entropy

MDPI

Article

MIGAN: Mutual-Interaction Graph Attention Network for Collaborative Filtering

Ahlem Drif [1,†] and **Hocine Cherifi** [2,*,†]

[1] Faculty of Sciences, Ferhat Abbas University, Setif 1, Setif 19000, Algeria
[2] Laboratoire d'Informatique de Bourgogne, University of Burgundy, 21078 Dijon, France
* Correspondence: hocine.cherifi@u-bourgogne.fr
† These authors contributed equally to this work.

Abstract: Many web platforms now include recommender systems. Network representation learning has been a successful approach for building these efficient recommender systems. However, learning the mutual influence of nodes in the network is challenging. Indeed, it carries collaborative signals accounting for complex user-item interactions on user decisions. For this purpose, in this paper, we develop a Mutual Interaction Graph Attention Network "MIGAN", a new algorithm based on self-supervised representation learning on a large-scale bipartite graph (BGNN). Experimental investigation with real-world data demonstrates that MIGAN compares favorably with the baselines in terms of prediction accuracy and recommendation efficiency.

Keywords: recommender systems; mutual influence; graph attention network; self-supervised; collaborative filtering

check for updates

Citation: Drif, A.; Cherifi, H. MIGAN: Mutual-Interaction Graph Attention Network for Collaborative Filtering. *Entropy* **2022**, *24*, 1084. https://doi.org/10.3390/e24081084

Academic Editor: David Geoffrey Green

Received: 9 June 2022
Accepted: 2 August 2022
Published: 5 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

In the literature, there are numerous techniques for building recommendation systems. They can be classified either as collaborative, content-based, or hybrid filtering approaches [1–4]. Collaborative filtering is the most influential. It relies on identifying users with similar tastes for item recommendations. It leverages their feedback to make suggestions to the active user. Collaborative recommender systems have been implemented in multiple application areas [5–7].

Learning effective user/item representations from their interactions and side information in recommender systems is a challenging issue. Since most data have a graph structure and the graph neural network (GNN) has superiority in representation learning, using GNN in recommender systems is a flourishing field of research. Several works are based on GNN to perform recommendations, and other tasks, such as the graph convolutional network (GCN) [8] and graph attention network (GAT) [9]. GAT computes the representation of nodes by adaptively combining their neighborhoods' vectors using a self-attention mechanism with trainable attention weights. Wang et al. [10] suggest a knowledge graph attention network (KGAT) for KG-based recommendations. To enrich the representation, the authors consider the implicit collaborative information of multi-hop neighbors. In the work [11], the authors propose a neighbor-aware graph attention network for recommendation tasks to model the implicit correlations of neighbors. Unlike previous attention networks, our current work determines the most relevant weights characterizing the mutual influence among item-users. In addition to learning the interaction between user interests and item embeddings, it integrates a new component accounting for the mutual influence of items carrying collaborative signals on user decisions. MIGAN learns deeply the most relevant weights representing the users mutual influence on an item. It exploits the complex relation between the user profile and the item attributes. Its main advantage lies in its ability to discriminate the relative influence of various interactions between nodes. Our main contributions summarize as follows:

- Our approach is based on self-supervised representation learning on a large-scale bipartite graph (BGNN). We have adapted this powerful representation for the recommendation task because its ability to model the dependencies between the nodes on a large scale.
- The collaborative filtering recommender based on interactive neural attention networks takes advantage of the encoding potential of interactive attention between users and items. It learns the most significant weights representing users' mutual effect on the item. Consequently, exploiting this information improves the recommender systems' accuracy.
- The empirical evaluation, including various real-world dataset, shows that MIGAN significantly outperforms the state-of-the-art baselines.

The rest of the paper is organized as follows. Section 2 describes the related literature. Section 3 presents in detail the proposed architecture. Section 4 discusses the experimental results. Section 5 summarizes the conclusions.

2. Related Work

Graph embedding models are one of machine learning's newest and fastest-growing subfields. Its strength is in its ability to take advantage of the intrinsic graph structure of a wide range of data types encountered in a wide range of applications. The graph format models a set of elements (represented by nodes) and their relationships (represented by edge) to capture structural information. Therefore, there have been proposed various graph embedding models in literature [12,13]. Node2Vec is an embedding model for converting graphs into numerical representations where each node in the network is used as a starting point to produce a corpus of random walks [14]. In a first-order random walk, each step is solely determined by the current state. The steps in a second-order random walk are determined by the current and prior states. The random walk corpus is fed through Word2Vec to build the node embeddings. Liang et al. [15] extend the variational autoencoders (VAEs) to collaborative filtering for implicit feedback. It models the collaborative information into multinomial likelihood (MultiVAE) for the data distribution to sample prediction for items on the long tail. Drif et al. [16] develop an ensemble variational autoencoder framework for recommendations (EnsVAE) that specifies a procedure to transform sub-recommenders' predicted utility matrix into interest probabilities that allow the VAE to represent the variation in their aggregation. This architecture is based on two components: (1) GloVe content-based filtering recommender (GloVe-CBF) that exploits the strengths of embedding-based representations and stacking ensemble learning techniques to extract features from the item-based side information, and (2) a variant of neural collaborative filtering recommender, named the Gate Recurrent Unit-based Matrix Factorization (GRU-MF) recommender. It models a high level of non-linearities and exhibits interactions between users and items in latent embeddings, reducing user biases towards items that are rated frequently by users. Weng Lo et al. [17] suggested the graph flow data's extensive structural information based on a graph neural networks (GNNs). The latter works with the message passing concept, where a node collects its neighbors features and sends them to a node as a message. In recent years, the graph attention network approach has developed rapidly. Wang et al. [18] introduced a multi-dimension interaction-based attentional knowledge graph neural network (MI-KGNN) to improve recommendations based on knowledge graph (KG). MI-KGNN explores the interaction between users and the neighborhood during embedding propagation. In [10], the authors proposed a knowledge graph attention network (KGAT) modeling the high-order connectivity in knowledge graphs. It exploits the attention mechanism to determine important neighbors. In [19], the authors propose a multi-view graph attention network (MV-GAN) based on the heterogeneous information networks for the recommendation. They create attention networks at the node and path levels to learn user and product representations from every view. A view-level attention mechanism is developed to integrate various relationship types in multiple views cooperatively. Liu et al. [20] propose a contextualized graph attention network (CGAT) based

