Published: 22 November 2025
Online AM: 04 June 2025
Accepted: 30 May 2025
Revised: 17 April 2025
Received: 21 November 2024

RESEARCH-ARTICLE

Sheaf4Rec: Sheaf Neural Networks for Graph-based Recommender Systems

ANTONIO PURIFICATO, Amazon.com, Inc., Seattle, WA, United States
GIULIA CASSARÀ, Sapienza University of Rome, Rome, RM, Italy
FEDERICO SICILIANO, Sapienza University of Rome, Rome, RM, Italy
PIETRO LIO', University of Cambridge, Cambridge, Cambridgeshire, U.K.
FABRIZIO SILVESTRI, Sapienza University of Rome, Rome, RM, Italy

Sheaf4Rec: Sheaf Neural Networks for Graph-based Recommender Systems

ANTONIO PURIFICATO*, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Italy and Eu Intech, Amazon Europe Core Sàrl, Luxembourg, Luxembourg
GIULIA CASSARÀ, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Italy
FEDERICO SICILIANO, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Italy
PIETRO LIÒ, Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom of Great Britain and Northern Ireland
FABRIZIO SILVESTRI, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Italy

Recent advancements in Graph Neural Networks (GNN) have facilitated their widespread adoption in various applications, including recommendation systems. GNNs have proven to be effective in addressing the challenges posed by recommendation systems by efficiently modeling graphs in which nodes represent users or items and edges denote preference relationships. However, current GNN techniques represent nodes by means of a single static vector, which may inadequately capture the intricate complexities of users and items. To overcome these limitations, we propose a solution integrating a cutting-edge model inspired by category theory: Sheaf4Rec. Unlike single vector representations, Sheaf Neural Networks and their corresponding Laplacians represent each node (and edge) using a vector space. Our approach takes advantage of this theory and results in a more comprehensive representation that can be effectively exploited during inference, providing a versatile method applicable to a wide range of graph-related tasks and demonstrating unparalleled performance. Our proposed model exhibits a noteworthy relative improvement of up to 8.53% on F1-Score@10 and an impressive increase of up to 11.29% on NDCG@10, outperforming existing state-of-the-art

*Work done before joining Amazon.

This work was partially supported by projects FAIR (PE0000013), under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and project NEREO (Neural Reasoning over Open Data), funded by the Italian Ministry of Education and Research (PRIN) Grant no. 2022AEFHAZ.

Authors' Contact Information: Antonio Purificato, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Lazio, Italy and Eu Intech, Amazon Europe Core Sàrl, Luxembourg, Luxembourg; e-mail: purificato@diag.uniroma1.it; Giulia Cassarà, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Lazio, Italy; e-mail: giulia.cassara@uniroma1.it; Federico Siciliano, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Lazio, Italy; e-mail: siciliano@diag.uniroma1.it; Pietro Liò, Computer Science and Technology, University of Cambridge, Cambridge, Cambridgeshire, United Kingdom of Great Britain and Northern Ireland; e-mail: pietro.lio@cl.cam.ac.uk; Fabrizio Silvestri, Department of Computer, Control and Management Engineering Antonio Ruberti, Universita degli Studi di Roma La Sapienza and Amazon, Rome, Italy; e-mail: fsilvestri@diag.uniroma1.it.

ACM 2770-6699/2025/11-ART29
https://doi.org/10.1145/3742898

models such as Neural Graph Collaborative Filtering (NGCF), KGTORe and other recently developed GNN-based models. In addition to its superior predictive capabilities, Sheaf4Rec shows remarkable improvements in terms of efficiency: we observe substantial runtime improvements ranging from 2.5% up to 37% when compared to other GNN-based competitor models, indicating a more efficient way of handling information while achieving better performance. Code is available at https://github.com/antoniopurificato/Sheaf4Rec.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Mathematics of computing** → **Algebraic topology**; • **Computing methodologies** → **Neural networks**; • **Human-centered computing** → Collaborative filtering;

Additional Key Words and Phrases: Recommender systems, graph neural networks, sheaf theory

ACM Reference Format:
Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. 2025. Sheaf4Rec: Sheaf Neural Networks for Graph-based Recommender Systems. *ACM Trans. Recomm. Syst.* 4, 2, Article 29 (November 2025), 26 pages. https://doi.org/10.1145/3742898

1 Introduction

Graph Neural Networks (GNNs) [45] have demonstrated remarkable capabilities, achieving outstanding performance in a variety of tasks, with notable examples including Applied Calculus [59], Drug discovery [12], and Natural Language Processing [36]. GNNs excel especially in contexts where relationships play a central role, such as Internet information services, which by their nature represent data in relational graphs structures [42]. For instance, social media relationships can be modelled as a unified graph, with nodes representing individuals and edges representing mutual connections.

Expanding on the idea of using graphs to represent complex relationships, GNNs find a standout real-world application in collaborative filtering for recommender systems [21, 39]. Collaborative filtering methods work by predicting user ratings for items based on the decisions of users with similar preferences. This approach is central to many contemporary recommender systems, seen both in operational environments [1] and at the forefront of research [6, 20]. The relations between users and items can in fact be visualized as a bipartite user-item graph, where labeled edges denote observed interactions. With the rise of Deep Learning techniques, including GNNs, recommendation models have reached unparalleled benchmarks [5, 54]. This reflects a transformative trend in the use of advanced graph-based models to improve the performance of recommender systems.

Despite their impressive performance, GNNs encounter significant limitations when it is necessary to represent the inherently complex interactions between users and items. Traditional GNNs model these relationships through single-vector representations, which often condense the intricate, diverse user-item features into a fixed, low-dimensional space. This vector-based reduction can lead to an oversimplified understanding of interactions, masking subtle yet crucial contextual and relational factors that influence recommendation quality. Such an approach tends to overlook higher-order relations and the varying significance of each interaction's contextual elements, as previously highlighted in studies on the bottleneck effect of GNNs [2].

A solution emerges with the introduction of a novel class of GNN architectures known as ***Sheaf Neural Networks*** **(SNNs)**. SNNs address these limitations drawing theoretical foundations from Category Theory [7]. Unlike traditional GNNs, which rely on static, globally shared vector spaces for all nodes, SNNs utilize dynamic, interaction-specific vector spaces that are instantiated only at inference time. This allows each user-item interaction to be characterized by its own multi-dimensional representation, which better captures the fine-grained semantics and context of that specific interaction. In other words, while conventional models map all users and items to a single latent space, SNNs can flexibly adjust the dimensionality and orientation of local feature spaces

across the graph. This leads to more expressive and adaptive modeling capabilities, which is especially important in recommendation scenarios where user preferences are diverse and often context-dependent. SNNs have been applied to various tasks and have demonstrated superiority over traditional GNNs [11]. This inherent flexibility makes them particularly suitable for recommender systems, where relationships between users and items are complex and context-dependent, benefiting from a nuanced, inference-time adaptation.

Our approach, Sheaf4Rec, links the theoretical insights of SNNs to the practical domain of item recommender systems. Specifically, our work shows that the use of SNNs in graph-based recommender systems significantly improves their performance.

The primary original contributions of our research are as follows:

— **Novel architecture for recommender system:** We introduce Sheaf4Rec, a novel architecture for recommender systems utilizing SNNs. This approach consistently achieves state-of-the-art performance in its designated tasks, highlighting not only the rigorous mathematical formalism of SNNs, but also their practical applicability in scenarios where relationship information is essential for constructing holistic representations of the objects involved. This is especially relevant when the inherent characteristics of these objects are ambiguous when considered in isolation.

— **Performance Evaluation Across Multiple Datasets:** Our extensive experimentation across multiple datasets for top-K recommendation underscores the viability and promise of our solution. Notably, Sheaf4Rec surpasses all existing benchmarks in the focal tasks of this study. For instance, we achieve relative improvements over state-of-the-art methods in terms of NDCG@10 by 11.29% on the Facebook Books dataset, by 8.77% on the Yahoo! Movies dataset in terms of F1@20 and by 8.24% on the MovieLens 1M dataset in terms of NDCG@20.

— **Efficiency gains in recommendation computation:** Leveraging the advancements of our sheaf-based architecture, recommendation computations are significantly expedited. Across the various datasets used in our studies, Sheaf4Rec shows relative efficiency improvements ranging from 2.5% to 37%, underscoring the practical efficiency gains that can be achieved by adopting our proposed SNN-based approach in item recommendation systems.

The rest of the article is organized as follows. Section 2 provides a comprehensive review of related works. In Section 3, we offer an overview of GNNs and sheaf theory, laying the groundwork for our approach. Our methodology is delineated in Section 4, followed by a discussion on implementation details, encompassing dataset selection and experimental configurations, in Section 5. Section 6 presents a thorough analysis of the efficacy of our proposed solution, while Section 7 encapsulates our findings and offers insights into potential avenues for future research.

2 Related Work

Recommender systems have been extensively studied, and various approaches have been proposed in the literature.

2.1 Matrix Factorization-Based Recommender Systems

Matrix Factorization (**MF**) is a popular collaborative filtering technique that aims to decompose a user-item interaction matrix into lower-dimensional latent factors. This low-rank approximation can capture the underlying structure of the data and yield better recommendations [8].

One of the most well-known MF-based methods for recommender systems is **Singular Value Decomposition** (**SVD**) [60]. However, traditional SVD struggles to deal with missing data, which is common in recommendation scenarios [26]. To address this issue, Koren et al. [26] introduced

SVD++, an extension of the basic SVD that incorporates implicit feedback into the factorization process.

Non-negative Matrix Factorization (**NMF**), introduced by Lee and Seung [28], represents another significant development in MF-based recommender systems. NMF imposes non-negativity constraints on the factor matrices, which can aid in interpretability and improve recommendation performance.

The idea of TrustMF [56] was to improve recommendation by using more data. In addition to explicit rating data, TrustMF also leverages the social trust relationships between users as auxiliary information. Trust is a crucial social cue because people are more inclined to accept the opinions of those they trust. By factorizing the trust network according to the directional feature of trust, the users are mapped to two low-dimensional spaces: trustee and truster. Next, separate models of the truster and trustee space are created to extract latent features of individuals and objects.

2.2 Autoencoder-based Recommender Systems

An inherent limitation of MF-based recommender systems is their linear nature, which constrains their ability to capture complex relationships between users and items [9]. Additionally, these systems generally disregard any side information (e.g., user demographics or item attributes) that could potentially benefit the recommendation process. Furthermore, MF methods can struggle in cold-start situations where little or no information about new users or items is available [9, 44]. To address these limitations, Autoencoder-based recommender systems have emerged as a promising solution. Autoencoders are unsupervised learning models that aim to learn a compressed representation of input data by encoding it into a lower-dimensional space and then reconstructing the original input from the encoded representation [3, 58].

