CATEGORICAL ENTITY FEATURES IN RECOMMENDATION SYSTEMS USING GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks are widely used in recommender engines and are commonly applied to user-item graphs augmented by various side information, including categorical entity features. It is established that a user selection process involves a complex framework of preferences and the importance of presented alternatives. For example, user's preferences might change depending on product category and/or brand. Thus, comprehending and modeling them effectively is essential in the recommender engines' context. Despite the significant influence of such categorical features on the user decision-making process, these have been incorporated in graph models in various ways without giving a clear indication of which method is most suitable. We investigate the capabilities of graph neural networks to extract and model categorical attribute-specific preferences effectively by systematically comparing existing techniques and graph models. These include one-hot encoding-based node features, category-value nodes, and categories as hyperedges. In addition, we introduce a novel hyperedge-based method designed to leverage categorical features more effectively compared to current approaches. The proposed model, which has a simple architecture and combines neighborhood aggregation with hyperedge aggregations, outperforms many complex and sophisticated methods. In extensive experiments using three real-world datasets, we compare existing methods and demonstrate the advantage of our approach in terms of commonly used quality metrics for recommender engines.

1 INTRODUCTION

E-commerce website users encounter the daunting challenge of sifting through an overwhelming number of products to find the right item. To address this issue, recommender system (RS) algorithms have been designed to understand user intentions and predict which items to shortlist. The central objective of these RS algorithms is to learn and extract user preferences effectively, enabling them to anticipate the next likely item of interest. This task poses significant difficulties as users' decision-making process involves quantifying preferences and the importance of presented alternatives (Dyer & Sarin, 1979). For instance, user price preferences are highly influenced by the brand or product category. The process of clicking on the next item is driven by a complex interplay of product attributes and user preferences. Thus, the importance of categorical features of entities is pivotal in effectively learning and modeling user preferences.

Given that user-item interactions can be naturally represented as graph data, where nodes represent users/items and edges correspond to interactions like clicks or purchases, many authors have successfully used graph neural networks (GNN) for recommender engines (He et al., 2020; van den Berg et al., 2017; Li et al., 2023; Sun et al., 2020; Guo et al., 2021; Zheng et al., 2023; Liu et al., 2022; Hu et al., 2020; Li et al., 2021). It is claimed that the advantage of GNN-based user-item recommender systems lies in their ability to incorporate information beyond user-item relations, including edges among users/items and diverse user and item features.

Although GNNs have been adopted for RS, it is noteworthy that there is limited research dedicated to understanding how to incorporate categorical features best and its capability to extract user preferences from such characteristics effectively (e.g., price preferences, brand preference, or interaction of those two).

In this paper, we investigate the role of categorical features in user-item recommender engines based on graph neural networks. We explore various techniques that are used to integrate categorical features of entities. Many papers include such information as binary encoded node features (Sun et al., 2020; Guo et al., 2021) or adding category value-nodes on graphs (Zheng et al., 2023; Liu et al., 2022; Hu et al., 2020; Li et al., 2021). However, authors usually do not explore or clarify why they selected a specific method. There are no definitive guidelines/studies on which approach is most suitable for integration with a particular GNN architecture and whether or not there are other ways to consider. Therefore, we examine existing practices from the literature and propose a new method - category values as hyperedges that demonstrate effective utilization of categorical features compared to current methods. Using hyperedges in recommender engines is not novel and has already been studied (Zhang et al., 2022; Wang et al., 2020; Xia et al., 2021). However, most of the research is focused on session-based recommender engines, where hyperedges are created by combining different attributes together (for example, all prices within sessions build a hyperedge).

It is to be noted that our examination focuses on user-item recommender systems and does not extend to session-based recommender systems. In addition, we concentrate on how entities' categorical features, e.g., users and/or items categorical features, can be effectively utilized and do not study context features, e.g., interactions categorical features.

The main contributions of this paper are as follows

- Examination of categorical feature integration: We review the literature and examine how categorical features are integrated into the models. Furthermore, we extensively compare different techniques to find out how different methodologies impact model performance.
- New architecture: We introduce a new approach where categorical features of entities are used directly as hyperedges in GNN-based user-item recommender engines. We demonstrate that even though our approach has a simple architecture, it surpasses the performance of more sophisticated methodologies.
- Empirical comparison and validation: We conduct extensive experiments on three real-world datasets and show that the hyperedge approach outperforms other methodologies (e.g., category-value nodes and binary-encoded features). In addition, we benchmark our approach against state-of-art models. The findings suggest that hyperedges can effectively be used to extract user preferences that improve model accuracy.

2 Related Work

We discuss the related work on categorical features in recommender engines in general and specifically for GNN-based methods.

2.1 Recommender Engines using Categorical Features

Early recommender systems used only user-item interaction data to generate new recommendations. In this context, categorical features were often considered in the pre and post-processing stages of recommendation generation (Mei et al., 2018; Sun et al., 2019). Several studies implemented item/user categories as pre and post-filters (Hwang et al., 2012; Panniello et al., 2009; Davidson et al., 2010; Baltrunas & Ricci, 2009; Wadhwa et al., 2020). For instance, Davidson et al. (2010), used categories as a post-processing step to further narrow down a subset of items for presentation to the users. Baltrunas & Ricci (2009) utilized contextual item information as a pre-processing step. Pre and post-filters were the first attempts to include additional information in recommender systems.