on an entity's local and non-local context data in a knowledge graph. CGAT implements a graph attention method to record local context information while considering users' unique preferences for entities. The non-local context of an entity is also extracted using a biased random walk sampling method by CGAT. In fact, propagating information from nodes across the network is done during numerous iterations. The aggregated information at each node (node embedding) is a memory- and time-consuming task. Due to this fact, many GNNs for recommendation suffer from scalability limitations, unpredictable memory, and computational resource requirements on large graphs. To overcome these drawbacks, our work considers node representation learning on large-scale bipartite graphs.

3. Mutual-Interaction Graph Attention Network Approach

The proposed architecture considers the mutual collaborative information of the user's preference on the whole neighboring item to enrich the representation. It is based on self-supervised representation learning on a large-scale bipartite graph (BGNN) [21]. Figure 1 illustrates the model architecture in detail. We first formulate the recommendation task on the bipartite graph. Secondly, we introduce the embedding representation based on BGNN and then describe the mutual interaction graph attention mechanism.

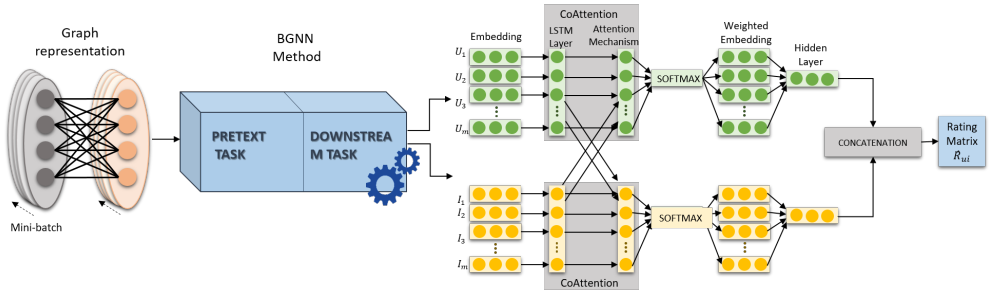

Figure 1. MIGAN Architecture.

3.1. Problem Formulation

Let $U = u_1, u_2, \ldots, u_n$ and $I = i_1, i_2, \ldots, i_m$ be the sets of users and items, respectively, where n is the number of users, and m is the number of items. We assume that $R^{n \times m}$ is the user-item rating matrix.

We formulate the recommendation task as a prediction problem as follows:

$\hat{R}$: utility matrix; $\hat{r}_u$: predicted rating for each user $u \in U$; $U = u_1, u_2, \ldots, u_n$: is the sets of users, where n is the number of users; $I = i_1, i_2, \ldots, i_m$ is the set of the items , where m is the number of items; R_{ui}: is the ground truth rating assigned by the user u on the item i.

We define the utility matrix as:

$$\hat{R}_{ui} = PQ^T = \sum_{k=1}^{K} p_{uk} q_{ki} \tag{1}$$

where: K is latent space's dimension.

$S'_{(n \times m)}$: is the inner product of both user and item latent vectors. It is decomposed by the matrix factorization method into $P \in \mathbb{R}^{N \times K}$ and $Q \in \mathbb{R}^{M \times K}$.

To normalize $\hat{R}$, we apply the min/max scaling:

$$minmax(x) = \frac{x - min}{max - min} \quad \forall x \in \hat{r}(ui) \tag{2}$$

where: $min = \min(r_{ui})$ is the minimal rating;

$max = \max(r_{ui})$: is the maximal rating. The normalization eliminates user bias. In other words, users have different ways to rate the items. Some would only give high ratings to items they like, while others do the opposite. Normalizing users' ratings hide

their bias by mapping them to values between 0 and 1. The lowest rating of each user is associated with 0, while 1 represents their highest value. It assists in leveraging more accurate collaborative-based recommendations.

Table 1 reports the notations used in the rest of this paper.

Table 1. Notation and descriptions.

