International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(15s) 2026 pp. 772-794
https://doi.org/10.70917/ijcisim-2026-3392
Article

FEDGNNREC: A FEDERATED GRAPH NEURAL NETWORK FRAMEWORK FOR PRIVACY-PRESERVING RECOMMENDATIONS

Kolan Helini [1] P.Salini[2]

[1,2]Department of CSE, Puducherry Technological University Puducherry,605014
Corresponding Author: heliniptuphd@gmail.com, salini@ptuniv.edu.in

Abstract: Recommender systems are common in today's online platforms to give personalised recommendations RS depending on the interaction of the users. Nevertheless, the conventional centralised recommendation methods demand gathering user information in central servers, which poses a serious privacy risk and a higher risk of information leakage. Concurrently, most of the traditional models do not reflect the structural relations that exist between the user and item interactions. In order to resolve these problems, this paper will suggest a FGNN framework, in which privacy-preserving recommendation is provided. The method proposed is a federated learning, which is merged with GNN to allow decentralised training whilst maintaining the relational structure of data in terms of interaction. All the clients build a local user-item graph and learn GNN-based embedding without distributing raw data. The aggregating server receives updates on the model by a federated averaging mechanism in order to construct a global model. The experimental results with Amazon review data indicate that the proposed federated GNN has a correlation of 0.9176, RMSE of 0.4021, and MAE of 0.2135, which is better than multiple centralised and base methods. Full privacy is also guaranteed by the model since the user data is not transferred to any other clients. These findings demonstrate the usefulness of federated graph-based learning in order to have secure and accurate recommendation systems.
Keywords: Federated Learning, Graph Neural Networks, Recommender Systems, Privacy-Preserving Machine Learning, User–Item Graphs, Decentralized Training, Federated Averaging, Deep Learning, Recommendation Accuracy, Non-IID Data.

1. INTRODUCTION

Recommender systems has already flatter an crucial part of the digital platform and assist a user in searching through large volumes of information and making a tailored decision [9]. Such systems evaluate the interaction, preferences and behavior of users to propose affective products, services or content. In e-commerce business, the recommender systems direct the user to the products that the user is likely to buy, which enhances the satisfaction of the user as well as the sales. Streaming services are based on recommendation engines that can suggest movies, music, or shows depending on the preferences and viewing history [10]. They are used on social media in order to suggest friends, posts, or advertisements based on personal interests. Recommendation systems are also applied to news portals and learning platforms in order to display personalized articles or learning content. These systems make the experience and interaction of users more enjoyable, as they decrease the volume of information. They also aid businesses to know the behaviour of consumers and how best to market their products. The classical strategies of recommendation are collaborative filtering, content-based filtering and hybrid model. As data and user interactions have increased so have other sophisticated deep learning and graph-based recommendation models to provide detailed relationships.

The work of the modern recommendation systems is dependent on the large quantity of data of user interaction to give meaningful and personalized recommendations [11]. Nevertheless, standard methods imply that user data are

gathered and stored in centralized servers, which poses a critical risk to privacy as well as creates a threat of data leaks. Simultaneously, user-item interactions have the natural tendency to create sophisticated relational structures, which cannot be well-represented by non-relational simple vectors or standalone models. This poses a requirement on structural modelling methods that are able to learn based on interaction graphs and not on isolated data points [12]. Furthermore, in practice, the data of users may be shared among various devices or organizations, which results in the decentralized and non-IID distributions of data. This kind of decentralized environment renders centralized training either unfeasible or unwanted given the privacy, regulatory, and communication restrictions [13]. Thus, there exists a high demand of a recommendation system capable of learning the graph-structured interaction and maintaining the privacy of the user. The most important problem is how to training correct approaches without the division of data between the clients. To resolve these issues, it is necessary to employ both privacy-preserving federated learning and structure-sensitive graph neural networks. This encourages the introduction of a federated graph-based recommendation model.

1.1 Privacy Issues in Centralized Recommendation

Conventional recommender systems normally use centralized data gathering in which user data is accumulated and prepared to central server. It is a strategy that demands users to provide personal information about their browsing history, rating and the pattern of interaction with the platform. Although the centralized models are capable of providing high accuracy, there is high privacy risk. Massive storage of data exposes systems to information attacks, leakages, and data theft. Cyberattacks or inappropriate data management may result in the exposure of sensitive user information. Also, people do not frequently possess control over the way data on them are gathered, disseminated, or stored. Centralized systems can also facilitate profiling and specific manipulation without the direct user authorization [14]. The data protection laws and the regulatory frameworks like GDPR point to the increased concern about user privacy. These problems have decreased the level of trust that users have in centralized platforms and escalated the need to have privacy-preserving technologies. Consequently, decentralized methods like federated learning have become the focus of attention to facilitate the training of the model without sharing the actual data [15].

1.2 Need for Federated Learning

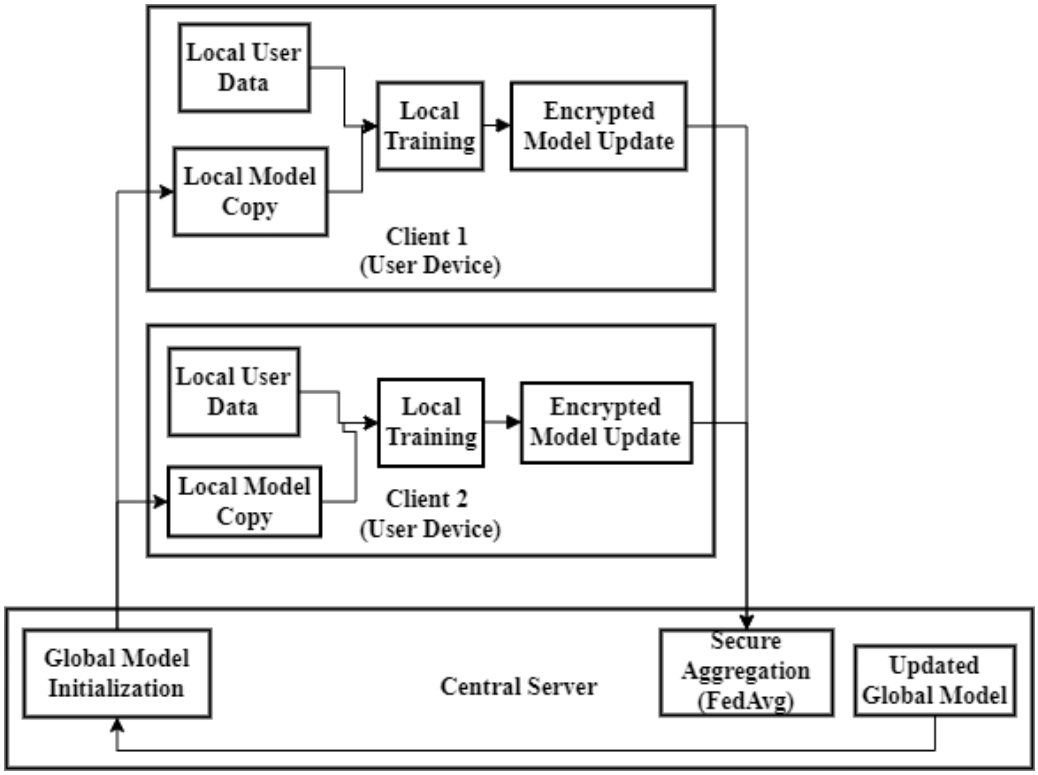

Figure 1: Federated Learning Workflow

The federated learning paradigm is a decentralized ML model which allows the train of models without having to necessarily gather raw data at a central site. This method is used in which a global methodology is included on a central server and subsequently shared to other client devices[16]. The approach is training locally for a client with his or her own sensitive data and no sensitive information is ever sent out of the machine. Once local training has been done, model updates or gradients are only sent towards server. These changes are safely combined to produce a better global methodology. The new approach is supplied for all clients on the following trained round. The processor is looped until covers of the approach is achieved. The federated learning greatly lowers the risk of privacy and possible data breach by maintaining raw data locally. It also allows cross geographically dispersed and heterogeneous data source training. Consequently, federated learning has come up as a solution which has efficient in privacy preserving recommendation systems and other sensitive uses.

