Comparative Analysis of Recommendation System Algorithms

Jiepei Chen
City University of HongKong (Dongguan)
72405876@cityu-dg.edu.cn

Hao Yang
City University of HongKong (Dongguan)
72403219@cityu-dg.edu.cn

Lu Peng
City University of HongKong (Dongguan)
72401459@cityu-dg.edu.cn

Xuanhao Yan
City University of HongKong (Dongguan)
72405380@cityu-dg.edu.cn

Zexuan He
City University of HongKong (Dongguan)
72403000@cityu-dg.edu.cn

ABSTRACT

This study presents a systematic evaluation of diverse recommendation algorithms across three benchmark datasets: MovieLens, Last.FM, and Yelp. We investigate traditional collaborative filtering approaches (user-based, item-based), matrix factorization techniques (SVD), and advanced graph neural network methods (LightGCN, NGCF). Additionally, we also employ content-based methods for the MovieLens dataset and friend-based approaches for the Last.FM dataset, tailored to the distinct characteristics of each dataset. Through rigorous empirical comparison using ranking-based metrics (HR@K, NDCG@K, MRR@K), we identify the strengths and limitations of each approach under varying data conditions, including sparsity levels and cold-start scenarios. Our findings reveal that while GNN-based methods generally outperform traditional approaches on denser datasets, collaborative filtering techniques maintain competitive performance in specific contexts, particularly with sparse interaction data. This comparative analysis provides practical insights for algorithm selection based on dataset characteristics and application requirements.

Source code for implementation is available at https://github.com/jasper7c/CS5481-RecommenderSystem.

1 INTRODUCTION

Personalized recommendation systems are essential tools to handle information overload, facilitating content filtering based on individual user interests and preferences. These systems significantly enhance user experience by predicting items of potential interest [1]. With the explosive growth of online information, recommender systems have become indispensable components in various domains including e-commerce, entertainment, social media, and information retrieval [2].

The field of recommender systems has evolved through several phases. Initially, content-based filtering methods analyzed item features to recommend similar items to users [3]. However, these approaches struggled with the "overspecialization" problem, limiting discovery of diverse items. Traditionally, collaborative filtering (CF) methods, particularly user-based and item-based algorithms [4], have dominated applied settings due to simplicity and interpretability. Yet, these methods face challenges with data sparsity, cold-start problems, and scalability issues [5].

Matrix factorization techniques such as Singular Value Decomposition (SVD) efficiently capture latent user and item characteristics through dimensionality reduction, thus addressing data sparsity challenges [6]. The Netflix Prize competition significantly advanced these methods, demonstrating their effectiveness in real-world recommendation tasks [7].

Recently, Graph Neural Networks (GNNs) have emerged, capturing complex user-item interactions via graph-structured data representations. Methods like LightGCN [8] and NGCF [9] overcome limitations inherent in traditional models by effectively leveraging neighborhood structural information. These approaches excel particularly in capturing higher-order connectivity patterns that are difficult to model with conventional matrix-based methods [10].

For our empirical analysis, we selected three widely-used benchmark datasets with distinct characteristics:

- **MovieLens (1M)**: Contains 1,000,209 ratings from 6,040 users on approximately 3,900 movies, representing a medium-density entertainment domain [11].
- **Last.FM**: Comprises music listening preferences of 1,892 users across 92,800 artists, characterized by high sparsity and diverse interaction patterns [12].
- **Yelp**: A large-scale dataset containing 6,990,280 reviews across 11 metropolitan areas for 150,346 businesses, featuring rich contextual information and geographic diversity [13].

This study makes several contributions:

- A comprehensive cross-method evaluation using consistent preprocessing and evaluation protocols, enabling direct performance comparisons across algorithmic approaches
- Analysis of algorithm performance under varying data conditions, including different sparsity levels and interaction patterns
- Practical guidelines for algorithm selection based on dataset characteristics and application requirements
- Reproducible implementation of recommendation algorithms with optimized hyperparameter configurations

2 RELATED WORK

2.1 Collaborative Filtering

CF approaches generally rely on computing similarity between users or items to generate recommendations. User-based CF measures user-user similarity to predict unknown user preferences. Item-based CF, conversely, calculates item-item similarity for recommendation [4].