Advancements in modeling recommendation engines have enabled the integration of categorical features in the learning process. In the context of user-item recommender engines, categorical features are either entity (user/item) specific or user-item interaction specific (Chen et al., 2019). User/item-specific attributes are called side information, for example, user age/gender, item category/brand. On the other hand, user-item interaction-specific features are called context (Meng et al., 2023; Adomavicius & Tuzhilin, 2015). Early studies have explored both context-aware and side information-aware recommender engines and suggested different methods to employ categorical features in the

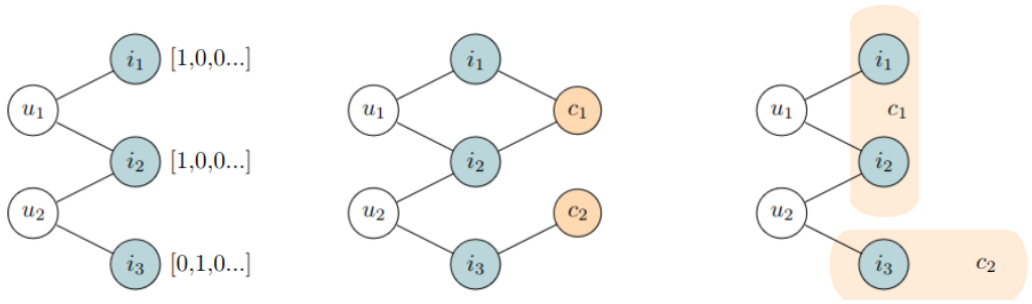

Figure 1: Illustration of three graph models incorporating categorical entity features. In the first graph, categories are considered as features of items by creating binary vectors encoding the categorical value. The second graph represents each categorical value as extra nodes. The graph on the right shows categories as hyperedges.

learning process. In the context of entity categorical features, early latent factor models have utilized them as auxiliary information, serving as sparse features to create a user/item side information matrix (Singh & Gordon, 2008; Veloso et al., 2019; Pasricha & McAuley, 2018).

Representation learning models also leverage user and item features to predict user-item connection (Maeng et al., 2022; Cheng et al., 2016; Covington et al., 2016). These methodologies construct an input feature matrix using dense and sparse user/item features. For example, Dong et el. Dong et al. (2017) constructed user and item feature matrix for the movie lens datasets where item features contain 18 movie genre categories encoded as binary vectors. Similarly, it utilizes the user's age, gender, and occupation.

2.2 Categorical Features in Graph Neural Networks

In the absence of rich, distinctive input features for items and users, it is well-established to use the identity matrix of a node as an input features matrix, e.g., each node is described as one hot encoding vector and is unique for every other nodes (He et al., 2020; van den Berg et al., 2017; Li et al., 2023). However, when relevant entity features exist, authors rely primarily on two methods.

The first commonly used technique is constructing binary-encoded vectors to represent categorical values. These binary vectors then are used directly as input features, or they are concatenated with the identity matrix (Sun et al., 2020; Guo et al., 2021). The latter is usually used when entities have insufficient unique features to differentiate users/items.

The second method used is category values as nodes. Several studies have adopted this technique (Zheng et al., 2023; Liu et al., 2022; Hu et al., 2020; Li et al., 2021). For example, Liu et al. (2022) created a use-item-attribute graph. Items were connected to attribute nodes, and user-attribute interest was extracted by an attribute-aware attention mechanism. Similarly, Zheng et al. (2023) included item categorical features (price and categories) as extra nodes on the graph. They designed a two-branch factorization machine to extract price preferences (Sun et al., 2019). Li et al. (2021) utilized item attributes such as categories and location as nodes.

The effectiveness of those methods is not very obvious. For example, some authors have pointed out the limitations of the binary-encoded category method (Zhang et al., 2022; Liu et al., 2022). When included as one-hot encoded features, it becomes very sparse where only a few entries are non-zero, which can lead to learning unreliable parameters (Liu et al., 2022). Similarly, creating category-value nodes and connecting them with item nodes might not directly extract user category preferences and dependences (Zhang et al., 2022).

Furthermore, there is a complex interdependence between the graph model used and the GNN architecture realizing the recommender engines. Various aggregation mechanisms for graphs and hypergraphs have been proposed. Moreover, approaches not only differ in their graph model for categorical features but also use various techniques, such as attention mechanisms, making it difficult

Table 1: Summary of different methods: $|V|$ is number of nodes, $|E|$ is number of edges, M is initial feature vector size, K is number of all categorical values, C_u is number of user category features, C_i number of item category features. $|V_u|$ number of user nodes, $|V_i|$ number of item nodes.

Method	Order of Graph	Size of Graph	Features
Without categorical features	$\|V\|$	$\|E\|$	$\|V\| \times M$
Categories as binary features	$\|V\|$	$\|E\|$	$\|V\| \times (M + K)$
Category value nodes	$\|V\| + K$	$\|E\| + (\|V_u\| \times C_u + \|V_i\| \times C_i)$	$(\|V\| + K) \times M$
Categories as Hyperedges	$\|V\|$	$\|E\| + (\|V_u\| \times C_u + \|V_i\| \times C_i)$	$\|V\| \times M$

to assess the impact of the representation of categorical features, although this is a crucial design decision.

