HANABI: Graph Embedding for Recommendation via Conditional Proximity

Kachun Lo
Tohoku Universiy
Sendai, Japan
argent.lo.hk@gmail.com

Tsukasa Ishigaki
Tohoku Universiy
Sendai, Japan
isgk@tohoku.ac.jp

ABSTRACT

Bipartite graph has been greatly studied to learn complex user-item relations on graph for personalized recommendation. Despite the great success achieved, several limitations exist. Firstly, most works treat indirect nodes as direct nodes, regardless of the intermediate nodes. This introduces massive noises, since direct and indirect neighbors are intrinsically distinct for a source node. Secondly, other than the necessary embedding parameters, most works also rely heavily on "extra parameters", resulting in verbose training processes and burdensome models with proneness to overfitting. Finally, the attention mechanism is mostly used to local key neighbors, while attention on feature level is rarely utilized.

In this work, we propose to model the **"conditional proximity"**, which emphasizes the decisive role of intermediate nodes playing in information propagation and helps filter or enhance signals conditionally. We measure the conditional proximity by feature-level attention, which propagates information through pivotal feature channels to learn meaningful embeddings. Without bringing in any "extra parameters", we summarize our ideas and develop a light graph-based framework for recommendation "***HANABI***" (HNB). Comprehensive experiments on 2 public datasets shows HNB's superiority over state-of-the-art models.

KEYWORDS

Recommender Systems, Graph Embedding, Collaborative Filtering

ACM Reference Format:
Kachun Lo and Tsukasa Ishigaki. 2021. HANABI: Graph Embedding for Recommendation via Conditional Proximity . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND RELATED WORK

Recommender systems (RSs) has been playing a non-negligible role nowadays in surprisingly many fields [7, 18], e.g. e-commerce, social media, search engine. RSs mainly studies how to connect entities within the system effectively and efficiently. Moreover, in real applications, overwhelmingly growing amount of data also require a reliable RS to be light in weight and simple in construction, such that fast inference can be achieved and building and updating can be done on a more frequent basis.

To estimate users' preferences for items, collaborative filtering (CF) is the most popular method among the literature. CF predicts preferences based on the assumption that similar users show resemblant behaviors and, as a result, would interact similar items. In order to capture latent user-item relations, most models embed both users and items as embedding vectors of same dimensions in a common space. Among many classical methods, matrix factorization (MF) utilizes inner products of users and items embeddings to infer observed interactions. Bayesian Personalized Ranking (BPR) frames the regression problem in MF as a binary classification problem for implicit interactions, where users' preferences are measured by a probabilistic pairwise ranking function. Since deep learning showed its superior capability of capturing complex patterns, researchers have also adopted the idea in RSs and made CF models deeper. Neural Collaborative Filtering (NCF) extends CF by stacking multiple layers with nonlinear activation functions. Collaborative Memory Network (CMN) combines CF with memory network, which helps memorize users' subtle preferences.

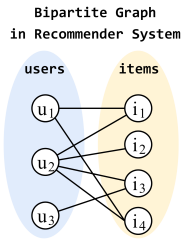

Figure 1: Illustration of Bipartite Graph.

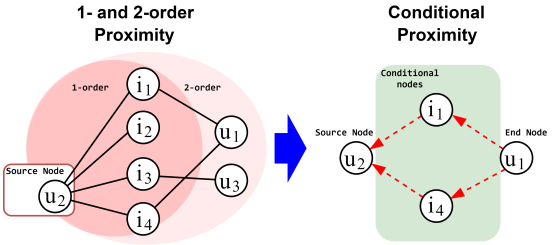

Figure 2: Illustration of how indirect 2-order information flow through intermediate nodes to source node.