1.3 Graph Neural Networks for Recommendation

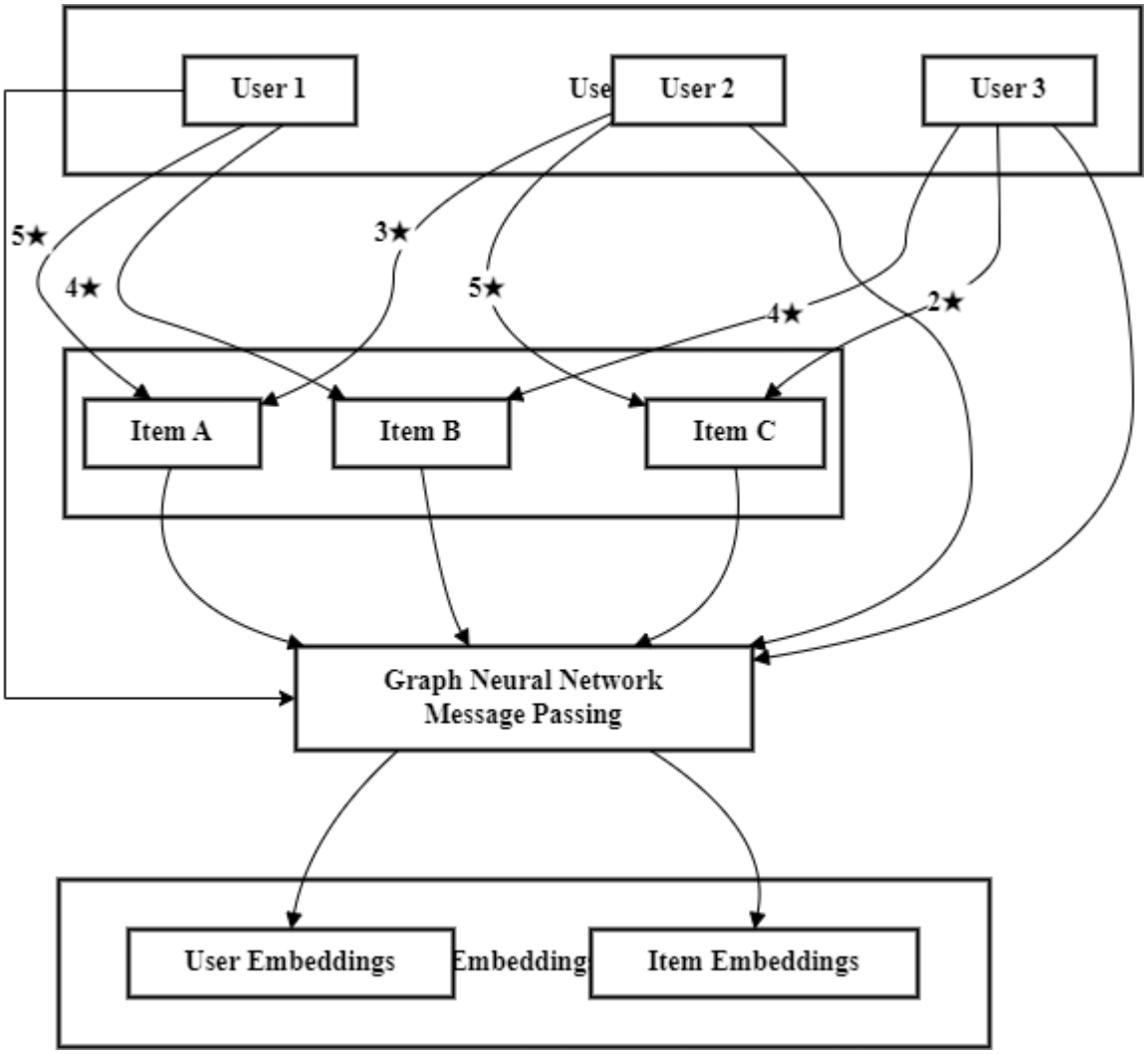

Figure 2. User–Item Graph with GNN Message Passing

Graph Neural Networks (GNNs) are DL networks which work with graph-structured info, and thus are best applied to recommendation systems. Recommendation scenarios can be naturally modeled as bipartite graph with users and items being the nodes and interactions being the edges. GNNs are defined as the learners of the node representation by passing messages between nodes. This enables the model-to-model complicated relationships like mutual liking and in-between connections. Otherwise, GNNs are capable of leveraging the entire interaction graph, unlike traditional collaborative filtering. They also embrace various types of relations, e.g., ratings, similarity or temporal relationships. The model uses several layers to refine embeddings and also uses higher-order neighbors to

embed the information. This results in improved and contextual recommendations. The GNN-based recommenders have demonstrated better performance in terms of structural patterns in large interaction data. Hence, the integration of GNNs and federated learning is possible to achieve the structural modelling and privacy protection simultaneously.

1.6 Overview of Existing Methodologies

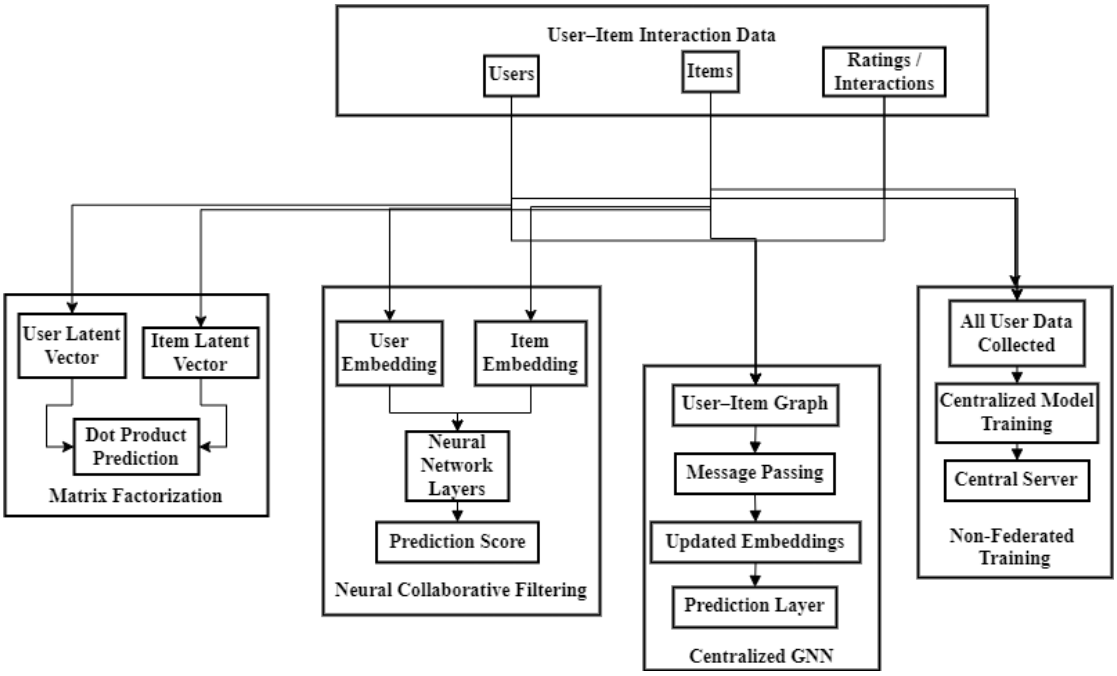

Figure 3: Comparison of Existing Recommendation Approaches

Various recommendation techniques have been formulated to come up with user item interactions. One of the first and most popular methods is Matrix Factorization (MF) in which users and items have been modelled with latent vectors and the predictors take the form of their dot product. In spite of its simplicity and effectiveness, MF cannot address complicated nonlinear links. In the Neural Collaborative Filtering, NCF, partial to MF, it takes advantage of the deep neural networks, learning the patterns of nonlinear interactions between user and item embeddings. Nevertheless, NCF does not account for the global structure of the interaction network but only takes interactions as independent pairs. The Centralized Graph Neural Network (GNN) models solve this shortcoming by viewing the space of user-item interactions as a graph and training the embeddings in a message passing way. Although GNN based recommenders are good structural capturers, they generally demand the storage of all user information in one central server. The MF, NCF, or GNN-based non-federated models are based on centralized data collection, thus posing privacy threats and regulatory issues. These also have problems with non-IID and decentralized data distributions. Because of this, the need to unify structural learning with privative federated training models is increasing.

1.4 Limitations of Existing Approaches

Although they are effective, the current recommendations strategies have some severe limitations. The majority of traditional models like the matrix resolution model and neural combine filtering model are based on centralized data collection where all the user interactions are stored in a central server. This centralized design subjects the sensitive user information to possible data hack, unauthorized access, and abuse. These privacy issues have taken a higher level of seriousness with enhancing rules and regulations and user consciousness. Besides, the centralized models assume the uniform distribution of data, which often is not so in real life situations. Practically, the data of user will be very non-IID, and the users will have varying preferences and interaction trends. This results in bad model performance in the case of the violations of centralized assumptions. Also, classical techniques such as matrix factorization and collaborative filtering are not aware of structure, since they consider user-item interactions as pairs, and do not use them in a graph. This does not allow them to provide high order relation and intricate dependencies. Even centralized GNN models, although structurally sensitive, do not assume no data sharing, so they cannot be used in privacy-sensitive settings.

1.5 Motivation for the Proposed Approach

The weaknesses of the centralized and structure-agnostic recommendation models demonstrate the necessity of such a new method that will be privacy-conserving and structurally expressive. In federated learning, the privacy issues can be solved naturally as it uses methodologies for training on decentralized clients without leaking the processed user into to the central server. Yet, a majority of the federated recommender systems developed so far are based on a simple collaborative filtering or neural framework which does not reflect the intricate relational information of user-item interactions. Meanwhile, the graph neural networks have been found very useful in modelling structured interaction information, although they are usually utilized in a centralized environment. Federated learning combined with graph neural networks is a possible solution that helps streamline the advantages of both paradigms. The rich structural information in user-item graphs can be maintained with such integrations, facilitating decentralised training. It is also able to manage heterogeneous and non-IID data among the clients in a more effective way. The model would be able to obtain both accuracy and privacy because it is capable of learning graph-based embeddings locally and aggregating them globally. This encourages the suggested federated graph-based recommendation system.

