There were fierce debates on whether the non-linear embedding propagation of GCNs is appropriate to GCN-based recommender systems. It was recently found that the linear embedding propagation shows better accuracy than the non-linear embedding propagation. Since this phenomenon was discovered especially in recommender systems, it is required that we carefully analyze the linearity and non-linearity issue. In this work, therefore, we revisit the issues of i) which of the linear or non-linear propagation is better and ii) which factors of users/items decide the linearity/non-linearity of the embedding propagation. We propose a novel Hybrid Method of Linear and non-linear collaborative filtering (CF). The proposed model yields the best accuracy in three public benchmark datasets. Moreover, we classify users/items into the following three classes depending on our gating modules’ selections: Full-Non-Linearity (FNL), Partial-Non-Linearity (PNL), and Full-Linearity (FL). We found that there exist strong correlations between nodes’ centrality and their class membership, i.e., important user/item nodes exhibit more preferences towards the non-linearity during the propagation steps. To our knowledge, we are the first who design a hybrid method and report the correlation between the graph centrality and the linearity/non-linearity of nodes. All HMLET codes and datasets are available at: https://github.com/qbxlvnf11/HMLET.

KEYWORDS
Recommender Systems, Collaborative Filtering, Embedding Propagation, Graph Neural Network

ACM Reference Format:
Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park, and Sang-Wook Kim. 2022. Linear, or Non-Linear, That is the Question!. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM ’22), February 21–25, 2022, Tempe, AZ, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498501

1 INTRODUCTION
Recommender systems, personalized information filtering (IF) technologies, can be applied to many services, ranging from E-commerce, advertising, and social media to many other online and offline service platforms [38]. One of the most popular recommender systems, collaborative filtering (CF), provides personalized preferred items to users by learning user and item embeddings from their historical user-item interactions [3, 4, 7, 8, 16, 18, 23, 24, 30, 34].

One of the mainstream research directions in recommender systems is how to learn high-order connectivity of user-item interactions while filtering out noises. Recently, GCN-based CF methods became popular in recommender systems because they show strong points to capture such latent high-order connectivity. Since existing GCNs are originally designed for graph or node classification tasks on attributed graphs, however, two limitations had been raised out when it comes to GCN-based CF methods: training difficulty [6, 15, 37] and over-smoothing [5, 6, 15]. Over-smoothing degrades the recommendation accuracy by considering the connectivity information too much [6]. To overcome these problems, a couple of linear GCNs (linear embedding propagation-based GCNs) were proposed [6, 15]. These methods effectively alleviate the aforementioned two limitations and show superior performance over non-linear GCNs (non-linear embedding propagation-based GCNs). Even though linear GCNs show the state-of-the-art performance in many benchmark CF datasets, it is questionable in our opinion whether they can properly handle users and items with various characteristics and whether linear GCNs are consistently superior to non-linear GCNs in all cases. In addition, we are curious about, if one outperforms the other, which factors of graphs decide it.

To this end, we propose a Hybrid Method of Linear and non-linear collaborative filtering (HMLET, pronounced as Hamlet), a GCN-based CF method. HMLET has the following key design points: i) We adopt a gating concept to decide between the linear
We let our gating modules decide which of the linear or non-linear propagation is used for a certain node at a certain layer instead of relying on manually designed architectures. This gating mechanism’s key point is how to generate appropriate one-hot vectors. For this purpose, we adopt the Gumbel-softmax [11, 27].

To our knowledge, we are the first who combines the linear and non-linear propagation in a systematic way, i.e., via the gating in our paper. Our gating mechanism can be considered as a sort of neural architecture search (NAS) for the GCN-based CF method. However, our proposed mechanism is more sophisticated because it provides the switching function for each user/item and the overall GCN architecture can be varied from a node’s perspective to another.