Symbols	Definitions and Descriptions
r_{ui}	User u's rating for item i
p_u	The user u's embedding.
q_i	The item i's embedding.
g	Long-Short-Term-Memory function
x_u	The user embedding layer followed by LSTM layer
x_i	The item embedding layer followed by LSTM layer
h	The Multi-Layer-Perception application
$lstm_u$	The user LSTM layer following by MLP
$lstm_i$	The item LSTM layer following by MLP
α_u^*	Attention network function for user u
α_i^*	Attention network function for item i
α_u	The last attention weights for user u
α_i	The last attention weights for item i
C_{ui}	User-item space
C_t	Text space
$\oplus$	The concatenation operator
$r_{ui}^{'}$	User u's rating expected value for item i
W, b	The weight and bias in neural network
U, I	Nodes of bipartite graph
X_u, X_i	lists of Features
B_u, B_i	adjacency matrix

A bipartite graph is composed of two independent sets of vertices, U_1 and I_1. The edges connect a vertex from one set U_1 to one in I_1.

We define Bipartite Graphs as follows: Let $G = (U_1, I_1, E)$ be a bipartite graph. e_{ij} represents the edge between u_i and i_j.

$B_u \in \mathbb{R}^{M \times N}$ is the incidence matrix for U_1. $B_i \in \mathbb{R}^{N \times M}$ is the incidence matrix for I_1. Where

$$B_u(i,j) = \begin{cases} 1 & \text{if } e_{ij} \in E, \\ 0 & \text{if } e_{ij} \notin E. \end{cases} \tag{3}$$

$X_u \in \mathbb{R}^{M \times P}$: is defined as a feature matrix of node u_i (X_i is similarly written).

Our work is based on the self-supervised node representation learning model [21] that can employ topology information as well as separate node attributes from two domains to increase the recommendation performance for large graph.

3.2. Embedding Representation Based on Bipartite Graph Neural Networks (BGNN)

He et al. [21] proposed a a self-supervised representation learning framework for large-scale bipartite graphs. In this section, we adapt the outputs of this BGNN architecture, thus,

we can deploy it in our recommendation system. Let us define the following notions: $H_u \in \mathbb{R}^{P\prime}$ ($H_i \in \mathbb{R}^{Q\prime}$, respectively): is nodes embedding representation for U_1 (I_1, respectively). f_{emb}: is an embedding model given by θ parameters.

The embedding of distinct node features X_u and X_i is written as:

$$H_u, H_i = f_{emb}(X_u, B_u, X_i, B_i, \theta) \tag{4}$$

The architecture of f_{emb} is based on two functions: (i) Inter-Domain Message Passing (IDMP) and (ii) Intra-Domain Alignment (IDA). We will describe briefly these functions and show how we prepare the outputs for the recommendation task (interested reader can refer to [21] for more details). IDMP enables one domain to aggregate information from the other domain, through the linked edges, as follows:

$$H_{i\to u} = f_u(X_i, B_u, \theta) \tag{5}$$

$$H_{u\to i} = f_i(X_u, B_i, \theta) \tag{6}$$

such as: f_u (resp. f_i): represents the IDMP function for this domain.

$H_{i\to u}$ (resp. $H_{u\to i}$): is the flow of aggregated information from I_1 (resp. U_1) to U_1 (resp. I_1). After that, the Intra-Domain Alignment (IDA) is deployed for theses two distinct features into a single representation. After the self-supervised training, the algorithm gives the domains representation of H_u^1 and H_I^1. The adversarial loss L_{adv} is used to compute the best results.

$$Loss_u = L_{adv}(H_{i\to u}, X_u) \tag{7}$$

$$Loss_i = L_{adv}(H_{u\to i}, X_i) \tag{8}$$

Thus, the Inter-Domain Message Passing (IDMP) is expressed as:

$$H_{i\to u}^{(K)} = \sigma(B_u^{'} H_i^{(K)} W_u^K) \tag{9}$$

$$H_{u\to i}^{(K)} = \sigma(B_i^{'} H_u^{(K)} W_i^K) \tag{10}$$

where: $B_u^{'} = D_u^{-1} B_u$ is the normalization of B_u (D_u is the degree matrix of B_u). By normalizing the incidence matrix of the graph, the algorithm can effectively reduce the computational cost. σ is the activation function ReLU [22]. K denotes the depth index of the hidden features of the nodes in set U_1 (resp. I_1).

Let σ is the Intra-Domain Alignment (IDA) discriminator and ϕ is the IDMP generator. The discriminator loss function is expressed as follows:

$$L_D(\sigma|\phi) = \frac{1}{M}\sum_{j=1}^{M} logP_{\sigma,\phi}(source = 0|h_{u(j)}) - \frac{1}{N}\sum_{j=1}^{N} logP_{\sigma,\phi}(source = 1|h_{i\to u(j)}) \tag{11}$$

where: $P_{\sigma,\phi}$ $(source = 1|h)$ is the probability that the input feature vector h is from the source domain $H_{i\to u}$.

The implementation for the self-supervised representation learning on large scale bipartite graph (BGNN) for the recommendation task is summarized in Algorithm 1.

Algorithm 1 Bipartite Graph Neural Networks for recommendation task.