These models are then further improved by graph-based frameworks, which enrich the sparse RSs with indirect information. DeepWalk, LINE and Node2Vec have successfully caught high-order indirect relations and generated representative nodes' embeddings through random surfer [3]. In the domain of RSs, user-item interactions are typically constructed as a bipartite graph first, where users and items have no direct edge with their own kind, as shown in Figure 1. Then, pairs of neighbor nodes would be collected by random walk sampling as training examples. An example of 1 and 2-order proximity can be found in Figure 2 (left).

Bipartite Network Embedding (BINE) is a graph embedding model for general purposes, featured by emphasizing homogeneous relations between user-user and item-item [3]. HOP-Rec is designed for RSs, which improves a basic BPR model by adding indirect higher-order neighbors as training examples [18]. CSE, inspired by HOP-Rec and LINE, adopts the idea of auxiliary context embedding which helps to simultaneously model higher-order user-user, item-item, and user-item similarity [1]. More recently, NGCF takes into account the node-to-node information propagation and explicitly model this process in a recursive style over layers [17]. Multi-GCCF takes a

Conference'17, July 2017, Washington, DC, USA

step forward by not only modelling the information propagation but also maintaining the intrinsic difference between users and items [13].

Graph-based methods empower CF to capture indirect high-order relations. There, however, still exist several **limitations**:

Limitation 1. Most graph-based models treat "high-order proximity" in the same way as "direct proximity" by maximizing inner products of pairs of neighbors' embeddings, regardless of their different orders. Such approach completely ignores intermediate node(s). Since direct and indirect interactions are intrinsically different, treating them the same way would introduce massive noises, resulting in sub-optimal solution. We argue that, especially for indirect interactions, the decisive role of intermediate node(s) should be modelled explicitly.

Limitation 2. To improve performance, most graph-based models employ "extra parameters/weights" in either training or both training and inference. To clarify, for graph embedding model, we refer to any parameters other than the "embedding matrix" as "extra weights". "Extra weights" include "context embedding", "weights for linear transformation", etc. First, "extra weights" are rather burdensome and often imply complex computations. A robust RSs should be light in weight for fast inference and fast updating. Second, heavier and deeper models are prone to overfitting, especially in extremely sparse settings such as RSs. Consequently, deep models exhibiting powerful performance in experimental setting might not generalize well in real applications. Finally, it's truth that training techniques like drop-out and early-stopping can effectively alleviate overfitting. These, however, require more laboring efforts on tuning hyperparameters, which again is at cost of human efforts and computer resources [11].

Limitation 3. Feature-level attentive mechanism has not been utilized in graph-based RSs studies. Although many works [15, 16] have employed "node-level" attention to locate key neighbors for source nodes, the attention on latent feature channels of embedding has not been explicitly exploited. We argue that, due to the extreme sparsity and the heterogeneity of nodes in RSs, a more fine-grained "feature-level" attention can better handle propagation of information signals than "node-level" attention.

To alleviate the aforementioned problems, we propose our light graph embedding model for RSs, "***HANABI***" (HNB) [1]. HNB not only gets rid of additional parameters both in training and inference, but also achieves superior performance by explicitly modelling intermediate nodes and conditionally measuring high-order proximity with feature-level attention.

Specifically, HNB first samples a pair of direct neighbors and treats them as two source nodes. Then, starting from source nodes, multiple indirect neighbors are sampled by an extended random walk for training. Next, especially different from other works, HNB models high-order information propagation **conditionally**. The process that information has to flow through intermediate node(s) before arriving to source nodes is modelled as "conditional proximity". We design a feature-level attention mechanism to capture these conditional dependencies.

To sum up, the main contributions of our work are as follows:

- We propose HNB, a light graph-based CF model for recommendation, which gets rid of additional parameters, achieving fast inference, allowing easy construction and being less prone to overfitting.
- We design "conditional proximity" to model the decisive process that, before arriving to source nodes, information signals are filtered or enhanced by intermediate nodes. Feature-level attention is used to capture the conditional dependencies.
- Extensive experiments conducted on 2 datasets examine the effectiveness of the conditional proximity and reveal the superiority of HNB over state-of-the-art models.

