applied sciences

Article

Incorporating Similarity Measures to Optimize Graph Convolutional Neural Networks for Product Recommendation

Wafa Shafqat and Yung-Cheol Byun *

Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; wafashafqat@jejunu.ac.kr
* Correspondence: ycb@jejunu.ac.kr

Abstract: With the ever-growing amount of online data and information, recommender systems are becoming overwhelmingly popular as an adequate approach for overcoming the challenge of information overload. Artificial Intelligence (AI) and Deep Learning (DL) have accumulated significant interest in many research areas, and recommender systems are one of them. In this paper, a Graph Convolutional Neural Network (GCNN)-based approach was used for online product recommendation. Graph-based methods have undergone substantial consideration for several recommendation tasks, with effective results. However, handling the computational complexities and training large datasets remain a challenge for such a model. Even though they are useful, the excessive measure of the model's boundaries obstructs their applications in real-world recommender frameworks to a great extent. The recursive way of generating neighbor node embeddings for each node in the graph makes it more challenging to train a deep and large GCNN model. Therefore, we propose a model that incorporates measures of similarity between two different nodes, and these similarity measures help us to sample the neighbors beforehand. We estimate the similarity based on their interaction probability distribution with other nodes. We use KL divergence on different probability distributions to find the distance between them. This way, we set a threshold criterion for neighbor selection and generate other clusters. These clusters are then converted to subgraphs and are used as input for the proposed GCNN model. This approach simplifies the task of neighbor sampling for GCNN, and hence, we can observe a significant improvement in the computational complexity of the GCNN model. Finally, we compared the results with those for the previously proposed OpGCN model, basic GCNN model, and other traditional approaches such as collaborative filtering and probabilistic matrix factorization. The experiments showed that the complexity and computational time were decreased by estimating the similarity among nodes and sampling the nodes before training.

Keywords: recommendations; probability distribution; graph convolutional neural networks; product recommendation system; Kullback–Leibler (KL) divergence; deep learning

check for updates

Citation: Shafqat, W.; Byun, Y.-C. Incorporating Similarity Measures to Optimize Graph Convolutional Neural Networks for Product Recommendation. *Appl. Sci.* **2021**, *11*, 1366. https://doi.org/10.3390/app11041366

Academic Editor: Ángel González-Prieto
Received: 8 December 2020
Accepted: 29 January 2021
Published: 3 February 2021

1. Introduction

Lately, recommender systems have been at the cutting edge and are becoming overwhelmingly popular in different fields. These systems are enticing a large community of researchers because of their noteworthy applications. Recommender systems primarily screen, organize, and proficiently convey significant data to ease the issue of data over-burden, which has become a possible issue for numerous internet clients. Users are presented with enormous information and choices to make every day, which complicates the decision-making process. Therefore, these intelligent systems aim to rescue such users from the daily fatigue of selection.

Recently, eCommerce has been becoming increasingly saturated by online customers, and companies are researching possible ways to develop trust and capture customers' loyalty by personalizing the shopping experience. One of the core approaches to dealing with advancement is through the implementation of product recommendations. Research shows that 56% of customers return to a website that provides personalized product recommendations that meet customers' expectations, and 75% of today's generation prefers a customized online shopping experience [1,2]. The COVID-19 pandemic has accelerated and shifted consumers more towards the digital world. This online shift has made it easier to control the outbreak of COVID-19 and maintain social distancing and has enabled firms to extend their business and strengthen the economy. Necessities and daily use products have become more demanded than luxurious goods in the United States, where online grocery sales have increased by 16%, and in the EU, they have increased by 30% [3]. South Korea, remarkably, has seen an increase of 90% for the online sale of food products during the pandemic. According to South Korea's Ministry of Trade, Industry, and Energy (MOTIE), sales for offline markets took the worst hit, with a decrease of 21.4% in sales, whereas online markets with personalized recommendation engines surged by 92.5% [4]. Multiple surveys suggest that the shift towards the digital world and online shopping will outlast the COVID-19 pandemic and requires the development of significant recommendation engines to stay in the competition for and retain customers [2].

Due to the massive availability of data and product information on the Internet, it becomes difficult to extract meaningful information to eliminate the tiresome decision-making process. Recommendation systems filter information relevant to the user based on a specific predefined set of rules. The recommendation engine uses data related to customer click patterns, the number of views for an item, reviews, ratings, the popularity of an item, the delivery time for the product, and many other factors to automatically present the list of the most related products without the customer needing to search for them. Recommendation systems are being used not only for products but also for endorsing services in the fields of banking, telecom, retail, and media. Collaborative filtering, one of the well-known product recommendation techniques, considers that customers having similar preferences for items will have identical click patterns and will end up buying many similar products. However, the sparsity of data requires more intelligent recommendation systems that consider intricate details to identify users' behavioral patterns and measure the similarity between users' activities.

This paper proposes a session-based Graph Convolutional Neural Network (GCNN)-based product recommendation model that incorporates measures of similarity between multiple users to produce an optimized, accurate, and intelligent recommendation system. The input data for the recommendation system are acquired from an e-shopping mall known as eJeju mall. They consist of the users' historical data, and we utilize these data to form session-based graphs that exhibit the users' dynamic behavior. The session graphs are directed graphs generated based on each user's sequence of clicks in each session. We feed the session graphs to the embedding layer that provides each graph's embedding that is applied as input to the graph convolutional neural network. Figure 1 shows the product IDs (P9, P3, P1, P45, P10) representing the click sequences of user n at a particular session and how the session graph is generated according to the click pattern. The double circle indicates the product brought by the user in that specific session. We also consider the number of paths to an item and give weightage to a product with multiple paths, indicating that the product must be of user interest. The methodology is repeated for each user and session. Therefore, our model captures complex user behavior within multiple sessions and for various users. After processing the data and generating the session graphs, our primary focus in the proposed work was to obtain the probability distribution and sample the users of similar behavior. We measure the similarity through distance metric learning and KL divergence, and finally calculate the similarity score. We regenerate and update the graphs based on the similarity scores and train our graph convolutional neural network model for prediction and recommendation.

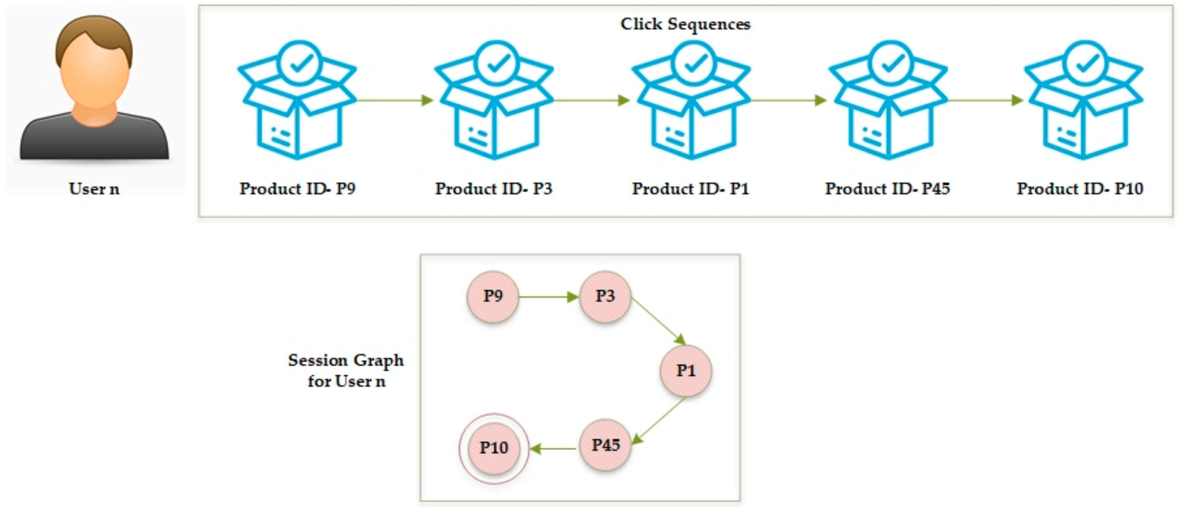


Figure 1. Session graphs of users.

The main contributions of the work can be summarized as follows:

- We propose a GCNN model for product recommendation that is based on the similarity among different nodes.
- We use the item click probability of a user to find their similarity with other users.
- We perform neighborhood sampling based on the similarity scores calculated through the KL divergence probability distribution formulation.
- The GCNN model is simplified by using neighbor modeling as a preprocessing step. The same neighbor modeling and embeddings learned are propagated through all the layers rather than the recursive aggregation of the neighbor's data.
- We conducted experiments on two different datasets and used other proposed models to verify our proposed GCNN model's performance. The results show that the proposed model performs better than the existing GCNN models and some traditional recommendation systems.

The rest of this paper includes related work in Section 2, the proposed method in Section 3, a data introduction in Section 4, the results in Section 5, the evaluation setup in Section 6, and the conclusions in Section 7.

2. Related Work

In this section, we explore relevant and recent work performed in the same domain. We have divided this section into two subsections. In the first part, we focus on the deep learning- and graph CNN-based approaches in recommendation systems. Then, we cover the work in the area of neighbor sampling and similarity estimation.

Recommendation systems are evolving with each passing day because of their constant necessity and importance in our daily lives. There have been many approaches proposed for recommendation systems, such as collaborative filtering [5], hybrid approaches [6], and deep learning-based approaches [7–10]. In [11,12], a collaborative filtering approach is considered in the viewpoint of Autoencoder. In [13], a context-aware deep learning-based model is proposed for paper citation recommendations. There have been multiple techniques derived from the traditional approaches; for example, in [14], a multilayer perceptron based approach known as Neural Collaborative Filtering (NCF) is used; in [15], DeepFM is introduced; and similarly, in [16], another multilayer perceptron approach is proposed, referred to as Collaborative Metric Learning (CML). Long Short-term Memory (LSTM) is also being used to learn different time-based or sequential links between other nodes [17].

Graph-based approaches are comparatively new in recommendations. GCNN models generally first learn the embeddings of nodes such as item and user embeddings based on the content knowledge and structure of graphs [18]. In many studies, GCNN-based approaches are used for feature extraction [17–19]. In [19], PinSage is introduced, which is a GCNN-based algorithm. It uses random walks and convolutions for the node embeddings. Besides this, variants of the GCNN model can be found in many other fields such as link prediction [20,21], node classification [22–24], and recommendations [25,26].