Input
X_u : *User features list*
X_i : *Item features list*
R : *Rating matrix*
Output
emb_u : *Users' Embedding representation based on BGNN.*

emb_i : *Items' Embedding representation based on BGNN.*

Begin
phase 1 ▷ Extract the graph from the rating matrix
$B_u, B_i = GetBipartiteGraph(R)$
phase 2 ▷ Computing embeddings for each item and user
$H_u^0 = X_u \qquad H_i^0 = X_i$

for $j = 0, 1, \ldots, K$ **do**
$H_{u \to i}^j = \textbf{IDMP}(B_i, H_u^K)$

$H_{i \to u}^j = \textbf{IDMP}(B_u, H_i^K)$

$H_u^K = \textbf{IDA}(H_{u \to i}^j)$

$H_i^K = \textbf{IDA}(H_{i \to u}^j)$
Endfor
phase 3 ▷ embeddings preparation
$emb_u = GetEmbeddings(H_u^K)$
$emb_i = GetEmbeddings(H_i^K)$
return emb_u, emb_i

3.3. The Interactive Attention Network Recommender

The interactive attention network recommendation system aim at identifying latent features that show the users and items mutual influence. The attention mechanism has been shown to be useful in a variety of machine learning applications, including image/video captioning [23,24]. Our proposed interactive concept extracts each participant's contribution from their compressed representation. Thus, it allows the proposed recommendation framework to model efficiently the interaction characteristic. This attention network model figures out which weights best represent the users' mutual effect on the item. Figure 2 explains the mutual interactions between users and items.

Algorithm 2 shows the architecture of the proposed interactive attention network. In order to anticipate a distribution over the items, we create combined user and item interactive attention maps. As a result, the co-attention mechanism detects a correlation between items and users and calculates the likelihood that an item will be of interest to indirect comparable individuals.

The first embedding layers e_u and e_i captures latent features of users p_u and items q_i. They are followed by Long-Short-Term-Memory (LSTM) layers to learn long sequences with long time lags. Each LSTM state includes two inputs: the current feature vector and the output vector h_{t-1} from the previous state. Its output vector is h_t. We chose to apply the LSTM model as it exhibits interactions between users and items in latent embeddings. Each node embedding layer is chained with an LSTM layer that contains recurrent modules enabling long-range learning. Information from nodes neighbors gradually enhances the subsequent feature representation because LSTM has an augmented hidden state with non-linear mechanisms. It allows propagating without modification, updating, or reset-

ting states using simple learned gating functions. The LSTM representation is expressed as follows:

$$h_{tu} = g_1(p) \tag{12}$$

$$h_{ti} = g_2(q) \tag{13}$$

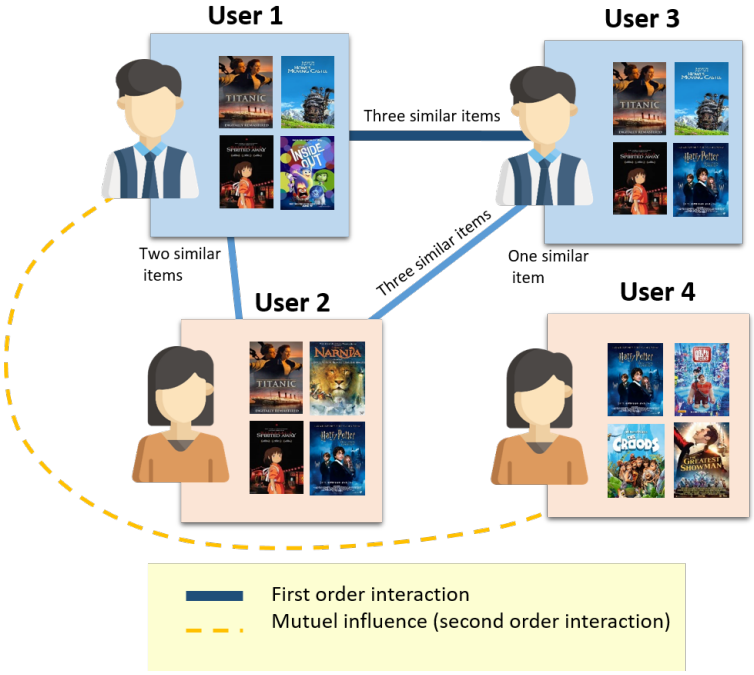

Figure 2. The interactive attention network recommender. In this example, user 1 and user 2 rate three similar items. They have strong interactive attention. User 2 and user 3 rate an item not seen by user 1. Therefore, one can deduce that this new item can also attract user 1. It is a first-order interaction. Moreover, one can deduce a mutual influence based on entities' dependencies at more than a first order interaction level. For example, user 1 influences user 4 (a similar user of user 3), generating a recommendation based on user 3 preferences.

The learned representation is H_p and H_q, respectively, with $d \times n$ dimensions for H_p and $d \times m$ for H_q. Users and items embedded inputs are projected into a vector representation space using the attention technique. In fact, this representation models the high-order non-linear mutual relationship. For the interactive attention mechanism, we build an attention maps in order to predict a distribution over the items. For this purpose, we compute a matrix $L = tanh(H_p^\top W_{pq} H_q)$, where $L \in \mathbb{R}^{n \times m}$, and W_{pq} is a $d \times d$ a learnable parameters matrix. The features co-attention maps is defined as:

$$\begin{aligned} \alpha_p^* &= tanh(W_p H_p + (W_q H_q) L^\top) \\ \alpha_q^* &= tanh(W_q H_q + (W_p H_p) L) \end{aligned} \tag{14}$$

The interactive attention model uses a tangent function to model the mutual interactions between users and items. Afterward, we compute the the probability distribution over the embedding space. The softmax function is used to generate the attention weights:

$$\alpha_u = Softmax(f(\alpha_p^*)) \tag{15}$$

$$\alpha_i = Softmax(f(\alpha_q^*)) \tag{16}$$

where f: is a multi-layer neural network.