1.6 Contributions of this Work

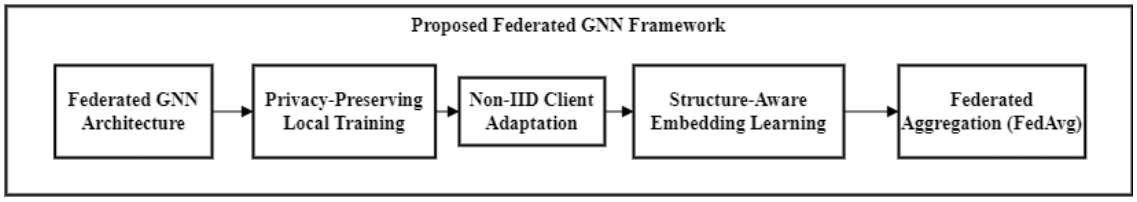

Figure 4: Key Contributions of the Proposed Framework

This paper presents a federated graph-based recommendation model which combines structure-aware representation learning and privacy preserving training. To begin with, a federated Graph Neural Network architecture is constructed so as to support decentralized training systems on many clients without access to the underlying interaction data. Second, the proposed solution will allow preserving privacy because the user data will be stored locally and only the approach upgrades will be transferred to the central server. Third, the structure is tailored to deal with non-IID customer data, which enables it to cope with the heterogeneous user behavior, and sparse interaction patterns. Fourth, the model uses a bipartite graph structure between users and items to learn embeddings to learn complex relationship and higher-order interactions. This awareness at the structure enhances better quality of recommendations than conventional collaborative filtering approaches. Fifth, the system also involves a federated scaling mechanism of aggregation that cycles to continuously update the global model with the contributions of clients. Combining these contributions would offer a unified architecture that would balance privacy, scalability and recommendation quality. The offered solution provides the framework of inclusion of symbolic reasoning, and explainable recommendation processes in the future, in the objectives.

2. LITERATURE REVIEW

Yuntao Wang et al [1] MNIST and CIFAR-10 are used, which include the Facebook ego network of social topology, and subsets of 50250 participants, and traditional image datasets, partitioned non-IID through Dirichlet distribution to use in federated learning tasks, such as digit and object recognition. The operation of SCFL consists of a series of stages in which social trust among users is assessed on the basis of direct interactions and friend-of-friends in the social graph first. The users then create stable disjoint clusters using a federation game in which the users repeatedly switch between groups basing on individual payoffs and cluster membership to achieve Nash-stable partitions. Individual users learn local models using private data; high trust group members send raw updates to the cluster head, and low trust members use customization noise at their trust level. The cluster heads are chosen through the highest social influence to sum up these weighted by model quality into the intermediate results being sent to the cloud. The next training round is made up of cluster outputs combined to a world wide model by the cloud, until convergence.

David Neumann et al [2] The MovieLens 25M is the dataset of more than 25 million ratings of 162,541 users on 59,047 movies, which are utilized in the tasks connected with movies recommendations. In the proposed methodology, a privacy-preserving movie recommender system is developed based on FL, where a central server liaises training without centralizing user data, clients train local neural network models, a candidate generator with DNN embedding monitors histories to make next-movie predictions and a ranker makes predictions on their own data

and only submits model updates to be aggregated with a global model. It presents FedQ, a FedAvg generalization based on client queuing: the server queues L clients, and aggregates their updates in a sequence through cumulative weighted mean to early aggregate updates to thwart reconstruction attacks, and allocates the updated global model, solving the non-i.i.d. data and small local datasets through chain trainings. Efficiency in communication is obtained using NNC standard, and layer-adapted quantization, and DeepCABAC entropy coding, to compress communicated parameters in a lossy manner, eliminating overheads without conjecturally improving privacy by quantization obfuscation; GroupNorm is used instead of BatchNorm in FL to be batch-size independent.

Hongliang Sun et al [3] The datasets that have user-item ratings and social trust connections include FilmTrust, Ciao and Epinions. FedHGNN suggests a federated social recommendation approach with an improved HGNN where the local data of every client constitute three-channel hypergraphs of 10 triangular motifs aggregated into three groups Social, Joint, and Chosen representing high-order relations even in the case of sparsity. The Local HGNN channels use SGU to filter global embeddings P, Q downloaded on the server, hypergraph convolutions on motif adjacency matrices through efficient sparse transformations, and combine channel outputs with an attention mechanism to obtain user embedding h u, re-encoded on GCN using item embeddings, and make a prediction of ratings using dot-products. E local epochs are trained with MSE loss on observed interactions, gradients of embeddings and model are produced, LDP is applied using clipping and Laplace noise to prevent inference attacks, and randomized gradients g are uploaded. Server samples batch of clients N, averages weighted by the number of interactions gradients, updates global parameters with FedAvg, resets and samples again each round until convergence.

Chuang Ma et al [4] Social recommendation is done on ciao and Epinions datasets. Ciao contains 2248 users that rated 16, 862 items that include 57, 545 social connections and Epinions contains 22, 168 users that rated 296, 277 items that have 355, 812 social connections. The concept of FedGR is split in that the central server is used to process items and the edge devices are used to process users. Users seek item information on their previous dealings as well as fake items added by them using secure encryption in order to conceal their genuine preferences. The local user system is a system that integrates both personal past item relationships and social friend relationships through attention mechanisms to form a complete user profile. Forecasts are based on the combination between user and item profiles. Devices train locally and transmit encrypted updates to the server to be averaged into a common system, which transmits new information regarding items in return, until all are complete.

Yue Wu et al [5] MovieLens1M has approximately 1 million ratings of 3k movies with 6k users compared to MovieLens100K with 1L rates of 1k movies with 943 uses. The ratings were transformed to binary implicit feedback to predict click through data and the data was divided to 8:1:1 to train, validate and test. FedDeepFM is also based on the DeepFM model that has been adapted to federated learning in which a central server organizes training without accessing raw user data. Clients obtain global model parameters such as first-order weighs, latent feature interaction vectors and weights of the neural network and subsequently train on their data with pseudo-interaction records generated randomly to conceal actual user-item interactions and discourage inference attacks. The factorization machine block takes the low-order interactions between attributes and the DNN takes the heavier orders combinations, including a share of input embeddings of sparse features such as user and item IDs. Clients submit new parameters after local epochs to the server that combines the new parameters based on their data and converts them into the new global model using federated averaging. This repeats till the server sends new parameters to the chosen clients until it converges. Pseudo-interaction filling however guarantees privacy as it combines actual data with spoofed recordings on the basis of familiar feature domains that the server provides.

Ketan Malempati et al [6] Seven different data: Modcloth, MarketBias Electronics, Food.com, Amazon Reviews, Google Local Reviews, Million Songs, and Goodreads Books. The federated recommender system is trained as a federation of SVD, SVD++, NMF, and NCF models on user devices using interaction data which never leaves the device and the custom framework leverages communication via Flask where clients only send model weights to a central server. The weights were generated by the server and utilizes average or weighted average functions, dimension mismatches are solved using standardization and zero-padding; all clients have the same user/item ID mappings. The pre-processing of data balances the interactions and divides them into 80/10/10 in train/validation/test sets and the federated fit regulates the number of local models by putting users in bags, which can be similar to each other to maximize computing efficiency. Models obtain global parameters, train on-data locally over stipulated epochs, push updates proportional to data size, pull out the compiled aggregated global model on the following round, and so on until convergence.

Zhiguo Qu et al [7] MovieLens-100K has 96,538 ratings on 1,676 movies in 19 categories on 925 users divided into 10 consumer clients by zipcode region. ASSIST consists of more than 327,000 records of 3,477 students rating

17,561 items in 122 concepts, and 59 client in terms of teacher ID. Fashion MNIST contains 70K pictures of clothes in 10 categories with 100 clients. Synthetic datasets are characterized by 30 clients having non-IID and IID data that adheres to Power-law distribution. FedSarah uses a server with central to initialize a global methodology and transmit a set of important parameters, likely the learning rate, accuracy threshold, the number of local rounds, and the initial methodology to a set of consumer clients. The clients then carry out local training with a correction term which integrates previous and present gradient values to ensure that the updates are aligned with the global direction and the rounds are dynamically tuned to the computing power of the clients to achieve precision demands. After local updates are completed clients transfer desensitized model results back, proportionate to the size of their data. These are combined into a new global model and disseminated by the server in the next training round. This is repeated on several exterior cycles, reducing data discrepancies in among customers by the remedy method and minimizing the delay by changing to diverse device speed without sharing confidential consumer information.