We conduct experiments with three benchmark CF datasets and compare our HMLET with various state-of-the-art CF methods in terms of the normalized discounted cumulative gain (NDCG), recall, and precision. We also define several variations of HMLET in terms of the locations of the non-linear propagation layers (see Fig. 1). Among all of them, HMLET(End) shows the best performance in all datasets. Furthermore, we define three classes of nodes, i.e., users and items, depending on their preferences on the linear or non-linear propagation: Full-Non-Linearity (FNL), Partial-Non-Linearity (PNL), and Full-Linearity (FL). An FNL (resp. FL) node means that our gating modules choose the non-linear (resp. linear) propagation every time for the node and a PNL node has a mixed characteristic. At the end, we analyze the class-specific characteristics in terms of various graph centrality metrics and reveal that there exist strong correlations between the graph centrality, i.e., the role of a node in a graph, and the linear/non-linear gating outcomes (see Table 1). Our discovery shows that recommendation datasets are complicated because the linearity and non-linearity are mixed.

Contributions of our paper can be summarized as follows:

- We propose HMLET, which dynamically selects the best propagation method for each node in a layer.
- We reveal that the role of a node in a graph is closely related to its linearity/non-linearity, e.g., our gating module prefers the non-linear embedding propagation for the nodes with strong connections to other nodes.
- Our experiments on three benchmark datasets show that HMLET outperforms baselines in yielding better performance.

2 RELATED WORKS

In this section, we review recommender systems and the Gumbel-softmax used in our proposed gating module.

2.1 Recommender Systems

Traditional recommender systems have focused on matrix factorization (MF) techniques [19, 23]. Typical MF-based methods include BPR [30] and WRMF [18]. These MF-based methods simply learn relationships between users and items via dot products. Therefore, they have limitations in considering potentially complex relationships between users and items inherent in user-item interactions [16]. To overcome these limitations, deep learning-based recommender systems, e.g., Autoencoders [21, 33] and GCNs [2, 12, 22, 37], have been proposed to effectively learn more complicated relationships between users and items [32, 35, 38, 39].

Recently, recommender systems using GCNs [32, 35, 38] are gathering much attention. GCN-based methods can effectively learn the behavioral patterns between users and items by directly capturing the collaborative signals inherent in the user-item interactions [35]. Typical GCN-based methods include GC-MC [32], PinSage [38], and NGCF [35]. In general, GCN-based methods model a set of user-item interactions as a user-item bipartite graph and then perform the following three steps:

(Step 1) Initialization Step: They randomly set the initial D-dimensional embedding $e^0_u$ of all user $u$ and item $v$, i.e., $e^0_u, e^0_v \in \mathbb{R}^D$.  

(Step 2) Propagation Step: First of all, this propagation step is iterated $K$ times, i.e., $K$ layers of embedding propagation. Given the $K$ layers, the embedding of a user node $u$ (resp. an item node $v$) in the $i$-th layer is updated based on the embeddings of $u$’s (resp. $v$’s) neighbors $N_u$ (resp. $N_v$) in the $(i-1)$-th layer as follows:

$$e^i_u = \sigma(\sum_{v \in N_u} e^{i-1}_v W_i), \quad e^i_v = \sigma(\sum_{u \in N_v} e^{i-1}_u W_i),$$

where $\sigma$ denotes a non-linear activation function, e.g., ReLU, and $W_i \in \mathbb{R}^{D \times D}$ is a trainable transformation matrix. There exist some other variations: i) including the self-embeddings, i.e., $N_u = N_u \cup \{u\}$ and $N_v = N_v \cup \{v\}$, ii) removing the transformation matrix,
We note that all existing methods consider only one of the linear or residual prediction. In each layer, the linear propagation steps and selects an appropriate embedding propagation for each node. The preference of user \( u \) to item \( v \) is typically predicted using the dot product between the user \( u \)'s and item \( v \)'s embeddings in the last layer \( K \), i.e., \( \hat{r}_{u,v} = \mathbf{e}_u^K \odot \mathbf{e}_v^K \).

However, these GCN-based methods have two limitations: i) training difficulty \([6, 15, 37]\) and ii) over-smoothing, i.e., too similar embeddings of nodes \([5, 6, 15]\). First, the training difficulty is caused by their use of a non-linear activation function in the propagation step \([6, 15, 37]\). Specifically, the non-linear activation function complicates the propagation step, and even worse, this operation is repeatedly performed whenever a new layer is created. Thus, they suffer from the training difficulty of the non-linear activation functions for large-scale user-item bipartite graphs \([6, 37]\).