We briefly mention that some authors used categorical features as edge features, mostly in context-aware recommender engines (Wu et al., 2022). Other research papers (Guo et al., 2021) built dual graphs to incorporate attribute information, one for user-item interactions and one for the attributes.

Another way, we suggest, categorical features can be utilized on user-item graphs is to use them directly as hyperedges. In graph theory, hyperedges are edges that connect any number of nodes simultaneously (Yadati et al., 2019; Huang & Yang, 2021). For example, two items can be linked via a hyperedge because they share the same brand and price level.

The concept of hyperedges is not new, and many studies have used hypergraphs and hyperedges to model recommender engines (Zhang et al., 2022; Wang et al., 2020; Xia et al., 2021). However, most studies are limited to session-based recommendation engines, and most importantly, those studies create hyperedges based on combinations of itemID and/or attributes, e.g. they introduce category value nodes into the graph. For example, Zhang et al. (2022) proposed session-based recommender engines, where nodes are price, category, and items. Hyperedges then connect some combination of those nodes, e.g., all price nodes within the session.

The main advantage of hyperedges is that it can naturally model high-order interactions, which is common in real-world scenarios and thus can be utilized to overcome the above-mentioned limitation.

3 PRELIMINARIES

3.1 GRAPH AND INPUT FORMULATION FOR DIFFERENT TECHNIQUES

Figure 1 depicts three discussed approaches for incorporating categorical features into GNN-based user-item recommender engines, e.g., binary encoding of categorical features, category value nodes, and categories as hyperedges.

Below, we describe how a graph changes when adopting different methods. We define an undirected bipartite graph $G = (V, E)$ with V consisting of user and item nodes $V = V_u \cup V_i$. Edge set E contains interaction edges between user-item nodes(u, i). Users and items have non-categorical feature vectors of size M associated. Let us assume that users and items have C_u and C_i categorical features, respectively. Finally, K is a number of all category values for both user and item. Table 1 summarizes how the order of the graph, size of the graph, and feature matrix transform with different methods. The order is defined as the number of nodes and size as the number of edges (Harris et al., 2008)

We can observe that in the hyperedge method, the size of the graph increases by the number of nodes times category features without increasing the number of nodes or feature matrix. In general, the size of the graph increases by the number of hyperedges. In the case of binary-encoded categorical values, input features grow by the number of all category values. While for category values as nodes, both the graph's size and the graph's order increase, as does the feature matrix.

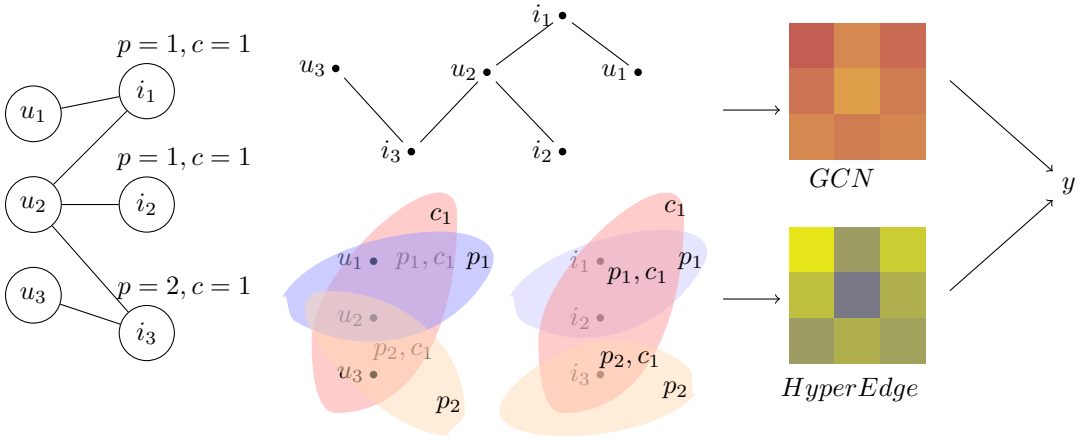

Figure 2: An example of incorporating price level and product category features as hyperedges. As an input, we have a bipartite graph with two types of nodes (users, items), and items have two categorical attributes. For example, i_3 has price level 1 and category value 1. On the bipartite graph, we have two types of aggregation. A simple GCN layer that aggregates neighboring information. The second is hyperedge aggregations. Finally, they are combined to make a final prediction.

4 METHODOLOGY

In previous sections, we discussed existing methods and motivated our new approach and the concept underlying it. Here, we discuss its concrete realization and present a unified framework to compare the different methodologies. We adopted categories as hyperedge concept for studying price and product category dependency for ecommerece recommender engine.

Figure 2 illustrates the proposed model architecture. Here, we have a standard undirected bipartite graph $G = (V, E)$ with V consists of user and item nodes, $u \in U, i \in I$. Items have two categorical features: $p \in P$ and $c \in C$ (p stands for the price level and c for the product category). Edge set E contains interaction edges (u, i) and hyperedges for each category value h_c, h_p, h_{cp}. Hyperedge construction is as follows: For every category value, one hyperedge is created. Then, all items that share the same category value are connected. Similarly, we create hyperedge for all users who interacted with items of the same category value. In addition, interactions hyperedges are constructed h_{cp} (e.g., price level=1 and product category='tablets' is one hyperedge).