Some research studies have utilized multilayer perceptron layers for prediction tasks such as IntentGC [26,27]. GraphRfi is proposed to generate robust predictions of ratings and the detection of fraudsters [28]. For item recommendation, contextual information is used by exploiting graph attention networks to understand social impacts on users' preferences [29]. In [30], we proposed a model that optimizes the GCNN model's performance for product recommendation by using objective functions that maximize the weights of desired features and minimize the weights of undesired features [30]. It aims to simplify the GCNN model's processing and to improve its performance in terms of computational complexity. In a recently proposed approach [31], it is exposed that removing some nonlinear transformations can simplify GCNN and can achieve better performance in terms of node classification.

Amar Budhiraja et al. [32] handle the users and items of a rich data structure with multiple entities and sources with GCNNs. Their work tried to formulate a generalized solution for complex interactions using multiple graph CNN-based novel deep-learning architectures (MEDRES). A ranking metric, namely, pAp@k, was introduced in this paper to judge the performance of MEDRES, which fed the multilayer perceptron network with multiple embedding layers along with dynamic content. Jianing Sun et al. [33] proposed a multigraph convolution collaborative filtering recommendation framework that considers the similarity between the user and item pairs and solves them using multiple bipartite user–item interaction graphs in the embedding layer. BasConv was recently developed, which considers within-basket recommendations by combining heterogeneous interactions using the GCNN [34]. Cross-graph convolution [35] has also been used recently for creating a text–picture shopping recommendation. In this work, the author considered text while recommending products and provided pictures to users as a shopping guide. Hyeungill Lee et al. proposed collaborative filtering based on a deep neural networks recommendation system with user-rating and item-rating vectors as inputs. The authors claimed that their proposed approach was more scalable than and performed superiorly to the traditional collaborative filtering method [36]. Aminu Da'u et al. proposed developing recommendation systems utilizing the multichannel deep convolutional neural network that serve in aspect-based opinion mining, which considers fine-grained users' opinions to enhance the accuracy of recommendation systems [37]. The authors also used tensor factorization to improve the rating prediction of the overall recommendation systems.

Though GCNNs have been successful in various fields and applications [19,38,39], it is still challenging to train a large graph. Some work has proposed simplifying the GCN model to improve the training task and reduce the model's complexity for larger datasets. Neighbor sampling is a critical task in such models, which can be achieved by measuring nodes' similarities. In [40], node-wise sampling was performed, i.e., only a fixed number of neighboring nodes were considered to generate the embedding of each node. However, this might result in redundancy issues when generating the embeddings [41]. In other words, a shared neighbor between two nodes is navigated twice to generate its embedding. That results in increased complexity and computational time. Different approaches have been introduced to resolve this issue, such as Cluster-GCN [42], a vectorized relational graph convolutional neural network model known as VR-GCN [43], and FastGCN [23]. In Cluster-GCN, before training the GCN model, a graph-clustering approach restricts the neighbors in a subgraph or cluster. In VR-GCN, the complexity of the sample is improved by using some variance reduction techniques. FastGCN estimates the sampling probability for each node to reduce the approximation variance.

3. Proposed Model

In this section, we present the proposed framework in detail. Figure 2 illustrates the conceptual flow of the proposed framework. The raw data of users' clicks on different products are processed to learn about the users and items. The click probabilities for the items are generated for each user. The distance between two probability distributions is calculated, which gives us a similarity score. The similarity scores help to create clusters of similar users. These clusters are used to regenerate graphs; i.e., the neighboring nodes are classified based on a threshold value for the similarity scores. These graphs are used for GCNN training.

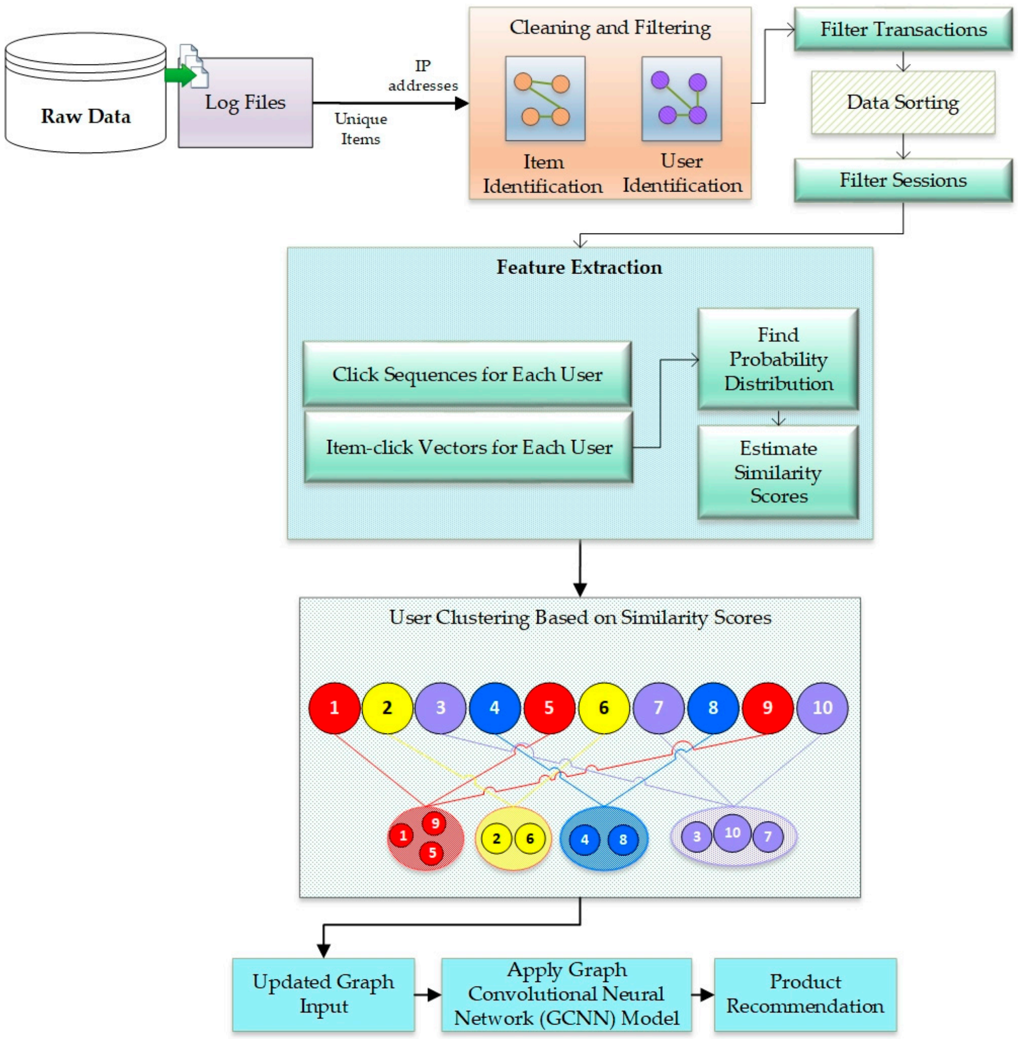

Raw Data
Log Files
IP addresses
Unique Items
Cleaning and Filtering
Item Identification
User Identification
Filter Transactions
Data Sorting
Filter Sessions
Feature Extraction
Click Sequences for Each User
Item-click Vectors for Each User
Find Probability Distribution
Estimate Similarity Scores
User Clustering Based on Similarity Scores
Updated Graph Input
Apply Graph Convolutional Neural Network (GCNN) Model
Product Recommendation

Figure 2. The conceptual flow of the proposed framework.

Each part of the proposed framework is introduced separately. Section 3.1 covers the frequency tables, Section 3.2 covers the probability distributions and similarity measures, Section 3.3 introduces the GCNN model, and Section 3.4 presents the recommendation mechanism.

3.1. Problem Definition

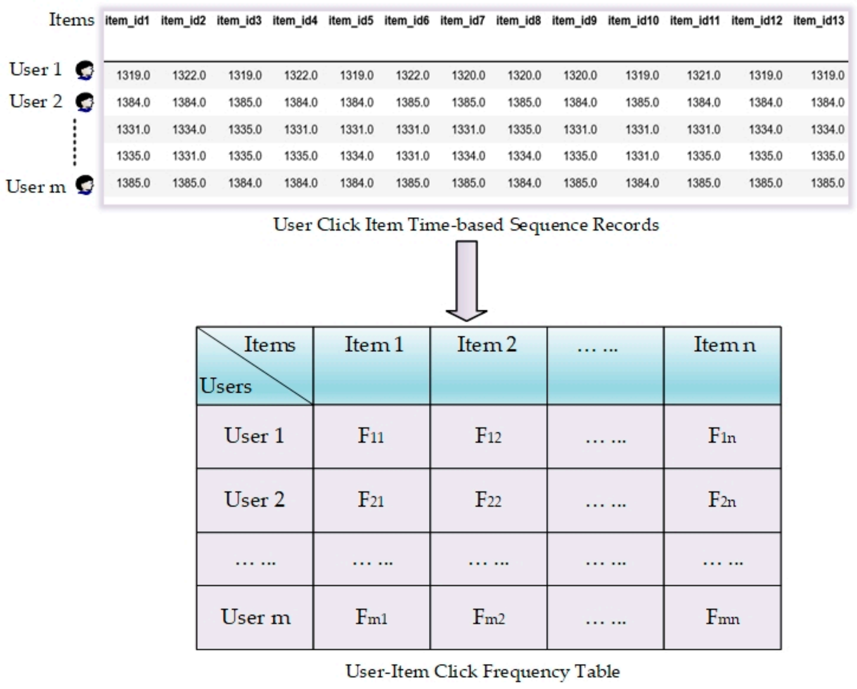
In this paper, we aimed to build a framework that could simplify the GCNN by identifying node neighbors based on their interaction similarities with the items. The task of item recommendation to a user is heterogeneous, as it consists of users, items, and the direct and indirect links among them. The set of users is denoted by $U = \{\text{u1, u2, u3,} \ldots\ldots, \text{um}\}$, where m is the total number of users. Similarly, the set of products is represented as $I = \{i1, i2, i3, \ldots.., in\}$, where n is the total number of products.

This recommendation model aims to recommend a product to the user that is most likely to be purchased based on the similarity of click patterns and distributions. Therefore, we can define our network as G = (V, E), where V is the set of nodes, and E is the set of edges between those nodes.

As the subgraphs are used to train GCN models sampled from a complete graph, many studies consider this for the recommendation systems [20,44]. One of the critical reasons for this is to reduce the model's computational complexity, as using the complete structure G = (V, E) can be computationally costly. Therefore, we also sample subgraphs from the whole graph structure, referred to as $G_{Sgh} =$ (V, E_{Sgh}). E_{Sgh} represents the edges between the node and its neighbors sampled from G. Both G and $G_{Sgh,}$ contain all the network nodes. The only difference is that G_{Sgh} will not have all the propagation paths and only have sampled edges. Therefore, this sampling process of generating subgraphs minimizes the neighbor's information. We identify similar neighbors based on the similarity scores.