Next, the over-smoothing is caused as they use only the embeddings updated through the last layer in the prediction layer \([6]\). Specifically, as the number of layers increases, the embedding of a node will be influenced more from its neighbors’ embeddings. As a result, the embedding of a node in the last layer becomes similar to the embeddings of many directly/indirectly connected nodes \([5, 6]\). This phenomenon prevents most of the existing GCN-based methods from effectively utilizing the information of high-order neighborhood. Empirically, this is also shown by the fact that most of non-linear GCN-based methods show better performance when using only a few layers instead of deep networks.

Recently, LR-GCCF \([6]\) and LightGCN \([15]\), which are GCN-based recommender systems to alleviate the problems, have been proposed. First, to alleviate the former problem, they perform a linear embedding propagation without using a non-linear activation function in the propagation step. In order to mitigate the latter problem, they utilize the embeddings from all layers for prediction. After that, they perform residual prediction \([6, 15]\), which predict each user’s preference to each item with the multiple embeddings from the multiple layers. In \([6, 15]\), the authors demonstrated that a GCN architecture with the linear embedding propagation and the residual prediction can significantly improve the recommendation accuracy by successfully addressing the two problems.

In summary, GCN-based recommender systems can be characterized by, as shown in Table 2, the propagation and prediction types. We note that all existing methods consider only one of the linear or non-linear propagation, i.e., they assume only one type of user-item interactions. However, we conjecture that user-item interactions are neither only linear nor only non-linear, for which we will conduct in-depth analyses in Section 4.3. In this paper, therefore, we propose a Hybrid Method of Linear and non-linear collaborative filtering method (HMLET), which considers both the two disparate propagation steps and selects an appropriate embedding propagation for each node in a layer.

### Table 2: GCN-based recommender systems. In each layer, the gating module in HMLET chooses either of the linear or the non-linear propagation for each node.

| Non-Linear Propagation | O | O | O | X | X | X |
|------------------------|---|---|---|---|---|---|
| Linear Propagation     | X | X | X | O | O | O |
| Residual Prediction    | X | X | O | O | O | O |

and iii) removing the non-linear activation function, which is in particular called as linear propagation \([6, 15]\).

**Step 3** Prediction Step: The preference of user \( u \) to item \( v \) is typically predicted using the dot product between the user \( u \)'s and item \( v \)'s embeddings in the last layer \( K \), i.e., \( \hat{r}_{u,v} = \mathbf{e}_u^K \odot \mathbf{e}_v^K \).

This Gumbel-softmax has been widely used to learn optimal categorical distributions. One such example is network architecture search (NAS) \([13, 17, 25, 36]\). In NAS, we let an algorithm find optimal operators (among many pre-determined candidates prepared by users) and their connections. All these processes can be modeled by generating optimal one-hot (or multi-hot) vectors via the Gumbel-softmax \([20]\). Another example is multi-generator-based generative adversarial networks (GANs) \([10]\). Park et al. showed that data is typically multi-modal, and it is necessary to separate modes and assign a generator to each mode of data, e.g., one generator for long-hair females, another generator for short-hair males, and so on for a GAN generating facial images \([29]\).

In our case, we try to separate the two modes, i.e., the linear and non-linear characteristics of nodes.

### 3 PROPOSED APPROACH

We first formulate our problem of top-\( N \) recommendation as follows: Let \( u \in U \) and \( v \in I \) denote a user and an item, respectively, where \( U \) and \( I \) denote the sets of all users and all items, respectively; \( \mathcal{N}_u \) denotes a set of items rated by user \( u \). For each user \( u \), the goal is to recommend the top-\( N \) items that are most likely to be preferred by \( u \) among her unrated items, i.e., \( I \setminus \mathcal{N}_u \).

In this section, among several variations of HMLET, we mainly describe HMLET(End) for ease of writing because it shows the best accuracy — other variations can be easily modified from HMLET(End) and we omit their descriptions. Its key concept is to adopt the gating between the linear and non-linear propagation in a layer. In other words, we prepare both the linear and non-linear propagation steps in a layer and let our gating module with STGS decide which one to use for each node. Table 3 summarizes a list of notations used in this paper.