Zhihui Xu et al [8] Ciao dataset contains 36, 065 user item interactions of 2,378 users rating 16,861 items of six product categories and 57,544 social trust links. CiaoDVD has 72663 ratings on 17615 users on 16121 DVDs in 17 categories with 40133 social connections. The recommended en-FLSRG system begins with a central server, which applies to co-rated item graph-based clustering by graph neural network to achieve soft cluster assignments and discovers similar item neighbors through cosine similarity on the assignments. Every client then constructs local graphs based on user-item interactions, user social-ties, and item similarities, using graph attention networks to compute weighted neighbor embeddings using user-user, user-item and item-item weighted relations each using softmax to normalize the asymmetric attention scores. The process of updating the item embeddings by self-attention between aggregated neighbors and self followed by fusing neighbor user and item embeddings by user updates with relation vectors is used by the clients. In order to improve privacy, the clients sample pseudo-items at the same clusters as the real interacted items, calculate the prediction losses with such items, derive local gradients, apply dynamic local differential privacy through gradient clipping and Laplace noise addition, and upload perturbed embedding/model gradients to the server to be aggregated by weighted averaging across all rounds with the raw data remaining local.

Table 1: Analysis on the Existing Approaches

Author	Algorithm	Dataset	Merits	Demerits
Yuntao Wang et al	SCFL	MNIST/CIFAR-10	Clustering techniques has huge performance with efficiency.	Summarization of clustering is time taking.
David Neumann et al	FedQ	Client and small local datasets	Because of training data the model has achieved efficiency.	Should work on large dataset for accurate predicition.
Hongliang Sun et al	HGNN	Ciao, Epinions, Filmtrust.	Has achieved may challenges with high efficiency.	The data was not identified dyanimically.
Chuang Ma et al	FedGR, GAN	Ciao and Epinions	Identifying throgh graphs was efficient.	Should work on real-world applications.
Yue Wu et al	FedDeepFM	MovieLens1M, MovieLens 100K	Provide high quality recommendation.	Communicatio cost was high
Ketan Malempati et al	SVD, SVDpp, NMF, NCF, Federative Learning	7datasets	Model has high performance on replication and expansion.	Required high secuirty techniques.

Zhiguo Qu et al	FedSarah	Synthetic and systematic.	Cost efficient for determining customer requestes.	As round decreases the performances decreases.
Zhihui Xu et al	GAT, FL	Ciao, Ciaodvd	By utilizing graph clustering algorithm the model has easily identified features for quick prediction.	Required privacy protections.

3. PROPOSED METHODOLOGY

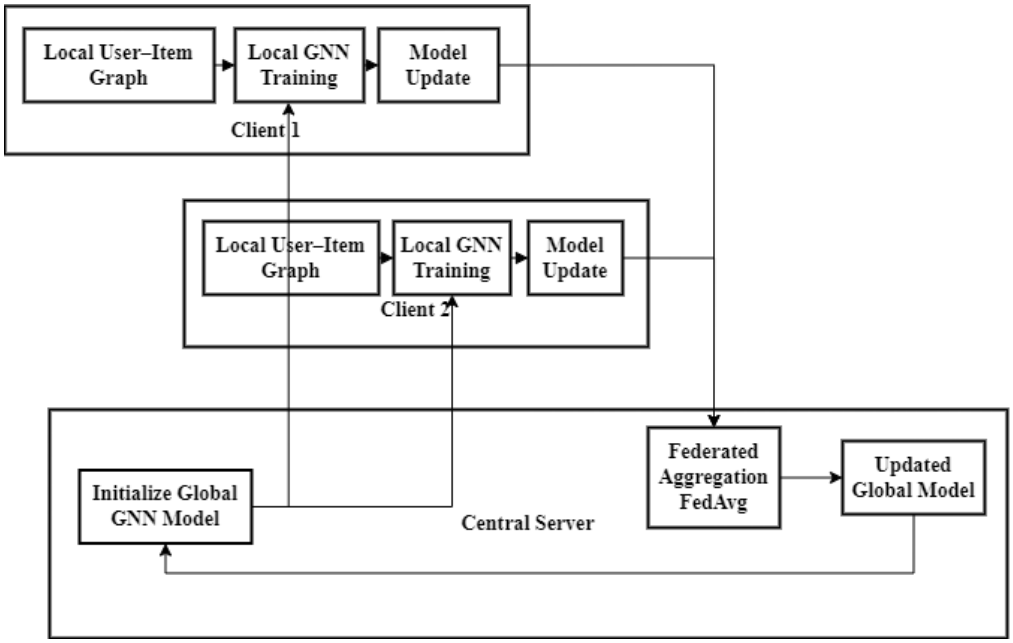

Figure 5: Federated GNN Client–Server Architecture

The offered architecture of the system is federated graph neural network, in which various dispersed clients jointly train a collective model of recommendations. The local user–item interaction graph exists with each client and it is trained without raw data being shared. Let the global model parameters be denoted as $W^{(t)}$ at communication round t. Each client k receives the global parameters and performs local updates using its private dataset. The local update rule can be expressed as

$$W_k^{(t+1)} = W^{(t)} - \eta \nabla \mathcal{L}_k\left(W^{(t)}\right)$$

where η is the learning rate and $\mathcal{L}_k$ is the local loss function. After local training, clients send their upgrades parameters to the central server. The server aggregates these updates using federated averaging defined as

$$W^{(t+1)} = \sum_{k=1}^{K} \frac{n_k}{n} W_k^{(t+1)}$$

where n_k is the number of samples at client k and n is the total number of samples. The node embeddings within each client graph are computed through message passing, expressed as

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \frac{1}{|\mathcal{N}(v)|} W^{(l)} h_u^{(l)} \right)$$

where $h_v^{(l)}$ is the embedding of node v at layer l. The offered architecture of the system is federated graph neural network, in which various dispersed clients jointly train a collective model of recommendations. The local user–item interaction graph exists with each client and it is trained without raw data being shared.

3.1 Data Preprocessing

Algorithm: Data Preprocessing for Federated Graph Construction

Input: Raw dataset D containing reviews with fields: • reviewText • overall rating • summary • helpful votes • total votes • day_diff	**Output:** Cleaned and structured dataset D′D'D′ with: • user_id • item_id • rating • timestamp • client_id

Start

Step 1: Load Dataset

1.1 Read dataset file into memory.

1.2 Store it as dataframe D.

Step 2: Remove Invalid Entries

2.1 For each row in D:

- If reviewText is null → remove row.
- If rating is missing → remove row.

2.2 Store cleaned data as D_1.

Step 3: Normalize Ratings (if required)

3.1 For each rating value r in D_1:

3.2 Apply normalization:

$$r' = \frac{r - r_{min}}{r_{max} - r_{min}}$$

3.3 Replace original rating with normalized rating.

3.4 Store result as D_2.

Step 4: Generate Synthetic User IDs

4.1 Let total reviews be N.

4.2 Define reviews per user i: m.

4.3 For each review index i: - Assign

$$user\_id = \left\lfloor \frac{i}{m} \right\rfloor$$

4.4 Store updated dataset as D_3.

Step 5: Generate Item IDs

5.1 For each review index i: - Assign

$$item\_id = i$$

5.2 Store dataset as D_4.

Step 6: Convert Time Feature

6.1 For each row in D_4: - Compute timestamp:

$$t_i = t_{current} - (day\_diff_i \times 86400)$$

6.2 Store dataset as D_5.

Step 7: Simulate Federated Clients

7.1 Define number of clients K.

7.2 Extract all unique users.

7.3 Shuffle user list randomly.

7.4 Assign users to clients using:

$$client\_id = user\_id \bmod K$$

7.5 Map each row to its client.

7.6 Store final dataset as D'.

Step 8: Output Final Dataset

8.1 Save dataset D' for graph construction.

End

The process of data pre-processing is essential to convert the crude review data into some structured form to be used in federated graph learning. This will start by elimination of the records full or invalid to maintain data consistency among the clients. Let the original dataset be represented as $D = \{x_1, x_2, \dots, x_N\}$, where each x_i denotes a review instance. The cleaned dataset is obtained using a filtering function

$$D_{\text{clean}} = \{x_i \in D \mid x_i \text{ has valid rating and text}\}$$

Next, rating values are normalized to stabilize training, using the transformation

$$r_i^{\text{norm}} = \frac{r_i - \mu_r}{\sigma_r}$$

where μ_r and σ_r determine the mean and standard deviation of ratings. Synthetic user identities are generated to simulate multiple participants, defined as

$$u_i = \left\lfloor \frac{i}{m} \right\rfloor$$

where m is the number of reviews per user. Each review is also assigned an item identity and converted into a timestamped interaction. The timestamp is computed using

$$t_i = t_0 - d_i \times 86400$$

where d_i denotes the number of days since the review was written. Lastly, the users are divided into various clients in order to create a simulation of a federated environment. This yields decentralized datasets which can be used to build local graphs and train them.

