acting effect can also to some extent solve the problem of newly added item nodes not being recommended by a collaborative filtering algorithm, enabling the inductive feature of PreferSAGE to be effectively utilized.

From the experimental results of various non-neural network-based collaborative filtering algorithms, we can see that the effectiveness of these algorithms is often weaker than that of neural network-based algorithms. Meanwhile, the hit items of neural network algorithms are usually concentrated in the head, and the density of hit items decreases as they rank lower. After the experiments, it was found that if the calculation results of neural network algorithms are directly combined with the results of collaborative filtering algorithms, the performance is not ideal in many cases, and it is also difficult to adjust parameters. If we can find a critical point in the middle place and make the results of the collaborative filtering algorithm work from here, we should be able to improve the model’s performance to a certain extent. The method used in this paper’s experiments is to manually adjust the position of this critical point after observing the results. After finding
this critical point, the first \( n \) items of PreferSAGE are retained as the first \( n \) items of the prediction results. Then, the first \( k - n \) items of the PreferSAGE prediction results are fused with the collaborative filtering results of user similarity using the weighted voting method, and the first \( k - n \) items from the fusion result are selected and added to the prediction results to generate the final prediction result. A schematic diagram of the fusion process is shown in Figure 4.

Figure 4. Fusion of results, assuming the critical point is 3 (beginning from 0) and the goal is to recommend five items to the user. Green represents the current user, purple represents recommended items that do not participate in collaborative filtering result fusion, white represents recommended items that participate in collaborative filtering result fusion, and gray represents placeholders.

The collaborative filtering method used in this paper’s experiments is based on user collaborative filtering using the cosine similarity method to calculate the similarity. Based on the calculation results, we obtained the highest rated \( m \) historical access items from \( n \) users with the highest similarity between users, attached these items with user similarity, and returned them to the current user as collaborative filtering results. The formula is as follows:

\[
\cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2} \sqrt{\sum_{k=1}^{n} x_{2k}^2}} \tag{4}
\]

4. Experiments

4.1. Datasets and Evaluation Metrics

The first experimental dataset used in this paper, DA, comes from our survey of the big data analysis department of a manufacturing enterprise. Due to the lack of built-in data collection capabilities in the data visualization used by the enterprise, the survey method used was to find offline statistics on the recent usage of several visualization analysis dashboards by users. The title, style (chart representation elements such as lists, histograms, line charts, etc.), and interactivity (whether data filtering can be performed, whether drilling is allowed, etc.) of the analysis charts included in the dashboard were used as chart attributes, and the recent chart usage of users were used as rating data. After sorting and enhancing results, the final dataset contained 2508 rating records of 344 charts from 50 users. The distribution of user charts access is shown in Figure 5.

Due to the fact that the model proposed in this paper can also be used for other similar tasks of recommending items with high user preferences, simulation experiments can be conducted using public datasets with user preferences or ratings to verify the effectiveness and universal applicability of the model to larger datasets. This paper selects the “ml-1m” [34] dataset as the second experimental dataset, which was published by GroupLens Labs and belongs to the MoveLens dataset series. It contains 1,000,209 rating records of 3883 movies from 6040 users. The structure of this dataset is relatively simple and easy to handle, while meeting the relevant conditions of the model in this paper; thus, this dataset
is suitable. Due to the purpose of the model in this paper being to recommend high-scoring items to users, a biased test sample extraction method was used to randomly select 30% of the items with the highest score of 40% or above from each user in two datasets, to be used as test data.

![Distribution of visits](image)

**Figure 5.** Distribution of visits in dataset DA.

From the perspective of users as a whole, how many users are hit by the correct results is an important factor that affects their experience. If the hit rate of the model is too low, it will result in some users not receiving any help. For individual users, the accuracy of chart recommendations and the ranking of recommendation results are the most influential factors in their experience. The accuracy determines whether chart recommendations have practical effects, and the correct ranking of recommendation results can greatly improve the user experience. As a contradictory and complementary indicator of accuracy, it is also valuable to consider the recall indicator.