Figure 2 illustrates the overall workflow of HMLET(End). After constructing the user-item interaction as a user-item bipartite graph,
3.1 Propagation Layer

We omit the description of the initialization step due to its obviousness. HMLET propagates the embedding $\mathbf{e}_u$ (resp. $\mathbf{e}_v$) of each user $u$ (resp. each item $v$) through the propagation layers. In this subsection, we describe the propagation process. We first formally define the linear and the non-linear propagation steps used in this paper. Then, we present our gating module.

3.1.1 Propagation. Recently, the authors of LightGCN [15] found that the feature transformation and the non-linear activation do not have a positive effect on the effectiveness of CF. So, LightGCN removed the feature transformation and the non-linear activation from Eq. (1), and it shows better performance than existing non-linear GCNs for recommendation. In HMLET, we adopt the linear layer definition of LightGCN. Therefore, our linear embedding propagation is performed as follows:

$$
\mathbf{e}_{u+1}^{L_i} = \sum_{v \in N_u} \frac{1}{\sqrt{|N_u||N_v|}} \mathbf{e}_v G_i^{u_i}, \quad \mathbf{e}_{v+1}^{L_i} = \sum_{u \in N_v} \frac{1}{\sqrt{|N_u||N_v|}} \mathbf{e}_u G_i^{v_v},
$$

where $\mathbf{e}_{u+1}^{L_i}$ and $\mathbf{e}_{v+1}^{L_i}$ are the linear embeddings for user $u$ and item $v$. Since our gating module, which will be described shortly, selects between the linear and the non-linear embeddings, $\mathbf{e}_{u_i}$ and $\mathbf{e}_{v_i}$ means the embeddings selected by our gating module in the previous $i$-th layer. If $i = 0$, $\mathbf{e}_{u_i}^{L_0} = \mathbf{e}_u^{0}$ and $\mathbf{e}_{v_i}^{L_0} = \mathbf{e}_v^{0}$, i.e., initial embeddings. $\frac{1}{\sqrt{|N_u||N_v|}}$ is a symmetric normalization term to restrict the scale of embeddings into a reasonable boundary.

For the non-linear embedding propagation, we design a variant of the linear embedding propagation by adding non-linear activation functions. Its propagation is preformed as follows:

$$
\begin{align*}
\mathbf{e}_{u+1}^{N_i} &= \begin{cases} 
\mathbf{e}_u, & \text{if bypass} \\
\phi\left(\sum_{v \in N_u} \frac{1}{\sqrt{|N_u||N_v|}} \mathbf{e}_v G_i^{u_i}\right), & \text{if propagate,}
\end{cases} \\
\mathbf{e}_{v+1}^{N_i} &= \begin{cases} 
\mathbf{e}_v, & \text{if bypass} \\
\phi\left(\sum_{u \in N_v} \frac{1}{\sqrt{|N_u||N_v|}} \mathbf{e}_u G_i^{v_v}\right), & \text{if propagate,}
\end{cases}
\end{align*}
$$

where $\phi$ is a non-linear activation function, e.g., ELU, Leaky ReLU.

For instance, as shown in Figure 2, HMLET(End) bypasses the non-linearity propagation on the first and second layers to address the over-smoothing problem and then propagates the non-linear embedding in the third and fourth layers.

3.1.2 Gating Module. Now, we have the two types of the embeddings for each node, created by the linear and non-linear propagation in Eqs. (4) and (5), respectively, in the previous $i$-th layer. Therefore, we should select one of the linear and non-linear embeddings for the propagation in the next $(i + 1)$-th layer.

Toward this end, we add a gating module, which dynamically selects either of the linear or non-linear embedding after understanding the inherent characteristics of nodes. A separate gating module should be added whenever the linear and the non-linear

\begin{table}[h]
\centering
\caption{Notations used in this paper}
\begin{tabular}{|l|l|}
\hline
\textbf{Notation} & \textbf{Description} \\
\hline
$K$ & The number of total layers \\
$\mathbf{e}^{L_i}_u, \mathbf{e}^{L_i}_v$ & $u$’s and $v$’s embeddings at $i$-th linear layer \\
$\mathbf{e}^{N_i}_u, \mathbf{e}^{N_i}_v$ & $u$’s and $v$’s embeddings at $i$-th non-linear layer \\
$\mathbf{e}^{G_i}_u, \mathbf{e}^{G_i}_v$ & $u$’s and $v$’s embeddings selected by the gating module at $i$-th layer \\
$D$ & The size (dimension) of embedding \\
$|U|, |I|$ & The number of users and items \\
$r_{uv}$ & The user $u$’s final preference on item $v$ \\
$N_u, N_v$ & The set of items rated to user $u$ and the set of users who rated item $v$. \\
\hline
\end{tabular}
\end{table}