3.2. User–Item Click Frequency Table

Figure 3 shows user click item time-based sequence records. These records show the item click sequences of each user in a session. The product IDs are encoded. We calculate the click frequencies of each user for each product across all the sessions to generate a user–item click frequency table of size m × n. Each user is represented as Fij, i.e., in terms of their clicks' frequency on an item. Fij represents the click frequency on item j by user i. Each row represents a user, and each column represents an item. If user i has not clicked an item j, then Fij = 0. The symbol m represents the total number of users, and the symbol n represents the total number of products.



Items	item_id1	item_id2	item_id3	item_id4	item_id5	item_id6	item_id7	item_id8	item_id9	item_id10	item_id11	item_id12	item_id13
User 1	1319.0	1322.0	1319.0	1322.0	1319.0	1322.0	1320.0	1320.0	1320.0	1319.0	1321.0	1319.0	1319.0
User 2	1384.0	1384.0	1385.0	1384.0	1384.0	1385.0	1385.0	1385.0	1384.0	1385.0	1384.0	1384.0	1384.0
	1331.0	1334.0	1335.0	1331.0	1331.0	1331.0	1331.0	1335.0	1331.0	1331.0	1331.0	1334.0	1334.0
	1335.0	1331.0	1335.0	1335.0	1334.0	1331.0	1334.0	1334.0	1335.0	1331.0	1335.0	1335.0	1335.0
User m	1385.0	1385.0	1384.0	1384.0	1384.0	1385.0	1385.0	1384.0	1385.0	1384.0	1385.0	1385.0	1385.0

User Click Item Time-based Sequence Records

Items / Users	Item 1	Item 2	… …	Item n
User 1	F_{11}	F_{12}	… …	F_{1n}
User 2	F_{21}	F_{22}	… …	F_{2n}
… …	… …	… …	… …	… …
User m	F_{m1}	F_{m2}	… …	F_{mn}

User-Item Click Frequency Table

Figure 3. Frequency table of user–item clicks.

3.3. Measuring Similarity Based on Probability Distribution of User Interactions

We measure the nodes' similarity based on their probability distributions in comparison with other nodes. It is difficult to find similarity between nodes in heterogeneous networks, as indirect links might exist between different nodes. The similarity is measured to identify the neighboring nodes.

Table 1 presents the notation we used to define different interaction paths. We denote the set of users as U and a single user as u_m. Similarly, the set of all the items is denoted by I, and a single item is represented by i_n. Now, as we are finding the probability distribution of user clicks for items, we define the click probability of a user u_m for a single item i_n as $P_{u_m}(i_n)$, and their click probability for all items I is defined as $P_{u_m}(\mathrm{I})$. After this, we find the similarity between two users by measuring the distance between their click distributions for items. Hence, we define it as $d(P_{u_1}(\mathrm{I}),\ P_{u_2}(\mathrm{I}))$. Xu_{ij} is a collection of all the clicked items in an individual user's set. It can be defined according to Equation (1).

$$Xu_{ij} = \mathrm{Q}u_i \cup \mathrm{Q}u_j\ ,\ (u_i\ ,\ u_j\ € \ U) \tag{1}$$

where $\mathrm{Q}u_i$ *and* $\mathrm{Q}u_j$ are sets of clicked items of any two users of U. To formulate this similarity, different distance metrics on the probability distribution can be used. We use both Jensen–Shannon divergence (JS divergence) and Kullback–Leibler divergence (KL divergence), as both have their pros and cons [45]. For example, JS divergence is symmetrical, while KL divergence is sensitive towards small changes in data. It is symmetrical. Wang et al. [46] used Kullback–Leibler (KL) divergence, also called KL distance, from the perspective of using probability distributions to evaluate the distance (similarity) between a pair of items. As JS divergence is derived from KL divergence, we first define and calculate the KL divergence.

Table 1. Different notation used with its respective definitions.

Notations	Definitions
U	Set of all users
I	Set of all products
$P_{u_m}(i_n)$	Click probability of user m for item i
$P_{u_m}(\mathrm{I})$	Click probability of user m for all items I
$d(P_{u_1}(\mathrm{I}),\ P_{u_2}(\mathrm{I}))$	The similarity between two users u_1 and u_2
$\mathrm{Q}u_i$	Set of user u_i's clicked items
Xu_i	Union set of all the items in any of two users from U consisting of clicked items

For the distance between two users u_i and u_j in terms of item clicks, the preference can be defined according to Equation (2).

$$D_{KL}\left(P_{u_i}(\mathrm{I}),\ P_{u_j}(\mathrm{I})\right) = \sum\nolimits_{i_m\ €\ Xu_{ij}} P_{u_i}\ (i_m)\ \ln \frac{P_{u_i}\ (i_m)}{P_{u_j}\ (i_m)} \tag{2}$$

Before formulating the JS divergence, we need to define an additional factor, i.e., P_M according to Equation (3).

$$P_M = \frac{1}{2} P_{u_i}(\mathrm{I}) +\ P_{u_j}(\mathrm{I}) \tag{3}$$

For the distance between two users u_i and u_j in terms of item clicks, the preference can be defined according to Equation (4).

$$D_{JS}\left(P_{u_i}(\mathrm{I}),\ P_{u_j}(\mathrm{I})\right) = \frac{1}{2}\ D(P_{u_i},\ P_M) + \ \frac{1}{2}\ D\left(P_{u_j},\ P_M\right) \tag{4}$$

As shown in Figure 4, once the distributions of the clicks' probability have been estimated, the similarity between two users is calculated based on their distribution distances.

According to Equation (5), we can say that users' similarity increases if the distance among their probability distributions decreases.

$$\text{Similarity} \propto \frac{1}{\text{Distance}} \tag{5}$$

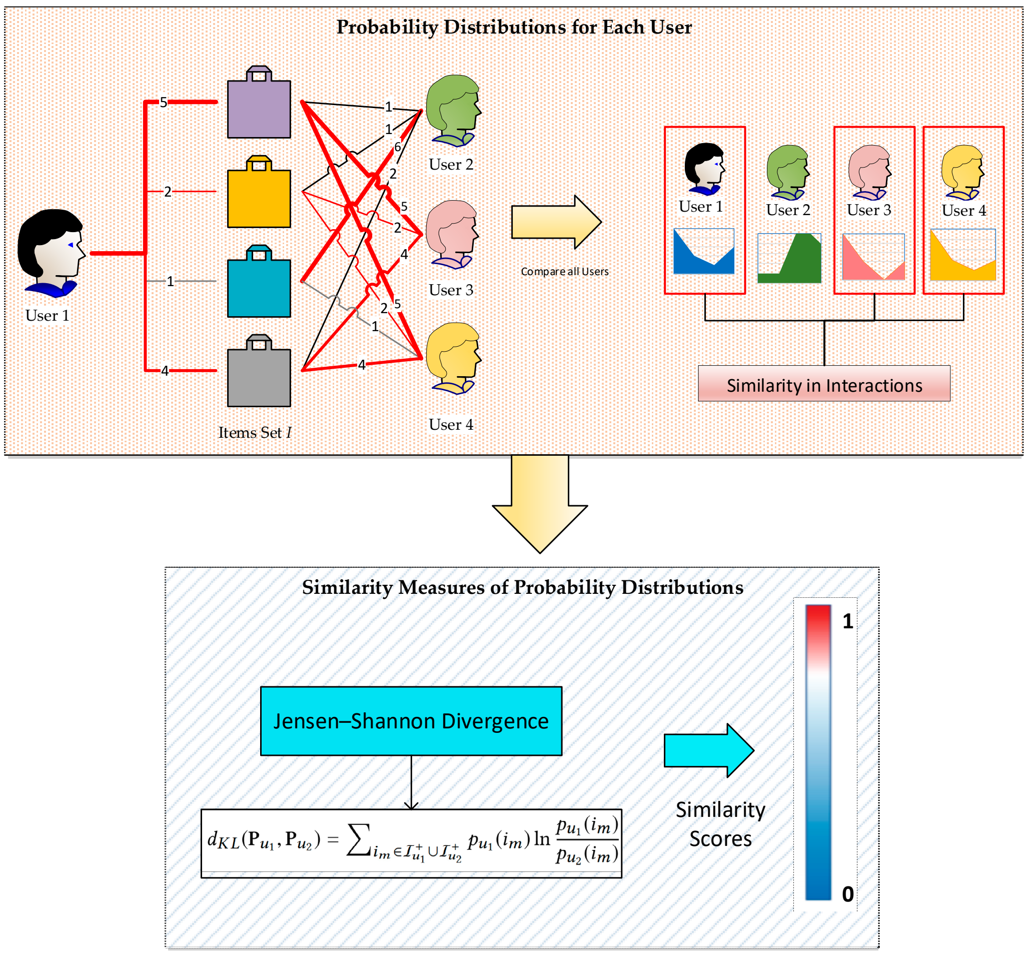
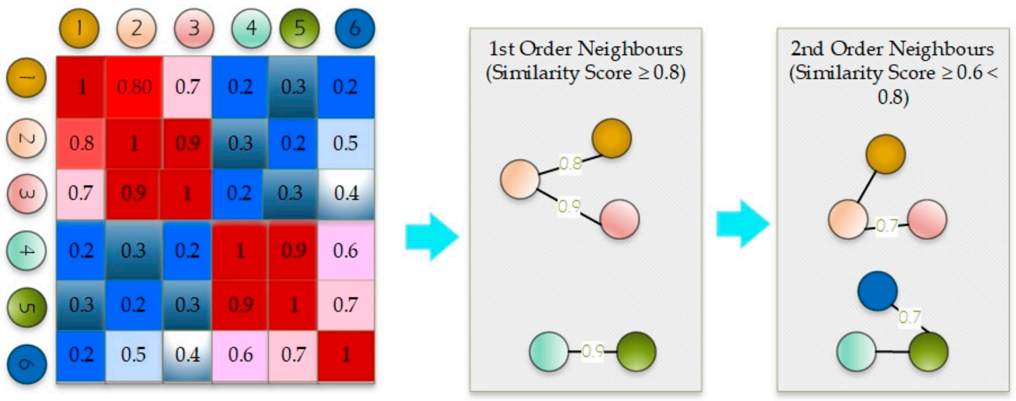

Probability Distributions for Each User
User 1
5
2
1
4
Items Set *I*
1
1
6
2
5
2
4
2
5
1
4
User 2
User 3
User 4
Compare all Users
User 1
User 2
User 3
User 4
Similarity in Interactions
Similarity Measures of Probability Distributions
Jensen–Shannon Divergence
$d_{KL}(\mathbf{P}_{u_1}, \mathbf{P}_{u_2}) = \sum_{i_m \in I^+_{u_1} \cup I^+_{u_2}} p_{u_1}(i_m) \ln \frac{p_{u_1}(i_m)}{p_{u_2}(i_m)}$
Similarity Scores
1
0

Figure 4. Similarity score estimation for users based on the probability distribution.