Based on the above requirements, this paper uses the widely used evaluation metrics, including Hit, F-score, and NDCG. Hit represent the probability of actual results appearing in the predicted results, that is, whether each user’s predicted results contain hit items. If any item hits, it is 1; otherwise, it is 0. The higher the value, the higher the Hit rate of the model. As shown in the formula, where $\Pi$ is the indicator function, if the condition of this function is 1, the return value of this function is 1; otherwise, it is 0.

$$\text{Hit} = \frac{1}{N} \sum_{i=1}^{N} \Pi(\text{rank}_i \leq k)$$ (5)

The precision indicator represents the proportion of true positive cases in the predicted positive cases, and the recall indicator represents the proportion of true positive cases in the predicted results among all true positive cases. The larger these two values, the better the model performance. F-score is a type of evaluation indicator that harmonizes precision and recall. The commonly used evaluation indicators for F-score include F1-score, F0.5-score, and F2-score. Due to the higher accuracy requirements for recommendations in data visualization, F0.5-score was chosen as one of the evaluation indicators in this paper. The formulas are as follows, where $TP$ represents true positive, $FP$ represents false positive, and $FN$ represents false negative:

$$\text{Precision} = \frac{TP}{TP + FP}$$ (6)

$$\text{Recall} = \frac{TP}{TP + FN}$$ (7)
\[ F - score = \left(1 + \beta^2\right) \frac{Precision \times Recall}{\beta^2 \times Precision + Recall} \] (8)

The NDCG indicator stands for Normalized Discounted Cumulative Gain, which not only evaluates the accuracy of recommendation results, but also evaluates whether the order of recommendation results is more reasonable. The NDCG indicator consists of two parts, DCG and IDCG, and DCG is composed of CG and the loss value at the current position of the result. CG is the cumulative gain, which is the sum of the correlation scores of the recommendation results; DCG is the Discounted Cumulative Gain, which means dividing each CG result by the discount value at the current position, so that the higher ranked result has a greater impact on the final result; and IDCG is the maximum DCG value under ideal conditions. The larger the value of this indicator, the better the model’s performance. The formula is as follows.

\[ DCG = \sum_{i=1}^{k} \frac{rel_i}{\log_2 (i+1)} \] (9)

\[ IDCG = \frac{|REL|}{\log_2 (|REL|+1)} \] (10)

\[ NDCG = \frac{DCG}{IDCG} \] (11)

4.2. Implementation Details

All experimental programs in this paper were written using the PyTorch framework, with an Intel Core i7-10870H CPU, NVIDIA GeForce RTX 3070 GPU, and 16 GB RAM. This paper selects several recommendation models that can be used for this situation as the baselines, including UserCF, ItemCF, GraphSAGE, GAT, ImprovedSAGEGraph, and PinSAGE. These models have all achieved good performance in their respective fields, and they are relatively easy to implement. For all models, the epoch is 50. The batch-per-epoch is 500 in DA and 2000 in ml-1m; the batch size is 128, the number of neighbor samples is 3, the node aggregation level is 3, the embedding size is 64, and the learning rate is 0.0001. GAT uses a two-headed attention mechanism. To ensure consistency in experimental conditions, ImprovedSAGEGraph used an item–user cluster but did not use a year–user cluster. The sampling random walk length for PinSAGE and PreferSAGE is 4. The critical point between the CF results and the calculated results of the graph embedding of the PreferSAGE-SCF model is 4 in DA and 3 in ml-1m. During the experiment, we used five items with the highest user ratings as the standard items for prediction.

4.3. Experimental Results

The experimental results of each model in the dataset DA are shown in Table 1. In this dataset, the four indicators of hit, F0.5-score, and NDCG predicted using the model proposed in this paper are 0.020, 0.083, and 0.076 higher than those of the baselines. This demonstrates the effectiveness of PreferSAGE-SCF in recommending BI’s visualization analysis charts.

The experimental results of each model in the dataset ml-1m are shown in Table 2. In this dataset, the four indicators of hit, F0.5-score, and NDCG predicted using the model proposed in this paper were higher than those of the baseline by 0.045, 0.019, and 0.023, which further proves the effectiveness of the PreferSAGE-SCF model in recommending high-scoring items, especially when the recommended result set is small.
Table 1. Experimental results on dataset DA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserCF</td>
<td>0.600</td>
<td>0.172</td>
<td>0.170</td>
</tr>
<tr>
<td>ItemCF</td>
<td>0.160</td>
<td>0.037</td>
<td>0.029</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>0.980</td>
<td>0.554</td>
<td>0.621</td>
</tr>
<tr>
<td>GAT</td>
<td>0.980</td>
<td>0.561</td>
<td>0.623</td>
</tr>
<tr>
<td>ImprovedSAGEGraph</td>
<td>0.920</td>
<td>0.565</td>
<td>0.631</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.960</td>
<td>0.564</td>
<td>0.622</td>
</tr>
<tr>
<td>PreferSAGE-SCF</td>
<td>1.000</td>
<td>0.648</td>
<td>0.707</td>
</tr>
</tbody>
</table>