The prediction of rating r_{ui} for user u and item i in user-based CF typically follows:

$$r_{ui} = \frac{\sum_{v \in N(u)} sim(u,v) \cdot r_{vi}}{\sum_{v \in N(u)} |sim(u,v)|} \tag{1}$$

where $N(u)$ denotes the neighborhood of user u, and $sim(u,v)$ refers to similarity between users.

Similarly, item-based CF predicts ratings as:

$$r_{ui} = \frac{\sum_{j \in N(i)} sim(i,j) \cdot r_{uj}}{\sum_{j \in N(i)} |sim(i,j)|} \tag{2}$$

where $N(i)$ represents similar items to item i.

Common similarity metrics include Pearson correlation coefficient:

$$sim_{pearson}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{3}$$

and cosine similarity:

$$sim_{cosine}(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}} \tag{4}$$

where I_{uv} represents items rated by both users u and v.

2.2 Matrix Factorization: SVD

Matrix factorization methods model user-item interactions into latent factor spaces. Singular Value Decomposition (SVD) factorizes the interaction matrix $R \in \mathbb{R}^{m \times n}$ into the product of three matrices:

$$R = U \Sigma V^T \tag{5}$$

where $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$ are orthogonal matrices, capturing latent user and item features respectively, and $\Sigma \in \mathbb{R}^{k \times k}$ is a diagonal matrix of singular values.

In practice, a regularized version known as Regularized SVD or probabilistic matrix factorization is often used, modeled as:

$$min_{p,q} \frac{1}{2} \sum_{(u,i) \in \Omega} (r_{ui} - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{6}$$

where p_u and q_i are latent vectors for user u and item i, Ω denotes observed ratings, and λ controls regularization.

2.3 Graph Neural Networks

Graph Neural Networks model the recommendation problem as a bipartite graph where users and items are nodes, and interactions form edges.

2.3.1 Neural Graph Collaborative Filtering (NGCF). NGCF explicitly incorporates higher-order connectivities between users and items by propagating embeddings through multiple graph convolutional layers [9]. In the initial step, each user and item is associated with an ID embedding. Let $\mathbf{e}_u^{(0)}$ denote the ID embedding of user u and $\mathbf{e}_i^{(0)}$ denote the ID embedding of item i. Then NGCF leverages the user-item interaction graph to propagate embeddings as:

$$\mathbf{e}_u^{(k+1)} = \sigma\Big(\mathbf{W}_1 \mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \cdot (\mathbf{W}_1 \mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))\Big), \tag{7}$$

$$\mathbf{e}_i^{(k+1)} = \sigma\Big(\mathbf{W}_1 \mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \cdot (\mathbf{W}_1 \mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}))\Big), \tag{8}$$

where $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_i^{(k)}$ respectively denote the refined embedding of user u and item i after k layers propagation, σ is the nonlinear activation function, $\mathcal{N}_u$ denotes the set of items that are interacted by user u, $\mathcal{N}_i$ denotes the set of users that interact with item i, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weight matrices to perform feature transformation in each layer. By propagating L layers, NGCF obtains $L+1$ embeddings to describe a user $(\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, ..., \mathbf{e}_u^{(L)})$ and an item $(\mathbf{e}_i^{(0)}, \mathbf{e}_i^{(1)}, ..., \mathbf{e}_i^{(L)})$. It then concatenates these $L+1$ embeddings to obtain the final user embedding and item embedding, using inner product to generate the prediction score.