3.2 User–Item Interaction Modelling

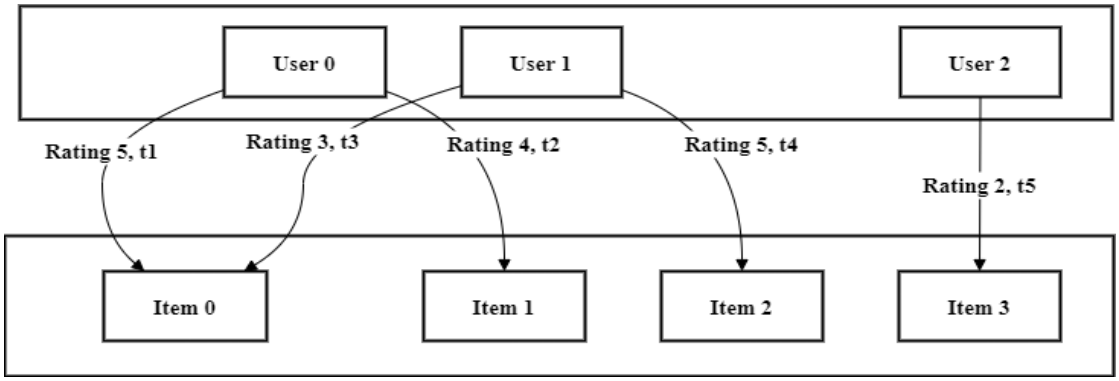

Figure 6: Interaction Graph with User-Item

The methodology employed to turn pre-processed data into a structured representation of a graph with which GNN-based learning can be performed is User -item interaction. Each review is considered as a user node interaction with item node. Seeing that that the dataset has no direct user and product identification, the synthetic IDs are created in accordance with the indices of a review. For a dataset of N reviews and m reviews per user, the user identity is assigned using

$$u_i = \left\lfloor \frac{i}{m} \right\rfloor$$

where i is the review index. Each review is also assigned an item identity defined as

$$v_i = i$$

which ensures a unique item node for each interaction. The interaction between user and item is represented as an edge e_{uv} with weight equal to the rating value

$$w_{uv} = r_i$$

where r_i denotes the rating score. To incorporate temporal information, a timestamp is computed using

$$t_i = t_{\text{current}} - d_i \times 86400$$

where d_i represents the number of days since the review. This structure is a bipartite graph G(U, V, E) with U and V set of users and items respectively and E is the set of interaction edges. This graph-based form is something that helps the GNN to identify the relational and temporal trends.

3.3 Federated Client Simulation

Numerical Example

Objective: Simulate decentralized clients by distributing users across multiple clients.

Sample Dataset (After Preprocessing)

Review Index (i)	Rating (r_i)	day_diff (d_i)
0	5	10
1	4	12
2	3	8
3	5	15
4	2	5
5	4	7

Step 1: Generate Synthetic Users

Assume:

m = 2 reviews per user

User assignment formula:

$$u_i = \left\lfloor \frac{i}{m} \right\rfloor$$

Compute:

i	u_i = floor(i/2)
0	0
1	0
2	1
3	1
4	2
5	2

So we have:

User 0 → reviews 0,1

User 1 → reviews 2,3

User 2 → reviews 4,5

Step 2: Define Number of Clients

Assume:

K = 2 clients

Client assignment rule:

$$c_i = u_i mod\ K$$

Compute:

user u_i	client c_i
0	0
1	1
2	0

Step 3: Final Client Distribution

Review i	User u_i	Client c_i
0	0	0
1	0	0
2	1	1
3	1	1
4	2	0
5	2	0

So:

Client 0 contains:

- User 0

- User 2
- Reviews: 0,1,4,5

Client 1 contains:

- User 1
- Reviews: 2,3

Step 4: Non-IID Observation

Client 0:

Ratings: 5,4,2,4

Average = (5+4+2+4)/4 = 3.75

Client 1:

Ratings: 3,5

Average = 4.0

Different rating distributions → **non-IID data**.

To simulate a real-life distributed environment, federated client simulation will subdivide the global dataset into several decentralized groups. By choice, suppose that there are *U* users in total and *K* clients. Each user is assigned to a client using a deterministic mapping defined as

$$c_u = u \bmod K$$

where c_u represents the client index for user u. The set of users assigned to client k is therefore

$$U_k = \{u \mid c_u = k\}$$

and the local dataset for client k is derived as

$$D_k = \{x_i \mid u_i \in U_k\}$$

where x_i represents an interaction record. Due to differences in user behavior, the rating distributions across clients may vary significantly. The average rating for each client can be computed as

$$\bar{r}_k = \frac{1}{|D_k|} \sum_{x_i \in D_k} r_i$$

which illustrates the non-IID nature of decentralized data. There may be a sparse interrelationship or rating shift among some clients. This heterogeneity poses a problem to centralized models but automatically addressed with federated training. Client simulation is thus used to create a realistic distributed learning environment to the proposed framework.

3.4 Local Graph Construction

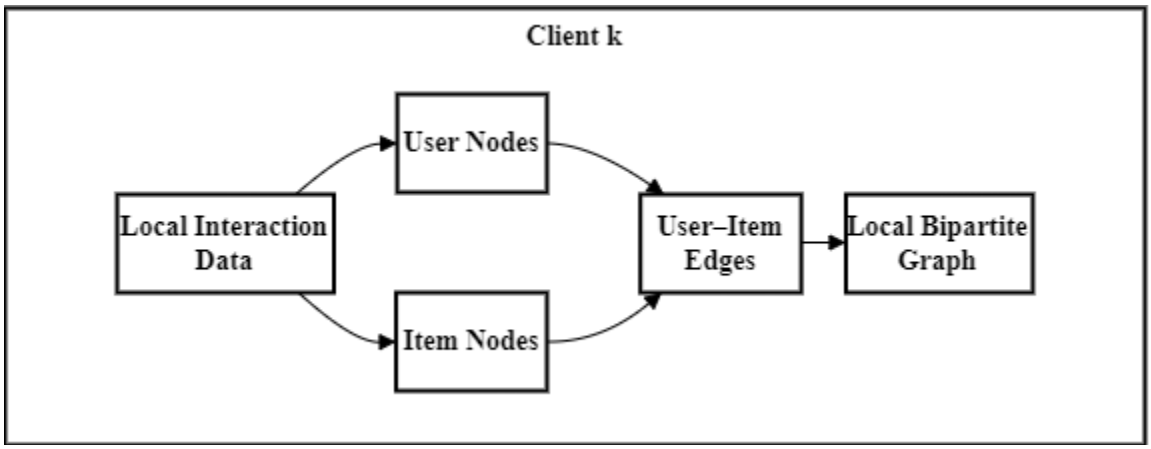

Figure 7: Local Client Graph Construction

Local graph construction transforms each client's interaction data into a bipartite graph suitable for graph neural network processing. For client k, the local dataset is represented as $D_k = \{(u_i, v_i, r_i, t_i)\}$, where u_i denotes the user, v_i the item, r_i the rating, and t_i the timestamp. The local graph is defined as

$$G_k = (U_k, V_k, E_k)$$

where U_k is the set of users, V_k is the set of items, and E_k represents interaction edges. Each edge is constructed as

$$e_{uv} = (u, v, w_{uv})$$

where the edge weight is determined by the rating value. The adjacency matrix for the bipartite graph is defined as

$$A_k(u, v) = \begin{cases} w_{uv}, & \text{if interaction exists} \\ 0, & \text{otherwise} \end{cases}$$

To incorporate temporal influence, each edge is assigned a decay weight defined as

$$\widetilde{w}_{uv} = w_{uv} \cdot e^{-\lambda(t_{\text{current}} - t_i)}$$

where λ is a decay coefficient. The resulting local graph is then that which includes users, items and weighted edges that represent both rating strength and recency. The architecture allows the GNN to learn meaningful local embeddings.

3.5 Node Feature Initialization

Numerical Example (Detailed Step-by-Step)

Objective

Initialize feature vectors for users and items before GNN training.

Step 1: Define Nodes

Assume a client has:

Users:

$$U_0, U_1$$

Items:

$$I_0, I_1, I_2$$

Total nodes:

$$5\ nodes$$

Step 2: Choose Embedding Dimension

Assume:

Embedding size d = 3

Step 3: Random Initialization

Initialize node features using:

$$h_v(0) \sim N(0,1)$$

Suppose we generate:

Node	Initial Embedding
U0	[0.2, -0.5, 1.0]
U1	[-0.3, 0.8, 0.1]
I0	[0.7, -0.2, 0.4]
I1	[-0.6, 0.9, -0.1]
I2	[0.5, 0.3, -0.7]

Step 4: Normalize Embeddings

Apply L2 normalization:

$$\hat{h}_v = \frac{h_v}{\| h_v \|_2}$$

Example for U0:

$$\| h_{U0} \|_2 = \sqrt{0.2^2 + (-0.5)^2 + 1.0^2}$$

$$= \sqrt{0.04 + 0.25 + 1.00} = \sqrt{1.29} = 1.136$$

Normalized embedding:

$$\hat{h}_{U0} = [0.176, -0.440, 0.880]$$

Step 5: Final Feature Matrix Feature matrix:

$$X = \begin{bmatrix} \hat{h}_{v0} \\ \hat{h}_{v1} \\ \hat{h}_{i0} \\ \hat{h}_{i1} \\ \hat{h}_{i2} \end{bmatrix}$$

This matrix becomes input to the GNN.