HMLET initializes the user and item embeddings in the initialization step. After that, each embedding is propagated to its neighbors through $K$ propagation layers. The gating module in HMLET selects either of the linear or the non-linear propagation in a layer for each node (Section 3.1). To this end, we use the gating module with STGS. In order to predict each user’s preference on each item, the dot product of the user embedding and the item embedding in each layer is aggregated and we use their sum for prediction (Section 3.2).
We support three gating types: i) choose the linear embedding $\mathbf{e}^L$, Non-linear embedding $\mathbf{e}^N$, Temperature $\tau$. Gating type $\xi$

1. **Function Gating Module** $(\mathbf{e}^L, \mathbf{e}^N, \tau, \xi)$:
   
   if $\xi = \text{linear then}$
   
   1. $\mathbf{e}^G \leftarrow \mathbf{e}^L$
   
   else if $\xi = \text{non-linear then}$
   
   1. $\mathbf{e}^G \leftarrow \mathbf{e}^N$
   
   else
   
   1. $\mathbf{e}_{\text{concat}} \leftarrow \mathbf{e}^L || \mathbf{e}^N$  
   2. $\mathbf{I} \leftarrow \text{MLP}(\mathbf{e}_{\text{concat}})$
   3. $\mathbf{g} \leftarrow \text{STGS}(\mathbf{I}, \tau)$
   4. $\mathbf{e}^G \leftarrow \mathbf{g} \cdot [\mathbf{e}^L, \mathbf{e}^N]$

   return $\mathbf{e}^G$

Algorithm 1: Gating Module

Input: Linear embedding $\mathbf{e}^L$, Non-linear embedding $\mathbf{e}^N$

3.1.3 **Variants of HMLET.** As shown in Figure 1 and Table 4, there can be four variants of HMLET, denoted as HMLET(All), HMLET(Front), HMLET(Middle), and HMLET(End), depending on the locations of the non-linear layers. Each method except HMLET(All) uses up to 2 non-linear layers since it is known that more than 2 non-linear layers cause the problem of over-smoothing [6]. Moreover, we test with various options of where to put them. First, HMLET(Front) focuses on the fact that GCNs are highly influenced by close neighborhood, i.e., in the first and second layers [31]. Therefore, HMLET(Front) adopts the gating module in the front and uses only the linear propagation layers afterwards. Second, HMLET(Middle) only uses the linear propagation in the front and last and then adopts the gating module in the second and third layers. Last, as the gating module is located in the third and fourth layers, HMLET(End) focuses on gating in the third and fourth layers — our experiments and analyses show that HMLET(End) is the best among the four variations of the proposed method. We select $\mathbf{e}^L_i$ or $\mathbf{e}^N_i$ at the third layer and $\mathbf{e}^L_4$ or $\mathbf{e}^N_4$ at the fourth layer via the gating modules, respectively. If the linear embeddings are selected for a node in all layers, it is the same as using a linear GCN with $K = 4$ for processing the node. If the non-linear embedding is selected for another node in all layers, it reduces to a non-linear GCN with $K = 2$. Likewise, HMLET(End) can be considered as a node-wise dynamic GCN (between the linear and non-linear propagation) with varying $K \in \{2, 4\}$.

3.2 **Prediction Layer**

After propagating through all $K$ layers, we predict a user $u$’s preference for an item $v$. To this end, we create a dot product value of $\mathbf{G}_u$ and $\mathbf{G}_v$ in each layer and use the following residual prediction:

$$\hat{r}_{uv} = \mathbf{G}_u \cdot \mathbf{G}_v.$$

In some layers, a gating module can be missing. In such a case, there is only one type of embeddings, but we also use $\mathbf{e}^L_i \mathbf{e}^N_i$ to denote these embeddings for ease of writing.