One of the early works in autoencoder-based recommender systems was presented by Sedhain et al. [46], who proposed AutoRec, an autoencoder framework for CF. AutoRec learns a latent feature representation of users or items, capturing the nonlinear relationships between them. Building on this, Liang et al. [30] introduced the **Variational Autoencoder** (**VAE**) for CF, which learns a probabilistic latent representation to better model the uncertainty in user-item preferences.

Recommendation via Dual-Autoencoder (**ReDa**) [61] takes a novel approach by using two AEs to simultaneously learn hidden latent representations for users and items, minimizing the deviations in the training data. However, ReDa considers only explicit feedback information between users and items, such as rating matrix and check-in matrix, and ignores side information about users or items.

Sachdeva et al. [41] implement a recurrent version of the VAE, where instead of passing a subset of the whole history regardless of temporal dependencies, they pass the consumption sequence subset through a recurrent neural network.

Drif et al. [10] propose EnsVAE, which is based on mapping the original data to higher-order features interactions. EnsVAE learns the interest probability distribution on a per-item basis, allowing the VAE to correctly learn and represent the distribution pattern. Moreover, by sampling from the distribution, complete rows of new users and columns of new resources are generated even with minimal interaction.

MacridVAE [32] aims to disentangle the underlying user intents that drive user behaviors. Specifically, it employs a β-VAE to model the generative process of a user's personal history. The model assumes that multiple latent factors influence user behaviors, and it seeks to achieve disentangled representations of these factors from the observed user behaviors. The main difference with respect to our approach is that they learn a set of prototypes and then capture the preference of each about the different intentions separately. Sheaf4Rec instead exploit an entire vector space, allowing the encapsulation of all aspects of users or items, without the need to learn prototypes.

2.3 Graph Neural Networks-based Recommender Systems

When dealing with sparse datasets, autoencoders struggle to accurately reconstruct the input [27], resulting in latent representations that fail to adequately generalize to unseen data or capture the nuances of user preferences, ultimately leading to poor recommendations.

In contrast, GNNs have emerged as a powerful class of deep learning models, gaining attention due to their ability to handle complex relational data. Their application in recommender systems has become increasingly popular, capitalising on the inherent graph structure of user-item relationships.

An early work in GNN-based recommender systems by van den Berg et al. [48] introduced the **Graph Convolutional Matrix Completion** (**GCMC**) method for link prediction in bipartite graphs. GCMC applies graph convolution operations to user-item interactions, effectively learning latent representations for recommendation, showing improved performance compared to traditional MF-based methods.

NGCF [51] designs an innovative embedding propagation layer that refines user or item embeddings by aggregating those of interacted items or users. The embedding refinement step explicitly injects collaborative signal into embeddings. By stacking multiple embedding propagation layers, the model captures collaborative signals in higher order connectivities.

STAR-GCN [57] adopts a multi-block architecture featuring a stack of GCN encoder-decoders combined with intermediate supervision. This approach allows the propagation of the initial node embeddings and features across the whole graph via an encoder-decoder architecture. The graph encoder captures semantic graph structures and input content features, while the decoder works to recover the input node embeddings. In contrast to GCMC [48], utilizing one-hot encoding node inputs, STAR-GCN employs low-dimensional user and item latent factors as inputs to limit the complexity of the model space.

LightGCN [19], based on the GCN approach, reduces the computational cost by adopting a straightforward weighted sum aggregator and discarding feature transformation and nonlinear activation. This streamlined variant reduces the trainable parameters to the embeddings in the initial layer. The model uses layer combination to avoid oversmoothing, make the final representation more comprehensive and capture the effect of graph convolution with self-connections.

Recently, different approaches based on contrastive learning, have been proposed. For example, **Candidate-aware Graph Contrastive Learning (CGCL)** for Recommendation [18] explores the relationship between the user and the candidate item in the embedding at different layers and use similar semantic embeddings to construct contrastive pairs, thereby enforcing structural neighborhood consistency while preserving semantic similarity. Similarly, Jiang et al. [23] introduce an adaptive self-supervised framework composed by a graph generative model and a graph denoising model to synthesize views tailored to the underlying data distribution. This procedure enhances the robustness of the graph collaborative filtering by distilling additional training signals derived from contrastive learning.

In summary, GNNs have shown promising results in recommender systems by successfully modeling the complex relationships between users and items, contributing to improved recommendation quality and user satisfaction [15].

3 Background

We provide a brief overview of the basic concepts that are essential for understanding our proposed solution. First, we outline the main characteristics of GNNs, followed by an introduction to cellular sheaf theory. Finally, we present the learning algorithm central to our approach: *Neural Sheaf Diffusion*. The following discussion summarizes the key theoretical definitions and methods introduced by Barbero et al. [4]. The notation used throughout this article is illustrated in Table 1.

Table 1. Notation Used in the Article

Symbol	Description
v_i	Node i
e_{ij}	Edge from v_i to v_j
$N(v)$	Neighbourhood of node v
$H(l)$	Hidden state at node l
G	Graph
V	Set of nodes
E	Set of edges
A	Adjacency matrix
$\mathcal{F}_{v \trianglelefteq e}$	Restriction map between node-edge pair $v \trianglelefteq e$
C^i	Space of i-cochains
I_n	Identity matrix of size n

3.1 Graph Neural Networks

The rise of GNNs can be attributed to the advancement in both **Neural Networks** (**NNs**) and **Graph Representation Learning** (**GRL**) [55]. NNs excel at extracting localized features from Euclidean data, such as images or texts. However, adapting NNs to non-Euclidean data, like graphs, requires generalization to accommodate variable object sizes (e.g., nodes in graphs) [29]. On the other hand, GRL focuses on generating low-dimensional vector representations for nodes, edges, or subgraphs of a given graph, effectively capturing the intricate connectivity structures inherent in graphs.

A directed graph $G = (V, E)$ consists of a set of nodes V and a set of edges E. Let $v_i \in V$ be a node and $e_{ij} = (v_i, v_j) \in E$ be an edge pointing from v_i to v_j. The set of edges can also be described by an adjacency matrix A as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

If the graph is undirected, we have $e_{ij} = e_{ji} \in E$ and $a_{ij} = a_{ji} \in A$, thus the adjacency matrix is symmetric.

The neighborhood of a node v is denoted as:

$$N(v) = \{u \in V \mid (v, u) \in E\}. \tag{2}$$

The degree matrix D of G contains information about the number of edges attached to each vertex:

$$D_{ij} = \begin{cases} |N(i)| & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Using the definitions of degree matrix D and adjacency matrix A from, respectively, Equations (1) and (3), the Laplacian matrix L is easily expressed as $L = D - A$. Given graph data, the primary objective of GNNs is to iteratively aggregate feature information from neighbors and incorporate this aggregated information with the current central node representation during the propagation process. From a network architecture standpoint, GNNs consist of several propagation layers, which involve aggregation and update operations.

The update operation at the lth layer is expressed as:

$$H^{(l)} = f(H^{(l-1)}, A), \tag{4}$$

where $H^{(l)} \in \mathbb{R}^{|V| \times d}$ denotes the hidden state of nodes at layer l, A is the adjacency matrix, and f is a function that combines, for each node, the previous layer's hidden state $H^{(l-1)}$ with its neighbours' aggregated information.

For example, in the case of GCN [25], the update operation can be defined as:

$$H^{(l)} = h\left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^{(l-1)} W^{(l)}\right), \tag{5}$$

where h is a non-linear activation function, D is the degree matrix, A is the adjacency matrix, $H^{(l-1)} \in \mathbb{R}^{|V| \times d}$ denotes the hidden state of nodes at layer $l-1$ and $W^{(l)}$ is the matrix of learnable weights at layer l.

By stacking multiple layers of aggregation and update operations, GNNs can learn expressive node representations that effectively capture both local and global graph properties, making them suitable for graph-related tasks such as node classification, link prediction, and graph classification. We can define local information the as that associated with a specific node or a small neighbourhood of nodes in the graph, while global relationships refer to the overall structure or patterns that emerge when considering the entire graph [33].

3.2 Cellular Sheaf Theory

Cellular sheaves provide a framework for analyzing graph-structured data by combining sheaf theory [47] with graph theory [53].

In a cellular sheaf, data is associated with both the nodes and edges of a graph, effectively representing local and global structures. This section comprehensively explains the formulation of cellular sheaves as applied to graphs.

Given an undirected graph $G = (V, E)$, for each node $v \in V$, we associate a vector space or algebraic structure $\mathcal{F}(v)$. Similarly, for each edge $e \in E$, we associate a vector space or algebraic structure, $\mathcal{F}(e)$. $\mathcal{F}(v)$ and $\mathcal{F}(e)$ the local data on the graph.

A restriction map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \rightarrow \mathcal{F}(e)$ represents the relationship between local data on nodes and edges for each incident node-edge pair $v \trianglelefteq e$. In this context, the vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$, corresponding to nodes and edges, are called stalks. The restriction maps play a key role in transforming cellular sheaves into a tangible representation that can be used to generate recommendations.

For a given sheaf $(G, \mathcal{F})$, the space of 0-cochains $C^0(G, \mathcal{F})$ is defined as the direct sum over the vertex stalks:

$$C^0(G, \mathcal{F}) = \bigoplus_{v \in V} \mathcal{F}(v). \tag{6}$$

Similarly, the space of 1-cochains $C^1(G, \mathcal{F})$ encompasses the direct sum over the edge stalks.

By assigning an arbitrary orientation to each edge e, the co-boundary map serves as a linear map associating 0-cochains to 1-cochains by capturing the difference between the data associated with the vertices connected by an edge. Formally, given two nodes $u, v \in V$ and some arbitrary orientation for each edge $e = u \rightarrow v \in E$, the co-boundary map is defined as:

$$\delta : C^0(G, \mathcal{F}) \rightarrow C^1(G, \mathcal{F}) = \delta(x)_e = \mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u. \tag{7}$$

The sheaf Laplacian is an operator that generalizes the concept of the graph Laplacian to the setting of sheaves on graphs [16]. The graph Laplacian is a matrix that captures the connectivity structure of a graph, and it is widely used in graph analysis and spectral graph theory [14]. The sheaf Laplacian extends these ideas to capture not only the graph's connectivity but also the local-to-global relationships of the data on the graph as described by the sheaf. It describes how the elements of a vector space are transported via rotations in another neighbouring vector space in

a way similar to how tangent vectors are moved across a manifold via parallel transport [7]. The sheaf Laplacian is an operator that maps 0-cochains to 0-cochains:

$$L_{\mathcal{F}} = \delta^T \delta = \sum_{v, u \trianglelefteq e} \mathcal{F}^T_{v \trianglelefteq e} (\mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u). \tag{8}$$

Notably, the sheaf Laplacian is closely related to the Laplacian, but it takes restriction maps into account instead of edges. Finally, the normalized sheaf Laplacian $\Delta_{\mathcal{F}}$ is expressed as:

$$\Delta_{\mathcal{F}} = D^{-\frac{1}{2}} L_{\mathcal{F}} D^{-\frac{1}{2}}. \tag{9}$$

Here, D represents the diagonal of $L_{\mathcal{F}}$.