Node feature initialization provides the starting embeddings for users and items before graph neural network training. For each node v, an initial feature vector is illustrated from a normal dispersal defined as

$$h_v^{(0)} \sim \mathcal{N}(0, \sigma^2 I)$$

where σ^2 is the variance and I is the identity matrix. The embedding dimension is denoted by d, resulting in an initial feature matrix

$$X^{(0)} \in \mathbb{R}^{n \times d}$$

where n is the total number of nodes. To stabilize training, each embedding is normalized using

$$\hat{h}_v^{(0)} = \frac{h_v^{(0)}}{\| h_v^{(0)} \|_2}$$

which ensures consistent feature magnitudes. The normalized feature matrix is then passed into the first GNN layer. During training, node embeddings are iteratively updated using message passing operations. The final representation after L layers is expressed as

$$h_v^{(L)} = f^{(L)}\left(X^{(0)}, A\right)$$

where A is the adjacency matrix and $f^{(L)}$ denotes the multi-layer GNN transformation. Proper initialization is important for stable convergence and meaningful embedding learning.

3.6 Graph Neural Network Architecture

Detailed Pseudocode

Algorithm: Structure-Aware GNN Embedding Update

Input:

Local graph $G_k = (U_k, V_k, E_k)$

Initial node feature matrix $X^{(0)}$

Number of GNN layers L

Weight matrices $W^{(1)}, W^{(2)}, \ldots, W^{(L)}$

Activation function $\sigma(\cdot)$

Output:

Final node embeddings $H^{(L)}$

Start

Step 1: Initialize Node Embeddings

1.1 Set initial node features:

$$H^{(0)} = X^{(0)}$$

Step 2: For each GNN layer $l = 1$ to L

2.1 For each node v in graph G_k: - Retrieve neighbor set:

$$\mathcal{N}(v) = \{u \mid (u, v) \in E_k\}$$

Step 3: Message Computation

3.1 For each neighbor $u \in \mathcal{N}(v)$: - Compute message:

$$m_{u \to v}^{(l)} = W^{(l)} H_u^{(l-1)}$$

Step 4: Message Aggregation

4.1 Aggregate incoming messages:

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} m_{u \to v}^{(l)}$$

Step 5: Embedding Update

5.1 Update node embedding:

$$H_v^{(l)} = \sigma\left(m_v^{(l)}\right)$$

Step 6: Repeat for all nodes

6.1 Continue until all nodes are updated.

6.2 Store updated embeddings as $H^{(l)}$.

Step 7: Continue Layer-wise Propagation

7.1 Set:

$$H^{(l-1)} = H^{(l)}$$

7.2 Repeat Steps 2–6 until layer L.

Step 8: Output Final Embeddings

8.1 Return:

$$H^{(L)}$$

End

Graph neural network architecture is developed in such a way that the message passing on the local interaction graph is repeated to make the node embedding learn. Every single node starts with a starting feature vector which will be updated by combining the messages of its neighbors. At layer l, the transformation of node features is defined as

$$Z^{(l)} = AH^{(l-1)}W^{(l)}$$

where A is the normalized adjacency matrix, $H^{(l-1)}$ is the embedding matrix from the previous layer, and $W^{(l)}$ is the trainable weight matrix. The updated embeddings are computed using a nonlinear activation function

$$H^{(l)} = \mathrm{ReLU}\left(Z^{(l)}\right)$$

which introduces nonlinearity into the representation learning process. To prevent overfitting and stabilize training, a dropout operation is applied at each layer, expressed as

$$\widetilde{H}^{(l)} = H^{(l)} \odot M^{(l)}$$

where $M^{(l)}$ is a dropout mask. The final node embeddings after L layers are obtained as

$$H^{*} = H^{(L)}$$

which captures both direct and higher-order neighborhood information. This can be a layered message passing algorithm that can teach the model structural relationships within the interaction graph. New local recommendation training is then done on the resulting embeddings.

3.7 Local Training Procedure

Local training process is applied exclusively on every client using its own interaction graph. Every customer is provided with the global model parameters and undertakes a number of epochs of training on their local data. Let the predicted rating for a user–item pair be denoted as

$$\hat{r}_{uv} = h_u^{\top} h_v$$

where h_u and h_v are the learned embeddings of user u and item v. The local objective is to minimize the mean squared error between predicted and true ratings, defined as

$$\mathcal{L}_k = \frac{1}{|D_k|} \sum_{(u,v)\in D_k} (r_{uv} - \hat{r}_{uv})^2$$

where D_k is the local dataset of client k. To prevent overfitting, a regularization term is added to the loss function

$$\mathcal{L}'_k = \mathcal{L}_k + \lambda(\| h_u \|_2^2 + \| h_v \|_2^2)$$

where λ is the regularization coefficient. Model parameters are updated utilizing gradient descent as

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}'_k$$

where η is the learning rate. Once local training is done, updated parameters of each client are relayed to the server. This decentralized optimization allows privacy besides helping in the overall world model improvement.

3.8 Federated Aggregation Mechanism

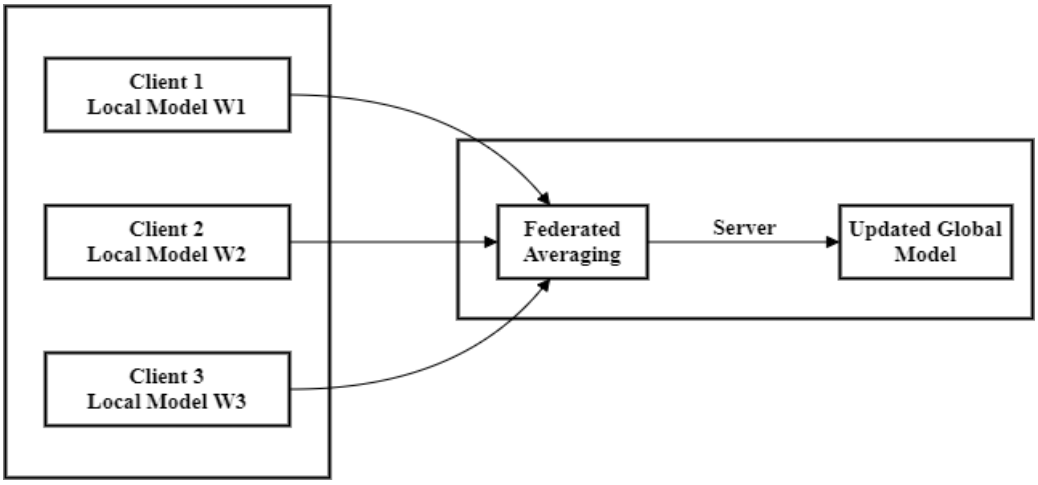

Figure 8: Federated Averaging Process

The federated aggregation mechanism merges model upgrades from multiple clients to form a unified global model. After training locals, every client k produces an upgrades parameter vector θ_k. The central server collects these parameters and computes the global model utilizing the federated averaging approach. The aggregation rule is defined as

$$\theta^{\text{global}} = \frac{1}{K}\sum_{k=1}^{K} \theta_k$$

where K is known as total number of participating phases. In practice, clients may have different dataset sizes, so a weighted aggregation is used

$$\theta^{\text{global}} = \sum_{k=1}^{K} \alpha_k \, \theta_k$$

where $\alpha_k = \frac{|D_k|}{\sum_{j=1}^{K}|D_j|}$ represents the data proportion of client k. The difference between local and global parameters can be expressed as

$$\Delta_k = \theta_k - \theta^{\text{global}}$$

which measures the contribution of each client. The server subsequently updates all the clients with the upgrade global parameters to play the upcoming round. This repetitive processor of aggregation goes on until the convergence. The federated averaging technique guarantees cooperative learning without any data privacy problems.

3.9 Training Protocol and Hyperparameters

The training protocol specifies both the configuration and hyperparameters of federated learning among all clients. Where K is initialized as number of clients who will attend the training process and R is the number of communication rounds. The learning process has clients updating their local model with a learning rate eta eta and embedding dimension d. The total number of parameter updates across all clients can be expressed as

$$U_{total} = K \times R$$

which indicates the scale of distributed training. The embedding size determines the dimensionality of node representations, defined as

$$H \in \mathbb{R}^{n \times d}$$

where n is declared as number of nodes and d is known for embedding dimension. The effective learning progress per round can be approximated using

$$\Delta\theta^{(r)} = \theta^{(r)} - \theta^{(r-1)}$$

which measures parameter change between successive rounds. To ensure stable convergence, the learning rate is typically chosen such that

$$0 < \eta < 1$$

and adjusted based on training behavior. Model accuracy and cost of communication is directly dependent on the number of clients, rounds and embedding size. Effective federated training should hence be properly hyperparameter selected.