2 OUR MODEL

Notation we use in this paper is summarized in Table 1.

[1]The model is named after the graph structure which looks like a firework and "HANABI" (HNB) is "firework" in Japanese.

Table 1: List of Notations.

Symbol	Definition
U, I, u, i	User, Item set, A specific user, item
N_j	Set of direct neighbors of node j
θ_j, Θ	Embedding of node j; Entire embedding matrix
d, k	Size of embedding; Length of random walk
$s, \mathbf{m}, e$	Given an information flowing path, the **s**ource node, inter**m**ediate node(s) and **e**nd node
e^+, e^-	Positive, negative end node
η	Number of high-order neighbors to sample
w_k, w_{od_j}	Length-aware decay factor; Outdegree-aware factor

2.1 Information Propagation in Graph

Graph-based method captures similarity between nodes by measuring the quality and quantity of information flows (or edges). For example, as shown in Figure 2(left), for a source node u_2, all items $i_{1\sim4}$ are equally important, because they are all direct neighbors and the edge weights, by default in most RSs, are the same. The 2-order neighbors, u_1 and u_3, are not as close to u_2 as $i_{1\sim4}$, since they take two steps to reach u_2. Furthermore, given that u_1 has more paths to reach u_2 than u_3, u_1 is more similar to u_2 than u_3.

Conditional Proximity. Although proximity can be partially obtained by measuring the length and number of paths, such simple approach is not enough to capture high-order complex relations, because indirect neighbors have intrinsically different way of propagating information from direct (1-order) neighbors. We argue that, intermediate node(s), which connect source and end nodes, play a decisive role in high-order proximity. Illustrated in Figure 2(right), a 2-order neighbor u_1 has 2 paths to the source node u_2, where each path carries fundamentally different information depending on the intermediate node, i_1 or i_4.

An **intuitive example** of how information propagates distinctly through 2 paths, $[e \rightarrow m_1 \rightarrow s]$ and $[e \rightarrow m_2 \rightarrow s]$, is as follows. Suppose a source user (node s) has two friends, a scientist (m_1) and an artist (m_2), and they both recommend a book (e) to s. User s would probably be very interested because s gets multiple suggestions of the same thing e from two 1-order close friends. However, more interestingly, user s might perceive the two suggestion messages very ***distinctively***. While the scientist m_1 might give suggestion for educational purpose, the artist m_2 might recommend e because of its aesthetic value. We, therefore, refer to the information propagation being conditional on intermediate node as **"conditional proximity"** and intermediate node as **"conditional node"**.

2.2 Node Embedding Layer

We first model both users and items embeddings in a common space as an embedding matrix, whose parameters are all learnable, as other works [1, 13, 17, 18]. The overall embedding matrix $\mathbf{\Theta}$ takes the following form:

$$\mathbf{\Theta} = [\theta_{u_1}, \cdots, \theta_{u_{|U|}}, \quad \theta_{i_1}, \cdots, \theta_{i_{|I|}}], \quad \mathbf{\Theta} \in \mathbb{R}^{(|U|+|I|)\times d}.$$

Then, for a node j, its embedding vector θ_j can be looked up from Θ (where $\theta_j \in \mathbb{R}^d$ and d is the embedding size). This embedding matrix is randomly initialized and the objective of our framework is to learn from the observed data and optimize $\mathbf{\Theta}$, such that the embeddings are representative of the graph structure and capture complex user-item relations.

It is also worth noting that to keep the model light, we refrain from using any additional parameters. The single embedding matrix $\mathbf{\Theta}$ contains all learnable parameters that are necessary for the proposed HNB model.

2.3 Conditional Proximity with Feature-Level Attention

Next, the measurement of various proximity is described. Especially, for high-order neighbors, their proximity will be "conditional" on intermediate node(s) and be measured by feature-level attention.