2.3.2 LightGCN. LightGCN simplifies NGCF by removing nonlinearities and feature transformations [8]. The embedding propagation rule becomes:

$$e_i^{(l+1)} = \sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}} e_j^{(l)} \tag{9}$$

The final embedding is a weighted sum of embeddings at different layers:

$$e_i = \sum_{l=0}^{L} \alpha_l e_i^{(l)} \tag{10}$$

where α_l denotes the importance of the l-th layer embedding.

Algorithm 1 LightGCN Algorithm

Input: User-item interaction matrix R, embedding size d, number of layers L
Output: User embeddings matrix E^U, item embeddings matrix E^I
Initialize $E^{U(0)}, E^{I(0)}$ randomly
$A = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$
$D_U = \text{diag}(A \cdot \mathbf{1})$
$D_I = \text{diag}(A^T \cdot \mathbf{1})$
for $l = 0$ to $L-1$ **do**
  $E^{U(l+1)} = (D_U^{-\frac{1}{2}} A D_I^{-\frac{1}{2}}) E^{I(l)}$
  $E^{I(l+1)} = (D_I^{-\frac{1}{2}} A^T D_U^{-\frac{1}{2}}) E^{U(l)}$
end for
$E^U = \sum_{l=0}^{L} \alpha_l E^{U(l)}$
$E^I = \sum_{l=0}^{L} \alpha_l E^{I(l)}$
return E^U, E^I

3 METHODOLOGY

3.1 Experimental Setup

For comprehensive evaluation, we implemented five recommendation algorithms:

(1) User-based Collaborative Filtering (UCF)
(2) Item-based Collaborative Filtering (ICF)
(3) Singular Value Decomposition (SVD)

(4) Neural Graph Collaborative Filtering (NGCF)
(5) LightGCN

Additionally, we explored dataset-specific approaches. For the MovieLens dataset, we experimented with content-based methods, utilizing movie-specific categories and movie titles (some popular movies have sequels, such as Iron Man 1, Iron Man 2, etc.). For the Last.fm dataset, we leveraged the user-friend relationship table unique to this dataset, attempting to recommend artists based on other users that a user follows.

Although these two methods did not perform well overall, we still attempted to build recommendation algorithms using other features from the datasets. The underwhelming performance may be attributed to several factors: For MovieLens, the complex features we constructed may not effectively reflect the common patterns of items liked by all users, making simple abstract item ratings more effective; For Last.fm, the preferences of friends that users follow do not completely represent the artistic styles that users like. While some users' friends have common interests, this is clearly not the case for all users, which might explain why this algorithm's overall performance was not satisfactory.

3.2 Dataset Processing

We applied consistent preprocessing across all datasets:

(1) Removal of users and items with fewer than 5 interactions
(2) Implementation of random, users and temporal splits to evaluate temporal generalization
(3) For the music dataset, we converted weights to 0-5 ratings, using logarithmic normalization to ensure a reasonable distribution of ratings.

3.3 Evaluation Metrics

Our evaluation focuses on ranking-based metrics, which better reflect real-world recommendation scenarios:

(1) **Hit Ratio at K (HR@K)**

$$\text{HR@K} = \frac{1}{|U|} \sum_{u \in U} \delta(r_u \leq K) \tag{11}$$

where $\delta(x) = 1$ if x is true, 0 otherwise, and r_u denotes the rank position of the relevant item for user u.

(2) **Normalized Discounted Cumulative Gain at K (NDCG@K)**

$$\text{NDCG@K} = \frac{1}{|U|} \sum_{u \in U} \frac{2^{rel_u} - 1}{\log_2(r_u + 1)} \tag{12}$$

where the relevant item is ranked at position r_u by the algorithm.

(3) **Mean Reciprocal Rank at K (MRR@K)**

$$\text{MRR@K} = \frac{1}{|U|} \sum_{u \in U} \begin{cases} \frac{1}{r_u} & \text{if } r_u \leq K \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Again, r_u denotes the rank of the first correct recommendation item to user u.