Table 2. Experimental results on dataset ml-1m.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserCF</td>
<td>0.377</td>
<td>0.062</td>
<td>0.093</td>
</tr>
<tr>
<td>ItemCF</td>
<td>0.513</td>
<td>0.098</td>
<td>0.145</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>0.609</td>
<td>0.148</td>
<td>0.261</td>
</tr>
<tr>
<td>GAT</td>
<td>0.617</td>
<td>0.150</td>
<td>0.266</td>
</tr>
<tr>
<td>ImprovedSAGEGraph</td>
<td>0.608</td>
<td>0.151</td>
<td>0.267</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.621</td>
<td>0.155</td>
<td>0.268</td>
</tr>
<tr>
<td>PreferSAGE-SCF</td>
<td>0.666</td>
<td>0.174</td>
<td>0.291</td>
</tr>
</tbody>
</table>

By observing the results of the above two sets of experiments, we can draw the following conclusion. The performance of UserCF and itemCF is not very stable, and their results are greatly influenced by the amount and distribution of data. Meanwhile, the content of work is often not determined by personal preferences, which affects the performance of ItemCF. The performance of GraphSAGE is far superior to non-neural-network-based collaborative filtering algorithms, but due to the indiscriminate feature propagation and aggregation of nodes, the final results are slightly worse. GAT has added an attention mechanism to GraphSAGE, thereby improving various indicators of the final result. ImprovedSAGEGraph not only utilizes node information but also uses user clustering information, resulting in better performance on two datasets. However, it is greatly affected by data volume and distribution and involves a large number of hyperparameters. This has resulted in the difficulty of making appropriate adjustments to these parameters. PinSAGE utilizes the spatial importance of nodes for sampling, which is essentially a special attention mechanism for graph space. The PreferSAGE-SCF model proposed in this paper reinforces this point of PinSAGE, significantly bringing the inner product between nodes of these high-rated items closer, while taking into account the knowledge of neighboring users, thus achieving better performance compared to other models.

However, there are still areas for improvement in the model proposed in this paper. The model directly divides titles based on words, lacking understanding of the overall meaning of titles. In addition, the critical point position of SCF still need to be manually adjusted, so automatic determination of this position is a problem that needs to be considered. These issues can be further studied and improved in the future.

4.4. Ablation Experiments

In order to better evaluate the effectiveness of each part of the model, comprehensive ablation experiments were conducted on two datasets for the model in this section. Firstly, we evaluated the two parts of PreferSAGE-SCF—PreferSAGE and SCF. Due to the inability of SCF to function independently, this paper divides PreferSAGE-SCF into three parts: PreferSAGE, PreferSAGE-UserCF, and PreferSAGE-SCF. Among them, PreferSAGE does not contain collaborative filtering; PreferSAGE-UserCF uses standard collaborative filtering methods. The experiments on dataset DA are shown in Table 3, and the results on dataset ml-1m are shown in Table 4. From the results, it can be seen that the SCF part is helpful in improving the predictive performance of PreferSAGE. After comparing these results with
Tables 1 and 2, we can also see that retaining only the PreferSAGE part can still achieve good results for the model.