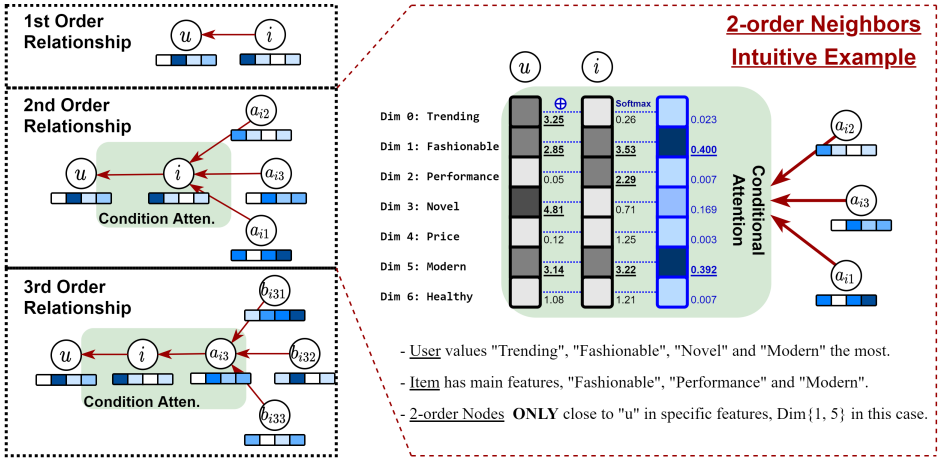

Figure 3: Intuitive Example of how Feature-level Attention models Conditional Information Flowing. Best view in color.

For 1-order proximity. Following prior works [3, 14], the proximity score between u and its direct neighbor i is measured by:

$$\mathbf{Prox}(u, i) = \theta_u^\top \theta_i.$$

Since direct user-item interaction is the strongest signal when measuring proximity, we use the inner product to ensure that the two embeddings should be close across all dimensions.

For 2-order proximity. Unlike direct neighbors, we model high-order proximity as conditional proximity, where the intermediate node is taken into account as condition. Illustrated in Figure 3, we denote the conditional proximity between a source node u and its 2-order neighbor a_{i3} as:

$$\mathbf{Prox}(u, a_{i3}|i),$$

where i is the intermediate node connecting u and a_{i3}.

To model $\mathbf{Prox}(u, a_{i3}|i)$ and to emphasize the effect of intermediate node, we propose to employ feature-level attentive mechanism. Ideally, such mechanism should first aggregate features' information from both source node and intermediate node and then guide the information flows to focus on only important features.

$$\mathbf{Prox}(u, a_{i3}|i) = [\mathbf{Atten}(\theta_u, \theta_i) \odot \theta_u]^\top \theta_{a_{i3}},$$
$$\text{where } \mathbf{Atten}(\theta_u, \theta_i) = softmax(\theta_u \oplus \theta_i).$$

The $\oplus$ and $\odot$ are element-wise addition and product operation, the $\mathbf{Atten}(\cdot)$ is the feature-level attention and the $softmax(\cdot)$ is:

$$softmax(\theta)_l = \frac{e^{\vartheta_l}}{\sum_{j=1}^{d} e^{\vartheta_j}},$$
$$\text{for } l = 1, \ldots, d \text{ and } \theta = (\vartheta_1, \cdots, \vartheta_d) \in \mathbb{R}^d.$$

We utilize the $\oplus$ to model feature-level attention $\mathbf{Atten}(\cdot)$, because when considering further neighbors (e.g. $\mathbf{Atten}(\theta_0, \theta_1, \theta_2, \theta_3)$), the product $\odot$ will produce very small numbers, while the simpler adding is stabler.

The mechanism is intuitively explained in Figure 3. Supposed each dimension of embedding encodes an interpretable feature of preference, e.g. dimension 0 for the feature "Trending", then the source node u has higher preference for dimensions {0, 1, 3, 5} , while the intermediate node i is featured by dimensions {1, 2, 5}.