Then, the high order interaction latent space of users and items is given by:

$$f_1 = [\beta\prime_u \oplus \beta\prime_i] \tag{17}$$

where β_p and β_q: are the derived attention weights.

As a result, the predicted matrix $\hat{R_{ui}}$ is defined as:

$$\hat{R_{ui}} = f(f_1) \tag{18}$$

where f: is a dense layer using a sigmoid activation function.

Finally, we train the model to minimize the loss function which is the Mean Absolute Error (MAE):

$$L(R_{ui}, \hat{R}_{ui}) = \frac{1}{|C|} \sum_{(u,i) \in C} |(R_{ui} - \hat{R_{ui}})| \tag{19}$$

Algorithm 2 CoAttention: The interactive attention network recommender.

Input
lstmU: user's lstm : size d × n
lstmI: item's lstm: size d × m
Output
att$_{ui}$: *The Interactive Attention between users and items*
Begin
phase 1 ▷ initialization of weights
W_u : *size* $n \times d$
W_i : *size* $m \times d$
W_{ui} : *size* $d \times d$
b_u : *size* $d \times n$
b_i : *size* $d \times m$
phase 2 ▷ tanh function application
S = lstmI
G = lstmU
$F = \mathbf{tanh}(S^t W_{ui} G)$
$\alpha_u^* = \mathbf{tanh}(W_u G + (W_i S) F^t)$
$\alpha_i^* = \mathbf{tanh}(W_i S + (W_u G) F)$
phase 3 ▷ Softmax function application
$\alpha_u = \mathbf{softmax}(f(b_u \alpha_u^*))$
$\alpha_i = \mathbf{softmax}(f(b_i \alpha_u^*))$
$\beta_u = O(\alpha_u)$
$\beta_i = O(\alpha_i)$
▷ O: is the batch.dot() function from Keras backend that is used between two tensors $((\alpha_p)^t$ and $(G)^t)$, $(\alpha_(q)^t$ and $(S)^t)$ respectively.
phase 4 ▷ Each output of function *O* are transposed and then used as input into a product function. After that, both results ($\beta\prime_u$ $\beta\prime_i$) can be summed by a concatenate function.
att$_{ui}$ = **concatenation**$(\beta\prime_u, \beta\prime_i)$
return att$_{ui}$

Algorithm 3 summarizes the overall mutual-interaction graph attention network approach. Network representation learning can tackle the recommendation problems by embedding nodes into a low-dimensional space $\mathbb{R}^d$. Furthermore, this adapted BGNN representation for recommendation task improves multi-hop relationship modeling and the training accuracy. Unlike previous research on graph neural networks for recommendation that only learn complex relationship between the target and their neighbors using attention network, our work learn the most important weights representing the users' mutual influence on the item based on the interactive attention.

Algorithm 3 Migan: Mutual-Interaction Graph Attention Network.

Input
X_u : *User features list*
X_i : *Item features list*
U : *List of user* : $size = n$
I : *List of item* : $size = m$
R : *Rating matrix*
Output P_{ui} : *prediction matrix*
Begin
phase 1 ▷ Preparing data to be passed to the BGNN
foreach u $\in U, i \in V$ **do** $Rj =$ **minmax(R)**
phase 2 ▷ extracting embeddings by BGNN-Class()

$$emb_u, emb_i = BGNN(X_u, X_i, R_j)$$

phase 3 ▷ User and Item embedding are followed by LSTM layers.

$$lstm_u = LSTM(emb_u)$$
$$lstm_i = LSTM(emb_i)$$

phase 4 ▷ Applying Attention mechanism

$$att_u = Attention(lstm_u)$$
$$att_i = Attention(lstm_i)$$
$$att_{ui} = CoAttention(lstm_u, lstm_i)$$

phase 5 ▷ Concatenating The outputs

$$ATT = concatenation(att_u, att_i, att_{ui})$$
InteractiveAttention = BuildModel(ATT) ;
InteractiveAttention.trainModel(D);
return $P_{ui} = InteractiveAttention.predict(\varphi)$

4. Experiments and Discussion

We conduct our experiments on MovieLens 1M that is commonly used for benchmarking recommendation frameworks. The MovieLens dataset [25] is a real, timestamped 5-star ratings of the MovieLens platform users on various films. The selected dataset contains 1 million ratings from 6000 users over 4000 movies. Table 2 shows the dataset description. We divide the datasets into 75% training data and 25% testing data in a stratified way.

Table 2. MovieLens 1M description.

# *Users*	6040
# *Movies*	3883
# *Ratings*	1000209
Sparsity %	95.5%
Item	Genomic Tags
User	Demographics

We evaluate MIGAN using Mean Average Precision and Normalized Discounted Cumulative Gain. The mean average precision (MAP) measures the accuracy of information retrieval. The results of recommender systems are frequently pruned to return the Top-k components. The value of k varies based on the application: a system might display the top three trending items or the best ten items that meet the current user's preferences. The Normalized Discounted Cumulative Gain (NDCG) calculates an item's normalized usefulness depending on its position in the final list. It is used to assess the Top-k recommended items' ranking quality. Interested readers can refer to [16] for more details.

4.1. Hyperparameters Analysis

Here, we report the hyperparameters analysis phase, which is performed separately for each MIGAN recommender variant. Evaluation is done with $MAP@k$ and $NDCG@k$.