3.4 Hyperparameter Optimization

To ensure fair comparison, we performed grid search on validation sets for each algorithm:

(1) UCF/ICF: Neighborhood size $K \in \{20, 30, 40, 50\}$, similarity metrics (Pearson, cosine)
(2) SVD: Latent factors $K \in \{50, 100, 200, 300, 500\}$, regularization $\lambda \in \{0.001, 0.01, 0.1\}$
(3) NGCF/LightGCN: Embedding size $d \in \{32, 64, 128, 256\}$, layer count $L \in \{2, 3, 4, 5\}$.

4 EXPERIMENTAL RESULTS

4.1 Performance Comparison

We conducted extensive experiments on MovieLens-1M, Last.FM and Yelp datasets using different splitting strategies: user-split (default), time-split, and random-split. Tables 1 to 6 present the comparative performance results using standard evaluation metrics.

For the cold-start problem, where the algorithm training data does not include information about users in the test set (the algorithm has no information about new users), we simply employed the Popularity method for recommendations.

Regarding data sparsity, which increases when some items are filtered from the data tables during preprocessing, we addressed this challenge through SVD algorithm matrix decomposition and graph-based embedding techniques. Additionally, we implemented dictionary mapping on the data to reduce memory consumption and optimize computational efficiency.

Table 1: Performance comparison on MovieLens-1M dataset using user-split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	**0.8352**	**0.2764**	**0.5048**	**0.9122**	**0.2798**	430.69
ItemCF	0.7785	0.2535	0.4699	0.8700	0.2516	291.47
SVD	0.4815	0.0950	0.2177	0.6197	0.0968	101.10
NGCF	0.6801	0.1870	0.3638	0.7973	0.1882	367.61
LightGCN	0.6122	0.1571	0.3111	0.7442	0.1585	256.67
Content-based	0.2286	0.0261	0.0589	0.4116	0.0337	1359.90
Random	0.0580	0.0062	0.0163	0.1014	0.0067	37.85
Popularity	0.6111	0.1565	0.3106	0.7512	0.1597	35.87

Table 2: Performance comparison on Last.FM dataset using user-split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	**0.5534**	**0.1861**	**0.2713**	**0.6696**	**0.2124**	97.22
ItemCF	0.3957	0.1365	0.2022	0.4831	0.1567	178.50
SVD	0.4947	0.1685	0.2512	0.6187	0.1985	16.09
NGCF	0.4073	0.1312	0.1889	0.5208	0.1528	19.80
LightGCN	0.3619	0.1274	0.1735	0.4859	0.1486	14.61
Friend-based	0.2302	0.0608	0.0894	0.3547	0.0787	57.26
Random	0.0055	0.0007	0.0015	0.0072	0.0009	8.16
Popularity	0.3116	0.0921	0.1142	0.4228	0.1117	6.49

Table 3: Performance comparison on MovieLens-1M dataset using time-split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	0.7729	0.3507	0.5527	0.8503	0.3246	331.87
ItemCF	0.7176	0.2793	0.4596	0.7978	0.2625	261.44
SVD	0.6594	0.2323	0.4011	0.7602	0.2176	90.65
NGCF	**0.7919**	**0.3638**	**0.5617**	**0.8592**	**0.3362**	401.96
LightGCN	0.7622	0.3444	0.5336	0.8362	0.3216	261.10
Content-based	0.4633	0.1922	0.3039	0.5648	0.1815	364.01
Random	0.1602	0.0200	0.0534	0.2689	0.0202	25.83
Popularity	0.7059	0.2726	0.4481	0.8017	0.2562	24.94

Table 4: Performance comparison on MovieLens-1M dataset using random-split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	**0.8177**	**0.2786**	**0.5010**	**0.8997**	**0.2799**	433.95
ItemCF	0.7670	0.2557	0.4798	0.8510	0.2522	285.64
SVD	0.4659	0.0935	0.2107	0.5978	0.0950	123.31
NGCF	0.6207	0.1625	0.3116	0.7399	0.1639	239.12
LightGCN	0.5955	0.1548	0.3072	0.7243	0.1564	99.26
Content-based	0.2272	0.0278	0.0640	0.4016	0.0344	1349.70
Random	0.0547	0.0060	0.0164	0.1023	0.0068	27.64
Popularity	0.5982	0.1547	0.3075	0.7261	0.1559	25.68