When modelling the 2-order proximity between $\{a_{i\cdot}\}$ and u, typical graph models [1, 18] would treat it like the 1-order proximity and ignore the intermediate node i, which leads to massive noises being brought in to u along the information flows. As discussed in section 2.1, a suggestion of an item could be perceived differently depending on the intermediate node. A feature-level attention can effectively fix this. It introduces the intermediate node i and constrains the information flows to pass through only significant features of both u and i. Eventually, noisy features are filtered and key features are enhanced. In the example, u finally receives information mainly about "dim 1: Fashionable" and "dim 5: Modern" from the 2-order neighbors $a_{i\cdot}$ through i.

For even higher-order proximity. The same logic generalizes to higher order. Take a 3-order proximity example from Figure 3. Consider a path of information flow $[b_{i32} \rightarrow a_{i3} \rightarrow i \rightarrow u]$, the 3-order proximity is computed:

$$\mathbf{Prox}(u, b_{i32}|i, a_{i3}) = [\mathbf{Atten}(\theta_u, \theta_i, \theta_{a_{i3}}) \odot \theta_u]^\top \theta_{b_{i32}},$$
$$\text{where } \mathbf{Atten}(\theta_u, \theta_i, \theta_{a_{i3}}) = softmax(\theta_u \oplus \theta_i \oplus \theta_{a_{i3}}).$$

General Proximity Notation. To simplify the notations, we use a general notation as follows for both conditional (high-order) and unconditional (1-order) proximity:

$$\mathbf{Prox}(s, e|\mathbf{m}) = \begin{cases} \mathbf{Prox}(s, e) & \text{if } k = 1, \\ \mathbf{Prox}(s, e|\text{ intermediate node(s)}) & \text{if } k > 1, \end{cases} \quad (1)$$

In Equation 1, $\mathbf{m}$ is a general symbol for intermediate node(s). In case of 1-order proximity ($k = 1$), $\mathbf{m}$ is $\emptyset$, indicating an unconditional proximity; for $k = 2$, $\mathbf{m}$ is the node between source s and end node e; for $k = 3$, $\mathbf{m}$ is the set of intermediate nodes; and so on.

2.4 Reinforcing Conditional Proximity

We propose three components to emphasize the role of intermediate nodes in high-order propagation and to reinforce the conditional proximity.

Length-Aware Decay Factor. As studied in [1, 17, 18], the strength of proximity decays as the order gets higher. We take into account such effect by integrating a length-aware decay factor: $w_k = (\frac{1}{k})^\tau$, where τ is a hyper-parameter and can be chosen in range $(1, 8)$. The purpose of w_k is to emphasize close relations and down-weight faraway pairs. w_k is introduced in the loss function in section 2.5.

Outdegree-Aware Factor. Moreover, node's outdegree plays an important role in measuring user-item relations as well [9, 17]. Given a node j, its outdegree-aware factor w_{od_j} is obtained by: $w_{od_j} = \frac{1}{\log(|N_j|+2)}$, where $|N_j|$ is the number of direct neighbors of j. The logarithm serves as a smoothing function and the $|N_j| + 2$ is to avoid situation where user or item has only 0 or 1 historical interaction. The w_{od_j} is designed to be inversely proportional to outdegree, because information coming from popular neighbor nodes (those having high outdegree) are less important than information from unpopular nodes (those having low outdegree) [9, 17].

In RSs, this is particularly important. For instance, when a user interacts with a popular item, this item actually does not provide much useful knowledge about this user, since almost all users are interested in it. Conversely, unpopular items (with low outdegree) carry much precious knowledge about users, because users must have specific and strong preferences to buy something others would ignore or dislike. Moreover, two users are likely to be especially similar (in some aspects), if they both have interaction with a very unpopular item.