An example often used to illustrare the concept of sheaves comes from opinion dynamics [17]. Given a social network represented as a graph G with vertices representing agents and (undirected) edges representing pairwise communication, a discourse sheaf $\mathcal{F}$ is naturally generated. Each agent v has an opinion space, a real vector space with basis some collection of topics. Points on each axis correspond to negative, neutral, or positive opinions on the topic, with a positive/negative intensity registered by the scalar value. This opinion space comprises the stalk $\mathcal{F}(v)$ of the discourse sheaf over v. Given an edge e between vertices u and v, it is assumed that there exists a certain set of basic topics about which the two agent discuss. Each agent represents its opinions on the topics under discussion by formulating stances as a linear combination of existing opinions on personal basis topics. These opinions are linear transformations $\mathcal{F}_{u \trianglelefteq e} : \mathcal{F}(u) \to \mathcal{F}(e)$ and $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$. If the agents have opinions $x_u \in \mathcal{F}(u)$ and $x_v \in \mathcal{F}(v)$, then they have expressed consensus when $\mathcal{F}_{u \trianglelefteq e}(x_u) = \mathcal{F}_{v \trianglelefteq e}(x_v)$.

3.3 Neural Sheaf Diffusion

Considering a graph $G = (V, E)$, each individual node $v \in V$ is associated with a d-dimensional feature vector $x_v \in \mathcal{F}(v)$. The individual vectors x_v are column-stacked to create an nd-dimensional vector $x \in C^0(G, \mathcal{F})$. The vectors belonging to $C^0(G, \mathcal{F})$ form the columns of the feature matrix $X \in \mathbb{R}^{(nd) \times f}$.

Sheaf diffusion can then be described as a process that operates on $(G, \mathcal{F})$, controlled, in each diffusion (or network layer) t, by the differential equation:

$$X(0) = X, \quad \dot{X}(t) = -\Delta_{\mathcal{F}} X(t). \tag{10}$$

This equation is discretized using the explicit Euler scheme, which employs a unit step size:

$$X(t+1) = X(t) - \Delta_{\mathcal{F}} X(t) = (I_{nd} - \Delta_{\mathcal{F}}) X(t). \tag{11}$$

In the model proposed in [7], the discretization of the above equation is carried out as follows:

$$X(t+1) = X(t) - \sigma(\Delta_{\mathcal{F}(t)} (I_n \otimes W_1^t) X_t W_2^t). \tag{12}$$

In this case, the sheaf $\mathcal{F}(t)$ and the weights $W_1^t \in \mathbb{R}^{d \times d}$ and $W_2^t \in \mathbb{R}^{f_1 \times f_2}$ are time-dependent. This implies that the underlying geometric structure of the graph changes over time.

4 Method

We combine cellular sheaves with the **Bayesian Personalized Ranking** (**BPR**) loss to implement our novel recommender system framework. By harnessing the sheaf structure, our objective is to compute latent factors, capitalizing on the rationale inherent to the BPR loss optimization. Our methodology introduces a fresh viewpoint on personalized ranking within recommender systems by merging the robust expressiveness of the cellular sheaf framework with the established efficacy of the BPR loss.

4.1 Data

The data consists of a set of users $U = \mathbb{N}^n$, a set of items $I = \mathbb{N}^m$ and a set of reviews $R = \{(i, j, r_{i,j}) | i \in U, j \in I, r_{i,j} \in \mathbb{R}\}$. Typically, the cardinality of R is significantly smaller than its possible maximum, i.e. $|R| << n \cdot m$, indicating that it captures only a subset of all possible interactions between users and items.

Given this data, we transform it into a bipartite graph $G = (V, E)$, where $V = U \cup I$ represents the disjoint union of the user vertex set U and the item vertex set I. The set of edges E contains edges connecting users to the items they have interacted with, such that if $(i, j) \in E$, then $(i, j, r_{i,j}) \in R$.

However, applying the sheaf framework directly to bipartite graphs poses challenges, as discussed in [4]. Therefore, we adapt the sheaf Laplacian operator to fit the structure of bipartite graphs. This adaptation involves computing a projection on the bipartite graph, transforming it into a structure that is not strictly bipartite but retains the same properties. This approach preserves the connection between data points and the intrinsic properties of the sheaf, while facilitating parameter updates.

4.2 Embedding

Since nodes are initially represented only by identifiers, which lack substantial information, it becomes necessary to map them into a more meaningful latent space where they can convey richer information. This process involves defining two embedding tables, Ψ_u and Ψ_v, which contain the embeddings for all users and items, respectively. These tables are formed as unions of individual embeddings: $\Psi_u = \bigcup_{j \in U} \psi_j^u$ and $\Psi_v = \bigcup_{i \in I} \psi_i^v$. To illustrate, consider the transformation of user u_1 into its embedding ψ_1^u and item v_1 into its embedding ψ_1^v.

At first, random embeddings are generated for each user and item in the graph to initialize the embedding tables. These are trained along with the model, adapting and refining their representations as the learning process progresses.

This process is visually illustrated in Figure 1, which shows the transformation of the bipartite graph into its embedded version.

4.3 Sheaf4Rec

Our novel recommender system, Sheaf4Rec, utilizes cellular sheaves as the foundational framework. The cellular sheaf provides a structured mathematically grounded representation of the underlying problem setting, allowing to effectively encode complex and localized relationships between users, items, and preference scores.

In our architecture, we associate multidimensional vector spaces (stalks) with both nodes and edges of the interaction graph: 0-cochains (vertex stalks) naturally correspond to the embeddings of users Ψ_u, while 1-cochains (edge stalks) naturally represent the embeddings of items Ψ_v. The co-boundary map $\delta : C^0(G, \mathcal{F}) \rightarrow C^1(G, \mathcal{F})$ allows user preferences to be projected linearly onto the space of item properties. In this framework, stalks over vertices encapsulate latent preferences of individual users, while stalks over edges encode the latent semantics of items. Crucially, the restriction maps define how user preferences interact with item semantics, dynamically adapting to the local structure of the graph. This structure allows for a more expressive representation of the relationship between users, their preferences, and the items they evaluate.

This setup allows the model to construct context-specific embeddings at each layer. The input to the first sheaf layer is the embedded interaction graph initialized from user and item embedding tables. Each sheaf layer aggregates a node embedding ψ_i with the semantic information propagated from its neighbours via the boundary maps, as defined in Equation (7). The node embedding is then updated with this new aggregated representation. This process is repeated over N sheaf layer, allowing the model to propagate personalized information over multiple hops.

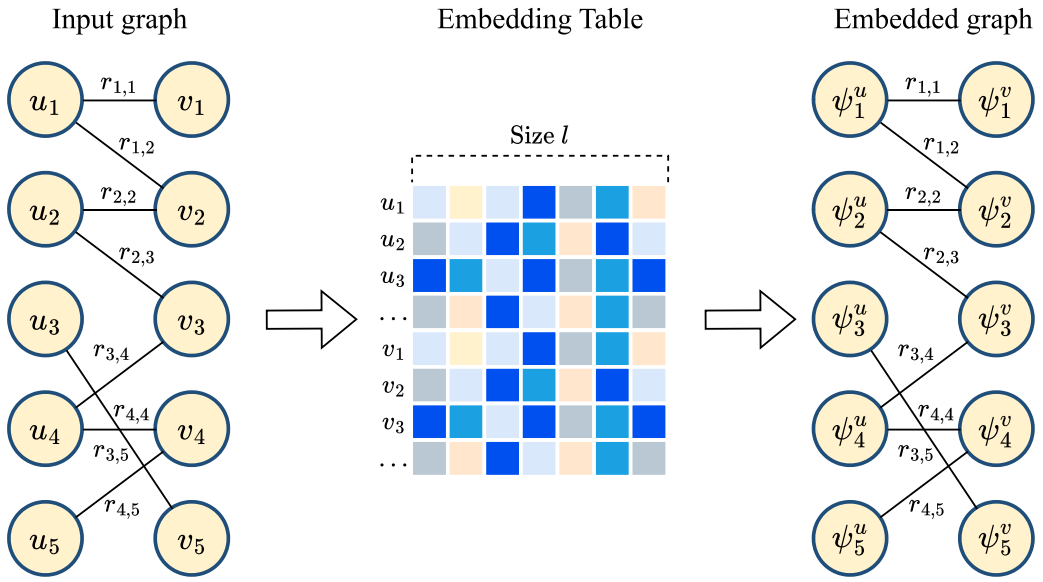

Fig. 1. The input graph comprises users and items, forming a bipartite structure, with review values on edges. Using two embedding tables, one for the users and one for the items, the graph is transformed to its embedded version.

As a result, the final user embeddings $\mathcal{F}_u$ in layer N encapsulate structural and semantic knowledge about users and items up to N-hops away in the graph. In the context of recommendations, this process increases the embedding similarity between users with shared item interests. A similar principle applies to the item embeddings $\mathcal{F}_v$, which represents the learned embeddings associated with each item v.

An inherent advantage of Sheaf4Rec lies in its seamless integration of localized embedding spaces with sheaf-based message passing. Unlike traditional models that force users and objects into a global latent space, our architecture preserves and exploits localized vector spaces and mappings, enhancing its ability to model fine-grained relational patterns. This cohesive and expressive design ensures that all pertinent information from the embeddings is stored in the corresponding vector spaces, contributing to more accurate and context-aware recommendations.

The process is shown in Figure 2.

4.4 Recommendation

The final sheaf layer outputs the restriction maps $\mathcal{F}_{u \trianglelefteq e}$ and $\mathcal{F}_{v \trianglelefteq e}$, which serve as the refined representations for users and items, respectively. For brevity, we denote these final representations without the $\trianglelefteq e$, i.e. $\mathcal{F}^u$ and $\mathcal{F}^v$, which are matrices of shapes $d_e \times d_u$ and $d_e \times d_v$. Here, d_e represents the dimensionality of the localized vector space associated with each constraint map, capturing the context-specific nuances of the interaction.