In most previous GCN-based recommender system research, only the embedding of the last layer was used to predict, but in HMLET, the above residual prediction $\hat{r}_{uv}$ with $\beta$ is used. Similar to LightGCN, $\beta$ is set to $1/(K + 1)$. This residual prediction can produce good performance by using not only the embedding in the last layer but also the embeddings in previous layers.

3.3 **Training Method**

For training HMLET, we employ the Bayesian Personalized Ranking (BPR) loss [30], denoted $L$, which is frequently used in many CF methods. The BPR loss is written as follows:

Table 4: Variants of HMLET in terms of their setting for the non-linear propagation in Eq. (5) and the gating type $\xi$

| Layer | 1 | 2 | 3 | 4 |
|-------|---|---|---|---|
| HMLET(All) | propagate/gating | propagate/gating | propagate/gating | propagate/gating |
| HMLET(Front) | propagate/gating | propagate/gating | gating | propagate/gating |
| HMLET(Middle) | bypass/linear | bypass/linear | gating | bypass/linear |
| HMLET(End) | bypass/linear | bypass/linear | gating | bypass/linear |

Algorithm 2: HMLET

Input: The number of total layers $K$, A bipartite graph $G$

1. **Function HMLET($K$, $G$):**
   
   1. Initialize $\mathbf{e}_u^L, \mathbf{e}_v^L$ for $u, v$
   2. $\text{iter} \leftarrow 0$
   3. while the BPR loss is not converged do
      
      1. $\tau \leftarrow 1.0 \exp(-0.001 \times \text{iter})$
      2. $\hat{r}_{u,v} = \mathbf{e}_u^L \cdot \mathbf{G}_v^L$, for $u, v$
      3. for $i \leftarrow 1$ to $K$ do
         
         1. $\mathbf{e}_u^{G_i}, \mathbf{e}_v^{G_i} = \text{Eq. (4), for } u, v$
         2. $\mathbf{e}_u^L, \mathbf{e}_v^L = \text{Eq. (5), for } u, v$
         3. $\hat{r}_{u,v} \leftarrow \mathbf{e}_u^{G_i} \cdot \mathbf{e}_v^{G_i}$, for $u, v$
         4. Update $\mathbf{e}_u^L, \mathbf{e}_v^L$ with the BPR Loss for $u, v$
         5. $\text{iter} \leftarrow 1$
      6. return $\hat{r}_{u,v}$, for $u, v$
We can train the network with annealing the temperature \( \tau \) we then split a dataset into training (80%), validation (10%), and test (10%) sets in the same way as in [35].

\[ L = - \sum_{i=1}^{N} \sum_{u \in N_i} \sum_{v \in R_u} \ln(\sigma(\hat{r}_{ui} - \hat{r}_{ui})) + \lambda \| \Theta \|^2, \]  

where \( \sigma \) is the sigmoid function. \( \Theta \) is the initial embeddings and the parameters of the gating modules, and \( \lambda \) controls the L2 regularization strength. We use each observed user-item interaction as a positive instance and employ the strategy used in [15] for sampling a negative instance.

We employ STGS for a smooth optimization of the gating module. We can train the network with annealing the temperature \( \tau \), and we use the temperature decay for each epoch (Line 5 in Algorithm 2). In order to calculate \( \tau_{\text{dec}} \) as in Eq. (6), we accumulate the dot product results (Lines 6 and 12). Then, we train the initial embeddings (Line 13) and the parameters of the gating modules (Line 14).

### 4 EXPERIMENTS

In this section, we evaluate our proposed approach via comprehensive experiments. We design our experiments, aiming at answering the following key research questions (RQs):

- **RQ1**: Which variation of HMLET is the most effective in terms of recommendation accuracy?
- **RQ2**: Does gating between the linear and non-linear propagation provide more accurate recommendations than baseline methods?
- **RQ3**: What are the characteristics of the nodes that use i) only the linear propagation, ii) only the non-linear propagation, or iii) different propagation steps in different layers?