During the learning process, we have two types of aggregation on graphs. One is a standard graph convolutional layer (Kipf & Welling, 2017) to capture neighborhoods, and the second is a hyperedge aggregation. Finally, we combine these two aggregations and use them for the prediction. The Pseudo algorithm algorithm is shown in Algorithm 1.

4.1 ENCODER

Below is the exact formulation of the encoding part of the model. As mentioned above, for neighborhood aggregation, we use the GCN layer. For the hyperedge aggregation, we adapt the UniSAGE aggregation (Huang & Yang, 2021) extending GraphSAGE (Hamilton et al., 2017) to hypergraphs. The exact node-level formulation for a node v is:

$$h_v^{l+1} = \sigma\left(\left(W_n^l \sum_{u \in N(v) \cup \{v\}} \frac{1}{\sqrt{\tilde{d}_j \tilde{d}_i}} h_u^l\right) \Big\| \left(W_h^l \left(h_v^l + \sum_{e \in E_v} h_e^l\right)\right)\right), \tag{1}$$

where the left term corresponds to the node-level formulation of GCN (Kipf & Welling, 2017), E_v is the set of hyperedges containing v, h_e^l is the embedding of the hyperedge e obtained as

Table 2: Statistics of the datasets

Datasets	#users	#items	#interaction	#price level	#category
Amazon Grocery	8535	12906	145755	21	24
Amazon Tools	17642	26087	291361	21	13
Yelp	19301	17587	452931	4	83

$h_e^l = \frac{1}{|e|}\sum_{u \in e} h_u^l$, W_n^l, W_h^l are learnable parameters for neighborhood and hyperedge aggregation, respectively, σ is a nonlinear activation function and $\|$ denotes concatenation.

4.2 Decoder and Loss function

To predict user preferences, we use the inner product of the final user and item representations. We combine this with the Bayesian Personalized Ranking (BPR) loss function (Rendle et al., 2009) to train the model. Combination of inner product and BPR loss is a well-established framework for training recommender engines (Yue et al., 2023; He et al., 2020; Liu et al., 2022; Wang et al., 2019; Li et al., 2021; Lin et al., 2022). The exact formulation of the decoder is as follows:

$$y_{ui} = z_u^T z_i$$

where z_u, z_i are the final user item representation. This approach implies that the similarity of a user to an item is proportional to the dot product of their representation (Hamilton, 2020).

BPR loss is a widely used method since it considers positive and negative user-items pairs. BPR encourages models to rank positive user-item interactions higher than negative user-item interactions. The precise formulation of the loss function is as follows:

$$L = \sum_{(u,i,j) \in O} -\ln(\sigma(s(u,i)) - \sigma(s(u,j))) + \lambda \|\Theta\|^2$$

Where O denotes the set of positive-negative sample pairs, representing user u with a positive item i and a negative item j, σ denotes the sigmoid function, which maps the predicted scores to probabilities between 0 and 1, $s(u,i), s(u,j)$ are predicted scores for positive and negative items, respectively, and Θ represents the model parameters, where λ controls L2 regularization.

5 Experimentation

5.1 Experimental settings

Research Questions: In our study, we performed extensive experimentation to evaluate various approaches and answer the following research questions:

- **RQ1** Do existing GNN-based user-item recommendation systems benefit from categorical entity (user/item) features?
- **RQ2** What is the best way to incorporate categorical features in a graph model?
- **RQ3** Can we develop GNN-based user-item recommender engines effectively using categorical features to improve their prediction accuracy?

Datasets: To examine model performances, we use three real-world data sets: Yelp2018, Amazon Tools 5core, and Amazon Grocery 5core datasets. Table 2 depicts a summary of datasets.

- Yelp2018[1] dataset is widely used for recommender engines. Here, restaurants are considered as items for which users have reviews. Price categories, e.g., how expensive the restaurant is, and restaurant subcategories are extracted. We follow the same approach as the PUP paper and use a 10-core setting, only keeping users and items with at least ten interactions.

[1] https://www.yelp.com/dataset/

- Amazon Tools 5 core[2] is adapted. Subcategories and prices are used to create categorical features. Price buckets are created by grouping values within an interval of 5. Furthermore, subcategories are used to create category features. The Same as above, we apply 10-core settings.
- Amazon Grocery 5 core[3] similar to Amazon Tools dataset we use subcategories and prices. Price categories are created by grouping prices into 5-euro buckets. The first-level subcategories are used as categories. The same as the above 10-core setting is applied.

For each dataset, we rank the interactions by timestamps. We then split consecutively 60/20/20 as training, validation, and testing datasets. We use 1:1 negative sampling, e.g., for every positive training edge, we create one negative sample. Item is considered negative if a user did not interact with it.

Evaluation Metrics: To evaluate the model performances, we adapted two widely used evaluation metrics, Recall at K and Normalized Discounted Cumulative Gain (NDCG) at K position (He et al., 2015). Recall@K measures how many items are in the top-K recommended items, while NDCG@K focuses on the quality of the ranking. NDCG@K takes into account the position in which item was recommended. We used 50 and 100 top-K ranks. The reported results are average values over the number of users. Furthermore, we run each method 10 times and mean values are reported in the tables.