To rank items for a given user i, we compute a relevance score $s_{i,j}$ for each item j. This can be achieved by using a dot product similarity measure between the final representations of users and items. Specifically, for user i with representation $\mathcal{F}_i^u$, and item j with representation $\mathcal{F}_j^u$, the similarity score is computed as: $\langle \mathcal{F}_i^u, \mathcal{F}_j^u \cdot \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the dot product. Extending this operation to all users and items in the graph, we express the process as a matrix multiplication: $(\mathcal{F}^u)^T \mathcal{F}^v$. The final result is a score matrix S of size $d_u \times d_v$. Each entry $S_{i,j}$ in this matrix

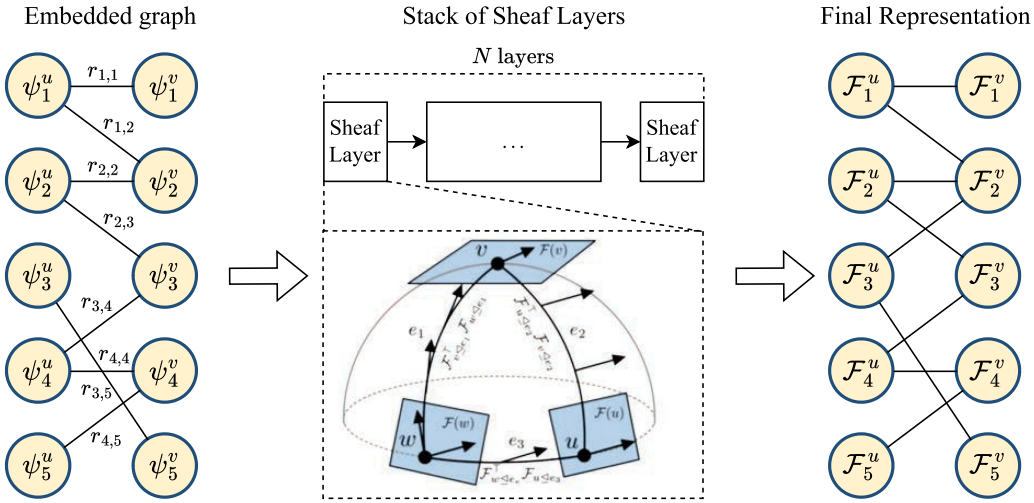

Fig. 2. The input to the sheaf layers is the embedded graph obtained from the two embedding tables. In the sheaf layers, 0-cochains are mapped to user embeddings and 1-cochains to item embeddings. These embeddings are continuously updated. To enhance the representative power of the proposed formalism, multiple sheaf layers are stacked, resulting in better recommendations. The output of the sheaf layers is still a graph, but containing the final representation.

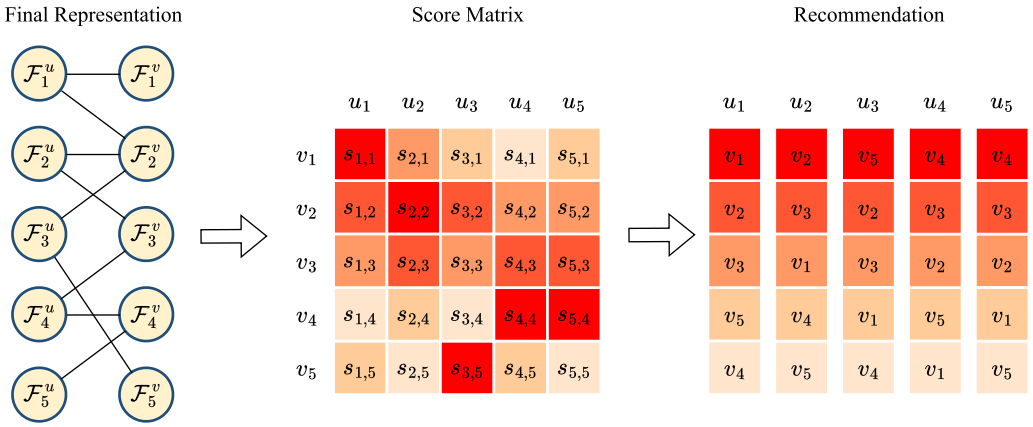

Fig. 3. The final step of the proposed pipelines consists in providing personalized recommendations to users. This involves computing a score $s_{i,j}$ for each user i and item j by performing matrix multiplication between the user embedding $\mathcal{F}^u$ and the item embedding $\mathcal{F}^v$: $(\mathcal{F}^u)^T\mathcal{F}^v$. Finally, scores are ranked to present each user with the most relevant items.

quantifies the relevance of item j to user i. These scores are then ranked to present each user i with a personalized list of the most relevant items.

This computation process is illustrated in Figure 3.

4.5 Training

The BPR loss is a natural choice for the loss function in recommender systems [40]. To understand the essence of BPR, it's important to define the concepts of positive and negative edges. Positive

edges correspond to user actions, such as purchasing or selecting an item, and are therefore present in the graph. In contrast, negative interactions are not observed, resulting in the absence of the edges in the graph.

Consequently, we define the set of positive scores $S_{pos} = \{s_{i,j}, s.t.(i,j) \in E, i \in U, j \in V\}$, which includes all scores corresponding to edges present in the graph. Conversely, the set of negative scores $S_{neg} = \{s_{i,j}, s.t.(i,j) \notin E, i \in U, j \in V\}$ comprises all scores corresponding to edges not present in E.

The BPR loss is then calculated by summing all the positive scores and negative scores, and subtracting the latter from the former:

$$BPR(S) = -ln\left(\sigma(\sum_{s_{i,j} \in S_{pos}} s_{i,j} - \sum_{s_{i,j} \in S_{neg}} s_{i,j})\right) = -ln\left(\sigma(s_{pos} - s_{neg}\right), \tag{13}$$

where S is the score matrix, s_{pos} and s_{neg} represent the two summation terms, respectively.

Due to computational complexity, we use mini-batch sampling during training to estimate the BPR loss. For each mini-batch, we sample a subset of users, and for each user, we randomly sample one positive item and one negative item. This training strategy emphasizes the probability of a user favoring a positively observed item over a randomly chosen negative counterpart.

The diffusion type used in our approach is a vanilla neural sheaf diffusion, as proposed in Bodnar et al. [7]. This means that $d = 1, W_1 = I_d$ and $W_2 = I_f$. X represents the graph features and the activation function is defined as $\sigma(x) = x$. Since the sheaf structure must be described as a learnable non-symmetric function of the data, we avoided using the similarity between nodes (items) in this step; we focused instead on the BPR loss. This formulation leverages current feature information to adjust the graph's underlying structure, consequently affecting the diffusion dynamics. The output of this learning process is a sheaf.

The training process is exemplified in Figure 4.

4.6 Challenges

Adapting Neural Sheaf Diffusion to recommendation systems introduced several unique challenges that required careful methodological adjustments.

First, traditional neural sheaf diffusion assumes asymmetric relationships between nodes [7], which is incompatible with the symmetric nature of user-item interactions in recommendation tasks. To address this, we modified the sheaf structure by introducing a BPR-based formulation that explicitly models non-symmetric relationships, ensuring that the learning dynamics remain faithful to the characteristics of recommendation data.

Second, there exists an fundamental disconnect between the intrinsic topological abstractions underlying sheaf theory and the practical demands of graph-based recommender systems. We resolved this by systematically adapting the sheaf formalism: we mapped cochains directly to user and item embeddings, and employed learnable restriction maps to encode user and item transformations. This adaptation preserves the expressive power of sheaf structures while making them applicable to the collaborative filtering context.

Third, the selection of the optimal diffusion process required careful consideration. Classical sheaf diffusion processes are tailored to specific graph properties that may not hold in recommendation datasets, such as homophily or strong local structures. We carefully designed a diffusion process suited to the heterogeneous and sparsely connected nature of recommendation graphs, balancing expressiveness and computational efficiency.

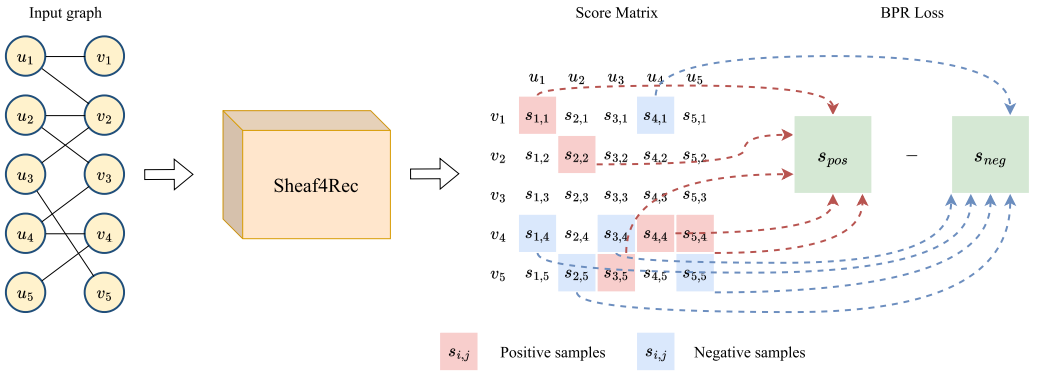

Fig. 4. During training, Sheaf4Rec takes as input the user-item graph, and generates a score matrix S that encapsulates the computed relevance of each item j for each user i. To compute the loss function, it is necessary to extract from this matrix the scores corresponding to both positive items ((those with which the user has already interacted) and negative items (those with which the user has not yet interacted). These scores are then used to compute the two terms of the BPR Loss, which is minimized during each training iteration.

5 Experiments

Our experiments on the performance of Sheaf4Rec on collaborative filtering aim to answer the following Research Questions:

— **RQ1**: How does Sheaf4Rec perform against state-of-the-art baselines?
— **RQ2**: What is the impact of hyperparameters on recommendation performance?
— **RQ3**: Does the categorical structure of sheaf architecture introduce more expressive power?

5.1 Datasets Description

We use four datasets: MovieLens 1M,[1] Yahoo! Movies,[2] Amazon Beauty, and Facebook Books.[3] These datasets were selected because they have been widely used in previous research [13, 34] and offer different characteristics suitable for evaluating our model. Their main characteristics are presented in Table 2.

Here is an overview of each dataset:

— **Facebook Books:** The Facebook Books dataset provides a relatively sparse collection of explicit feedback related to books. It contains 1,878 comments from 1,398 people about 2,933 products (books).
— **Yahoo! Movies:** The Yahoo! Movies dataset contains 69,846 explicit movie ratings on a scale of 1 to 5, gathered from the Yahoo! Movies website as of November 2003. It contains information about 2,626 items and includes 4,000 user reviews.
— **MovieLens 1M:** The MovieLens 1M dataset comprises around 1 million ratings provided by 6,040 users for approximately 3,900 movies. Each user has rated at least 20 movies on a 5-star scale with half-star increments. Demographic information such as age, gender, and occupation is also available for each user.