The main idea of our model is to pass the results of the BGNN through a neural network and then apply the attention model to it. Consequently, we perform the analysis on 1–6 variants and focus on the learning algorithms used by each hyperparameter. We have 3 hyperparameters and two kinds of recurrent networks: Embedding output size = 50, 75, 100, Neural network = LSTM, GRU. It gives us six variants. The architecture of each variant is as follows:

- **variant 1:** BGNN output size =75 $\times$ 1 | Neural Network = LSTM.
- **variant 2:** BGNN output size = 100 $\times$ 1 | Neural Network = LSTM.
- **variant 3:** BGNN output size =50 $\times$ 1 | Neural Network = LSTM.
- **variant 4:** BGNN output size = 75 $\times$ 1 | Neural Network = GRU.
- **variant 5:** BGNN output size = 100 $\times$ 1 | Neural Network = GRU.
- **variant 6:** BGNN output size = 50 $\times$ 1 | Neural Network = GRU.

As shown in Table 3, **Variant 1** scores better than the other variants. Thus, it is picked for further tweaking.

Table 3. The best scoring MIGAN variant.

	Mean Average Precision			**Normalized DCG**		
Variant	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
Variant 1	**0.85**	**0.83**	**0.81**	**0.71**	**0.78**	**0.79**
Variant 2	0.82	0.78	0.76	0.65	0.72	0.76
Variant 3	0.80	0.77	0.76	0.66	0.73	0.76
Variant 4	0.79	0.74	0.773	0.62	0.71	0.73
Variant 5	0.79	0.73	0.70	0.60	0.71	0.72
Variant 6	0.77	0.76	0.73	0.63	0.73	0.75

We generate the utility matrix based on the learned embeddings. Figure 3 illustrates the hyperparameters analysis.

We explore a range of values for each hyperparameter as reported below:

- Dimensions of the embedding $\alpha \in [30, 100]$;
- Number of dense layers after the co-attention $\theta \in [2, 20]$;
- Number of neurons per dense layer $\tau \in [30, 150]$;
- Activation function used in the dense layers $\sigma \in \{selu, elu, relu\}$;
- Optimizer $\lambda \in \{sgd, adam, adagrad\}$.

Results illustrate that we achieve the best performance for the following settings: $\alpha = 50$, $\theta = 3$, $\tau = 100$, $\sigma = elu$, $\lambda = Adam$.

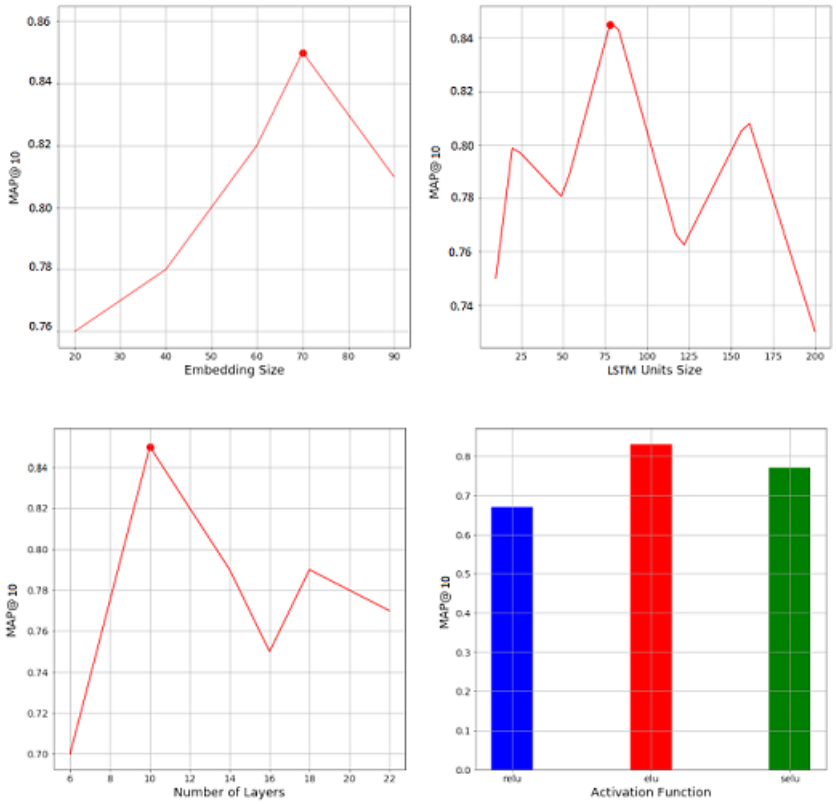

Figure 3. Hyperparameter searching MIGAN filtering recommendation system.

4.2. Performance Comparison with the Baselines

In this subsection, MIGAN's final benchmark results are compared to the outcomes of certain baseline recommender systems. We execute the baselines in the same evaluation environment as MIGAN to guarantee that comparisons are fair. Furthermore, we deploy the *NeuRec* library. Furthermore, we deploy the *NeuRec* library. It is released on GitHub as open-source software under an MIT license, and it implements 33 neural recommender systems [26].

