Generative Graph Models for Cold-Start in Recommendation Systems

GNN+XAI M2025
Dr. Amilpur Santhosh

ShreeRaj Mummidivarapu
S20220010144

Rahul Tarachand
S20220010223

Clarity of the Problem Statement

Core Issue:

- Recommender systems rely on user–item interaction graphs to suggest relevant items.
- The **cold-start problem** arises when new users or items have no historical interactions.
- Traditional recommender models struggle to predict meaningful links for these unseen nodes.

Objective:

- Build a generative graph model that, given the features of a new user or item, can generate plausible interaction edges.
- Use these generated edges to bootstrap recommendations in cold-start and sparse data scenarios.

Example:

- A new user joins movie platform $\rightarrow$ model predicts likely movie connections based on user profile and graph patterns.

Motivation for Choosing the Problem

Why this problem matters:

- Cold-start is one of the most persistent challenges in real-world recommender systems (e.g., Netflix, Amazon).
- Existing content-based and collaborative filtering methods often fail to generalize for new entities.

Why a Graph Generative Approach?

- Graph structure captures rich relational information between users and items.
- Generative Graph Models (GraphVAE, GraphRNN) can *learn the underlying distribution of edges* and generate new, realistic ones.
- Helps leverage latent patterns beyond simple feature similarity.

Expected Impact:

- Improved recommendations for new users/items.
- Enhanced adaptability in sparse or evolving datasets.

Relevance to Graph Neural Networks (GNNs)

Why GNNs?

- GNNs capture complex relational dependencies via message passing on graph structures.
- Ideal for learning embeddings of users and items that encode both feature and neighborhood information.

Relevant GNN Tasks:

- **Node classification:** Predict user preferences or item categories.
- **Link prediction:** Estimate the likelihood of a user–item interaction (core of recommendation).
- **Graph generation:** Model and sample new edges key for solving cold-start problems.

Integration in This Work:

- Combine *graph generative models* with GNN embeddings to generate realistic edges for unseen nodes.
- Evaluate generated edges for accuracy, diversity, and relevance.

Awareness of Related Works

Existing Cold-Start Approaches:

- **Content-based filtering:** Uses user/item metadata (e.g., genres, tags) but lacks relational context.
- **Collaborative filtering:** Depends on historical interactions ineffective for new users/items.
- **Hybrid methods:** Combine content and interaction data but still struggle with sparse connections.

Graph-based Advances:

- **Graph Neural Networks (GNNs):** Learn node representations through message passing.
- **Graph Autoencoders (GAE, VGAE):** Reconstruct or generate edges useful for link prediction.
- **Graph Generative Models:** (GraphRNN, GraphVAE) model graph distribution to synthesize new structures.

Gap:

- Few explore **graph generation** specifically for *cold-start recommendation*.

How the Chosen Problem Fits Into Existing Research

Connection to Prior Work:

- Builds upon **GNN-based recommenders** that model user–item graphs for link prediction.
- Extends **Graph Autoencoder frameworks** to generative modeling of new edges.
- Uses side information (knowledge graph data) to enhance feature-rich embeddings.

Novelty:

- Moves from predicting existing links to **generating new edges** for unseen entities.
- Bridges **cold-start recommendation** and **graph generation research**.

Applications:

- Personalized content recommendation for new users.
- E-commerce and streaming platforms facing rapid catalog growth.
- Any domain with evolving entity graphs (e.g., social, citation, or biomedical networks).

Identification of Appropriate Dataset(s)

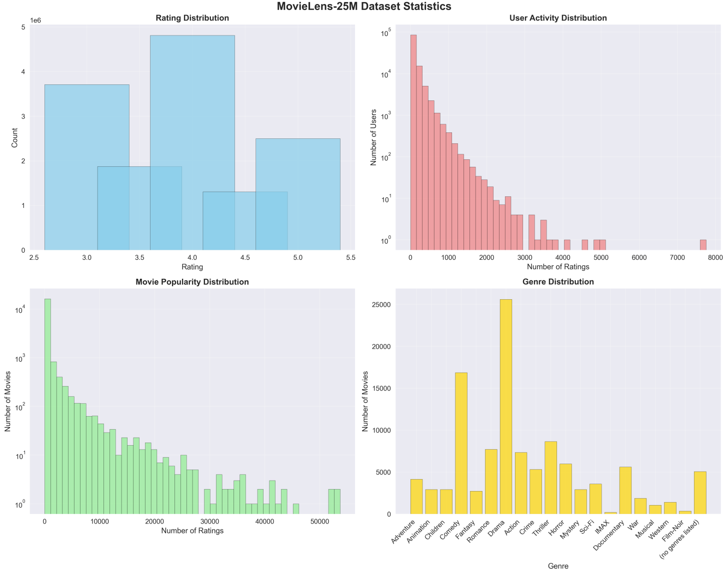
Chosen Dataset:

- **MovieLens-25M** — a large-scale, benchmark dataset for recommender systems.
- Contains over 25 million ratings across 62,000 movies by 162,000 users.
- Enriched with side information from a **knowledge graph** (e.g., DBpedia, IMDb).