[1]MovieLens 1M Dataset: https://grouplens.org/datasets/movielens/1m/
[2]Yahoo! Movies Dataset: https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[3]Facebook Books Dataset: https://2015.eswc-conferences.org/important-dates/call-RecSys.html

Table 2. Number of Users, Number of Items, Number of Ratings and Density (Percentage) for Each Dataset

	Facebook Books	Yahoo! Movies	MovieLens 1M	Amazon Beauty
Users	1,398	4,000	6,040	1210,271
Items	2,933	2,626	3,900	249,274
Ratings	1,878	69,846	1000,000	2023,070
Density [%]	0.025	0.664	4.24	0.067

— **Amazon Beauty:** The Amazon datasets consist of product reviews collected from *Amazon.com* [37]. The data are organized into distinct sets based on Amazon's primary product categories. For our study, we focus on the "Beauty" category.

5.2 Baselines

We compare our model to several baselines, including both established state-of-the-art models and recently proposed neural architectures. Here is a brief summary of the baselines used in our experiments.

— **MostPop:** MostPop recommends the most popular items to each user, where popularity is defined by the number of observed interactions in the training data.
— **ItemKNN:** Item-based nearest-neighbor algorithms were discussed in [43] and later successfully applied in industry [31].
— **NGCF: Neural Graph Collaborative Filtering (NGCF)** [51] integrates user-item interactions into the embedding process, incorporating collaborative signals directly into the embedding computation.
— **LightGCN:** LightGCN [19] is a variant of the **Graph Convolutional Neural (GCN)** network [25], which includes neighborhood aggregation for collaborative filtering.
— **GAT:** We adapted the popular **Graph Attention Network (GAT)** architecture [49], which exploits attention mechanisms to capture relational dependencies, to serve as a collaborative filtering method on graph data.
— **UltraGCN:** UltraGCN [35] provides a mathematical proxy to the GCN [25] layer to mitigate the negative impact of over-smoothing in recommendation tasks.
— **KGTORe: Knowledge-Graph and Tree-Oriented Recommendation (KGTORe)** [34] architecture leverages a knowledge graph to learn latent representations for semantic features, interpreting user decisions as a personal distillation of the item feature representations.

5.3 Metrics

In our evaluation, we employ three frequently adopted metrics from previous work [34, 35] to assess the performance of recommendation systems::

— **F1-Score@K:** This metric calculates the harmonic mean of precision and recall at rank K. Precision measures the proportion of recommended items relevant to the user, while Recall quantifies the proportion of relevant items successfully recommended. The harmonic mean balances Precision and Recall, ensuring neither dominates the score disproportionately.
— **NDCG@K: Normalized Discounted Cumulative Gain (NDCG)** measures the effectiveness of a ranking system by taking into account the position of K relevant items in the ranked list. It assigns higher scores to items ranked higher in the list, reflecting their perceived importance.

Table 3. Hyperparameter Values for Our Model Architecture

	Hyperparameter	Value
l	Latent dimensions	64
N	Number of layers	5
	Learning rate	10^{-3}
	Number of epochs	100
	Batch size	1024
	Weight decay	10^{-4}

— **MRR@K:** The **Mean Reciprocal Rank** (**MRR**) calculates the average reciprocal of the rank at which the first relevant recommendation appears.

In addition, we report the time required to generate 100 recommendations as an evaluation criterion.

5.4 Reproducibility

We ensure the reproducibility of our experiments by providing access to our codebase, datasets and detailed procedures on our Github Repository.[4]

To partition the datasets, we decided not to use negative sampling strategy [22, 24] in the evaluation phase to provide more precise metrics, which is also done in other works [19, 50]. We randomly select 80% of each user's past interactions for training, reserving 10% for testing. We allocate the remaining 10% of interactions for validation in order to effectively tune the hyperparameters.

We use Bayesian hyperparameter optimisation, exploiting a search space defined by optimal parameters identified in previous studies. Six distinct hyperparameter configurations are explored. The optimal hyperparameters used to train Sheaf4Rec are outlined in Table 3.

Our experiments are performed on a workstation equipped with an Intel Core i9-10940X (14-core CPU running at 3.3 GHz), 256 GB of RAM, and a single Nvidia RTX A6000 with 48 GB of VRAM.

6 Results

To answer the research questions and evaluate the effectiveness of Sheaf4Rec, we compare its performance against recent baselines. Subsequently, we perform hyperparameter and ablation tests in order to evaluate our components.

6.1 Performance Comparisons with Baselines (RQ1)

Table 4 presents a comprehensive overview of the performance comparison between Sheaf4Rec and the competing baselines, showing their performance on the test set. We highlight the best result in **bold** and second best result with underline.

Statistical validation of the results was performed using paired Wilxocon tests with a significance level of 0.01 and Bonferroni correction [52]. Significant improvements are marked with †.

It is worth noting that models based on graph architectures, i.e., LightGCN, NGCF, and UltraGCN, regularly achieve competing performance on all four datasets. The same holds for models based on knowledge graphs, such as KGTORe, which incorporate additional side information, like

[4] https://github.com/antoniopurificato/Sheaf4Rec

Table 4. Performance Comparison of the Six Models on the Four Tested Datasets, Measured in Terms of NDCG@K (N) and F1@K (F), with $K = \{10, 20\}$

	Facebook				Yahoo				MovieLens 1M				Beauty			
	F@10	N@10	F@20	N@20	F@10	N@10	F@20	N@20	F@10	N@10	F@20	N@20	F@10	N@10	F@20	N@20
MostPop	0.008†	0.018†	0.009†	0.028†	0.043†	0.093†	0.028†	0.101†	0.055†	0.097†	0.088†	0.101†	0.053†	0.089†	0.078†	0.112†
ItemKNN	0.012†	0.022†	0.014†	0.034†	0.041†	0.088†	0.023†	0.095†	0.066†	0.100†	0.097†	0.118†	0.068†	0.103†	0.086†	0.119†
GAT	0.019†	0.036†	0.018†	0.051†	0.069†	0.119†	0.055†	0.142†	0.109†	0.143†	0.122†	0.156†	0.057†	0.097†	0.081†	0.113†
NGCF	0.027†	0.062	0.022†	0.078†	0.073†	0.129†	0.057†	0.155	0.129†	0.169†	0.141†	0.182†	0.084†	0.141†	0.095†	0.156†
LightGCN	0.017†	0.052†	0.020†	0.050†	0.047†	0.120†	0.035†	0.121†	0.127†	0.145†	0.103†	0.125†	0.089†	0.146†	0.099†	0.158†
KGTORe	0.022†	0.035†	0.019†	0.044†	0.071†	0.137†	0.052†	0.159†	0.076†	0.125†	0.089†	0.130†	0.086†	0.138†	0.098†	0.157†
UltraGCN	0.027†	0.061†	0.019†	0.066†	0.057†	0.101†	0.048†	0.117†	0.083†	0.147	0.104†	0.155†	0.091†	0.150	**0.107**	0.171
Sheaf4Rec (ours)	**0.029**	**0.066**	**0.024**	**0.081**	**0.076**	**0.147**	**0.062**	**0.162**	**0.140**	**0.182**	**0.151**	**0.197**	**0.096**	**0.154**	0.104	**0.176**
Improvement (%)	7.41	11.29	9.09	3.85	4.11	7.30	8.77	1.82	8.53	7.69	7.09	8.24	5.49	2.67	−2.88	2.92

Statistically significant differences between Sheaf4Rec and the baselines, determined by paired Wilcoxon tests ($p < 0.01$) with Bonferroni correction, are marked with †. Our approach consistently outperforms all baselines in terms of both metrics across all datasets.

Table 5. Performance Comparison of the Six Models on the Four Tested Datasets, Measured in Terms of Precision@K (P) and Recall@K (R), with $K = \{10, 20\}$

	Facebook				Yahoo				MovieLens 1M				Beauty			
	P@10	R@10	P@20	R@20	P@10	R@10	P@20	R@20	P@10	R@10	P@20	R@20	P@10	R@10	P@20	R@20
MostPop	0.010†	0.008†	**0.021**	0.005†	0.030†	0.071†	0.030†	0.026†	0.062†	0.050†	0.100	0.080†	0.050†	0.057†	0.061†	0.112†
ItemKNN	0.015†	0.011†	0.012†	0.028†	0.029†	0.081†	0.021†	0.047†	0.078†	0.056†	0.069†	0.143†	0.067†	0.069†	0.079†	0.095†
GAT	0.015†	0.096†	0.013†	**0.155**†	0.036†	0.155†	0.028†	0.225†	0.111†	0.103†	0.087†	0.153†	0.050†	0.067†	0.062†	0.125†
NGCF	0.016	0.097†	0.011†	0.143†	0.044†	0.191†	0.033	0.274†	0.129†	0.123†	0.106†	0.192†	0.070†	0.111†	0.083†	0.111†
LightGCN	0.010†	0.085†	0.013†	0.035†	0.039†	0.061†	0.028†	0.047†	0.160†	0.105†	0.085†	0.131†	0.090†	0.089†	0.115†	0.087†
KGTORe	**0.021**†	0.100†	0.011†	0.082†	0.044†	0.189†	0.031†	0.221†	0.153†	0.063†	**0.122**	0.101†	0.099†	0.039†	0.075†	0.140†
UltraGCN	0.020†	0.060†	0.011†	0.910†	0.045†	0.143†	0.019†	0.079†	**0.170**†	0.072†	0.144†	0.120†	0.100	**0.084**	0.085	**0.145**
Sheaf4Rec (ours)	0.017	**0.104**	0.013	0.147	**0.047**	**0.202**	**0.034**	**0.286**	0.141	**0.138**	0.116	**0.214**	**0.133**	0.075	**0.090**	0.122
Improvement (%)	−23.52	4.00	−61.53	−5.44	4.45	5.76	3.03	4.38	−20.57	12.19	−5.17	11.46	33.00	−12.00	5.88	−18.85

Statistically significant differences between Sheaf4Rec and the baselines, determined by paired Wilcoxon tests ($p < 0.01$) with Bonferroni correction, are marked with †. Sheaf4Rec is able to obtain a good trade-off between precision and recall, while competitors, like UltraGCN, have high precision and low recall, indicating some problems in their recommendation.

user gender or age. In terms of NDCG, Sheaf4Rec surpasses all baselines. In particular, Sheaf4Rec achieves an impressive 11.29% improvement in NDCG@10 over NGCF on the Facebook Books dataset. Similarly, Sheaf4Rec shows an 8.29% improvement in NDCG@20 on the MovieLens 1M dataset compared to NGCF, which is the second-best performing model. On the Amazon Beauty dataset, Sheaf4Rec has a 5.49% improvement compared to UltraGCN.

In terms of F1, our model consistently outperforms all the competing baselines across all the evaluated datasets. As can be seen from Table 5, the main strength of our approach is its ability to strike a good trade-off between precision and recall. Some approaches, like LightGCN, have high precision and low recall, suggesting that the system is cautious in suggesting items and it might not cover the entire breadth of relevant items available. Other approaches, such as KGTORe, show the opposite trend, emphasizing recall at the expense of precision, indicating a system that manages to include a lot of potentially relevant items in its recommendations, but also many items that do not align well with the user's preferences or needs.