We compare the proposed recommender framework with the following baselines:

- *The stacked content-based filtering recommender:* the work [16] developed a content-based recommender system based on the stacking ensemble learning.
- *Neural collaborative filtering (NCF)* [27]: this work developed a recommender framework that uses the multi-layer perceptron to exploit the user-item interaction.
- *Variational Autoencoders for Collaborative Filtering* (MultiVAE) [15]: This approach investigates the collaborative information in a multinomial distribution to recommend items on the long tail.
- *Node2Vec embedding:* We propose a variant of MIGAN architecture, which deploys Node2Vec embedding representation instead of BGNN.

Table 4 reports the performance of the various approaches under investigation using the MovieLens dataset. According to mean average precision, MIGAN outperforms the baselines. Indeed, it models the higher-order feature interactions. Figure 4 shows the MAP@k evaluation versus k-top items. As the MAP measure indicates the fraction of relevant articles in the top k suggestions averaged over all users, MIGAN creates a tailored recommendation. To put it differently, MIGAN outperforms the other models in recalling relevant items for the user. It retains the user–item interaction and generates a user-specific task recommendation. Furthermore, both recommender-based BGNN and the recommender-based Node2Vec are quite competitive. For example, MIGAN and Node2Vec achieve MAP@10 = 0.85, and MAP@10 = 0.84, respectively, outperforming the other baselines. The training loss equals 0.23 with dimensions d = 70 and 0.32 with d = 50 for BGGN

and Node2vec, respectively. Node2Vec is a random walk-based node embedding method producing high memory consumption for large graphs. In contrast, the cascaded training used in BGGN does not involve loading the entire graph into memory. Consequently, it reduces the memory cost and training time. The MultiVAE recommender scores poorly. Indeed, it does not use a enough rich representation of data semantic. The neural collaborative filtering approach (Neural CF) shows a good score with k = 10, MAP@10 = 0.74 due to the appropriate representation of the interaction between user and item.

Table 4. Recommendation performance (%) of compared approaches conducted on MovieLens 1M dataset. We generate Top 10, 30, and 50 items for each user. The best score of MAP@k and NDCG@k are highlighted with a bold font.

	Mean Average Precision			**Normalized DCG**		
Rec sys	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
Glove-Cbf	0.82	0.78	0.77	0.67	0.74	0.77
Node2Vec	0.84	0.82	0.81	0.55	0.65	0.69
MultiVAE	0.62	0.58	0.54	0.57	0.62	0.65
Neural CF	0.74	0.68	0.65	0.68	0.73	0.76
MIGAN	**0.85**	**0.83**	**0.81**	**0.71**	**0.78**	**0.79**

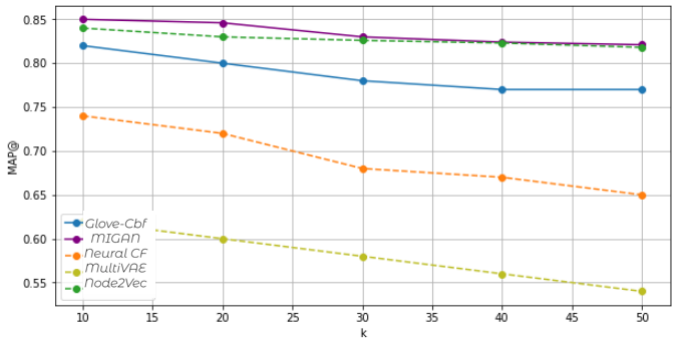

Figure 4. Performance results of Top-K recommended lists, according to MAP. The ranking position K ranges from 1 to 50.

Figure 5 shows that MIGAN exhibit a high NDCG score on MovieLens. Its Top-k recommendation list is quite similar to the ground-truth list. MIGAN boosts the modeling of user–item interaction. Indeed, the more interested the users are in an item, the more likely users with similar preferences to recommend it. Note that the Node2Vec also presents good NDCG scores. The graph recommender effectively obtains the user's overall interest built by the neighborhood representation.

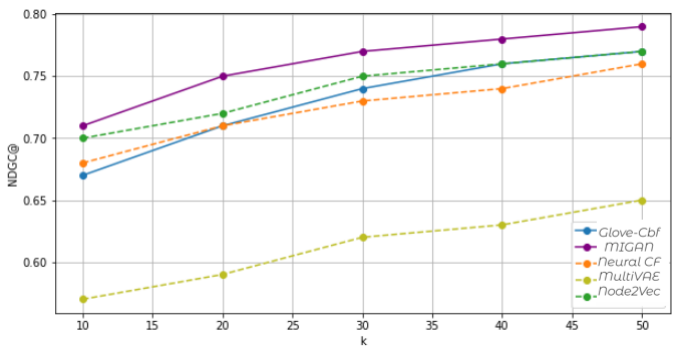

Figure 5. Performance results of Top-K recommended lists according to NDGC. The ranking position K ranges from 1 to 50.

5. Conclusions

We propose and investigate a graph recommender where each user's recommended content is accurate and personalized. MIGAN is a collaborative filtering (CF) system. A neural graph represents the item and users. It computes a representation of nodes by combining their neighborhoods' vectors according to their mutual influence interaction by utilizing a co-attention mechanism with trainable attention weights. The attention weights are adjustable parameters computed by aggregating neighbor vectors. This neural graph architecture predicts the ratings assigned by the users to the items.