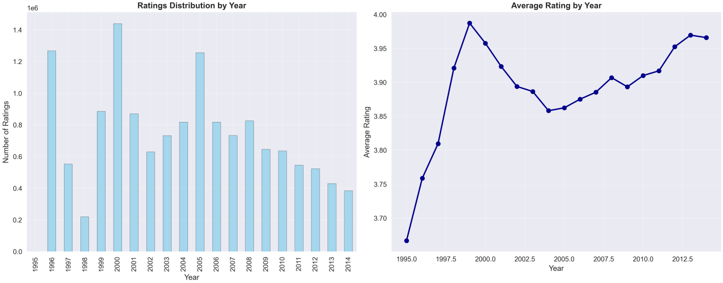
Why MovieLens-25M?

- Rich user–item interaction data ideal for graph-based modeling.
- Publicly available and well-studied in recommendation research.
- Allows for cold-start simulation by masking new users or items.

MovieLens-25M Dataset Overview



Temporal Distribution of Ratings



Key Insights:

- **Rating Volume:** Shows number of ratings per year (1995-2014)
- **Rating Patterns:** Reveals seasonal trends and popularity shifts
- **Average Ratings:** Displays how average ratings evolve over time
- **Cold-start Context:** Helps understand temporal biases for new users/items

Understanding the Graph Structure

Graph Representation:

- Model the system as a **bipartite graph**: $G = (U, I, E)$ where $U =$ users, $I =$ items (movies), and $E =$ interaction edges.

Nodes:

- **User nodes:** Encoded using demographic and behavior-based embeddings.
- **Item nodes:** Represented using content features (genre, cast, directors, etc.).

Edges:

- Each edge represents a user–movie interaction (rating or implicit feedback).
- Edge weight $=$ rating value or confidence score.

Node and Edge Features

Node Features:

- **Users:** Age group, gender, occupation, average rating behavior.
- **Movies:** Genres, keywords, director, and embeddings from external knowledge graphs.
- Optional: Pretrained embeddings (e.g., Word2Vec on movie descriptions).

Edge Features:

- Rating value (explicit feedback) or binary indicator (implicit feedback).
- Temporal component: timestamp of interaction for dynamic graph extensions.

Feature Goal:

- Enable GNNs to learn high-dimensional relational patterns between users and items.

Preprocessing Strategy & Justification for Dataset Choice

Preprocessing Steps:

1. Clean and normalize rating data (remove inactive users/items).
2. Integrate side information from knowledge graphs (entity linking).
3. Encode categorical attributes using embeddings or one-hot representations.
4. Split dataset into training, validation, and cold-start test sets.
5. Build adjacency matrices and feature tensors for GNN input.

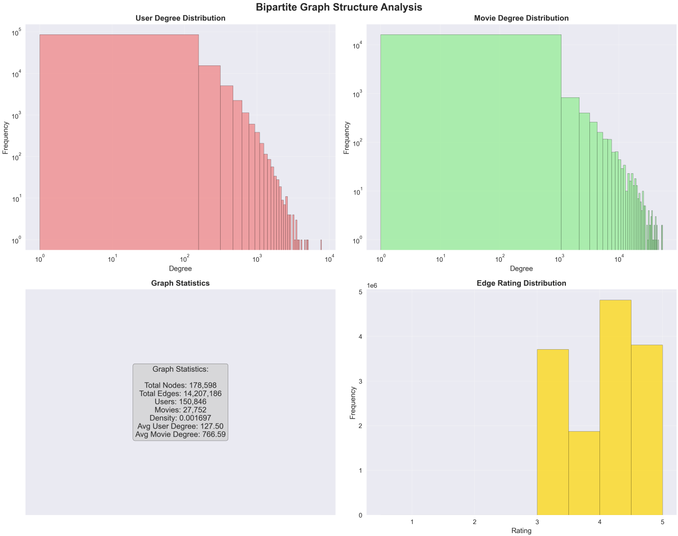
Justification for Dataset Choice:

- MovieLens-25M provides a strong balance of scale, diversity, and structure.
- Extensive prior work allows direct comparison with standard baselines.
- Easy to simulate realistic cold-start conditions for users/items.