Extended Random Walks. There are 2 sampling methods often used to augment training data in recent graph-based CF models. One is random walk, which starts from a source node and collects one sample per step until reaching walk length k [1, 18]. The other one [13, 17], gathers all possible neighbors on each order, similar to breadth-first search (BFS), generating exponentially great number of training samples. The simple random walk sampling is fast but is likely to miss meaningful neighbors, while the second sampling method maintains perfect neighborhood information but is very expensive to optimize in training.

To balance the two methods and to reinforce the conditional proximity, we use a straightforward "extended random walk". The random walk is extended by collecting not only 1, but $\eta > 1$ "indirect neighbors" in each step. Note that, since we aim at collecting training samples to reinforce the conditional proximity, the random walk is only extended when sampling high-order neighbors. For example, in Figure 3, given an edge $[i \rightarrow u]$ in dataset, the source node u will eventually have one direct neighbor i, three 2-order neighbors and nine 3-order neighbors when length $k = 3$ and $\eta = 3$. This simple sampling can collect enough information without forcing too heavy optimization in training. In experiments, we sample training data for HNB this way in preprocessing.

2.5 Optimization and Complexity Analysis

Optimization is based on Bayesian Pairwise Ranking loss function (BPR) [10], which utilizes positive and negative examples and has been widely used in RSs [1, 17, 18].

$$\begin{aligned}\textbf{Loss} = - \sum_{(s,e^+,e^-)} & \sqrt[3]{w_{od_s} w_{od_{e^+}} w_{od_{e^-}}} \\ & \cdot \log \sigma(w_k \cdot [\textbf{Prox}(s, e^+|\mathbf{m}) - \textbf{Prox}(s, e^-|\mathbf{m})]) \\ & + \lambda \|\Theta\|_2^2,\end{aligned}$$

where the set (s, e^+, e^-) is the collection of sampled source node s, positive end node e^+ and negative end node e^-; the first term is the geometric mean of the "w_{od}" of s, e^+ and e^-; the $\sigma(\cdot)$ is the *sigmoid function*; Θ is the embedding matrix; L_2 regularization $\|\Theta\|_2^2$ with strength λ is uesd. The estimated score is measured as the similarity between the target user and candidate item: $\hat{y}_{ui} = \theta_u^T \theta_i$.

Complexity analysis. Depending on the implementation, the complexity may vary with $\{d, |E|\}$, where E are the edges of a graph. During training, HNB performs vector operations $O(|E|dk)$, softmax $O(d)$ and SGD optimization $O(d)$. As for the space of model storage, since the embedding matrix contains all parameters of HNB, the space is $O((|U| + |I|) \cdot d)$. **Empirical computation times** on Gowalla dataset are summarized. When training, BPR, CSE, NGCF and HNB take about $\{21s, 50s, 75s, 53s\}$ per epoch. when inference, BPR, CSE, NGCF and HNB take about $\{17s, 17s, 86s, 17s\}$. HNB has the same inference time as CSE and the simplest BPR, since they have the same size of embedding matrix for inference.

3 EXPERIMENTS

We compare **7 baselines** with the proposed HNB on **2 datasets** with **3 metrics**.

Table 2: Statistics of Datasets.

Dataset	Sparsity	Users	Items	Interactions
Amazon-Book	99.94%	52,643	91,599	2,984,108
Gowalla	99.92%	29,858	40,981	1,027,370

3.1 Experimental Settings

3.1.1 ***Datasets.*** Each dataset, summarized in Table 2, is representative of a particular scenario in RSs (varies in sparsity, size). For all datasets, 72% of a user's historical items are randomly sampled as training set, 8% as validation set for hyper-parameters tuning and 20% as test set. **Amazon-Book (large; sparse)**. Amazon-Book is the largest and most sparse dataset in our experiments [5], reflecting real situations in e-commence and music store. As in [6, 13, 17], we keep nodes with at least ten data and binarize the ratings to {0, 1}. **Gowalla (mid; sparse)**. Gowalla is collected by Gowalla [8], where users interact with locations by checking-in. As in [6, 13, 17], we only keep users with at least ten locations.