In terms of the KL divergence, the negative distance between similarities can be defined as the negative distance between $P_{u_i}(\mathrm{I})$ and $P_{u_j}(\mathrm{I})$, which is defined as the measure of similarity between u_i and u_j as shown in Equation (6).

$$Sim_{\left(P_{u_i}(\mathrm{I}),\ P_{u_j}(\mathrm{I})\right)} = -\,D_{KL}\left(P_{u_i}(\mathrm{I}),\ P_{u_j}(\mathrm{I})\right) \tag{6}$$

The similarity measure is normalized between 0 and 1 for simplicity. After estimating the similarity between all the users, we set a similarity score threshold to select the neighbors. Neighbor sampling is a critical yet vital task for graphs. The value of the similarity score ranges between 0 and 1. If the node's probability distribution is 100% the same as that of other nodes, the score will be 1; otherwise, 0. We defined threshold values for the similarity scores to estimate the level of the neighboring node. In Figure 5, different colors represent different nodes, and there are a total of six nodes. If the score of similarity between two nodes is $\geq$0.8, they are considered first-order neighbors and are at a distance of only one click from each other. If the similarity score is $\geq$0.6 but less than 0.8, then these nodes are second-order neighbors.



	1	2	3	4	5	6
1	1	0.80	0.7	0.2	0.3	0.2
2	0.8	1	0.9	0.3	0.2	0.5
3	0.7	0.9	1	0.2	0.3	0.4
4	0.2	0.3	0.2	1	0.9	0.6
5	0.3	0.2	0.3	0.9	1	0.7
6	0.2	0.5	0.4	0.6	0.7	1

Figure 5. Threshold-based neighbor sampling of nodes.

Consider a scenario where there are multiple first-order neighbors and then how we can select to which node the second-level neighbor might become attached. In such scenarios, the similarity score is calculated in loops. The closer a node is in terms of the score, the more likely it is to get selected as a direct connection. For example, in Figure 5, the similarity score of node 5 and node 4 is 0.9, node 6 and node 5 is 0.7, and node 4 and node 6 is 0.6. In this case, node 4 and node 5 are direct neighbors according to the rule. On the other hand, node 6 is a second-order neighbor with both node 4 and node 5, as the similarity score between node 5 and node 6 is greater than the score between node 4 and 6. Therefore, node 6 will relate to node 5. If the score is the same, then the node can be attached to any node. All the other nodes do not have the similarity score criteria and, hence, are considered as isolated clusters.

Now, we have generated clusters of nodes that have similar item-click patterns and learned the neighborhood sampling. We regenerate session graphs based on this knowledge and use these graphs as input for our GCNN model. Previously, the GCNN model learned neighbors' sampling directly from the raw session graphs, but now, we use updated session graphs where we already know about the node neighbors. This knowledge helps to decrease the computational complexity of the GCNN. The next section explains the proposed model's complete architecture and describes the GCNN model used for training.

3.4. GCNN-Based System Architecture

Figure 6 shows the complete framework. This proposed model comprises multiple layers, i.e., an input data layer, a data preprocessing layer, similarity estimation, a neighbor sampling layer, a deep learning layer, and a recommendations layer. All the layers play a vital role. The first layer handles the input data. The raw data are in the form of log files, which are later converted to session graphs of items and users. We extracted multiple features from the raw data, such as the $Address_{IP}$ and location. The preprocessing layer is the primary pillar of this framework. We analyzed our data thoroughly, investigated different trends, and set some criteria for session selection. We discarded the session that had fewer than four user clicks. We also separated sales sessions from non-sales sessions.

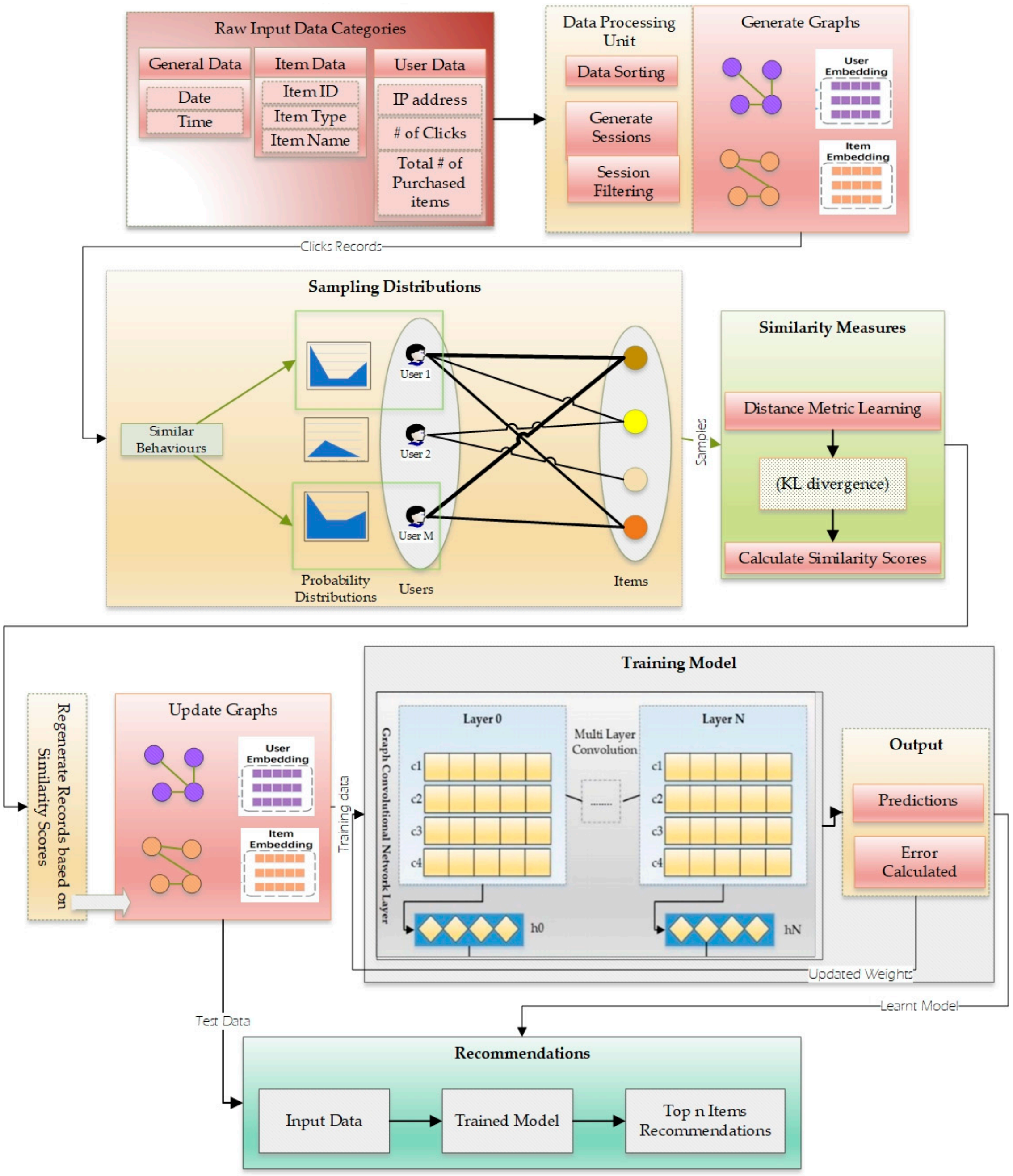


Figure 6. Detailed view of the proposed framework.

Data processing is an essential part of any machine learning model and plays an important role in improving any algorithm's accuracy. We used the processing layer to analyze the underlying behavior and pattern of our input data. In the processing stage, we look for data anomalies, irregularities in the data, patterns, and statistics. Various features such as the clicks for each session, clicks for each product, total and unique IPs, and sales history per session are analyzed in the processing phase. In the data sorting phase, we sort the data based on IPs or perform time-based sorting. For the statistics, we extract the unique IPs, the number of clicks for each IP, the number of items bought by a specific user, the session duration, and the hit rate. We then perform data filtering, where we filter unwanted data responsible for irregular patterns and anomalies. We also filter sessions that have no sales history. At last, the sessions are generated based on the click sequence pattern for each user.

After this, each user's activity is analyzed by finding his/her item-click probability distributions. Then, we compare these distributions with other distributions and find similarities between nodes based on similarity scores. We cluster similar nodes together as described in the previous section.

We use the GCNN model, which uses the subgraphs for training. It is challenging to train the traditional deep GCNN models. Therefore, to reduce the excessive computational complexity, we prelearned the neighbors and generated subgraphs. We used three layers for initial training, and this structure was useful in our case. The GCNN predicts the next most likely item clicked by the user and calculates the model error. Once the model is trained, it is used to recommend the top n items to the new user.

In the proposed model, we consider that both the item and user modeling are the same. Therefore, we will primarily refer to user modeling for explaining node representations in the network. The embedding vector of a user u from the set of users U is represented as $\boldsymbol{h}_{ui} = W_{u_i}$, where W_{u_i} is the set of features for the user ui. Once we have the embedding vector of user ui, the next step is to obtain neighboring nodes' embeddings. We first find the features of all the adjacent nodes of user ui. Then, these features are aggregated by the following Equation (7):

$$W_{N_{u_i}} = \text{AGGREGATE}\left\{W_{vi}, \forall_{vi} \in N_{u_i}\right\} \tag{7}$$

where N_{u_i} is the set of sampled neighbors discovered by using the probability distribution-based similarity and W_{vi} is the feature vector of neighbor vi, which belongs to the set N_{u_i}. Additionally, we use a pooling function defined as AGGREGATE{.}. Once we have the aggregated feature set of the sampled neighbors feature, the next step is to concatenate these features with the features of user u, i.e., W_{u_i}. This can be represented by Equation (8).

$$\overline{W}_{ui} = W_{u_i} \oplus W_{N_{u_i}} \tag{8}$$

Both W_{ui} and $W_{N_{u_i}}$ are the feature vectors of a node ui and its neighboring nodes N_{u_i}, respectively. The symbol $\oplus$ represents the concatenation between these two vectors. In the final step, we obtain the absolute node representation by feeding this aggregated feature set defined as $\overline{W}_{ui}$ to a simple neural network, and we represent it as Equation (9).

$$h_{ui} = \sigma\left(\text{W}.\,\overline{W}_{ui}\right) \tag{9}$$

In the same way, we perform the item modeling.