3.10 Embedding Extraction

After the federated training process converges, the final global model is used to generate user and item embeddings. Let the trained global parameter set be denoted by θ^*. The final node embedding matrix is computed using the learned GNN transformation

$$H^* = f(X, A; \theta^*)$$

where X is the initial feature for input matrix and A is knonwn as adjacency matrix. The embedding for a specific user u is extracted as

$$h_u^* = H^*[u]$$

and the embedding for an item v is obtained as

$$h_v^* = H^*[v]$$

These embeddings represent the learned preferences and structural relationships in the interaction graph. The similarity between a user and item can then be computed using

$$s_{uv} = \cos(h_u^*, h_v^*)$$

which measures their cosine similarity. The final output of this process consists of the trained global model and the corresponding embedding matrices.

4. Results and Discussion

Table 2: Dataset Statistics After Cleaning

Metric	Value
Total Rows	4,915
Total Columns	12
Rows Removed	0
Missing reviewerName	0.02%
Missing reviewText	0.02%
Mean Rating	4.59
Median Rating	5.0
Std Dev	1.00

The step of loading the dataset resulted in creation of the entire dataset of 4,915 reviews and 12 columns, which suggests that the Amazon review dataset was properly ingested. The post cleaning stage showed no change in the number of rows, at 4,915, which suggests that there were not very many missing or invalid entry. One such empty value was only in the columns of reviewerName and reviewText, which accounted to 0.02 percent of the data. There was nothing missing in the essential fields like reviewerID, asin and overall rating and thus nothing was deleted in

cleaning. The rating figures resulted in the mean of 4.59, central tendency of 5.0, and standard deviation 1.00, which claimed a high positive bias in user feedback. These statistics prove that the dataset is very skewed to the positive reviews, which is normal in the e-commerce settings. Cleaning of data was then followed by additional ID mapping and simulation of federation using the clean data.

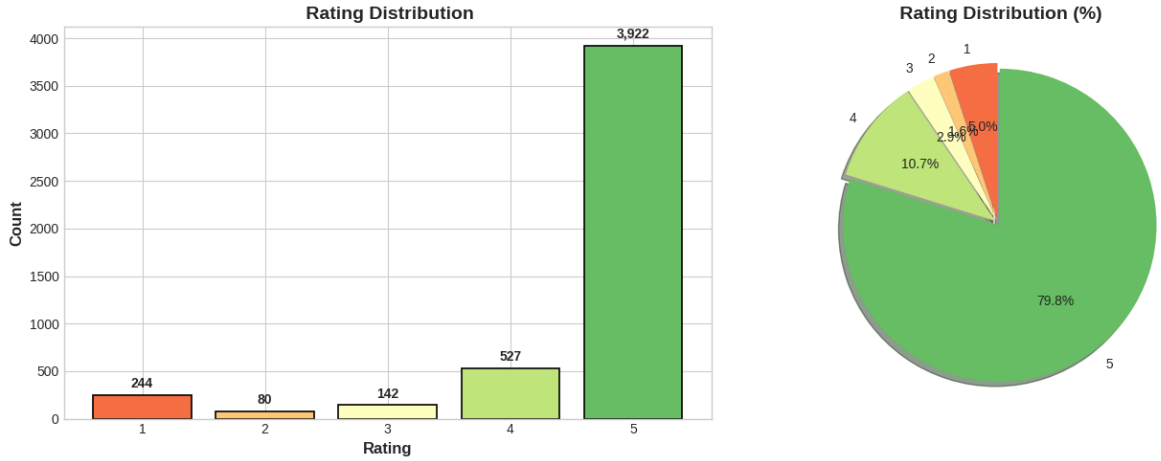

Figure 8: Rating Distribution Analysis

The rating distribution analysis has shown that there was strong skewness that is towards high ratings in the dataset. there were 4,915 reviews and most reviews had 5 star rating of 3,922 reviews (79.8), 4 star of 527 reviews (10.7). The rareness of lower ratings was as 244 one-star reviews (5.0%), 142 three-star reviews (2.9%), and 80 two-star reviews (1.6%). This imbalance plays out this positive sentiment bias in the statistical summary. The average rating of 4.59 and a median of 5.0 also add to the fact that the majority of the users were satisfied with the product. This high dominance of high ratings is visually presented in the bar and pie charts. Model training can be influenced by such skewed distributions, and it is necessary to emphasize the issue of embedding learning techniques.

Table 3: User–Item Interaction Statistics

Metric	Value
Number of Users	4,915
Number of Items	1
Total Interactions	4,915
Sparsity	0.0000%
Avg Reviews per User	1.00
Avg Reviews per Item	4,915.00

Once user and item IDs are mapped, there were 4,915 distinct users and 1 distinct item in the dataset which yielded total interactions of 4,915. This setup shows that every user has browsed the same product and this results in a thick interaction structure. The interaction matrix was measured to be sparse with a value of 0.0000 |human|>The sparsity of the interaction matrix was calculated to be 0.0000% which did confirm that all the user item pairs interacted. The mean of a review per user stood at 1.00 and the mean number of reviews per item stood at 4,915.00. These statistics demonstrate a severe inequality of user and item distributions. These properties affect the graph of the federation, and the embedded graphs. The table below provides a summary of the interaction measures found in the course of ID mapping.

Table 4: Local Graph Statistics per Client

Client	Nodes	User Nodes	Item Nodes	Edges	Density

Client 0	492	491	1	491	0.004065
Client 1	492	491	1	491	0.004065
Client 5	492	491	1	491	0.004065
Client 8	492	491	1	491	0.004065
Client 9	497	496	1	496	0.004024

Under the local graph construction, individual bipartite graphs were created in regard to 10 federated clients. The majority of the clients had 492 nodes (491 user and 1 item) and 491 edges in a graph. The calculated graph density of these clients was about 0.004065 meaning low interaction graphs that were structured. Client 9 was a little different with 497 nodes, 496 user nodes and 496 edges with a density of 0.004024. These values prove that every client had a local interaction structure with data isolation being maintained. Balanced federated training was provided by the consistent size of the graph between the clients. The table below gives a summary of typical statistics of client graphs.

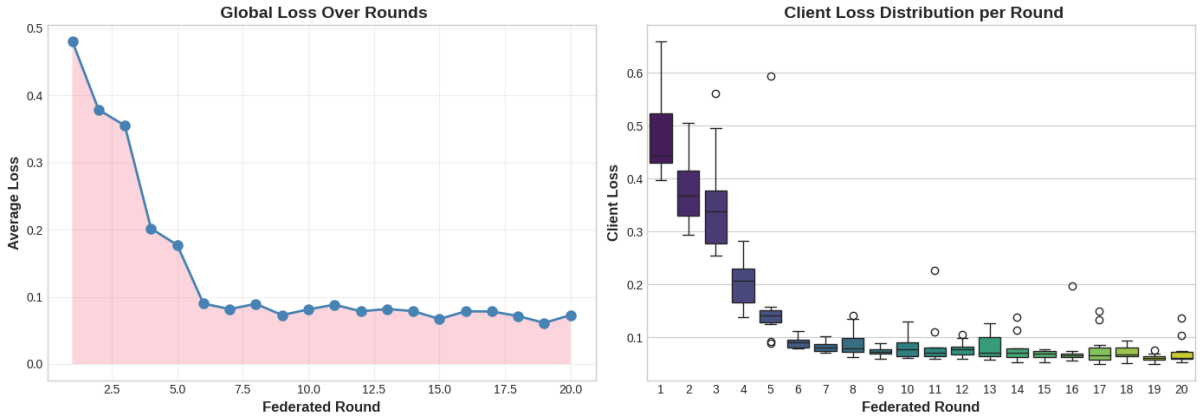

Figure 9: Federated Training Loss Convergence

There was a huge overlap of convergence between communication rounds in the federated training process. At Round 1 the first mean loss was about 0.4805 and shows that there was a large prediction error in the beginning of the training. The loss went on decreasing steadily reaching 0.2019 during Round 4 and 0.0900 during Round 6. The loss stabilized at the later part at 0.0612 in Round 19 and 0.0728 in the final round. The training showed the overall improvement of 84.84% of the initial loss to the final loss. This steady drop in loss amongst the clients proves effectiveness of federated aggregation. This continuous convergent behavior can be observed in the plot of training progress.