#### 4.1 Experimental Environments

**Datasets.** For evaluation, we used the following three real-world datasets: Gowalla, Yelp2018, and Amazon-Book from various domains. They are all publicly available. Table 5 shows the detailed statistics of the three datasets.

| Dataset     | # User | # Item | # Interaction | Sparsity   |
|-------------|--------|--------|---------------|------------|
| Gowalla     | 29,858 | 40,981 | 1,027,370     | 99.916%    |
| Yelp2018    | 31,688 | 38,048 | 1,561,406     | 99.870%    |
| Amazon-Book | 52,643 | 91,599 | 2,984,108     | 99.938%    |

We compare [30] is a matrix factorization (MF) trained by the Bayesian Personalized Ranking (BPR) loss.

### 4.2 Experimental Results

#### 4.2.1 Comparison among Model Variations (RQ1).

For answering RQ1, we first compare the accuracies of HMLET(All), HMLET(Front), HMLET(Middle), and HMLET(End). Figures 3 illustrates the results where X-axis represents the types of HMLET, and Y-axis represents NDCG@20.

HMLET(End) is the best among all variations of HMLET. The accuracies of all variations except HMLET(End) are similar. Specifically, the difference between HMLET(End) and other variations is around 7%, 0.9%, 1% in Amazon-Book, Yelp2018, and Gowalla.

---

1https://github.com/wubinzuzu/NeuRec.
respectively. However, the differences in accuracy among HMLET(All), HMLET(Front), and HMLET(Middle) are as small as 0.2%.

These results indicate that i) the effectiveness of the gating module greatly depends on the location where the gating module exists, and ii) the non-linear propagation is useful to capture distant neighborhood information — note that we added the gating modules at the last two layers in HMLET(End). As shown in Table 7, each variation has a quite different linear/non-linear embedding selection ratio. HMLET(End), the best model, uses the non-linear propagation in the layers 3 and 4, and their selection ratios are significant, e.g., 5.19% of linear vs. 94.81% of non-linear in the third layer. This observation also applies to the second best model, HMLET(All). Similar selection ratio patterns are observed in the other two datasets.

4.3 Linearity or Non-Linearity

In this subsection, we define three different classes of nodes, depending on their preferences on the linear or non-linear propagation, and perform in-depth analyses on them. In order to analyze accurately, we use the embeddings learned by HMLET(End), the best performing variation of HMLET, for Amazon-Book. Due to space limitations, we omit the results for the other datasets, which show similar patterns to those in Amazon-Book.

4.3.1 Node Class and Graph Centrality. We first classify all nodes into one of the following three classes according to the embedding types selected by the gating modules:

- **Full-Non-Linearity (FNL)** is a class of nodes in which all embeddings selected by the gating modules are non-linear embeddings.
- **Partial-Non-Linearity (PNL)** is a class of nodes in which the embeddings selected by the gating modules are mixed with linear embeddings and non-linear embeddings.
- **Full-Linearity (FL)** is a class of nodes in which all embeddings selected by the gating modules are linear embeddings.

We next introduce three graph centrality metrics to study the characteristics of the classes:

- **PageRank** [28] measures the relative importance of nodes in a graph. A node is considered as important, even though its connectivity with other nodes is not that strong, if connected to other important nodes.
- **Betweenness Centrality** [1] measures the centrality of a node as an intermediary in a graph. The more a node appears in multiple shortest paths, the higher the betweenness centrality of the node.
- **Closeness Centrality** [9] measures the centrality of a node by considering general connections to other nodes in a graph. The less hops it takes for a node to reach all other nodes, the higher the closeness centrality.

4.3.2 Characteristics of Node Classes (RQ3). In this subsection, we analyze the characteristics of the nodes in each class in terms of the various centrality metrics. Table 8 shows the relative class size in our three datasets. In Amazon-Book and Yelp2018, most nodes were classified as FNL and PNL (about 47-48% and 50-52%, respectively), and a few nodes were classified as FL (about 1-2%). However, in Gowalla, the ratio of FL is about 12%, which is relatively higher compared to the other two datasets. Figure 4 shows the relative sizes of the three classes by degree, and Figure 5 shows the statistics of the centrality scores in each class. From them, it can be seen that the degree and centrality scores increase in order of FL, PNL, and FNL. Now, we deliver the meaning of the above results for each class.
Table 8: The relative class sizes in three datasets

| Class | Amazon-Book | Yelp2018 | Gowalla |
|-------|-------------|----------|---------|
| FNL   | 46.78%      | 47.95%   | 27.49%  |
| PNL   | 51.77%      | 49.59%   | 60.84%  |
| FL    | 1.45%       | 2.46%    | 11.67%  |

Figure 4: The class ratio of nodes sorted by degree. i-th bin in X-axis means a range of [(10 * (i−1))-th percentile, (10 * i)-th percentile) in terms of degree. Nodes with a high degree are the most likely to be in FNL (10th), and nodes with a small degree are likely to be in FL (1st). We also find that nodes classified as PNL are more evenly distributed than other classes.