We attribute the superiority of our model to several factors, including:

— Improved expressiveness: Our model, employing cellular sheaves, offers a higher degree of expressiveness, which allows it to capture complex user-item relationships more effectively.
— Better alignment with the recommendation domain: The inherent characteristics of a sheaf-based approach are likely to be better aligned with the nuances of recommendation tasks, resulting in more accurate and relevant recommendations.

In terms of MRR, our solution shows notable improvements over all baseline models, with the exception of KGTORe on the Facebook dataset. The higher MRR values indicate our model's ability

Table 6. MRR@10 and MRR@20 of Our Approach Compared with the Other Approaches on the Four Datasets

	Facebook		Yahoo		MovieLens 1M		Beauty	
	MRR@10	MRR@20	MRR@10	MRR@20	MRR@10	MRR@20	MRR@10	MRR@20
MostPop	0.019†	0.022†	0.048†	0.059†	0.029†	0.032†	0.053†	0.057†
ItemKNN	0.024†	0.029†	0.045†	0.051†	0.036†	0.038†	0.051†	0.053†
GAT	0.037†	0.032†	0.066†	0.071†	0.043†	0.038†	0.058†	0.062
NGCF	0.037†	0.049	0.072†	0.077	0.060†	0.054†	0.061†	0.064†
LightGCN	0.038†	0.041†	0.076†	0.078†	0.050†	0.052†	0.069†	0.073†
KGTORe	0.036†	**0.054**†	0.081†	0.084†	0.055†	0.044†	0.078†	0.084†
UltraGCN	0.041†	0.045†	0.070†	0.083†	0.057†	0.047†	0.073†	0.076†
Sheaf4Rec (ours)	**0.044**	0.051	**0.085**	**0.090**	**0.067**	**0.059**	**0.082**	**0.090**
Improvement (%)	7.32	−5.88	4.94	7.14	11.67	9.26	5.12	7.14

Our solution outperforms all the baselines on all the metrics, except for KGTORe on Facebook Books. The results which are statistically significant based on paired Wilcoxon tests ($p < 0.01$) with Bonferroni correction are denoted with †. On the Facebook Books dataset, Sheaf4Rec has a 7.32 % improvement with respect to UltraGCN, the second-best model.

to prioritize relevant recommendations early in the ranking, potentially boosting user satisfaction and overall experience with the recommendation system. This positive outcome can be attributed to our model's ability to learn an adjacency matrix based on node features. By establishing a robust correlation between the underlying graph structure and node attributes, our model sorts items effectively, enhancing their positioning in the recommendation list.

6.2 Ablation Studies (RQ2)

6.2.1 Values of K. In Figure 5, we analyse the performance in terms of NDCG of Sheaf4Rec alongside other state-of-the-art systems as we vary the parameter K, i.e. the number of recommendations generated. It's evident that Sheaf4Rec consistently surpasses the baselines, as its NDCG@K remains superior to all other methods across all values of K.

In particular, outperforming baselines at higher values of K, showcases our system's ability to scale effectively. As the number of items to recommend increases, our system maintains or even improves its recommendation quality, which is crucial in scenarios where extensive top-K recommendations are required.

6.2.2 Recommendation Time. Table 7 provides a detailed comparison of the recommendation time between Sheaf4Rec and other state-of-the-art methods. To ensure fairness in our evaluation, we maintained consistency by using the same number of layers for each network (2 and 5 layers) and keeping the hyperparameters constant. The computation time, measured in seconds, reveals the remarkable efficiency of Sheaf4Rec, as it consistently outperforms all other methods, showcasing the shortest computational time. Despite its inherently complex topological structure, our method remains stable and consistently achieves top-tier performance. To further highlight the stability of our solution, we present the standard deviation computed over 10 attempts. Notably, Sheaf4Rec's standard deviations are persistently lower than those of the other methods, demonstrating great consistency and stability. Strikingly, LightGCN shows significant instability, as evidenced by the higher standard deviation values in Table 7. In this table, we avoid to introduce the ItemKNN and MostPop algorithms since their approach is not GNN-based but is simpler, and their computational time can not be compared to approaches based on GNNs. As can be seen from Table 8, while Sheaf4Rec exhibits competitive or superior performance in terms of recommendation accuracy, its computational behavior reveals an interesting trade-off: slightly slower training but faster inference compared to baseline models. This behavior stems from the

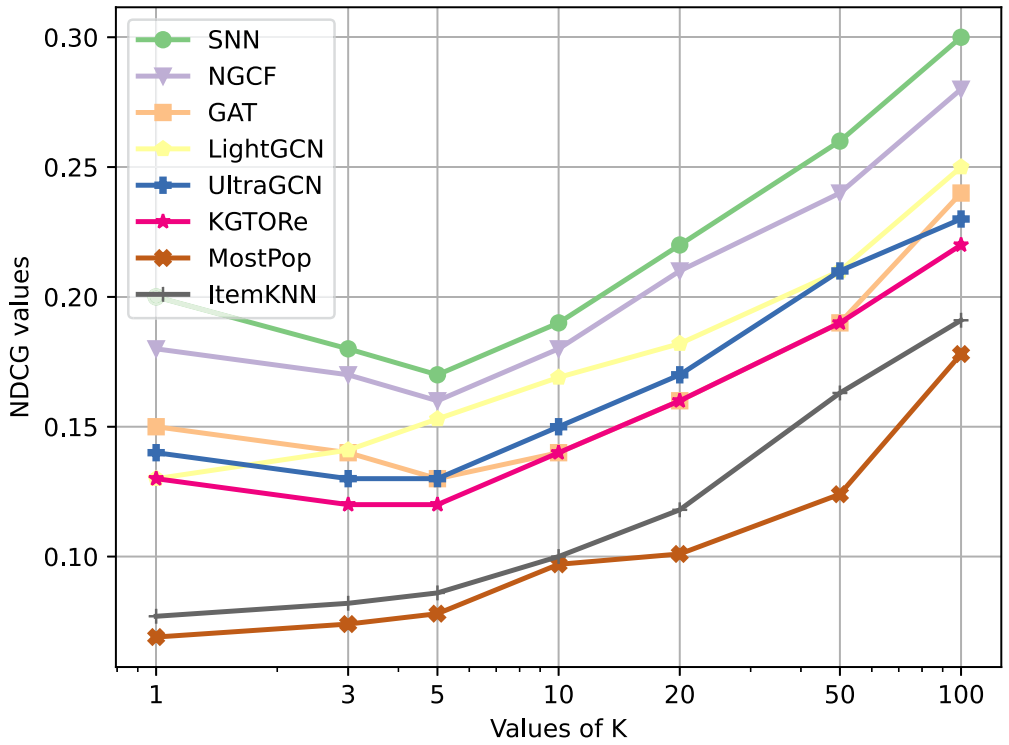

Fig. 5. NDCG@K across different values of K for the six models tested on the ML-1M dataset. Sheaf4Rec consistently outperforms other baselines, maintaining its superiority as the parameter K increases.

core architectural differences introduced by the sheaf-based parameterization. During training, Sheaf4Rec incorporates a richer set of learnable parameters associated with the local feature compatibility across edges in the graph. These parameters introduce additional degrees of freedom and require the model to optimize more complex structures, especially when enforcing consistency constraints or learning edge-specific projections. Consequently, the training phase involves more computation per batch, leading to increased training times. In contrast, the inference phase benefits significantly from the learned sheaf structures. Once training is complete, the model performs inference using pre-optimized local transformations, which do not rely on recursive neighborhood aggregation typical of message-passing GNNs. This results in reduced inference time per batch, making Sheaf4Rec particularly attractive for real-time recommendation scenarios where prediction latency is critical. Overall, this trade-off highlights the model's suitability for deployment contexts where training can be performed offline, while inference efficiency is a priority.

6.2.3 Impact of the BPR Loss. We investigate the effectiveness of different popularly used loss functions in training Sheaf4Rec, comparing the **Bayesian Personalised Ranking** (**BPR**) loss against the **Root Mean Squared Error** (**RMSE**) and the **Binary Cross-Entropy** (**BCE**) losses. We compute F1-Score@10 on the MovieLens 1M dataset, as summarised in Table 9.

Our results show that Sheaf4Rec trained with BPR loss achieves better performance than trained using RMSE or BCE. To ensure a complete analysis, we varied the number of layers in each architecture to mitigate any confounding effect of network depth on performance.

Table 7. Time (in Seconds) Required to Provide the Top 100 Recommendations to a User Compared to All the other Baselines

Model	Facebook Rec Time [s]	Yahoo Rec Time [s]	ML-1M Rec Time [s]	Beauty Rec Time [s]
GAT	0.594 ± 0.090	1.344 ± 0.077	3.796 ± 0.068	6.103 ± 0.021
NGCF	0.607 ± 0.057	1.329 ± 0.073	3.821 ± 0.060	6.783 ± 0.044
LightGCN	0.804 ± 0.349	1.434 ± 0.377	**3.633** ± 0.297	5.832 ± 0.370
KGTORe	0.763 ± 0.110	1.458 ± 0.066	4.001 ± 0.035	7.101 ± 0.009
UltraGCN	1.278 ± 0.044	1.388 ± 0.111	3.823 ± 0.076	6.543 ± 0.012
Sheaf4Rec (ours)	**0.586** ± 0.045	**1.309** ± 0.045	3.770 ± 0.051	**5.617**± 0.023
Upper Bound Deviation	−4.643%	−3.545%	−1.571%	−8.581%

Our solution outperforms all approaches on two of the four datasets. On the third dataset, ML-1M, LightGCN surpasses Sheaf4Rec, but its standard deviation is remarkably high. In **bold**, the lowest average time, underlined the one with the lowest upper limit.

Table 8. Time (in Seconds) Required to Train Each Model Across Datasets

Model	Facebook Train Time [s]	Yahoo Train Time [s]	ML-1M Train Time [s]	Beauty Train Time [s]
GAT	155.431	**556.579**	13753.223	16118.544
NGCF	157.561	592.530	16931.284	16336.093
LightGCN	**152.403**	567.441	**13333.496**	**16003.217**
KGTORe	180.337	627.313	17122.132	17254.432
UltraGCN	165.494	615.564	16001.127	16961.018
Sheaf4Rec (ours)	163.168	603.189	15974.531	16420.153
Deviation	+7.061%	+10.613%	+19.814%	+2.605%

Our model, Sheaf4Rec, is generally slower during training due to the increased complexity introduced by sheaf parameterization. However, its performance remains competitive across datasets.

Table 9. Performance in Terms of F1-Score@10 and Training Time (in Minutes) of Sheaf4Rec on ML-1M when using Different Loss Functions and Numbers of Layers

Loss	#Layers	F1@10	Time [min]
RMSE	2	0.112	283
	5	0.118	371
BCE	2	0.087	198
	5	0.101	262
BPR	2	0.123	**163**
	5	**0.140**	210

BPR Loss obtains both the highest F1@10 and the lowest training time, for both number of layers.