Table 5: Performance comparison on Last.FM dataset using random-split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	**0.5302**	**0.1885**	**0.2924**	**0.6255**	**0.2124**	89.10
ItemCF	0.3604	0.1357	0.2104	0.4380	0.1510	181.25
SVD	0.4801	0.1653	0.2579	0.5913	0.1913	13.40
NGCF	0.3818	0.1272	0.1873	0.4930	0.1496	17.13
LightGCN	0.3464	0.1213	0.1806	0.4539	0.1390	12.04
Friend-based	0.2376	0.0636	0.0956	0.3482	0.0804	48.43
Random	0.0031	0.0004	0.0012	0.0061	0.0008	5.36
Popularity	0.2877	0.0838	0.1184	0.3818	0.1004	3.73

Table 6: Performance comparison on Yelp dataset using default split

Algorithm	HR@10	NDCG@10	MRR@10	HR@20	NDCG@20	Time (s)
UserCF	0.1060	0.0400	0.0430	0.1550	0.0500	957.43
SVD	0.0810	0.0270	0.0290	0.1330	0.0370	95.39
NGCF	**0.1310**	**0.0480**	**0.0490**	**0.2010**	**0.0620**	260.17
LightGCN	0.0040	0.0010	0.0010	0.0090	0.0020	132.38
Random	0.0020	0.0000	0.0000	0.0030	0.0010	81.28

4.2 Analysis of Results

4.2.1 Traditional Collaborative Filtering vs. Neural Methods. On both datasets, traditional collaborative filtering methods, particularly UserCF, demonstrated superior performance compared to more complex neural approaches. In the MovieLens-1M dataset, UserCF achieved the highest performance across all metrics with an HR@10 of 0.8352, NDCG@10 of 0.2764, and MRR@10 of 0.5048. Similarly, on the Last.FM dataset, UserCF outperformed other methods with an HR@10 of 0.5534.

This observation contradicts some recent literature suggesting that neural approaches consistently outperform traditional methods. Our analysis suggests that for moderately sized datasets with dense interaction patterns, the simplicity and interpretability of memory-based CF provide competitive advantages. The success of UserCF can be attributed to its ability to directly leverage user similarity patterns without requiring complex embedding optimizations.

Certainly, this may also be related to our less than perfect implementation of advanced algorithms. Nonetheless, the current results indicate that, regardless of whether using the MovieLens or LastFM dataset, the UserCF method outperforms both the baseline algorithms: random, popular and other competing algorithms. This suggests that the UserCF algorithm demonstrates exceptional efficacy in relation to the simple user rating matrices presented in the datasets.

4.2.2 Impact of Dataset Characteristics. The performance gap between traditional and neural methods differs significantly between datasets. On MovieLens-1M, UserCF outperforms NGCF by a substantial margin (HR@10: 0.8352 vs. 0.6801), whereas on Last.FM the gap is smaller (HR@10: 0.5534 vs. 0.4073). This variance can be attributed to different interaction patterns and sparsity levels in the datasets.

MovieLens contains more consistent rating behaviors, benefiting memory-based approaches. Last.FM, representing music listening behavior, exhibits more diverse and sparse interaction patterns, which moderately reduces the effectiveness advantage of traditional methods over neural approaches.

4.2.3 Efficiency Considerations. Runtime performance presents another critical dimension for algorithm selection in practical applications. SVD and LightGCN demonstrated significantly better computational efficiency compared to memory-based CF methods. On the MovieLens dataset, SVD completed in 101.10 seconds compared to UserCF's 430.69 seconds. Similarly, on Last.FM, LightGCN required only 14.61 seconds for training and evaluation.