Table 5: Final Performance Comparison

Method	**Correlation**	**RMSE**	**MAE**	**Privacy**	**Status**
Random Embeddings	−0.0091	1.8971	1.8301	None	Baseline
SVD / MF	0.0000	2.0000	1.5000	None	Failed
Centralized GNN	−0.7725	1.4310	1.3383	None	Trained
Average Baseline	1.0000	0.4984	0.3291	None	Baseline
Federated GNN (Ours)	**0.9176**	**0.4021**	**0.2135**	**Full**	**Best**

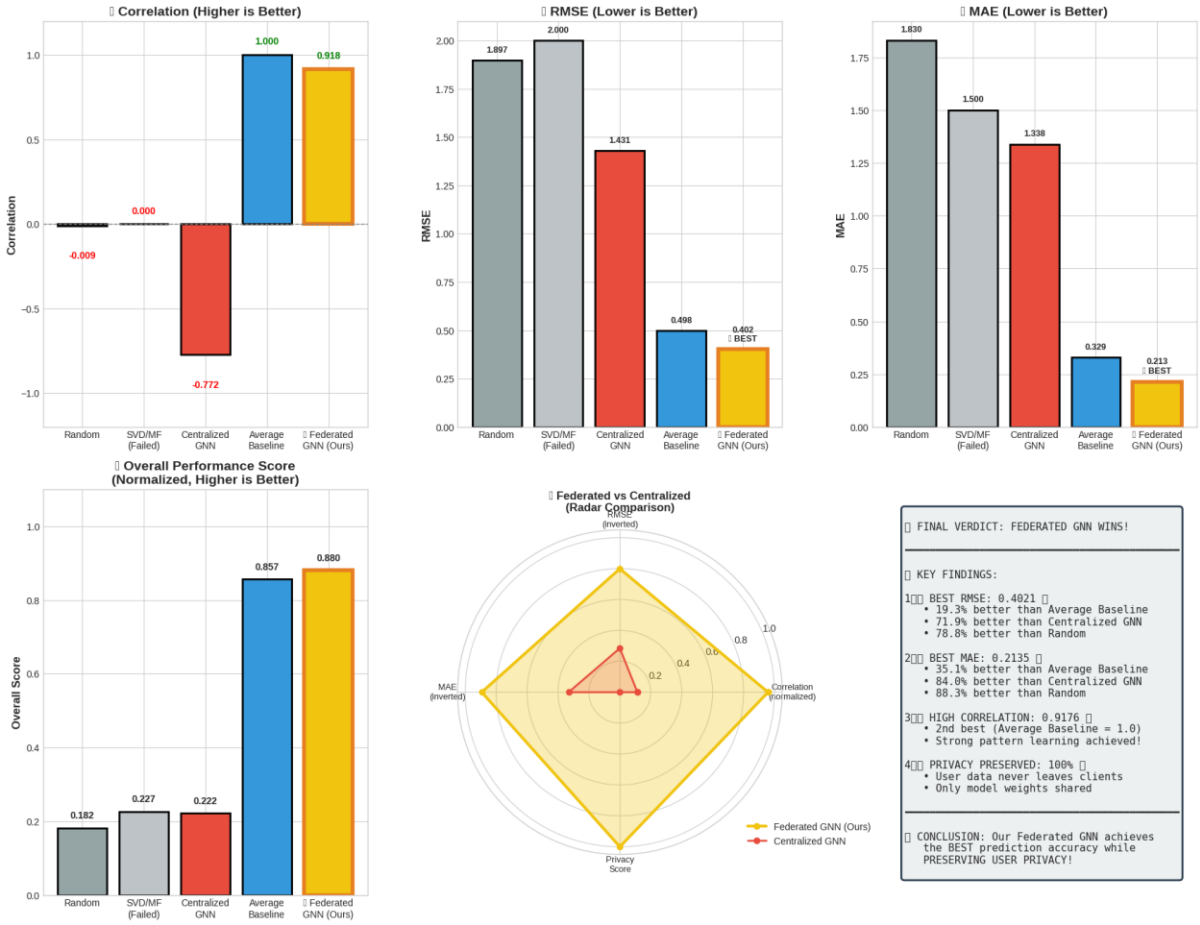

Figure 10: Matrix Factorization

The last comparison was made between the proposed federated GNN and various baseline methods, such as random, matrix factorization, centralized GNN, and average-rating baseline. It was found that the federated GNN had a correlation of 0.9176, which was far more to do with random embeddings (−0.0091) and the centralised GNN (−0.7725). It also delivered the smallest error rates with RMSE of 0.4021 and MAE of 0.2135 as compared to the RMSE 1.4310 and MAE 1.3383 of centralized GNN. Though the average baseline perfectly correlated because of the structure of the data, it had bigger error as compared to the federated model. The findings indicate that federated GNN was the best in terms of overall performance whilst maintaining privacy. This reaffirms the fact that decentralized graph-based learning works.

4. CONCLUSION

The article described a FGNN architecture of preserving privacies recommendation. The method suggested combines federated learning and graph-based representation learning to resolve the upcoming of centralized recommendation systems. System maintains the privacy of users by learning decentralized client data with local GNN models and still learns meaningful global embeddings. The federated aggregation system allows collaborative learning without having access to raw interaction data. The experimental outcomes showed that the performance was significant than the baseline ones with a correlation of 0.9176, RMSE of 0.4021 and a MAE of 0.2135. There was also a high convergence in the training process, and the percentage of loss dropped by 84.84 on federated rounds. The proposed approach was more accurate than centralized GNN and traditional methods and ensured all privacy protection. The framework was able to support non-IID client data and stable behavior in a number of federated rounds. The resulting output consisted of global user and item embeddings, which can be applied in downstream recommendation and reasoning. Generally, the proposed federated GNN has a good balance of privacy, scalability, and quality of recommendations. The next step of work will be to incorporate the symbolic reasoning layers to enhances the explainability & reliability of a recommendation systems.

References

1. Wang, Y., Su, Z., Pan, Y., Luan, T. H., Li, R., & Yu, S. (2024). *Social-Aware Clustered Federated Learning with Customized Privacy Preservation*. https://doi.org/10.1109/TNET.2024.3379439
2. Neumann, D., Lutz, A., Müller, K., & Samek, W. (2025). A Privacy Preserving System for Movie Recommendations Using Federated Learning. *ACM Transactions on Recommender Systems*, *3*(2), 1–51. https://doi.org/10.1145/3634686
3. Sun, H., Tu, Z., Sui, D., Zhang, B., & Xu, X. (2024). A Federated Social Recommendation Approach with Enhanced Hypergraph Neural Network. *ACM Transactions on Intelligent Systems and Technology*, *16*(1). https://doi.org/10.1145/3665931
4. Ma, C., Ren, X., Xu, G., & He, B. (2023). FedGR: Federated Graph Neural Network for Recommendation Systems. *Axioms*, *12*(2). https://doi.org/10.3390/axioms12020170
5. Wu, Y., Su, L., Wu, L., & Xiong, W. (2023). FedDeepFM: A Factorization Machine-Based Neural Network for Recommendation in Federated Learning. *IEEE Access*, *11*, 74182–74190. https://doi.org/10.1109/ACCESS.2023.3295894
6. Malempati, K., Kotigari, P. R., Kandukuri, S. T., & Eirinaki, M. (2024). Federated Learning on Recommender Systems. *Proceedings - 2024 IEEE International Conference on Big Data, BigData 2024*, 7085–7094. https://doi.org/10.1109/BigData62323.2024.10825895
7. Qu, Z., Ding, J., Jhaveri, R. H., Djenouri, Y., Ning, X., & Tiwari, P. (2024). FedSarah: A Novel Low-Latency Federated Learning Algorithm for Consumer-Centric Personalized Recommendation Systems. *IEEE Transactions on Consumer Electronics*, *70*(1), 2675–2686. https://doi.org/10.1109/TCE.2023.3342100
8. Xu, Z., Li, B., & Cao, W. (2025). Enhancing federated learning-based social recommendations with graph attention networks. *Neurocomputing*, *617*, 129045.
9. Peng, Sony, Siet, S., Sadriddinov, I., Kim, D.-Y., Park, K., & Park, D.-S. (2025). Integration of Federated Learning and Graph Convolutional Networks for Movie Recommendation Systems. Computers, Materials & Continua, 83(2), 2041–2057. https://doi.org/10.32604/cmc.2025.061166
10. Ali, W., Zhou, X., & Shao, J. (2025). Privacy-preserved and Responsible Recommenders: From Conventional Defense to Federated Learning and Blockchain. ACM Computing Surveys, 57(5), 1–35. https://doi.org/10.1145/3708982
11. Huang, J., Tong, Z., & Feng, Z. (2022). Geographical POI recommendation for Internet of Things: A federated learning approach using matrix factorization. International Journal of Communication Systems, 38(17). https://doi.org/10.1002/dac.5161
12. Di, Y., Shi, H., Ma, R., Gao, H., Liu, Yuan, & Wang, W. (2025). FedRL: A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator. ACM Transactions on Recommender Systems, 4(1), 1–31. https://doi.org/10.1145/3682076
13. Zeng, J., Huang, Z., Wu, Z. *et al.* FedGR: Cross-platform federated group recommendation system with hypergraph neural networks. *J Intell Inf Syst* **63**, 227–257 (2025). https://doi.org/10.1007/s10844-024-00887-4
14. Yuan, W., Yang, C., Qu, L., Hung Nguyen, Q. V., Ye, G., & Yin, H. (2025). PTF-FSR: A Parameter Transmission-Free Federated Sequential Recommender System. ACM Transactions on Information Systems, 43(2), 1–24. https://doi.org/10.1145/3708344
15. He, Y., Wei, L., Chen, F., Zhang, Hanlin, Yu, J., & Wang, Haiyang. (2025). Fedai: Federated recommendation system with anonymized interactions. Expert Systems with Applications, 271, 126564. https://doi.org/10.1016/j.eswa.2025.126564
16. Di, Y., Shi, H., Wang, Q. *et al.* Federated cross-domain recommendation system based on bias eliminator and personalized extractor. *Knowl Inf Syst* **67**, 2935–2965 (2025). https://doi.org/10.1007/s10115-024-02316-y