- **FNL Attributes**: A node in FNL is either an active user or a popular item with more direct/indirect interaction information, i.e., a high degree and closeness centrality, and higher influence, i.e., a high PageRank and betweenness centrality, than nodes in other classes. So, they will receive a lot of information during the propagation step. Therefore, the sophisticated non-linear propagation is required to correctly extract useful information from much potentially noisy information.

- **FL Attributes**: A node in FL is either a user or an item that does not have much direct/indirect interaction information, i.e., a low degree and closeness centrality, and little influence, i.e., a low PageRank and betweenness centrality, compared to nodes in other classes. The information they receive during the propagation step may mostly consist of useful information related to themselves with little noise. Therefore, the simple linear propagation is required to take useful information as it is, rather than refining it.

- **PNL Attributes**: A node in PNL, compared to nodes in other classes, is a user or an item with neither too large nor too small direct/indirect interaction information, i.e., a moderate degree and closeness centrality, and influence, i.e., a moderate PageRank and betweenness centrality. In other words, although they have many direct neighbors, there are few indirect neighbors connected to the direct neighbors, or even if there are few direct neighbors, their indirect neighbors can be many. Therefore, they need to perform one of the non-linear or linear operations depending on the information they receive from neighbors.

In order to double-check our interpretations, we show the statistics of the similarity of embeddings for each class. So, we calculate the cosine similarity between a node and its direct neighbors by using the embeddings learned by HMLET(End). The results are shown in Table 9. From these results, we can confirm that the neighbors of a node in FNL consist of diversified nodes, i.e., a low mean and high variance. Also, FL nodes’ neighbors mainly consist of similar nodes, i.e., a high mean and low variance. Lastly, PNL nodes’ neighbors are in between the previous two cases, i.e., a moderate mean and variance between nodes in other classes.

Figure 5: The statistics of PageRank, betweenness centrality, and closeness centrality.

Table 9: The statistics of the cosine similarity between a node’s embedding and its direct neighbors’ embeddings in each node class

| Similarity | FNL | PNL | FL |
|------------|-----|-----|----|
| Mean       | 0.6369 | 0.6449 | 0.7157 |
| Variance   | 0.0179  | 0.0162  | 0.0160  |

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel GCN-based CF method, named as HMLET, that can select the linear or non-linear propagation step in a layer for each node. We further analyzed how the linear/non-linear selection mechanism works using various graph analytics techniques. To this end, we first designed our linear and non-linear propagation steps, being inspired by various state-of-the-art linear and non-linear GCNs for CF. Then, we used STGS to learn the optimal selection between the linear and non-linear propagation steps. The intuition behind such design choice is that it is not optimal to put both the linear and the non-linear propagation in every layer. In this sense, we have defined several variations of HMLET in terms of combining the linear and non-linear propagation steps.

Through extensive experiments using three standard benchmark datasets, we demonstrated that HMLET shows the best accuracy in all datasets. Furthermore, we presented in-depth analyses of how the linearity and non-linearity of nodes are decided in a graph. Toward this end, we classified nodes into three classes, i.e., Full-Non-Linearity, Partial-Non-Linearity, and Full-Linearity, depending on our gating module’s selections and studied correlations between nodes’ centrality scores and their class membership.

We conjecture that GCNs for CF should somehow consider both linear and non-linear operations. We do not say that our specific mechanism to combine the linear and the non-linear propagation steps is optimal. We hope that our discovery encourages much follow-up research work.

ACKNOWLEDGMENT

The work of Sang-Wook Kim was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1901-03. The work of Noseong Park was supported by the Yonsei University Research Fund of 2021 and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University)).