3.1.2 ***Baselines and Settings.***
Model-base models:
• *BPR* [10] is a MF model specialized for implicit feedback data, which replaces the MSE with the BPR loss function.
• *NMF* [7] is a deep-learning CF model incorporating the features of MF and multilayer perceptron (MLP) with nonlinear activation.
• *CMN* [2] is a state-of-the-art model that simulates the memory network [12] by integrating 2 "memories components" which enables reading and writing latent features flexibly.
Graph-base models:
• *RANK-CSE* [1] is a state-of-the-art graph-based model. Inspired by LINE [14], CSE employs context embedding to improve training quality. We implement the "ranking" variant of CSE in our experiments.
• *PinSage* [19] adopts the CNN structure from GraphSage [4] , a general graph algorightm, for recommendation.
• *HOP-Rec* [18] is a graph-based CF model which approximates high-order proximity by random walk sampling.
• *NGCF* [17] is a state-of-the-art graph-based model. which models the massage propagation by stacking recursively "propagation layer".
Parameter Settings. For fair comparison, the embedding size d is set to 100 for all baselines, except for NGCF, which requires $64 \times 3 (194)$ [17]. Other hyper-parameters of baselines are set as the same as their original works. HNB_2 and HNB_3 are HNB with length $k = \{2, 3\}$ respectively. The number of neighbors $\eta = 3$; embedding size $d = \{100, 150\}$; $\tau = 3$; the negative ratio is set as 5 for all models using BPR loss function. **Evaluation Metrics**. 3 widely-used metrics are used: Recall (*Rec@20*), Precision (*Prec@20*) and Normalized Discounted Cumulative Gain (*NDCG@20*).

3.1.3 ***Comparative Experiments.*** Comparative experimental results are summerized in Table 3. **Amazon-Book (large; sparse).** A large and sparse dataset is representative of most e-commerce and music RS. Among baseline models, RANK-CSE outperforms all others, attributed to its ability in modelling user-user, item-item and user-item relations simultaneously. Compared with HOP-Rec and PinSage, RANK-CSE takes into account the effect of outdegree and also utilizes high-order neighbors to augment training set. RANK-CSE also has an extra "context embedding" when computing high-order proximity. The purposes of the context embedding are to avoid directly optimizing the vertex embedding because of the noisy signals from high-order neighbors, and to help put attentions on key neighbors, resembling a node-level attention. Based on our experiments, we argue that these purposes can be better realized by the feature-level attention in HNB.

HNB makes greatest improvements over baselines on Amazon-Book dataset than on other datasets. Specifically, HNB_2 (d=150) improves 12.5%, 12.4% and 17.9% on *Recall*, *Precision* and *NDCG*. Compared to other

Table 3: Comparative experiments for HNB. Best results are in bold. The symbol ‡ denotes the best baseline results.

	@20	BPR	NMF	CMN	RANK-CSE	PinSage	HOP-Rec	NGCF	HNB_2 (d=100)	HNB_2 (d=150)	HNB_3 (d=150)
Amazon-Book (large; sparse)	**Rec.**	0.0256	0.0261	0.0284	0.0351‡	0.0288	0.0309	0.0344	**0.0372** (+6.0%)	**0.0395** (+12.5%)	**0.0397** (+12.7%)
	Prec.	0.0121	0.0122	0.0130	0.0145‡	0.0128	0.0124	0.0138	**0.0153** (+5.5%)	**0.0163** (+12.4%)	**0.0164** (+12.9%)
	NDCG	0.0520	0.0526	0.0597	0.0638‡	0.0558	0.0606	0.0630	**0.0700** (+9.7%)	**0.0746** (+17.9%)	**0.0760** (+19.0%)
Gowalla (mid; sparse)	**Rec.**	0.1359	0.1388	0.1404	0.1049	0.1382	0.1399	0.1547‡	**0.1600** (+3.4%)	**0.1646** (+6.4%)	**0.1654** (+6.9%)
	Prec.	0.0417	0.0430	0.0445	0.0324	0.0451	0.0456	0.0470‡	**0.0487** (+3.6%)	**0.0500** (+6.4%)	**0.0502** (+7.0%)
	NDCG	0.1890	0.2023	0.2129	0.1712	0.1944	0.2128	0.2237‡	**0.2315** (+3.5%)	**0.2382** (+6.6%)	**0.2391** (+6.9%)

Table 4: Ablation Study.