Baselines: To answer RQ1, we construct different variations of the same model where only the input is different, e.g., The categorical features are added either as binary encoded input features or we create category-value nodes on the graph or using them as hyperedges. In addition, one extra model is constructed as a complete baseline where no categorical features are included, only relying upon the user-item identity feature matrix as input features. We describe the exact model formulations.

In the methodology chapter, we described in detail how GCN_h is aggregated. For all other methods in RQ1, we use a simple GCN layer for the model encoding part, e.g., we use the first part of the equation 1, followed by the activation function. The model prediction and training process is identical to GCN_h, which is described in the decoder and loss function section.

- GCN_w $\mathbf{F} \in \mathbb{R}^{n \times n}$ input is the user-item identity feature matrix, where n is the number of nodes.
- GCN_n $\mathbf{F} \in \mathbb{R}^{n+c \times n+c}$ considers categorical values as extra nodes on the graph. e.g., the size of the input matrix is increased by a number of categorical values.
- GCN_f $\mathbf{F} \in \mathbb{R}^{n \times n+c}$ adds categorical values in the feature matrix.
- GCN_h $\mathbf{F} \in \mathbb{R}^{n \times n}$ does not increase the size of the input features matrix but uses categorical features for hyperedge construction.

In each dataset, there are two category features: price level and product category. For RQ1, we test three scenarios per dataset, e.g., only price level, only product category, and both together price level and category. Hence, we have nine different frameworks to compare in total.

To test RQ3, we compare our hyperedge approach with the state-of-the-art models. The competitive models we picked are BPR-MF, A2-GCN, PUP, and CatGCN. All except BPR-MF are incorporating categorical features into the model learning process.

- **BPR-MF** Koren et al. (2009) is a classical matrix factorization method combined with Bayesian personalized ranking loss for optimization. It is only based on user-item interactions and ignores side information.
- **A2 GCN** Liu et al. (2022) is an attribute-aware recommender engine that incorporates categorical attributes as extra nodes in the graph. It uses an attention mechanism to model user preferences.
- **PUP** Zheng et al. (2023) is price aware recommender engine. This method considers categories as nodes and deploys a custom decoder to capture the global and local influence of prices and categories.

[2] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/
[3] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

Table 3: Performance comparison with different approaches to include categorical features at K=50

		Price		Category		Price And Category	
Dataset	**Model**	**Recall@50**	**nDCG@50**	**Recall@50**	**nDCG@50**	**Recall@50**	**nDCG@50**
Amazon Grocery	GCN_w	0.0745	0.0342	0.0745	0.0342	0.0745	0.0342
	GCN_n	0.0769	0.0352	0.0751	0.0342	0.0782	0.0357
	GCN_f	0.0728	0.0328	0.0720	0.0328	0.0700	0.0317
	GCN_h	**0.0802**	**0.0370**	**0.0813**	**0.0377**	**0.0822**	**0.0377**
Amazon Tools	GCN_w	0.0321	0.0139	0.0321	0.0139	0.0321	0.0139
	GCN_n	0.0346	0.0150	0.0320	0.0139	0.0342	0.0149
	GCN_f	0.0307	0.0132	0.0306	0.0131	0.0289	0.0124
	GCN_h	**0.0383**	**0.0164**	**0.0379**	**0.0165**	**0.0383**	**0.0166**
Yelp	GCN_w	0.2137	0.0984	0.2137	0.0984	0.2137	0.0984
	GCN_n	0.2157	**0.1001**	0.2138	0.0983	0.2158	0.1003
	GCN_f	0.2137	0.0983	0.2133	0.0979	0.2136	0.0980
	GCN_h	**0.2150**	**0.1001**	**0.2172**	**0.1011**	**0.2204**	**0.1024**

- **CatGCN** Chen et al. (2023) approach uses item categorical side information to enrich initial user feature representation. CatGCN is implemented for user node classification tasks. We adopt this approach for link prediction tasks. To adapt this approach for the link prediction, we do as follows: we use items categorical features to enrich users' initial representation. In the case of item features, we adopt the identity matrix. We then combine user and item features and pass them into GCN layers. The training process for the link prediction is identical to our hyperedge approach, e.g., we use the same decoder mechanism.

Implementation Details: For all baselines, we used the publicly available original implementations with their default parameters. We set the maximum epoch for training to 200. For our hyperedge model, we did hyperparameter search for the learning rate in (0.1, 0.01, 0.001, 0.0001) and L2 normalization in (1e-10, 1e-8, 1e-5, 1e-4) using the BPR loss function. The embedding size is fixed for 64. Adam optimizer is used for the optimization. The training happens in full batch mode. We use a one-layer model and report average values over ten runs.

5.2 Performance Comparison RQ1 and RQ2

Table 3 shows model performances at top-K=50 position. In the table, we highlight in bold the best performances. There are several interesting observations. First, we see that adding categorical features to the model is not always beneficial. In all datasets GCN_w is better than GCN_f. This is contrary to the expectation that more features in the model the better. This does not necessarily mean that features are meaningless. Rather, it could be that the model cannot learn reliable parameters for sparse input features.

Including categorical values as extra nodes is usually better than not including them at all. In 7 out of 9 scenarios, GCN_n is better than GCN_w. Furthermore, the results show that for almost all cases, including categorical features as nodes is superior to the binary-encoded method.