4. Data

In this part, we introduce and clarify the information we utilized for our tests. It essentially incorporated the data type, size, source, and preprocessing. Data determination is a significant part of the test cycle. Subsequently, we invested a lot of time and energy in examining and preprocessing the data, as the point of this investigation was to prescribe products to users depending on their past click interactions. The input data for the recommendation system were acquired from an e-shopping mall known as eJeju mall. The eJeju mall is an online shopping store that only delivers locally on Jeju Island, South Korea. The data consist of the users' historical data, and we utilize this data to form session-based graphs that exhibit the users' dynamic behavior. A portrayal of the data source is given in Table 2 beneath.

4.1. Item Clicks of User

We have customers' item-click history information for various things that incorporate the client's recently clicked products in a session. Customer's item clicks are utilized to comprehend the customer's behavior. We additionally allude to this information as the click sequence of a customer. This information was then prepared to find novel clicks and the bought things per customer, and afterward, all the information was encoded into unique product IDs.

Table 2. Description of data.

Data	Definition
$Address_{IP}$	User IP address
$Date_{accessed}$	The date when user accessed the website
$Time_{accessed}$	The time when the user accessed the website
$URL_{accessed}$	URL of the specific page accessed by the user
$ID_{Product}$	The ID of the product clicked by the user
$Name_{Product}$	Name of the product clicked by the user
$Type_{Product}$	Type of the product clicked by the user

4.2. Customers' Profile Data

Other than the click patterns of each customer, there are other features such as the customer's IP address, referred to as $Address_{IP}$, and the time and date when a customer clicks a product, referred to as $Time_{accessed}$ and $Date_{accessed}$, respectively. The relevant profile features of the customers are derived from these features; for example, we used $Address_{IP}$ to derive the customer's location. From this access_ip, we derived the location of each user. Additionally, we utilized $Time_{accessed}$ and $Date_{accessed}$ to produce various sessions.

4.3. Product Data

Our dataset contained some other product features such as the product ID (referred to as $ID_{Product}$), product name (referred to as $Name_{Product}$), and product click type (referred to as $Type_{Product}$).

4.4. Experimental Data

Table 3 presents the statistics of the data used for the experiments. The data were divided into three parts, i.e., the training data (60%), test data (20%), and validation data (20%). We also evaluated and compared the proposed model using other publicly available datasets such as YOOCHOOSE [47] and DIGINETICA [48].

Table 3. Description of experimental data.

Data	Statistics
Total number of instances	~300 K
Period of data	One Year (2019–2020)
Number of unique customers	~50 K
Total purchased products	7701
Total number of unique clicks	244,533
Total number of visited item description pages	42 K

5. Implementation and Experimental Setup

Here, we present the implementation environment details along with the experimental setup. The evaluation metrics are also defined in detail.

5.1. Experimental Setup

The implementation environment was based on (i) Ubuntu as an operating system (18.04.1 LTS version), (ii) memory of size 32 Gb, and (iii) Nvidia GeForce 1080 as a graphics processing unit (GPU). The development language was Python, along with Tensorflow and Scipy.

5.2. Evaluation Metrics

The performance of our system was measured by using the following evaluation metrics.

1. Accuracy: the following formula was used to calculate the model's accuracy, as shown in Equation (10).

$$Accuracy = 1 - \frac{\| Y - \hat{Y} \| F}{\| Y \| \ F} \tag{10}$$

where Y represents the actual data and $\hat{Y}$ represents the predicted data.

2. Root Mean Square Error (RMSE): the RMSE was calculated using Equation (11).

$$RMSE = \sqrt{\frac{1}{MN} \sum_{j=1}^{M} \sum_{i=1}^{N} \left(yi^{j} - \hat{yi}^{j} \right)^{2}} \tag{11}$$

where yi^{j} *and* $\widehat{yi}^{j}$ are subsets of Y and $\hat{Y}$ and represent the actual data and predicted data for the jth time sample in the ith session, respectively. M is the total time samples, and N is the number of products. The RMSE is precisely used to evaluate the prediction error. The smaller the value of the RMSE is, the better the prediction rate or score is according to Equation (12).

$$\text{Prediction rate} \propto \frac{1}{\text{RMSE}} \tag{12}$$

While the accuracy is used to detect the predictions' precision, it has an opposite effect to the RMSE on the prediction rate, as shown in Equation (13). The higher the value of the accuracy is, the better the prediction rate is.

$$\text{Prediction rate} \propto \text{Accuracy} \tag{13}$$

3. Recall@20: This refers to the total number of correctly recommended items among the first 20 items, making this one of the essential evaluation metrics for building accurate recommendation systems.
4. F-Score: This analyzes the classification of a recommendation model and provides the model's test accuracy computed from the precision and recall score.
5. MRR@20: This represents the mean reciprocal rank, and we computed the average reciprocal rank of the top 20 products. A higher value of MRR indicates that the model has higher accuracy for correctly recommending products.
6. ROC: The ROC curve (receiver operating characteristic curve) is a graph used to present the model's classification performance. The two key parameters, known as the true-positive rate and false-positive rate, are plotted against each other. We used this performance metric to evaluate our model's performance and compare it with a few traditional models.

6. Results

This section presents the analysis and results of our proposed GCNN methodology for product recommendation depending on the user click sequence, purchased items, and measures of similarity between customers with related and similar behaviors. We present our proposed model's time complexity, memory complexity, and convergence rate in the next subsections compared to those of other state-of-the-art models. We also present the root mean squared error rate, accuracy with respect to the hidden units (HU), and accuracy of the training and testing phase of our proposed GCNN. In the later subsection, we describe the training and testing loss and the impact of the length of the click sequence on accuracy.

6.1. Comparison of Complexities and Convergence Rate

Table 4 shows the time and memory complexity comparison of our proposed GCNN with other state-of-the-art methods such as GCNN, OpGCN, DeepWalk [49], and version of Knowledge Graph Convolutional Networks known as KGCN-neighbor [50]. Here, *L* represents the number of layers, *n* is the total number of nodes in the graph, *e* is the total number of edges, and *b* is the batch size. Additionally, *r* represents the number of neighbors clustered for each node, and *h* is a constant that refers to the feature dimension of hidden nodes. The implementation depicts that our proposed GCNN realizes the lowest time and memory complexity and converges at a faster rate than the other methods.

Table 4. Complexity (time and memory) and convergence rate comparison.

Model	Time Complexity	Memory Complexity	Convergence Rate
DeepWalk	O (Lneh.log(n))	O (Lnh + Lh^2)	2200 Epochs
KGCN-Neighbour	O (Leh + Lnh^2 + rLnh^2)	O (Lnh + Lh^2)	2100 Epochs
GCNN	O (Leh + Lnh^2)	O (Lnh + Lh^2)	1800 Epochs
OpGCN	O (Lrnh^2)	O (Lbrh + Lh^2)	1650 Epochs
Proposed GCNN	O (Leh + Lnh^2)	O (Lbh + Lh^2)	1400 Epochs

6.2. Training and Testing Accuracy

This section compares the training and testing accuracy of the proposed model against those of other models, primarily Probabilistic Matrix Factorization (PMF) [51], DeepWalk, GCNN, and OpGCN. The accuracy of these models was estimated in terms of the number of epochs, number of hidden units, and different batch sizes.

Figure 7 shows that measuring the similarity can optimize the GCNN models to a greater extent. Thus, our proposed GCNN achieved higher accuracy as compared to the simple GCNN model and OpGCN model. Our proposed GCNN obtained a training accuracy rate of around 95% after 1200 epochs, compared to simple GCNN, with an accuracy rate of 78%, and OpGCN, with an accuracy of 93%.

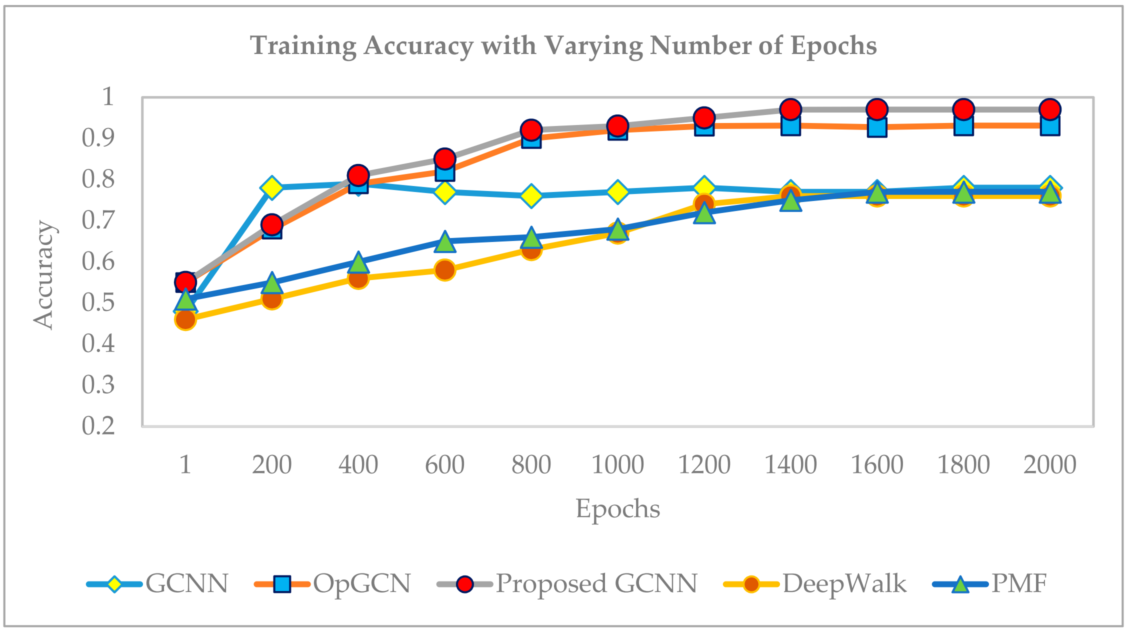


Figure 7. Training accuracy against the number of epochs.

Similarly, Figure 8 shows the comparison of the testing accuracy of all the previously mentioned models against the number of epochs. As we can see, our proposed model achieved a testing accuracy of 95.8%, as compared to OpGCN, with an accuracy of 92%, and GCNN, with an 88% accuracy. The testing accuracy of DeepWalk was improved compared with its training accuracy. The PMF model did not show satisfactory improvements in testing. The model's behavior in testing was estimated to be similar to the training behavior.