This investigation emphasizes the intrinsic relationship between the sheaf structure and the BPR loss, as discussed in Section 4, underscoring the importance of matching the model architecture with an appropriate loss function, ultimately optimising performance and effectiveness in real-world applications.

Additionally, the last column of Table 9 highlights that BPR has a considerable computational edge over RMSE and BCE in terms of the training time.

6.2.4 Latent Dimensions Analysis. Our architecture, as discussed in Section 4, allows stacking multiple layers to propagate new user and item features, facilitating the creation of distinct

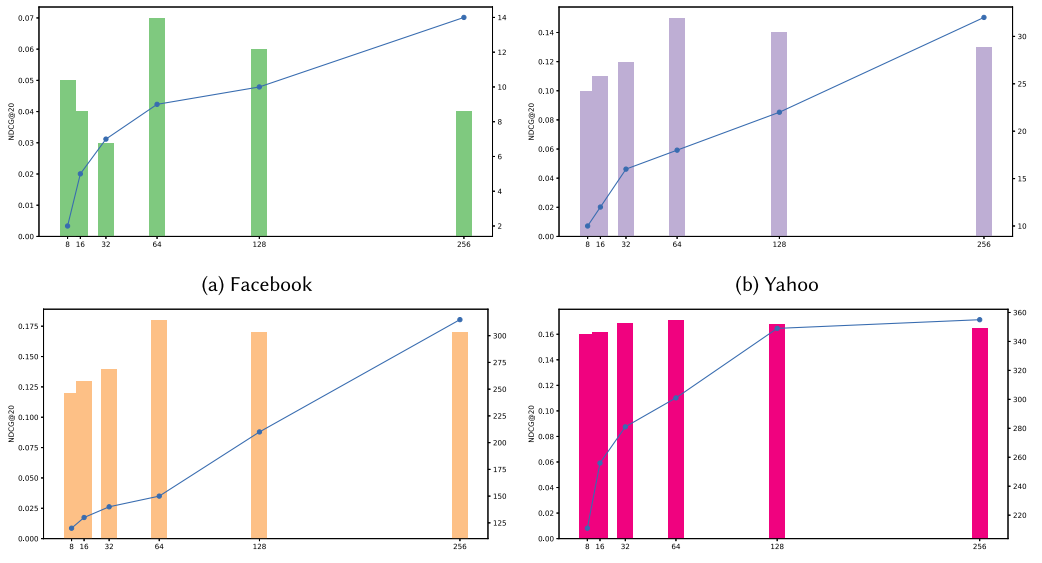

Fig. 6. NDCG@20 for Sheaf4Rec across four datasets while varying the latent dimension size. The computational time is shown on a secondary y-axis. Optimal performance is achieved when using a latent dimension of 64. As expected, as the latent dimension increases, so does the computational time.

clusters of users with similar preferences. The update of each latent feature vector as a function of its local neighbourhood is determined by a parameter called *latent dimension* l. This parameter represents the dimensionality of the latent space within each sheaf layer, thereby affecting the size of the learned representations and transformations within the network, ultimately influencing the expressiveness and capacity of the model.

We tested 5 different values for l, as depicted in Figure 6, and found that a latent dimension of 64 yields the best results across all the datasets. From the plot, we can draw several insights:

— **Optimal latent dimension l:** Increasing the latent dimension beyond a certain threshold negatively affects performance, indicating the importance of finding a suitable trade-off based on the specific task.
— **Computational considerations:** Higher latent dimension corresponds to increased computational time, especially evident for larger datasets such as MovieLens 1M. This phenomenon is due to the larger latent space, which in turn increases the size and complexity of the model, leading to longer training and inference times.

In essence, the choice of l serves as a critical design parameter, directly affecting both the performance and computational efficiency of the model.

6.3 The Advantages of the Categorical Structure (RQ3)

6.3.1 Number of Layers. One of the distinctive advantages of Sheaf4Rec lies in its capability to mitigate over-smoothing by incorporating numerous sheaf layers [7]. In contrast, several GNN architectures have shown a decrease in performance as the layer count increases [38].

To assess the impact of the number of layers on performance, we evaluated the NDCG@20 across varying layer counts, ranging from $L = 2$ to $L = 5$. As depicted in Figure 7, optimal performance tends to be achieved when utilizing $L = 5$ layers. This empirically finding underscores

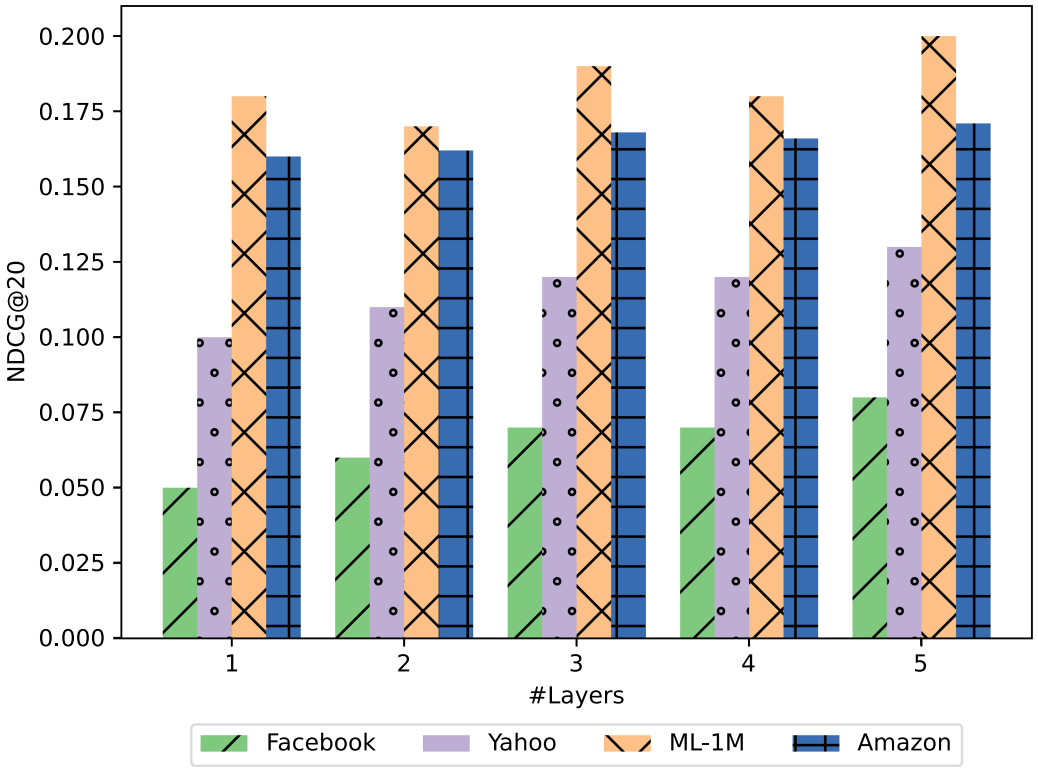

Fig. 7. NDCG@20 values of Sheaf4Rec across four datasets depending on the number of layers. For all the dataset the best results are obtained using five layers, highlighting that our model is robust to over-smoothing problems commonly encountered in GNNs.

the effectiveness of sheaf diffusion processes in mitigating oversmoothing, as further elucidated in [7].

6.3.2 Expressive Power of the Proposed Formalism. The objective of this experiment is to empirically showcase the expressive capacity of the sheaf architecture. By changing the size of node and edge stalks, we investigate four different configurations:

- **GAT equivalent:** $\dim(\mathcal{F}(v)) = 1$ and $\dim(\mathcal{F}(e)) = N$. This configuration mirrors the GAT architecture [49]. In GAT, the vector space associated to nodes is one-dimensional, allowing each node to attend every other node, ignoring structural information. Edge stalks have size N, allowing to compute the self-attention for each pair of nodes, a feature unattainable with edge stalks of size 1, which limits self-attention computation only to neighbouring nodes.
- **GCN equivalent:** $\dim(\mathcal{F}(v)) = N$ and $\dim(\mathcal{F}(e)) = 1$. This configuration reflects the GCN architecture [25]. In GCN, each node aggregates information about its neighbours by applying a graph convolution operation. In fact, a size of edge stalks of 1, we have no vector spaces associated to each edge and we use only neighbours' information.
- **Sheaf4Rec:** $\dim(\mathcal{F}(v)) = N$ and $\dim(\mathcal{F}(e)) = N$. Our proposed architecture features vector spaces of size N for both node and edge stalks.

The size of the node (or edge) stalks N represents the size of vector space that describes a certain element of the graph. If $\dim(\mathcal{F}(v)) = N$, this means that the vectors representing the

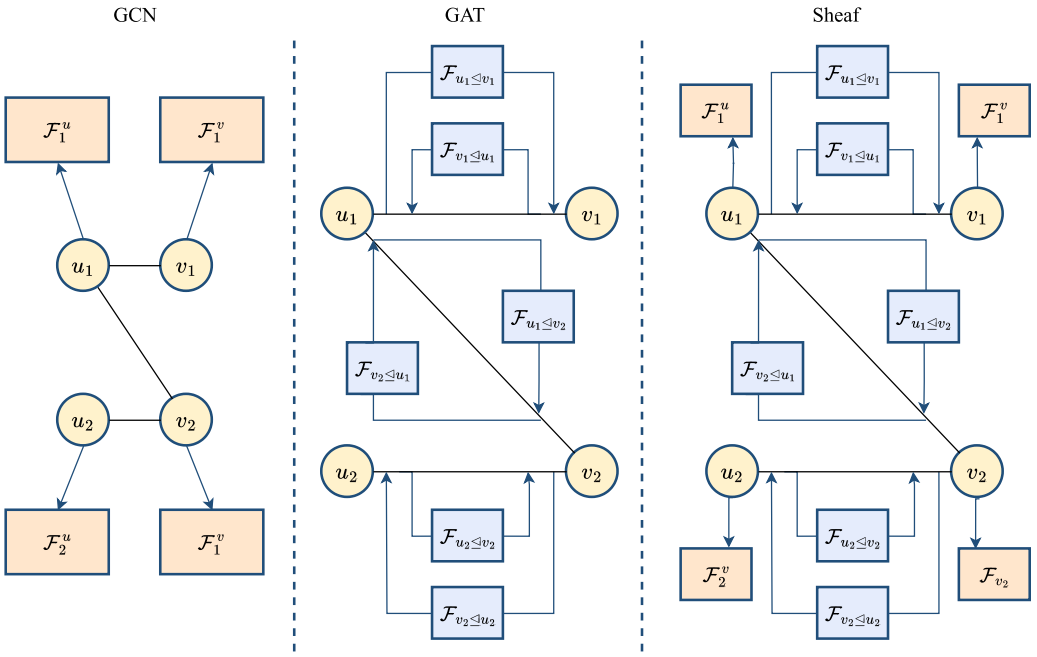

Fig. 8. GCN, GAT, and Sheaf architectures, differentiated by their node and edge stalks configurations. In the GCN architecture (left), the edge size is set to 1, while the size of node stalks is N. By setting to 1 the size of node stalks and to N the size of edge stalks, we characterise the GAT architecture (center). The Sheaf architecture (right) is obtained when both node and edge stalks sizes are equal to N.

nodes are vectors in $\mathbb{R}^N$. By fixing the size of node stalks and edge stalks to 1 we are implicitly modifying the number of parameters. To ensure comparability between architectures, we force all the architectures to have the same number of parameters. This is obtained by choosing the maximum number of parameters, dictated by our model ($d \times d$), and distributing the same amount of parameters in all the competitors. It should also be noticed that while it is possible thanks to the sheaf formalism to derive the GAT and GCN models, the opposite would not be possible. In fact Sheaf4Rec is a generalization and not a combinations of both the approaches, so by combining them it will not be possible to obtain our model.