The second research question focuses on identifying the best way to include categorical features. Our results suggest that including category features as hyperedges is always better than not including them at all, and by large, the hyperedge method outperforms other methods in almost all scenarios. Only in one case, GCN_n has better results than GCN_h.

Furthermore, performance varies across different datasets, indicating that the efficacy of model selection is influenced by dataset structure.

The results for top-K=100 can be found in the Appendix. We make similar observations as in top-K=50. This finding suggests that models generally do not necessarily and automatically benefit from categorical features. And it should be part of the model selection to decide how to integrate categorical features.

Table 4: Performance comparision with competitive baselines

Datasets	Model	Recall@50	Recall@100	nDCG@50	nDCG@100
Amazon Grocery	BPR-MF	0.0569	0.0834	0.0276	0.0337
	CatGCN	0.0349	0.0607	0.0139	0.0199
	A2-GCN	0.0510	0.0853	0.0212	0.0291
	PUP	0.0745	0.1106	0.0340	0.0424
	GCN_h	**0.0822**	**0.1209**	**0.0377**	**0.0467**
Amazon Tools	BPR-MF	0.0282	0.0443	0.0123	0.0160
	CatGCN	0.0123	0.0232	0.0047	0.0073
	A2-GCN	0.0236	0.0404	0.0097	0.0135
	PUP	0.0321	0.0511	0.0140	0.0184
	GCN_h	**0.0383**	**0.060**9	**0.0166**	**0.0218**
Yelp	BPR-MF	0.2123	0.3280	0.0999	0.1291
	CatGCN	0.1054	0.1831	0.0462	0.0663
	A2-GCN	0.1883	0.2979	0.0889	0.1167
	PUP	**0.2221**	**0.3417**	**0.1024**	**0.1326**
	GCN_h	0.2204	0.3384	**0.1024**	**0.1322**

5.3 Performance Comparison RQ3

In RQ1 and RQ2, we were solely interested in understanding if there is any difference in how categorical features are included in GNNs. Hence, we used standard GCN approaches to compare various techniques.

To answer RQ3, we further compare the hyperedge model with current state-of-the-art models. Table 4 summarizes the experiment results and shows that our approach is, by large, the most effective way to model categorical features with PUP having competitive results. The model performance of GCN_h is particularly strong in the Amazon Grocery and Amazon Tools dataset. The Amazon Grocery dataset GCN_h outperforms second-best results by 10 percent. In Amazon Tools, improvement is almost 18 percent compared to second-best results. In the Yelp dataset, our model has competitive performance. It is notable that in some cases even simple BPR-MF outperforms competitive baselines such as A2-GCN and CatGCN.

The hyperedge model has the simplest architecture compared to A2-GCC, PUP, and CatGCN, which rely on attention mechanisms, customized decoder, or local and global embedding learnings. Still, our approach outperforms those methods and, in some cases, has a significant margin.

6 Conclusions and future work

This research paper examined different methods to incorporate categorical features of entities into GNN-based user-item recommender engines. Extensive experimentation was conducted to compare traditional approaches, such as category-value nodes and binary-encoded category features, to category-value hyperedges, as well as using no categorical features at all. We tested in three datasets with three different scenarios (e.g., including only product category, price level, or both of them together). Our findings suggest that the hyperedge approach outperforms other techniques in all cases. Another interesting observation is that including categorical binary-encoded features makes the model almost always worse than not including them at all. Furthermore, we compared the hyperedge approach to competitive baselines such as PUP, A2-GCN, and CatGCN, which studied categorical features in GNN-based user-item recommender engines. By large, the findings demonstrate the superiority of the hyperedge approach.

For future work, further investigation is needed into how model architecture influences the most effective method for incorporating categorical features. Moreover, we hope that our study will motivate other researchers to dive deep into GNNs' ability to extract complex user preferences as well as category dependencies.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira (eds.), *Recommender Systems Handbook*, pp. 191–226. Springer, 2015. doi: 10.1007/978-1-4899-7637-6_6. URL https://doi.org/10.1007/978-1-4899-7637-6_6.

Linas Baltrunas and Francesco Ricci. Context-based splitting of item ratings in collaborative filtering. In Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme (eds.), *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pp. 245–248. ACM, 2009. doi: 10.1145/1639714.1639759. URL https://doi.org/10.1145/1639714.1639759.

Weijian Chen, Fuli Feng, Qifan Wang, Xiangnan He, Chonggang Song, Guohui Ling, and Yongdong Zhang. Catgcn: Graph convolutional networks with categorical node features. *IEEE Trans. Knowl. Data Eng.*, 35(4):3500–3511, 2023. doi: 10.1109/TKDE.2021.3133013. URL https://doi.org/10.1109/TKDE.2021.3133013.

Wen-Hao Chen, Chin-Chi Hsu, Yi-An Lai, Vincent Liu, Mi-Yen Yeh, and Shou-De Lin. Attribute-aware recommender system based on collaborative filtering: Survey and classification. *Frontiers Big Data*, 2:49, 2019. doi: 10.3389/fdata.2019.00049. URL https://doi.org/10.3389/fdata.2019.00049.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach (eds.), *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pp. 7–10. ACM, 2016. doi: 10.1145/2988450.2988454. URL https://doi.org/10.1145/2988450.2988454.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pp. 191–198, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pp. 293–296, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589060. doi: 10.1145/1864708.1864770. URL https://doi.org/10.1145/1864708.1864770.

Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. A hybrid collaborative filtering model with deep structure for recommender systems. In Satinder Singh and Shaul Markovitch (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1309–1315. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14676.

James S Dyer and Rakesh K Sarin. Measurable multiattribute value functions. *Operations research*, 27(4):810–822, 1979.

Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. Dual graph enhanced embedding neural network for CTR prediction. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 496–504. ACM, 2021. doi: 10.1145/3447548.3467384. URL https://doi.org/10.1145/3447548.3467384.

William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020. doi:

10.2200/S01045ED1V01Y202009AIM046. URL https://doi.org/10.2200/S01045ED1V01Y202009AIM046.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

John M. Harris, Jeffry L. Hirst, and Michael J. Mossinghoff. *Combinatorics and Graph Theory, Second Edition*. Undergraduate Texts in Mathematics. Springer, 2008. ISBN 978-0-387-79710-6.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pp. 1661–1670, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi: 10.1145/2806416.2806504. URL https://doi.org/10.1145/2806416.2806504.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (eds.), *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pp. 639–648. ACM, 2020. doi: 10.1145/3397271.3401063. URL https://doi.org/10.1145/3397271.3401063.

Linmei Hu, Chen Li, Chuan Shi, Cheng Yang, and Chao Shao. Graph neural news recommendation with long-term and short-term interest modeling. *Inf. Process. Manag.*, 57(2):102142, 2020. doi: 10.1016/j.ipm.2019.102142. URL https://doi.org/10.1016/j.ipm.2019.102142.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 2563–2569. ijcai.org, 2021. doi: 10.24963/ijcai.2021/353. URL https://doi.org/10.24963/ijcai.2021/353.

Won-Seok Hwang, Ho-Jong Lee, Sang-Wook Kim, and Minsoo Lee. On using category experts for improving the performance and accuracy in recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pp. 2355–2358, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311564. doi: 10.1145/2396761.2398639. URL https://doi.org/10.1145/2396761.2398639.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

Chen Li, Linmei Hu, Chuan Shi, Guojie Song, and Yuanfu Lu. Sequence-aware heterogeneous graph neural collaborative filtering. In Carlotta Demeniconi and Ian Davidson (eds.), *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pp. 64–72. SIAM, 2021. doi: 10.1137/1.9781611976700.8. URL https://doi.org/10.1137/1.9781611976700.8.

Yang Li, Fangtao Zhao, Zheng Chen, Yingxun Fu, and Li Ma. Multi-behavior enhanced heterogeneous graph convolutional networks recommendation algorithm based on feature-interaction. *Applied Artificial Intelligence*, 37(1):2201144, 2023.

Chuan Lin, Wei Zhou, and Junhao Wen. Interest-aware contrastive-learning-based GCN for recommendation. *IEEE Access*, 10:126315–126325, 2022. doi: 10.1109/ACCESS.2022.3226369. URL https://doi.org/10.1109/ACCESS.2022.3226369.

Fan Liu, Zhiyong Cheng, Lei Zhu, Chenghao Liu, and Liqiang Nie. An attribute-aware attentive GCN model for attribute missing in recommendation. *IEEE Trans. Knowl. Data Eng.*, 34(9): 4077–4088, 2022. doi: 10.1109/TKDE.2020.3040772. URL https://doi.org/10.1109/TKDE.2020.3040772.

Kiwan Maeng, Haiyu Lu, Luca Melis, John Nguyen, Mike Rabbat, and Carole-Jean Wu. Towards fair federated recommendation learning: Characterizing the inter-dependence of system and data heterogeneity. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, pp. 156–167, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392785. doi: 10.1145/3523227.3546759. URL https://doi.org/10.1145/3523227.3546759.

Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. An attentive interaction network for context-aware recommendations. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (eds.), *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pp. 157–166. ACM, 2018. doi: 10.1145/3269206.3271813. URL https://doi.org/10.1145/3269206.3271813.

Xiangwu Meng, Yulu Du, Yujie Zhang, and Xiaofeng Han. A survey of context-aware recommender systems: From an evaluation perspective. *IEEE Trans. Knowl. Data Eng.*, 35(7):6575–6594, 2023. doi: 10.1109/TKDE.2022.3187434. URL https://doi.org/10.1109/TKDE.2022.3187434.

Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In Lawrence D. Bergman, Alexander Tuzhilin, Robin D. Burke, Alexander Felfernig, and Lars Schmidt-Thieme (eds.), *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pp. 265–268. ACM, 2009. doi: 10.1145/1639714.1639764. URL https://doi.org/10.1145/1639714.1639764.

Rajiv Pasricha and Julian J. McAuley. Translation-based factorization machines for sequential recommendation. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan (eds.), *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pp. 63–71. ACM, 2018. doi: 10.1145/3240323.3240356. URL https://doi.org/10.1145/3240323.3240356.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng (eds.), *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pp. 452–461. AUAI Press, 2009. URL https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf.

Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 650–658, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581934. doi: 10.1145/1401890.1401969. URL https://doi.org/10.1145/1401890.1401969.

Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. Neighbor interaction aware graph convolution networks for recommendation. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (eds.), *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pp.

1289–1298. ACM, 2020. doi: 10.1145/3397271.3401123. URL https://doi.org/10.1145/3397271.3401123.

Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. Research commentary on recommendations with side information: A survey and research directions. *Electron. Commer. Res. Appl.*, 37, 2019. doi: 10.1016/j.elerap.2019.100879. URL https://doi.org/10.1016/j.elerap.2019.100879.

Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017. URL http://arxiv.org/abs/1706.02263.

Bruno M. Veloso, Fátima Leal, Benedita Malheiro, and Juan-Carlos Burguillo. On-line guest profiling and hotel recommendation. *Electron. Commer. Res. Appl.*, 34, 2019. doi: 10.1016/j.elerap.2019.100832. URL https://doi.org/10.1016/j.elerap.2019.100832.

Soumya Wadhwa, Ashish Ranjan, Selene Xu, Jason H. D. Cho, Sushant Kumar, and Kannan Achan. Personalizing item recommendation via price understanding. In Toine Bogers, Marijn Koolen, Casper Petersen, Bamshad Mobasher, Alexander Tuzhilin, Oren Sar Shalom, Dietmar Jannach, and Joseph A. Konstan (eds.), *Proceedings of the Workshops on Recommendation in Complex Scenarios and the Impact of Recommender Systems co-located with 14th ACM Conference on Recommender Systems (RecSys 2020), Online, September 25, 2020*, volume 2697 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020. URL https://ceur-ws.org/Vol-2697/paper4_complexrec.pdf.

Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (eds.), *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pp. 1101–1110. ACM, 2020. doi: 10.1145/3397271.3401133. URL https://doi.org/10.1145/3397271.3401133.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (eds.), *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pp. 165–174. ACM, 2019. doi: 10.1145/3331184.3331267. URL https://doi.org/10.1145/3331184.3331267.

Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. Graph convolution machine for context-aware recommender system. *Frontiers Comput. Sci.*, 16 (6):166614, 2022. doi: 10.1007/s11704-021-0261-8. URL https://doi.org/10.1007/s11704-021-0261-8.

Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 4503–4511. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16578.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha P. Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1509–1520, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1efa39bcaec6f3900149160693694536-Abstract.html.

Guowei Yue, Rui Xiao, Zhongying Zhao, and Chao Li. AF-GCN: attribute-fusing graph convolution network for recommendation. *IEEE Trans. Big Data*, 9(2):597–607, 2023. doi: 10.1109/TBDATA.2022.3192598. URL https://doi.org/10.1109/TBDATA.2022.3192598.

Xiaokun Zhang, Bo Xu, Liang Yang, Chenliang Li, Fenglong Ma, Haifeng Liu, and Hongfei Lin. Price DOES matter!: Modeling price and interest preferences in session-based recommendation. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (eds.), *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pp. 1684–1693. ACM, 2022. doi: 10.1145/3477495.3532043. URL https://doi.org/10.1145/3477495.3532043.

Yu Zheng, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. Incorporating price into recommendation with graph convolutional networks. *IEEE Trans. Knowl. Data Eng.*, 35(2):1609–1623, 2023. doi: 10.1109/TKDE.2021.3091160. URL https://doi.org/10.1109/TKDE.2021.3091160.

A Appendix

Table 5: Performance comparison with different approaches to include categorical features at K=100

		Price		Category		Price And Category	
Dataset	**Model**	**Recall@100**	**nDCG@100**	**Recall@100**	**nDCG@100**	**Recall@100**	**nDCG@100**
Amazon Grocery	GCN_w	0.0745	0.0342	0.0745	0.0342	0.0745	0.0342
	GCN_n	0.1144	0.0439	0.1122	0.0428	0.1153	0.0443
	GCN_f	0.1098	0.0414	0.1074	0.0410	0.1058	0.0400
	GCN_h	0.1191	0.0461	0.1204	0.0468	0.1209	0.0467
Amazon Tools	GCN_w	0.0321	0.0139	0.0321	0.0139	0.0321	0.0139
	GCN_n	0.0552	0.0197	0.0515	0.0184	0.0547	0.0196
	GCN_f	0.0493	0.0175	0.0496	0.0175	0.0470	0.0165
	GCN_h	0.0608	0.0216	0.0604	0.0216	0.0609	0.0218
Yelp	GCN_w	0.2137	0.0984	0.2137	0.0984	0.2137	0.0984
	GCN_n	0.3325	0.1296	0.3301	0.1277	0.3327	0.1298
	GCN_f	0.3306	0.1279	0.3299	0.1274	0.3297	0.1274
	GCN_h	0.3316	0.1296	0.3347	0.1308	0.3384	0.1322

Algorithm 1 An algorithm

Input: $G = (V, E)$**,** $L = 200$
Output: Z
Initialize model parameters
Construct **A** adjacency matrix for neighbourhood
Construct hyperedges $h^u_c, h^u_p, h^u_{cp}, h^i_c, h^i_p, h^i_{cp}$ for users and items respectevly
for $i = 1, \ldots, L$ **do**
  Obtain X^n using GCN layer for neighbourhood aggregation
  Obtain all X^j_k where $j \in \{u, i\}, k \in \{c, p, cp\}$ by hyperedge convolution
  Obtain X^{hyper} by summing all hyperedge convolutions
  Obtain final node embeddings **Z** by concatinating X^n and X^{hyper}
  Update model parameters by loss function
end for
Return final node Embeddings **Z**