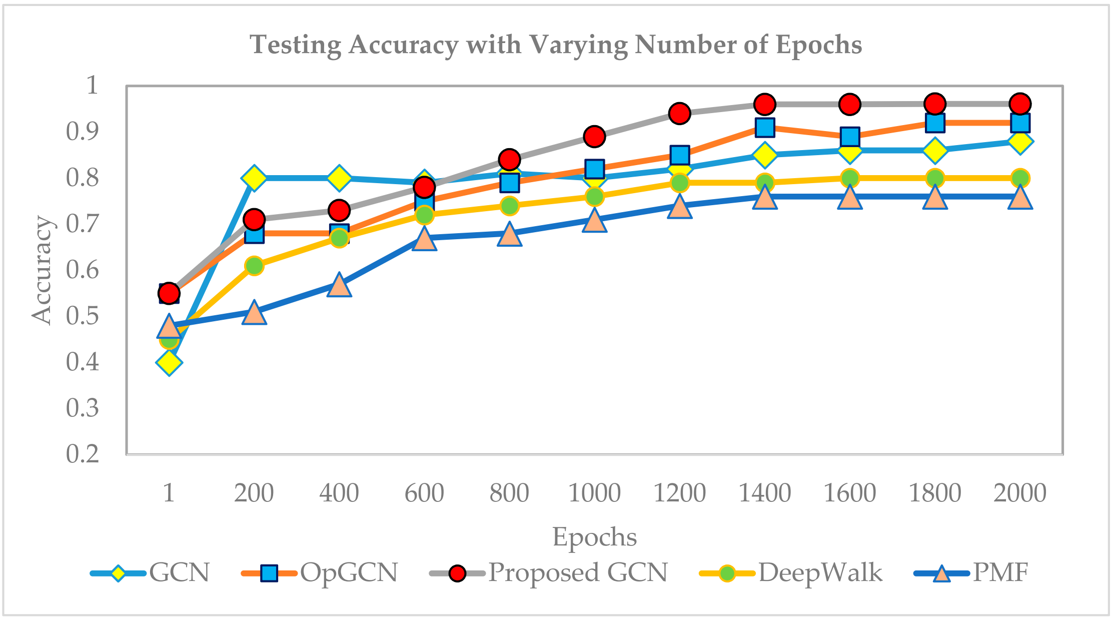


Figure 8. Testing accuracy against the number of epochs.

Hidden units play an essential role in impacting the model's performance, and therefore, there is a need to analyze the accuracy and error rate compared to various hidden units. Figure 9 shows that the proposed GCNN's accuracy is the highest at HU_64 and can be considered optimal. We can observe from the graph that our proposed GCNN is quite impressive in comparison to simple GCNN and better than OpGCN due to the optimization and processing techniques applied to the model and data.

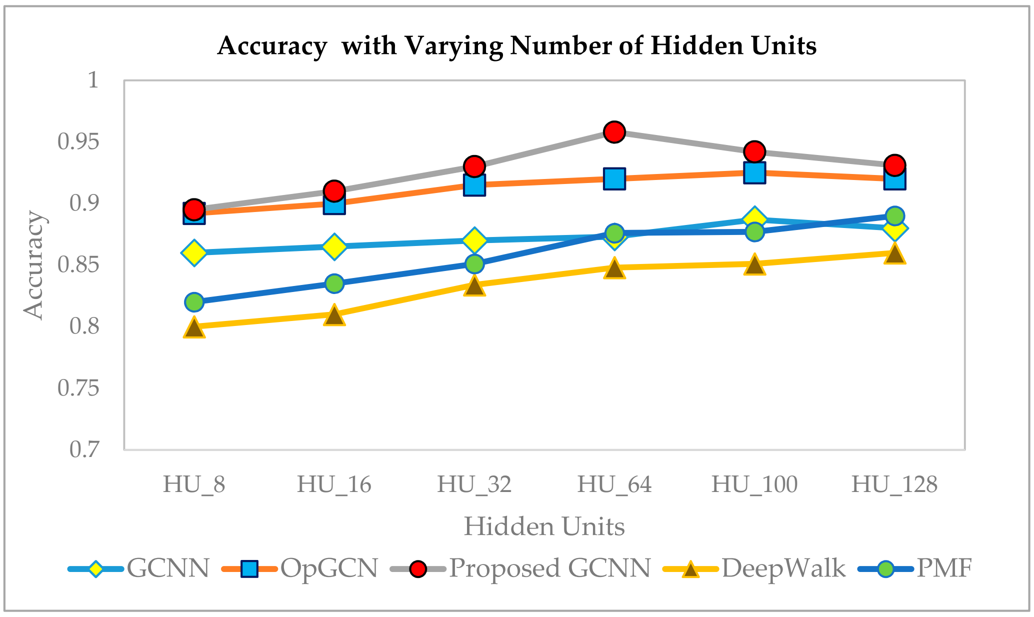


Figure 9. Testing accuracy against hidden units.

Figure 10 shows the experimental results of selecting different batch sizes for testing the model's accuracy for new data. Then, this accuracy is compared with that of other models such as GCNN and DeepWalk. As we can see, our proposed model achieved a testing accuracy of 95.8% for Batch_256. This shows that the model's performance is better with a reasonably small size of batch. Similar behavior is observed for the other models.

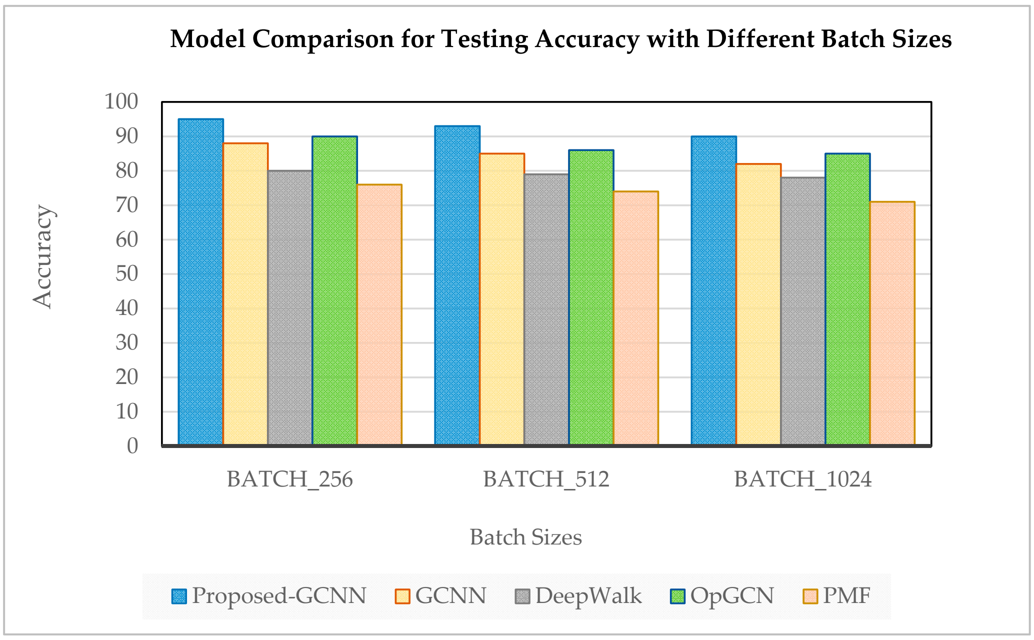


Figure 10. Testing accuracy for different batch sizes.

6.3. *Loss for Training and Testing Model*

In this section, we evaluate the proposed model's performance based on the loss values. In Figure 11, we compare the training loss against the number of epochs for GCNN, OpGCN, DeepWalk, PMF, and our proposed methodology.

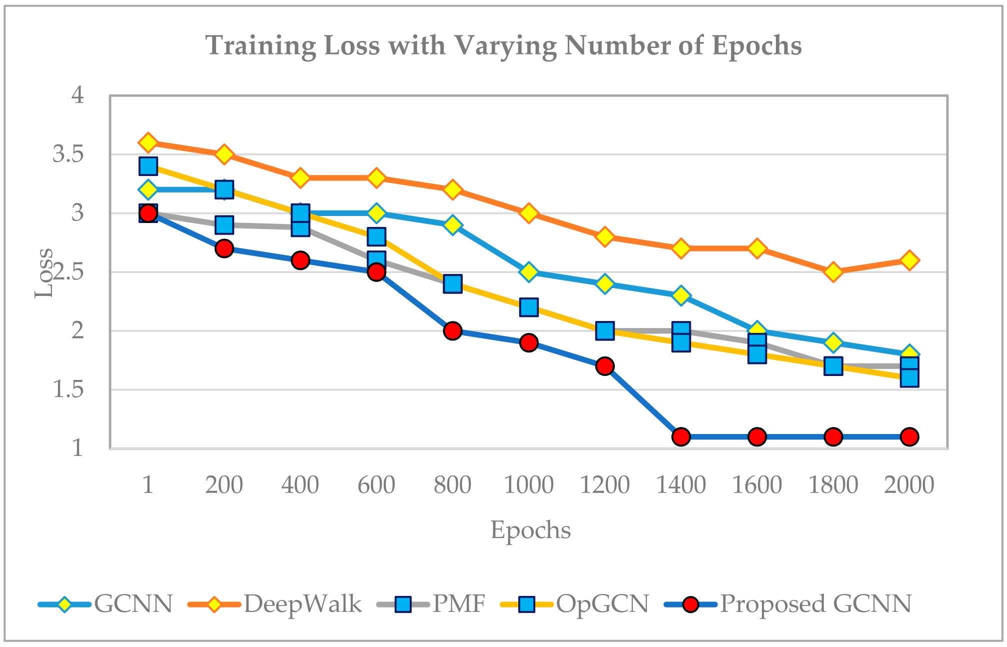


Figure 11. Training loss with respect to epochs.

The training loss for both simple GCNN and OpGCN is higher than that for our proposed GCNN, which also converges at a higher rate. It can be observed that the training loss for our proposed GCNN model is higher at the start and that it decreases as the number of epochs reaches 1400. In Figure 12, we show the results for the experiment where we used different batch sizes for testing and evaluated the model's loss value. Similar to the accuracy, a smaller batch size has a positive impact on the model's loss value.