Table 3. Model ablation experiments results on dataset DA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreferSAGE</td>
<td>1.000</td>
<td>0.633</td>
<td>0.698</td>
</tr>
<tr>
<td>PreferSAGE-UserCF</td>
<td>0.960</td>
<td>0.516</td>
<td>0.605</td>
</tr>
<tr>
<td>PreferSAGE-SCF</td>
<td>1.000</td>
<td>0.648</td>
<td>0.707</td>
</tr>
</tbody>
</table>

Table 4. Model ablation experiments results on dataset ml-1m.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreferSAGE</td>
<td>0.665</td>
<td>0.165</td>
<td>0.288</td>
</tr>
<tr>
<td>PreferSAGE-UserCF</td>
<td>0.666</td>
<td>0.170</td>
<td>0.288</td>
</tr>
<tr>
<td>PreferSAGE-SCF</td>
<td>0.666</td>
<td>0.171</td>
<td>0.291</td>
</tr>
</tbody>
</table>

The next step is to conduct ablation experiments on the various components of the PreferSAGE algorithm. This paper divides PreferSAGE into four parts of the experiment, including No-Skip-Connection, No-Reference, and PreferSAGE. Among them, No-Skip-Connection does not use skip connections in the process of information aggregation; No-Prefer does not contain preference weights, that is, only spatial importance is used in importance sampling. The experiments on dataset DA are shown in Table 5, and the results on dataset ml-1m are shown in Table 6.

Table 5. PreferSAGE ablation experiment results on dataset DA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Skip-Connection</td>
<td>1.000</td>
<td>0.627</td>
<td>0.691</td>
</tr>
<tr>
<td>No-Prefer</td>
<td>1.000</td>
<td>0.565</td>
<td>0.646</td>
</tr>
<tr>
<td>PreferSAGE</td>
<td>1.000</td>
<td>0.633</td>
<td>0.698</td>
</tr>
</tbody>
</table>

Table 6. PreferSAGE ablation experiment results on dataset ml-1m.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit</th>
<th>F0.5-Score</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Skip-Connection</td>
<td>0.631</td>
<td>0.164</td>
<td>0.286</td>
</tr>
<tr>
<td>No-Prefer</td>
<td>0.634</td>
<td>0.159</td>
<td>0.281</td>
</tr>
<tr>
<td>PreferSAGE</td>
<td>0.651</td>
<td>0.165</td>
<td>0.288</td>
</tr>
</tbody>
</table>

5. Conclusions

The data visualization of BI has significant value in the business field and can effectively guide industrial production and plan business goals. If we can solve some of the inconveniences that analysts encounter when using the data visualization of BI, we can effectively help to improve their analysis results. Due to the consideration of the characteristics of data visualization, the recommendation models applied to them should pay more attention to recommending high-scoring items compared with traditional recommendation models. The model proposed in this paper has been specifically designed based on the characteristics of analysts when using BI for data visualization. Node sampling uses importance sampling, which is performed through spatial importance (spatial key nodes) and logical importance (preference weights), and a propagation and aggregation method combining layer-by-layer feature transmission and a skip connection is used. Combined with a slow-acting collaborative filtering algorithm, the PreferSAGE-SCF model is established. The experiment proved that the model has good performance in recommending data visualization charts and other similar scenarios that require inductive recommendations based on user preferences.
Certainly, there are also some inconveniences in the practical application of this recommendation model applied to BI. The first issue is data collection. In order to determine users’ preferences during the experimental phase, an eye tracker can be used; charts can be displayed one by one or offline surveys can be conducted. They are not practical for actual use. Therefore, the determination of charts that users are using and collecting usage data is an issue that needs to be considered. This paper proposes that starting from the current position of the mouse and the position of the chart on the current screen can roughly determine the user’s usage, or the BI backend can automatically record the user’s click on the charts, which is another option. The second issue is the fusion of recommendation results. The following points need to be considered: displaying on a separate dashboard or integrating with a regular dashboard, where to integrate into the page, in what form to display, and so on. These points require further detailed investigation and design, and finding better hybrid recommendation models is also a direction that requires further research. If the recommendation results are combined with charts generated by other methods to dynamically adjust or generate the analysis page after correctly handling the above issues, they will have practical significance for improving the final analysis effect of analysts.

Author Contributions: Conceptualization, S.T., Y.Y. and L.Y.; investigation, S.T.; software, S.T.; writing—original draft preparation, S.T.; writing—review and editing, Y.Y. and L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Guangxi innovation-driven development project “Internet + Engine Intelligent Manufacturing Platform R&D and Industrialization Application Demonstration” (Guike AA20302002), and the Guangxi Science and Technology Base and Talent Special “China-Cambodia Intelligent Manufacturing Technology Joint Laboratory Construction” (Guike AD21076002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy. Publicly available datasets were analyzed in this study. These data can be found here: https://grouplens.org/datasets/movielens (accessed on 15 August 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

References