For industrial applications requiring real-time updates or dealing with massive datasets, this efficiency advantage becomes crucial, potentially justifying some trade-off in recommendation quality.

4.3 Temporal Generalization

To evaluate the algorithms' ability to predict future interactions, we also conducted experiments using temporal splits. Table 3 shows the performance on the MovieLens-1M dataset with time-based splitting.

Interestingly, in the temporal split scenario, NGCF outperformed traditional methods, achieving an HR@10 of 0.7919, NDCG@10 of 0.3638, and MRR@10 of 0.5617. This suggests that neural methods may have advantages in capturing temporal dynamics and evolving user preferences, making them more suitable for future prediction tasks.

4.4 Effect of Different Splitting Strategies

Our experiments across different data splitting strategies (Tables 1, 4, and 3 for MovieLens-1M; Tables 2 and 5 for Last.FM) reveal interesting patterns in algorithm performance.

For MovieLens-1M, UserCF maintains the top performance in both user-split and random-split scenarios, while NGCF takes the lead in the time-split scenario. The performance gap between traditional and neural methods narrows considerably in the time-split setting, suggesting that GNN-based approaches may better capture evolving user preferences over time.

In the Last.FM dataset, UserCF consistently outperforms other methods across both splitting strategies, though the performance metrics show slight variations between user-split and random-split. This consistent advantage suggests that the user similarity patterns in music preference data are particularly well-captured by memory-based collaborative filtering approaches.

4.5 Hyperparameter Analysis

We conducted an extensive hyperparameter study for the neural methods to ensure optimal performance. Tables 7 and 8 illustrate the impact of embedding dimension and layer count on NGCF and LightGCN performance.

Table 7: Effect of hyperparameters on NGCF performance (MovieLens-1M, default split)

Configuration	HR@10	NDCG@10	MRR@10	HR@20
embed_dim=32, n_layers=3	0.5928	0.1357	0.2780	0.7393
embed_dim=64, n_layers=2	0.6251	0.1620	0.3137	0.7593
embed_dim=64, n_layers=3	0.6308	0.1641	0.3190	0.7606
embed_dim=64, n_layers=4	0.6298	0.1634	0.3117	0.7654
embed_dim=64, n_layers=5	0.6250	0.1597	0.3098	0.7615
embed_dim=128, n_layers=3	0.6286	0.1646	0.3198	0.7633
embed_dim=256, n_layers=3	**0.6806**	**0.1940**	**0.3767**	**0.8035**

Table 8: Effect of hyperparameters on LightGCN performance (MovieLens-1M, default split)

Configuration	HR@10	NDCG@10	MRR@10	HR@20
embed_dim=32, n_layers=3	0.6137	0.1573	0.3113	0.7444
embed_dim=64, n_layers=2	0.6149	0.1578	0.3114	0.7429
embed_dim=64, n_layers=3	0.6135	0.1574	0.3112	0.7444
embed_dim=64, n_layers=4	0.6134	0.1571	0.3113	0.7469
embed_dim=64, n_layers=5	0.6144	0.1572	0.3113	0.7461
embed_dim=128, n_layers=3	**0.6144**	**0.1576**	**0.3124**	0.7431
embed_dim=256, n_layers=3	0.6142	0.1578	0.3121	0.7426

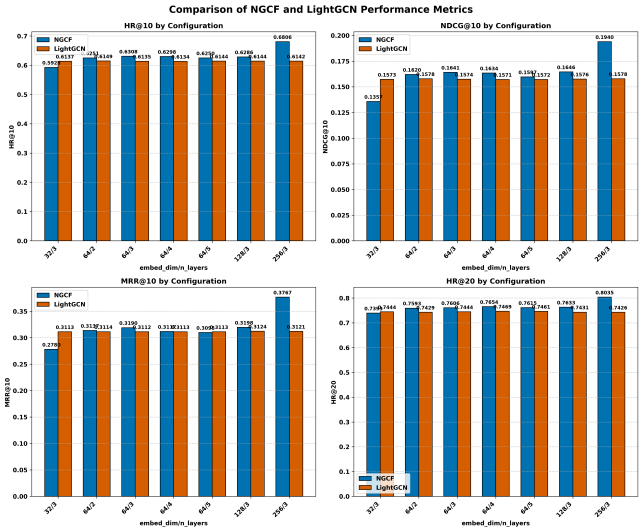

Figure 1: Comparison of NGCF and LightGCN performance metrics across different embedding dimensions and layer configurations. The charts illustrate HR@10, NDCG@10, MRR@10, and HR@20 metrics, showing how both models respond to architecture variations.