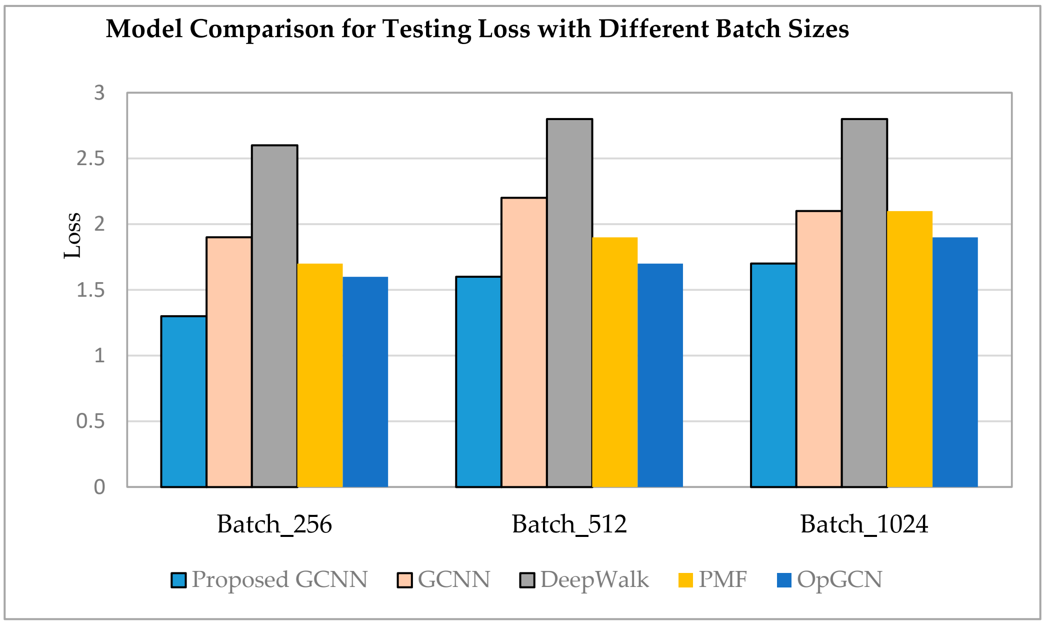


Figure 12. Testing loss with respect to different batch sizes.

6.4. Error Score and ROC Comparisons

The root mean squared error (RMSE) measures the difference between the predicted and the original value and denotes the error rate. Figure 13 shows the error scores for GCNN, OpGCN, and our proposed GCNN compared to different hidden units. Our model's RMSE score was the lowest compared to GCNN's and OpGCN's, with a score of 3.9, and performed best at HU_100.

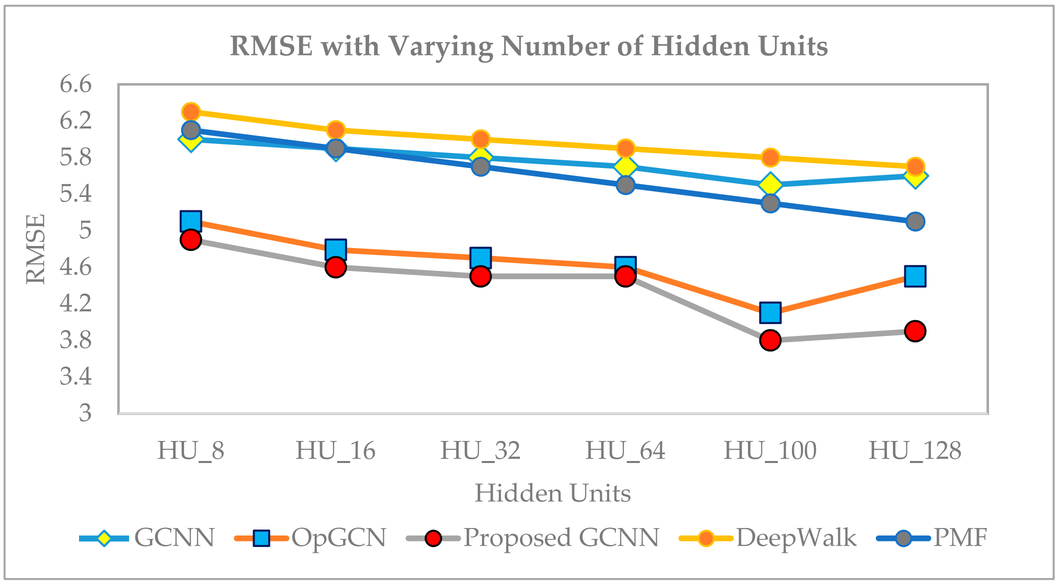


Figure 13. Root Mean Square Error (RMSE) against different numbers of hidden units.

We compared the ROC curves of four different models with the curve for our proposed GCNN model, applied to our dataset. The ROC is an important tool when predicting the probability of the results achieved. The ROC is basically a graph plotted between the False Positive Rate (FPR) and the True Positive Rate (TPR). The FPR is usually plotted on the x-axis, and the TPR is plotted on the y-axis. A model is considered efficient when the values on the plot are gathered towards the top left of the plot and away from the 45-degree diagonal, i.e., when there are higher values on the y-axis. In other words, it means that the model produced more correctly predicted values than falsely predicted ones. As we can see from Figure 14, our proposed model is the farthest from the 45-degree diagonal of the ROC space and performed better than the existing models, with higher accuracy. The models used for comparison include Association Rule Mining [52], GCNN, DeepWalk, PMF, and OpGCN.

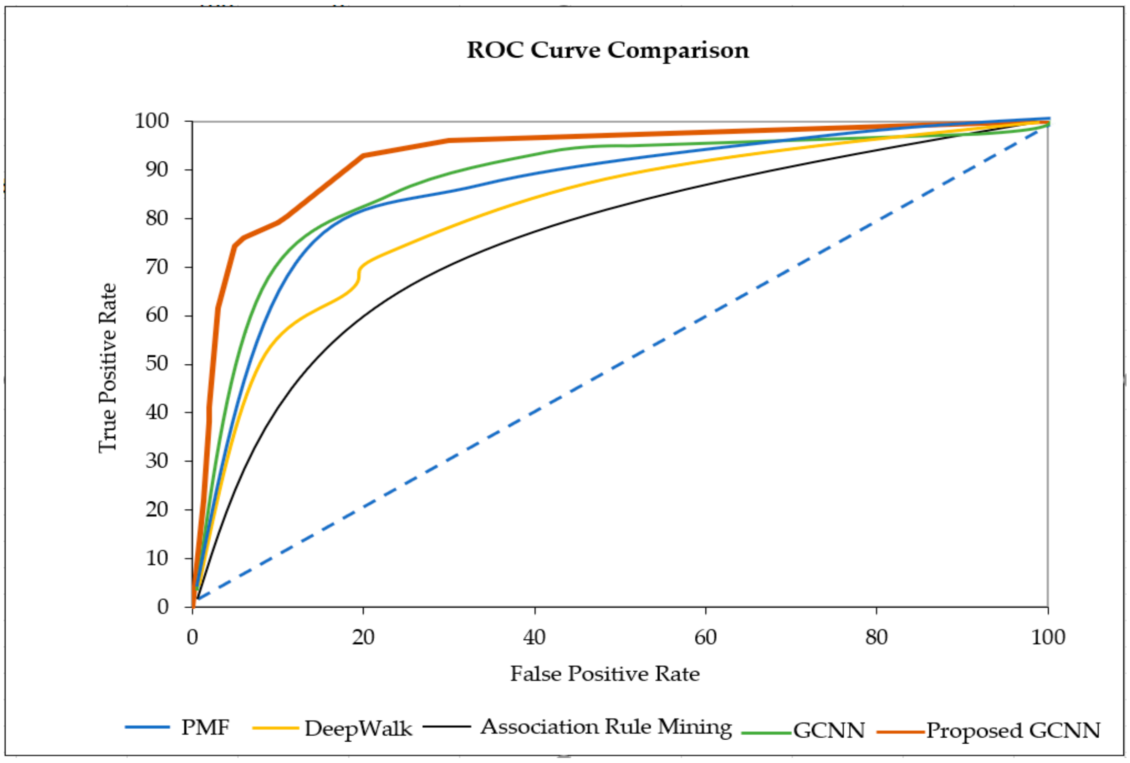


Figure 14. Receiver operating characteristic curve (ROC) comparison.

6.5. Performance Metric Comparisons of Different Models with Different Datasets

As it is difficult to plot all the performance measures for all the state-of-the-art models due to the limited ability of our model's architecture to accommodate all the algorithms, we present a comparison of our proposed model with different models on different datasets in Table 5. The models used for the comparisons include traditional models for recommendations such as Association Rule mining, a collaborative filtering-based approach known as Singular Value Decomposition (SVD) [53], and other methods such as DeepWalk, KGCN-neighbor, GCNN, OpGCN, and the proposed GCNN model. The performance metrics are the Recall@20, F-Score, MRR@20, and testing accuracy. Our proposed approach seems to perform better as compared with the other models.

Table 5. Performance comparisons for different datasets with different models.

Data	**Metrics**	**SVD**	**KGCN-Neighbor**	**DeepWalk**	**Association Rule Mining**	**GCNN**	**OpGCN**	**Proposed GCNN**
YOOCHOOSE	Recall@20	17.19	28.31	45.16	31.19	50.64	69.48	70.18
	F-Score	0.617	0.669	0.712	0.613	0.718	0.841	0.892
	MRR@20	8.11	19.21	17.59	27.83	37.15	38.12	40.55
	Testing Accuracy	0.65	0.69	0.71	0.61	0.77	0.76	0.81
DIGINETICA	Recall@20	20.13	27.44	46.98	31.26	48.13	65.42	69.87
	F-Score	0.691	0.627	0.698	0.713	0.785	0.794	0.847
	MRR@20	9.19	20.18	17.11	25.56	34.13	37.40	39.98
	Testing Accuracy	0.68	0.72	0.77	0.74	0.82	0.86	0.93
Our Data	Recall@20	20.21	29.82	45.38	38.56	50.18	69.34	78.91
	F-Score	0.647	0.624	0.791	0.792	0.869	0.856	0.914
	MRR@20	8.02	19.34	20.45	28.01	39.11	37.14	39.17
	Testing Accuracy	0.61	0.72	0.75	0.75	0.81	0.88	0.97

SVD: Singular Value Decomposition; KGCN-neighbor: Knowledge-Graph Convolutional Networks based approach.

7. Conclusions

In this paper, we introduced a graph-CNN-based model for online product recommendation. The proposed model incorporates the user–item click interaction as a probability distribution and uses this measure to find the similarity among nodes. We calculate the similarity between the nodes using divergence methods and use the results for clustering the nodes based on their interaction similarity. This way, we understand and sample the neighboring nodes beforehand and generate session graphs based on this knowledge. Afterward, these subgraphs are used as input for the GCNN model. With the simple GCNN model, the system finds it difficult to achieve more than 80% prediction accuracy. Additionally, without finetuning the model's structure, it becomes computationally rich and complex. That affects the performance of the model in terms of low prediction rates and model overfitting. Different variations of GCNN have been proposed previously to handle the performance-related issues of the model. We aimed to explore the effect of finding users' similarity based on their interactions with items such as clicks and then use this knowledge to make clusters. This way, we do not have to make extensive changes in the GCNN model. We achieved comparatively better computational complexity results and time than with our previously proposed model, i.e., OpGCN [30].