Table 10 indicates that, even with the same exact number of parameters, our proposed solution outperforms all the competitors. This underlines the expressive power of the sheaf formalism, which allows our model to effectively capture and exploit structural information within the graph.

7 Conclusions and Future Work

Our Sheaf4Rec architecture introduces a paradigm shift by using vector spaces rather than simple vectors to represent nodes and edges within bipartite graphs of user-item interactions. Experimental results highlight Sheaf4Rec's consistent superiority over top-performing baseline models. For example, Sheaf4Rec achieves a relative improvement of 8.6% in F1@10 and 7.6% on NDCG@10 on MovieLens 1M, while on Yahoo! Movies it boosts NDCG@10 by 7.3% and F1@20 by 8.7%.

The central argument of this article lies in the potential of novel architectures grounded in Sheaf theory. These architectures are particularly suited for contexts where relationships and their representations are complex and inherently ambiguous. Take, for instance, a user's behaviour, which is influenced by the items they interact with. Traditional vector representations may lack expressive

Table 10. Performance of Sheaf4Rec in Terms of F1 and NDCG Metrics while Changing the Size of Node and Edge Stalks

$\dim(\mathcal{F}(v))$	$\dim(\mathcal{F}(e))$	F1@10	NDCG@10	F1@20	NDCG@20
1	N	0.048	0.093	0.056	0.111
1	1	0.045	0.093	0.055	0.114
N	N	**0.051**	**0.105**	**0.061**	**0.129**

The best results are obtained with the N, N setting, which represents the sheaf architecture.

power to capture the nuances of such behaviours, underscoring the need for a full vector space. This reasoning could extend to various application domains.

This work shows how Sheaf4Rec is particularly suited to the task of recommendation by accepting an embedded graph as input that incorporates modified representations of users and items. This cohesive integration ensures that all relevant information from the embeddings is preserved and stored within the corresponding vector spaces of the sheaf architecture.

Looking forward, we aim to adapt this architecture to broader research domains. Its unique topological nature implies versatility, especially when embedding side information in related stalks. A viable avenue worth exploring is the Next Point-Of-Interest recommendation system, which could leverage historical data alongside immediate inclinations to offer personalized recommendations that enhance user experiences and satisfaction.

References

[1] G. Linden, B. Smith, and J. York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[2] Uri Alon and Eran Yahav. 2021. On the bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*.

[3] Andrea Bacciu, Federico Siciliano, Nicola Tonellotto, and Fabrizio Silvestri. 2023. Integrating item relevance in training loss for sequential recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1114–1119.

[4] Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. 2022. Sheaf neural networks with connection laplacians. In *Proceedings of the Topological, Algebraic and Geometric Learning Workshops 2022*. PMLR, 28–36.

[5] Filippo Betello, Antonio Purificato, Federico Siciliano, Giovanni Trappolini, Andrea Bacciu, Nicola Tonellotto, and Fabrizio Silvestri. 2024. A reproducible analysis of sequential recommender systems. *IEEE Access* (2024).

[6] Filippo Betello, Federico Siciliano, Pushkar Mishra, and Fabrizio Silvestri. 2024. Investigating the robustness of sequential recommender systems against training data perturbations. In *Proceedings of the European Conference on Information Retrieval*. Springer, 205–220.

[7] Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Lio, and Michael Bronstein. 2022. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems* 35 (2022), 18527–18541.

[8] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. 2015. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science* 49 (2015), 136–146.

[9] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web* 5, 1, Article 2 (Feb 2011), 33 pages. DOI: https://doi.org/10.1145/1921591.1921593

[10] Ahlem Drif, Houssem Eddine Zerrad, and Hocine Cherifi. 2020. EnsVAE: Ensemble variational autoencoders for recommendations. *IEEE Access* 8 (2020), 188335–188351. DOI: https://doi.org/10.1109/ACCESS.2020.3030693

[11] Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Liò. 2023. Sheaf hypergraph networks. *Advances in Neural Information Processing Systems* 36 (2023), 12087–12099.

[12] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 2224–2232.

[13] Antonio Ferrara, Vito Walter Anelli, Alberto Carlo Maria Mancino, Tommaso Di Noia, and Eugenio Di Sciascio. 2023. KGFlex: Efficient recommendation with sparse feature factorization and knowledge graphs. *ACM Transactions on Recommender Systems* 1, 4, Article 19 (Nov 2023), 30 pages. DOI: https://doi.org/10.1145/3588901

[14] Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal* 23, 2 (1973), 298–305. Retrieved from http://eudml.org/doc/12723

[15] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1, Article 3 (Mar 2023), 51 pages. DOI: https://doi.org/10.1145/3568022

[16] Jakob Hansen and Robert Ghrist. 2019. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology* 3 (2019), 315–358.

[17] Jakob Hansen and Robert Ghrist. 2021. Opinion dynamics on discourse sheaves. *SIAM Journal on Applied Mathematics* 81, 5 (2021), 2033–2060.

[18] Wei He, Guohao Sun, Jinhu Lu, and Xiu Susie Fang. 2023. Candidate-aware graph contrastive learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR'23)*. Association for Computing Machinery, New York, NY, USA, 1670–1679. DOI: https://doi.org/10.1145/3539618.3591647

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[20] 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*. 173–182.

[21] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (Philadelphia, Pennsylvania, USA) *(CSCW'00)*. Association for Computing Machinery, New York, NY, USA, 241–250. DOI: https://doi.org/10.1145/358916.358995

[22] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv:1511.06939. Retrieved from https://arxiv.org/abs/1511.06939 (2015).

[23] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4252–4261.

[24] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM'18)*. IEEE, 197–206.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

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

[27] Oleksii Kuchaiev and Boris Ginsburg. 2017. Training deep autoencoders for collaborative filtering. arXiv:1708.01715. Retrieved from https://arxiv.org/abs/1708.01715 (2017).

[28] D. D. Lee and H. S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (Oct. 1999), 788–791.

[29] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9267–9276.

[30] 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*. 951–961.

[31] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. DOI: https://doi.org/10.1109/MIC.2003.1167344

[32] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. *Advances in Neural Information Processing Systems* 32 (2019).

[33] Saunders MacLane and Ieke Moerdijk. 2012. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer Science & Business Media.

[34] Alberto Carlo Maria Mancino, Antonio Ferrara, Salvatore Bufi, Daniele Malitesta, Tommaso Di Noia, and Eugenio Di Sciascio. 2023. KGTORe: Tailored recommendations through knowledge-aware GNN models. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys'23)*. Association for Computing Machinery, New York, NY, USA, 576–587. DOI: https://doi.org/10.1145/3604915.3608804

[35] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. UltraGCN He. 2023. Ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1253–1262.

[36] Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1506–1515. DOI: https://doi.org/10.18653/v1/D17-1159

[37] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR'15)*. Association for Computing Machinery, New York, NY, USA, 43–52. DOI: https://doi.org/10.1145/2766462.2767755

[38] Kenta Oono and Taiji Suzuki. 2019. Graph neuralnetworks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*.

[39] Antonio Purificato and Fabrizio Silvestri. 2024. Eco-aware graph neural networks for sustainable recommendations. In *Proceedings of the International Workshop on Recommender Systems for Sustainability and Social Good*. Springer, 111–122.

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. arXiv:1205.2618. Retrieved from https://arxiv.org/abs/1205.2618

[41] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 600–608.

[42] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference 2021*. 2535–2546.

[43] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW'01)*. Association for Computing Machinery, New York, NY, USA, 285–295. DOI: https://doi.org/10.1145/371920.372071

[44] Alessandro Sbandi, Federico Siciliano, and Fabrizio Silvestri. 2024. Mitigating extreme cold start in graph-based RecSys through re-ranking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4844–4851.

[45] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. DOI: https://doi.org/10.1109/TNN.2008.2005605

[46] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW'15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112.

[47] A. D. Shepard. 1985. *A Cellular Description of the Derived Category of a Stratified Space*. Brown University. Retrieved from https://books.google.it/books?id=0qbyGwAACAAJ

[48] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. arXiv:1706.02263. Retrieved from https://arxiv.org/abs/1706.02263

[49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[50] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. DOI: https://doi.org/10.1145/3269206.3271739

[51] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[52] Eric W. Weisstein. 2004. Bonferroni correction. Retrieved from https://mathworld.wolfram.com/ (2004).

[53] Douglas Brent West. 2001. *Introduction to Graph Theory*. Vol. 2. Prentice hall Upper Saddle River.

[54] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: A survey. *Computing Surveys* 55, 5 (2022), 1–37.

[55] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. DOI: https://doi.org/10.1109/TNNLS.2020.2978386

[56] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. 2017. Social collaborative filtering by trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 8 (Aug 2017), 1633–1647. DOI: https://doi.org/10.1109/TPAMI.2016.2605085

[57] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4264–4270.

[58] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system. *Computing Surveys* 52, 1 (Feb 2019), 1–38. DOI: https://doi.org/10.1145/3285029

[59] Qingqing Zhao, David B. Lindell, and Gordon Wetzstein. 2022. Learning to solve PDE-constrained inverse problems with graph networks. In *International Conference on Machine Learning*. PMLR, 26895–26910.

[60] Xun Zhou, Jing He, Guangyan Huang, and Yanchun Zhang. 2015. SVD-based incremental approaches for recommender systems. Journal of Computer and System Sciences, 4 (2015), 717–733. DOI: https://doi.org/10.1016/j.jcss.2014.11.016

[61] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. 2017. Representation learning via dual-autoencoder for recommendation. *Neural Networks* 90 (2017), 83–89.

Received 21 November 2024; revised 17 April 2025; accepted 30 May 2025