Amazon-Book (@20)	Recall	Precision	NDCG
BPR ($d = 100$)	0.0256	0.0121	0.0520
+ k-order Proximity ($k = 3$)	0.0263	0.0119	0.0517
+ Length-Aware Decay Factor	0.0298	0.0124	0.0576
+ Outdegree-Aware Factor	0.0315	0.0126	0.0603
+ Feature-level Attention	0.0367	0.0151	0.0702
+ Full HNB ($\eta = 3$)	**0.0379**	**0.0155**	**0.0718**

models, HNB considers the influence of intermediate node in the process of information propagation and models it through feature-level attention. These turn out to help capture more meaningful user-item interactions and boost performance.

Gowalla (mid; sparse). Medium and sparse dataset is typical in many RSs scenarios. NGCF is the best baseline model showing the benefits of adding high-order neighbors and using training techniques like "message and node dropout". HNB outperforms baselines thanks to the feature-level attention, which allows the information to propagate conditionally only through important feature channels and filters noisy signals flowing from high-order neighbors to source nodes.

3.1.4 ***Ablation Study.*** To evaluate the effectiveness of each component, an ablation study is conducted. As shown in Table 4, experiments are on Amazon-Book and the base model is BPR ($d = 100$). First, it's interesting to notice that, without considering the effects of path length and outdegree, the k-order proximity alone actually jeopardizes the performance. Second, the designed feature-level attention significantly boosts the performance, indicating the effectiveness of the conditional flow mechanism and the intermediate nodes are indeed critical when optimizing with indirect neighbors. Finally, the full HNB model is completed with training examples collected by "extended random walks" in preprossing. The further improvement shows that the conditional proximity is reinforced by adding more indirect samples, which helps encode meaningful features in embedding.

4 CONCLUSION

In the work, we explore the importance of intermediate nodes and introduce the concept of "conditional proximity" for high-order information propagation. The process of conditional flowing is captured by a feature-level attention mechanism, which helps model to focus on decisive feature channels when measuring high-order proximity. We summarize the ideas and propose our HNB framework for RS. HNB is light in weight, with no extra parameters, and superior in performance, supported by the comprehensive experiments against state-of-the-art baselines. As future work, we plan to extend HNB to a general graph embedding model not just for RSs, as "conditional proximity" is a general concept for graph analysis.

REFERENCES

[1] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *The World Wide Web Conference*. ACM, 2637–2643.

[2] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 515–524.

[3] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. Bine: Bipartite network embedding. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 715–724.

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

[5] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*.

[6] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*.

[8] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*.

[9] Kachun Lo and Tsukasa Ishigaki. 2019. Matching Novelty while Training: Novel Recommendation based on Personalized Pairwise Loss Weighting. In *ICDM 2019*.

[10] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* (2014).

[12] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS 2015, Montreal, Quebec, Canada*.

[13] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-Graph Convolution Collaborative Filtering. In *ICDM 2019*. ACM.

[14] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*.

[15] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. [n.d.]. Graph Attention Networks. In *ICLR 2018*.

[16] Xiang Wang, Xiangnan He, Yixin Cao, and Meng Liu. 2019. Knowledge Graph Attention Network for Recommendation. *arXiv:1905.07854* (2019).

[17] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *arXiv preprint arXiv:1905.08108* (2019).

[18] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 140–144.

[19] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD*.