During the cycle, records with under three ticks are disposed of alongside the records that show some bizarre examples. In particular, we first cycle the log information documents; the irregular information records are changed over to client click arrangements. These click groupings of customers are changed over to unknown customers' sessions. At that point, coordinated charts are built from these sessions alluded to as session graphs. To sample the neighbors, we performed similarity measures to find the similarities between two nodes based on their products' interactions. For this, we estimate the probability distribution of each node and use divergence techniques to measure the distance between these nodes. Based on their similarity scores, we set a threshold and define our criteria for a neighbor's selection. If the similarity score does not fit the requirements, those nodes are not considered neighbors and are classified into different clusters. These clusters are then converted to subgraphs, and these subgraphs are used as input to the GCNN model.

To summarize, this research work contributes in the following ways: (1) It helps to elucidate the challenges the GCNN model might face while handling large datasets such as complexity and memory-related difficulties; (2) we propose a framework that helps GCNN models to simplify their tasks; (3) we introduced probability distribution based-similarity measures to sample neighboring nodes; (4) the proposed recommendation model simplifies the graph representation of nodes and items as preprocessing tasks for the GCNN to learn. We integrated KL divergence to build node clusters based on their similarities, and the results confirm that this helps to improve the complexity and computation time. (5) We compared our results with those for the previously proposed OpGCN model [30] and proved that simplifying the data beforehand can improve GCNN performance for recommendation tasks.

Author Contributions: W.S. conceived the idea for this paper, designed the experiments, wrote the article, assisted in the algorithm implementation, and assisted with the design and simulation; Y.-C.B. proofread the manuscript and supervised the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research was financially supported by the Ministry of Small and Medium-sized Enterprises (SMEs) and Startups (MSS), Korea, under the "Regional Specialized Industry Development Program (R&D, S2855401)" supervised by the Korea Institute for Advancement of Technology (KIAT).

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

References

1. Alexander, J.E.; Tate, M.A. *Web Wisdom: How to Evaluate and Create Web Page Quality*; Erlbaum, L., Ed.; Associates Inc.: USA, 1999.
2. Kräuter, S.G.; Kaluscha, E.A. Empirical research in on-line trust: A review and critical assessment. *Int. J. Hum. Comput. Stud.* **2003**, *58*, 783–812. [CrossRef]
3. CROWDFUNDING MARKET-GROWTH, TRENDS, COVID-19 IMPACT, AND FORECASTS (2021–2026). Available online: https://www.mordorintelligence.com/industry-reports/crowdfunding-market (accessed on 2 December 2020).
4. Crowdfunding. Available online: https://www.statista.com/outlook/335/100/crowdfunding/worldwide (accessed on 2 December 2020).
5. Shih, Y.Y.; Liu, D.R. Product recommendation approaches: Collaborative filtering via customer lifetime value and customer demands. *Expert Syst. Appl.* **2008**, *35*, 350–360. [CrossRef]
6. Liu, D.R.; Shih, Y.Y. Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences. *J. Syst. Softw.* **2005**, *77*, 181–191. [CrossRef]
7. Chen, H.; Yin, H.; Chen, T.; Nguyen, Q.V.H.; Peng, W.C.; Li, X. Exploiting Centrality Information with Graph Convolutions for Network Representation Learning. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering, Macau, China, 8–11 April 2019; pp. 590–601.
8. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.D.; Wang, M. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv* **2020**, arXiv:2002.02126.
9. Wang, W.; Yin, H.; Huang, Z.; Wang, Q.; Du, X.; Nguyen, Q.V.H. Streaming Ranking Based Recommender Systems. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 525–534.
10. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.* **2019**, *5*, 3285029. [CrossRef]
11. Sedhain, S.; Menon, A.K.; Sanner, S.; Xie, L. Autorec: Autoencoders Meet Collaborative Filtering. In Proceedings of the 24th International Conference on World Wide Web 2015, Florence, Italy, 18–22 May 2015; pp. 111–112.
12. Wu, Y.; DuBois, C.; Zheng, A.X.; Ester, M. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining CD-ROM, San Francisco, CA, USA, 22–25 February 2016; pp. 153–162.
13. He, Q.; Pei, J.; Kifer, D.; Mitra, P.; Giles, L. Context-Aware Citation Recommendation. In Proceedings of the 19th International Conference on World Wide Web 2010, Raleigh, NC, USA, 26–30 April 2010; pp. 421–430.
14. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. *World Wide Web Internet Web Inf. Syst.* **2017**, 173–182.
15. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *IJCAI* **2017**, 1725–1731.
16. Hsieh, C.K.; Yang, L.; Cui, Y.; Lin, T.Y.; Belongie, S.; Estrin, D. Collaborative Metric Learning. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 May 2017; pp. 193–201.
17. Chen, J.; Xu, X.; Wu, Y.; Zheng, H. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv* **2018**, arXiv:1812.04206.
18. Van den Berg, R.; Kipf, T.N.; Welling, M. Graph convolutional matrix completion. *arXiv* **2017**, arXiv:1706.02263.

19. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; Yin, D. Graph Neural Networks for Social Recommendation. *arXiv* **2019**, *1902*, 07243.
20. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
21. Chami, I.; Ying, Z.; Ré, C.; Leskovec, J. Hyperbolic Graph Convolutional Neural Networks. In Proceedings of the 2019 Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 4869–4880.
22. Zhang, M.; Chen, Y. Link Prediction Based on Graph Neural Networks. In Proceedings of the 2018 Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 5165–5175.
23. Chen, J.; Ma, T.; Xiao, C. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
24. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 2017 Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034.
25. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
26. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Metagraph Based Recommendation Fusion Over Heterogeneous Information Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 635–644.
27. Zhao, J.; Zhou, Z.; Guan, Z.; Zhao, W.; Ning, W.; Qiu, G.; He, X. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2347–2357.
28. Hu, B.; Shi, C.; Zhao, W.X.; Yu, P.S. Leveraging Meta-Path Based Context for Top-N Tecommendation with A Neural Co-Attention Model. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1531–1540.
29. Wu, Q.; Zhang, H.; Gao, X.; He, P.; Weng, P.; Gao, H.; Chen, G. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In Proceedings of the International World Wide Web Conferences, San Francisco, CA, USA, 13–17 May 2019; pp. 2091–2102.
30. Shafqat, W.; Byun, C.Y. Enabling "Untact" Culture via Online Product Recommendations: An Optimized Graph-CNN based Approach. *Appl. Sci.* **2020**, *10*, 5445. [CrossRef]
31. Wu, F.; Zhang, T.; de Souza, A.H., Jr.; Fifty, C.; Yu, T.; Weinberger, K.Q. Simplifying Graph Convolutional Networks. In Proceedings of the 2019 International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
32. Budhiraja, A.; Hiranandani, G.; Yarrabelly, N.; Choure, A.; Koyejo, O.; Jain, P. Rich-Item Recommendations for Rich-Users via GCNN: Exploiting Dynamic and Static Side Information. *arXiv* **2020**, arXiv:2001.10495.
33. Jianing, S.; Zhang, Y.; Ma, C.; Coates, M.; Guo, H.; Tang, R.; He, X. Multi-Graph Convolution Collaborative Filtering. In Proceedings of the 2019 IEEE International Conference on Data Mining, Beijing, China, 8–11 November 2019; pp. 1306–1311.
34. Zhiwei, L.; Wan, M.; Guo, S.; Achan, K.; Yu, P.S. BasConv: Aggregating Heterogeneous Interactions for Basket Recommendation with Graph Convolutional Neural Network. In Proceedings of the 2020 SIAM International Conference on Data Mining, Cincinnati, OH, USA, 7–9 May 2020; pp. 64–72.
35. Tong, Z.; Cui, B.; Cui, Z.; Huang, H.; Yang, J.; Deng, H.; Zheng, B. Cross-Graph Convolution Learning for Large-Scale Text-Picture Shopping Guide in E-Commerce Search. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1657–1666.
36. Lee, H.; Lee, J. Scalable deep learning-based recommendation systems. *ICT Express* **2019**, *5*, 84–88. [CrossRef]
37. Aminu, D.; Salim, N.; Rabiu, I.; Osman, A. Recommendation system exploiting aspect-based opinion mining with deep learning method. *Inf. Sci.* **2020**, *512*, 1279–1292.
38. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
39. Schlichtkrull, M.S.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the Semantic Web—15th International Conference, Heraklion, Crete, Greece, 3–7 June 2018; pp. 593–607.
40. Hamilton, W.L.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30, Proceedings of the Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017*; Neural Information Processing Systems Foundation Inc.: La Jolla, CA, USA, 2017; pp. 1025–1035.
41. Huang, W.B.; Zhang, T.; Rong, Y.; Huang, J. Adaptive Sampling towards Fast Graph Representation Learning. In *Advances in Neural Information Processing Systems 31, Proceedings of the Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, QC, Canada, 3–8 December 2018*; Neural Information Processing Systems Foundation Inc.: La Jolla, CA, USA, 2018; pp. 4563–4572.
42. Chiang, W.L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; Hsieh, C.J. Cluster-gcn: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 257–266.

43. Chen, J.; Zhu, J.; Song, L. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 941–949.
44. Fan, S.; Zhu, J.; Han, X.; Shi, C.; Hu, L.; Ma, B.; Li, Y. Metapath-Guided Heterogeneous Graph Neural Network for Intent Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2478–2486.
45. Nielsen, F. On the Jensen–Shannon symmetrization of distances relying on abstract means. *Entropy* **2019**, *21*, 485. [CrossRef] [PubMed]
46. Feng, L.; Wang, H.; Jin, B.; Li, H.; Xue, M.; Wang, L. Learning a distance metric by balancing kl-divergence for imbalanced datasets. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *49*, 2384–2395. [CrossRef]
47. Kaggle, RecSys Challenge 2015. Available online: https://www.kaggle.com/chadgostopp/recsys-challenge-2015 (accessed on 10 November 2020).
48. CodaLab, CIKM Cup 2016 Track 2: Personalized E-Commerce Search Challenge. Available online: https://competitions.codalab.org/competitions/11161 (accessed on 10 November 2020).
49. Perozzi, B.; Rfou, R.A.; Skiena, S. DeepWalk: Online learning of social representations. *arXiv* **2014**, arXiv:1403.6652.
50. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the World Wide Web Conference 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.
51. Mnih, A.; Salakhutdinov, R.R. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*; MIT Press: Vancouver, BC, Canada, 2007; Volume 20, pp. 1257–1264.
52. Sandvig, J.J.; Mobasher, B.; Burke, R. Robustness of Collaborative Recommendation Based on Association Rule Mining. In Proceedings of the 2007 ACM Conference on Recommender Systems, Minneapolis, MN, USA, 19–20 October 2007; pp. 105–112.
53. Koren, Y. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434.