For NGCF, we observed that increasing the embedding dimension significantly improves performance, with the best results achieved at embed_dim=256 and n_layers=3. This suggests that NGCF benefits from richer representation capacity to capture complex user-item interactions.

Interestingly, LightGCN showed less sensitivity to hyperparameter variations, with relatively stable performance across different configurations. This aligns with the design philosophy behind LightGCN, which aimed to simplify the NGCF architecture by removing feature transformations and non-linearities.

5 DISCUSSION AND IMPLICATIONS

5.1 Key Findings and Insights

Our experimental results yield several important insights:

(1) Traditional collaborative filtering methods remain highly competitive for recommendation tasks, particularly in domains with dense user-item interactions. However, this holds true for medium or small datasets; when dealing with very large datasets, the efficiency issues make it difficult for these algorithms to be applied directly in industrial settings.
(2) The relative performance of algorithms varies significantly with dataset characteristics and evaluation protocols. No single algorithm consistently outperforms others across all settings.
(3) Neural methods demonstrate advantages in temporal prediction scenarios, suggesting their ability to better capture evolving preference patterns.
(4) The efficiency-effectiveness trade-off is an important consideration, with simpler methods like SVD offering substantial computational advantages with acceptable performance.
(5) Hyperparameter tuning significantly impacts the performance of neural methods, with NGCF showing higher sensitivity compared to LightGCN.

5.2 Practical Recommendations

Based on our findings, we offer the following practical recommendations for recommendation system implementation:

(1) For small to medium-sized datasets with dense interactions, consider UserCF as a strong baseline that may outperform more complex approaches.
(2) When computational efficiency is critical or for very large-scale applications, SVD and LightGCN offer good performance-efficiency trade-offs.
(3) For applications where prediction of future interactions is important, neural methods like NGCF are preferred despite their higher computational requirements.
(4) LightGCN is recommended over NGCF when computational resources are limited, as it offers comparable performance with fewer parameters and lower training time.
(5) Careful hyperparameter tuning is essential for neural methods, particularly for NGCF where higher embedding dimensions can significantly improve performance.

5.3 Limitations and Future Work

While our study provides valuable insights, several limitations should be acknowledged:

(1) Our evaluation focused on two widely-used but moderately sized datasets. Future work should extend to larger-scale datasets with different sparsity characteristics.
(2) We primarily evaluated general recommendation performance. Domain-specific requirements like diversity, novelty, or fairness were not considered.
(3) Recent innovations in transformer-based recommendation algorithms were not included in our comparison.

(4) Our evaluation metrics focused on ranking quality rather than other aspects like recommendation diversity or serendipity.

Future research directions include investigating hybrid approaches that combine the strengths of traditional and neural methods, exploring the impact of different data augmentation techniques, and evaluating recommendation algorithms on multi-modal data incorporating textual, visual, and behavioral signals.

REFERENCES

[1] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender Systems Handbook*. Springer US, 2011.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[3] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. *The adaptive web*, pages 325–341, 2007.

[4] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[5] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[7] Robert M Bell and Yehuda Koren. The bellkor solution to the netflix prize. *KorBell Team's Report to Netflix*, 2007.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 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*, pages 639–648, 2020.

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

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

[11] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context, 2015.

[12] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM Conference on Recommender Systems*. ACM, 2011.

[13] Yelp. Yelp Dataset Challenge, 2023.