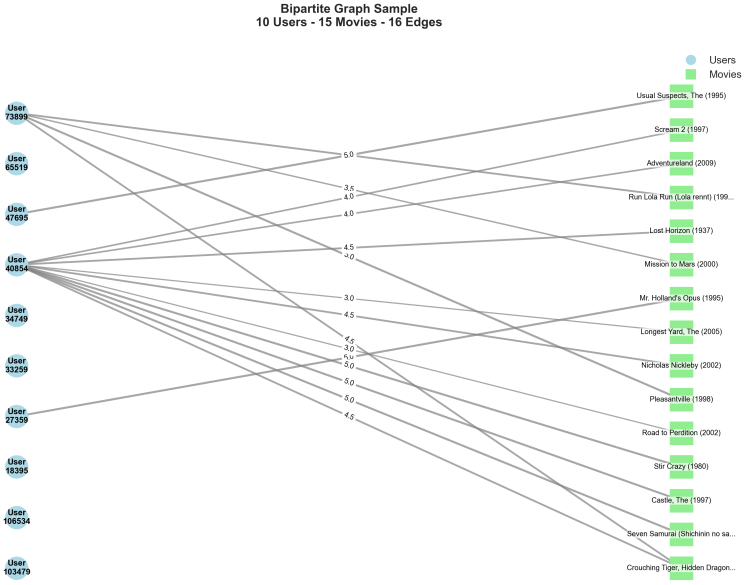
Outcome:

- A clean, feature-rich, graph-structured dataset suitable for generative GNN modeling.

Bipartite Graph Structure Analysis



Bipartite Graph Sample Visualization



Clarity in Model Design

Objective:

- Design a **Generative Graph Neural Network** that can infer missing or potential edges between users and items.
- Model combines **graph representation learning** and **edge generation**.

Model Overview:

- Input: User–item interaction graph with node features.
- Encoder: GNN-based embedding of users and items.
- Decoder: Generative component (e.g., GraphVAE or autoregressive edge generator) predicts new edges.
- Output: Probabilistic scores for potential user–item links.

Design Focus:

- Scalability to large graphs.
- Ability to generalize to unseen (cold-start) nodes.

Model Architecture Visualization

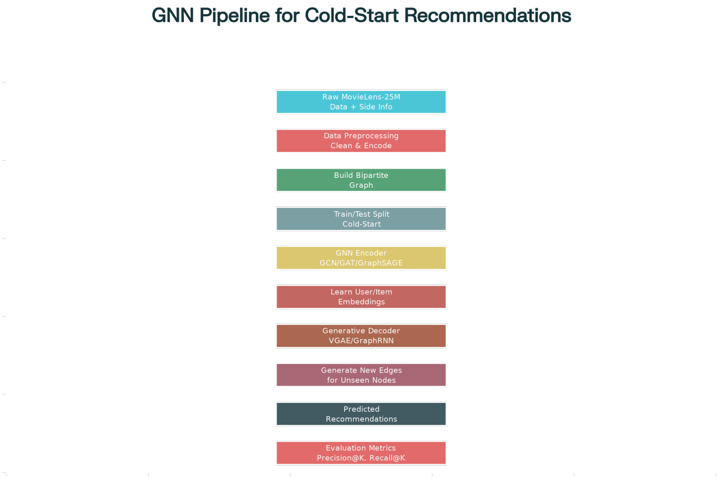


Figure: Proposed Generative Graph Neural Network Architecture

Architecture Choice

Why GNN-based Encoder?

- Captures both local and higher-order connectivity patterns.
- Learns embeddings that encode user/item relationships effectively.

Chosen Architectures:

- **Graph Convolutional Network (GCN):** Simple and efficient for initial graph embedding.
- **Graph Attention Network (GAT):** Adds attention weights for importance-based neighbor aggregation.

Decoder Design:

- **Graph Variational Autoencoder (GraphVAE):** Generates edges based on latent embeddings.
- **Autoregressive Edge Generator:** Sequentially predicts new user–item interactions.

Model Pipeline Flow

Step-by-Step Workflow:

1. **Data Preparation:** Process MovieLens-25M + side information $\rightarrow$ build bipartite graph.
2. **Graph Embedding:** Use GCN/GAT to learn user/item node embeddings.
3. **Edge Generation:** Apply GraphVAE or autoregressive model to predict probable new edges.
4. **Evaluation:** Compare generated edges with ground truth (real interactions). Metrics: Precision@K, Recall@K, Diversity, Novelty.

End-to-End Flow:

Data (Graph) $\Rightarrow$ GNN Encoder $\Rightarrow$ Generative Decoder $\Rightarrow$ Recommendations

Feasibility of the Proposed Approach

Practical Feasibility:

- GNN-based encoders (GCN, GAT) are computationally efficient and well-supported in frameworks like PyTorch Geometric.
- GraphVAE architectures can be implemented for edge generation using existing libraries.

Scalability:

- Mini-batch and sampling techniques (GraphSAGE-style) can handle large graphs.
- Parallel training feasible on GPU environments.

Expected Outcomes:

- Improved recommendation accuracy for new users/items.
- Enhanced diversity and relevance of generated links.
- Demonstrates viability of **graph generation** for real-world recommender systems.