We perform a comparative evaluation of several configurations for the RS using the well-known dataset MovieLens. We use two metrics to quantify its accuracy: the mean average precision (MAP) and the normalized discounted cumulative gain (NDCG). The first is about the accuracy of the recommendation. The second is about the ranking of the recommended items. Comparing MIGAN performance with some baselines shows that it outperforms all its alternatives in MAP and NDGC scores. However, it would be very interesting to focus on a specific domain for recommendation tasks, such as taking the knowledge graph to distill attribute-based collaborative signals and compare MIGAN performance with knowledge graph attention network models. Thus, future work will investigate this framework for collaborative knowledge graphs involving contextual and semantic data. Future work will investigate this framework for other recommendation tasks involving contextual data.

Author Contributions: A.D. carried out the study, designed the methodology, conceived the framework, and drafted and finalized the manuscript. H.C. supervised the study, participated in the analysis of the results, and reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The links to publicly datasets: https://dl.acm.org/doi/10.1145/2827872 (accessed on 1 March 2022).

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

Abbreviations

The following abbreviations are used in this manuscript:

RS	Recommender Systems
MIGAN	Mutual-Interaction Graph Attention Network
BGNN	Bipartite Graph Neural Networks
IDMP	Inter-Domain Message Passing
IDA	Intra-Domain Alignment
LSTM	Long-Short-Term-Memory
GRU	Gate Recurrent Unit
MAP	Mean Average Precision
NDCG	Normalized Discounted Cumulative Gain
Neural-CF	Neural Collaborative Filtering
Glove-Cbf	Glove Content-based filtering recommender
MultiVAE	Variational Autoencoders recommender

References

1. Kulkarni, S.; Rodd, S.F. Context Aware Recommendation Systems: A review of the state of the art techniques. *Comput. Sci. Rev.* **2020**, *37*, 100255. [CrossRef]
2. Fayyaz, Z.; Ebrahimian, M.; Nawara, D.; Ibrahim, A.; Kashef, R. Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Appl. Sci.* **2020**, *10*, 7748. [CrossRef]
3. Chen, R.; Hua, Q.; Chang, Y.S.; Wang, B.; Zhang, L.; Kong, X. A Survey of Collaborative Filtering-Based Recommender Systems: From Traditional Methods to Hybrid Methods Based on Social Networks. *IEEE Access* **2018**, *6*, 64301–64320. [CrossRef]

4. Berkani, L.; Belkacem, S.; Ouafi, M.; Guessoum, A. Recommendation of users in social networks: A semantic and social based classification approach. *Expert Syst.* **2021**, *38*, e12634. [CrossRef]
5. Drif, A.; Zerrad, H.E.; Cherifi, H. Context-Awareness in Ensemble Recommender System Framework. In Proceedings of the 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia, 12–13 June 2021; pp. 1–6.
6. 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.
7. Drif, A.; Guembour, S.; Cherifi, H. A Sentiment Enhanced Deep Collaborative Filtering Recommender System. In Proceedings of the International Conference on Complex Networks and Their Applications, Madrid, Spain, 1–3 December 2020; pp. 66–78.
8. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
9. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
10. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.S. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 950–958.
11. Song, J.; Chang, C.; Sun, F.; Song, X.; Jiang, P. NGAT4Rec: Neighbor-Aware Graph Attention Network For Recommendation. *arXiv* **2020**, arXiv:2010.12256.
12. Goyal, P.; Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowl.-Based Syst.* **2018**, *151*, 78–94. [CrossRef]
13. Cai, H.; Zheng, V.W.; Chang, K.C.C. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1616–1637. [CrossRef]
14. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
15. Liang, D.; Krishnan, R.G.; Hoffman, M.D.; Jebara, T. Variational autoencoders for collaborative filtering. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 689–698.
16. Drif, A.; Zerrad, H.E.; Cherifi, H. EnsVAE: Ensemble Variational Autoencoders for Recommendations. *IEEE Access* **2020**, *8*, 188335–188351. [CrossRef]
17. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. E-GraphSAGE: A Graph Neural Network based Intrusion Detection System. *arXiv* **2021**, arXiv:2103.16329.
18. Wang, Z.; Wang, Z.; Li, X.; Yu, Z.; Guo, B.; Chen, L.; Zhou, X. Exploring Multi-dimension User-Item Interactions with Attentional Knowledge Graph Neural Networks for Recommendation. *IEEE Trans. Big Data* **2022**, 155–170. [CrossRef]
19. Chen, L.; Cao, J.; Wang, Y.; Liang, W.; Zhu, G. Multi-view graph attention network for travel recommendation. *Expert Syst. Appl.* **2022**, *191*, 116234. [CrossRef]
20. Liu, Y.; Yang, S.; Xu, Y.; Miao, C.; Wu, M.; Zhang, J. Contextualized graph attention network for recommendation with item knowledge graph. *IEEE Trans. Knowl. Data Eng. arXiv***2021**, arXiv:2004.11529v1.
21. He, C.; Xie, T.; Rong, Y.; Huang, W.; Huang, J.; Ren, X.; Shahabi, C. Cascade-BGNN: Toward Efficient Self-supervised Representation Learning on Large-scale Bipartite Graphs. *arXiv* **2019**, arXiv:1906.11994.
22. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
23. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2048–2057.
24. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
25. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst. (TIIS)* **2015**, *5*, 1–19. [CrossRef]
26. Wu, B.; Sun, Z.; He, X.; Wang, X.; Staniforth, J. *NeuRec: Next RecSys Library*; National Natural Science Foundation: Beijing, China, 2019.
27. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.