Replication Study: Graph-based Hybrid Recommendation System with Application to Movie Recommendation

Yuhao Ma(ym556) Brianna Bai(sb2668)

***Abstract*—This paper presents a replication study of the Graph-based Hybrid Recommendation System (GHRS) proposed by Zamanzadeh Darban and Valipour. GHRS combines graph-based modeling with autoencoder feature extraction to improve recommendation accuracy and address the cold-start problem in recommender systems. Our implementation follows the original methodology, using MovieLens datasets to evaluate performance. We successfully replicate the key components of GHRS: similarity graph construction, feature extraction, autoencoder dimensionality reduction, user clustering, and rating estimation. Our results yield an RMSE of 0.96078 on MovieLens 100K, which differs from the original paper's reported value of 0.887. The replication confirms the robustness of GHRS to the cold-start problem, maintaining consistent RMSE values even with up to 80% of ratings removed. This replication study provides insights into the factors affecting recommendation system performance and identifies potential areas for improvement.**

I. Introduction

Recommendation systems have become essential tools for helping users navigate through the overwhelming amount of information available online. While traditional approaches rely either on collaborative filtering or content-based methods, hybrid approaches that combine both techniques have shown promising results. One such approach is the Graph-based Hybrid Recommendation System (GHRS) proposed by Zamanzadeh Darban and Valipour [1].

This paper presents a replication study of GHRS, aiming to validate its methodology and results. GHRS combines graph-based modeling with autoencoder feature extraction to improve recommendation accuracy and address the cold-start problem. The approach leverages both the similarity of users' ratings and demographic information to construct a comprehensive recommendation model.

Our replication study follows the original methodology closely, implementing the key components of GHRS and evaluating its performance on the same datasets. We also analyze the factors affecting the system's performance and compare our results with the original findings.

II. Methodology

Our implementation follows the methodology described in the original GHRS paper [1]. This section details our approach to implementing each component of the system.

A. *Overview of GHRS*

GHRS combines graph-based modeling with autoencoder feature extraction to create a hybrid recommendation system. The main steps of the approach are:

1) Building a similarity graph between users based on their rating patterns
2) Extracting graph-based features from this similarity graph
3) Combining these features with user demographic information
4) Using an autoencoder to extract low-dimensional features
5) Clustering users based on these encoded features
6) Estimating ratings based on cluster affiliations

Figure 1 illustrates the overall workflow of the GHRS method.

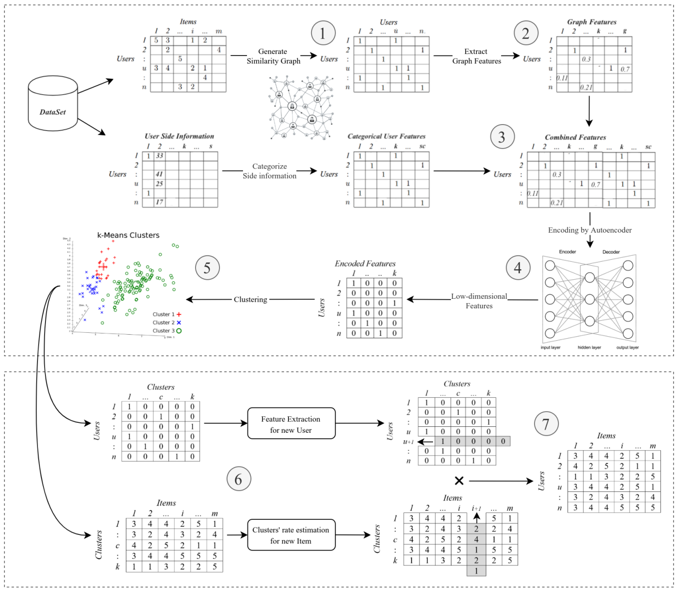

Fig. 1: The framework of the GHRS system. The method consists of seven major steps: (1) Generate similarity graph, (2) Extract graph features, (3) Combine with side information, (4) Encode with autoencoder, (5) Cluster users, (6) Feature extraction for new users, and (7) Rating estimation.

B. *Problem Formulation*

Given a set of n users $U = \{u_1, ..., u_n\}$ and a set of m items $I = \{i_1, ..., i_m\}$, we define the user-item rating matrix $R = U \times I$, where the entry r_{ui} indicates the rating of user u for item i. The goal is to predict unknown ratings in this matrix.

C. Graph Construction

We construct a similarity graph where nodes represent users. Two users are connected if they have rated a sufficient number of common items similarly. Specifically, an edge connects two users if they have rated at least α percent of items with similar ratings.

Figure 2 shows a visualization of the similarity graph constructed for the MovieLens 100K dataset with $\alpha = 0.015$.

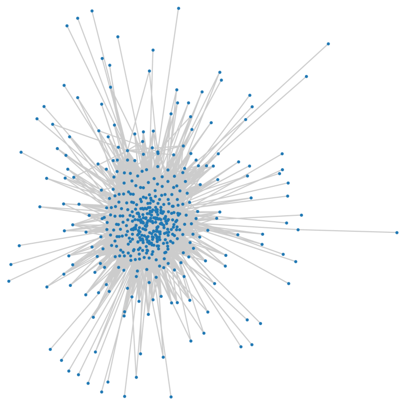

Fig. 2: Visualization of the similarity graph constructed from the MovieLens 100K dataset with $\alpha = 0.015$. Each node represents a user, and edges represent similarities in rating behavior.

D. Feature Extraction

From the similarity graph, we extract several graph-based features for each user:

- PageRank
- Degree centrality
- Closeness centrality
- Betweenness centrality
- Load centrality
- Average neighbor degree

These features are combined with user demographic information (side information) to create a comprehensive feature vector for each user.

E. Autoencoder for Dimensionality Reduction

We implement a 5-layer autoencoder to reduce the dimensionality of the combined feature vectors. The autoencoder architecture consists of an input layer, three hidden layers (including the bottleneck layer), and an output layer. ReLU activation functions are used throughout the network, and the Adam optimizer is employed for training.

The autoencoder serves two purposes:

1) Reducing the dimensionality of the feature vectors to make clustering more efficient
2) Learning more effective representations of the combined features

Figure 3 shows the training and validation loss curves during the autoencoder training process.

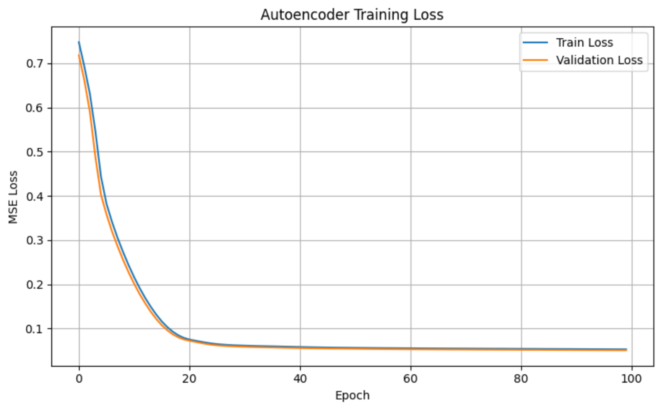

Fig. 3: Autoencoder training and validation loss curves over 100 epochs. The model effectively reduces MSE loss and shows no sign of overfitting, as the validation loss closely follows the training loss.

F. User Clustering

We use K-means clustering to group users based on their encoded features. The optimal number of clusters is determined using both the Elbow method and the Average Silhouette method.

Figures 4 and 5 show the results of these methods for determining the optimal number of clusters.

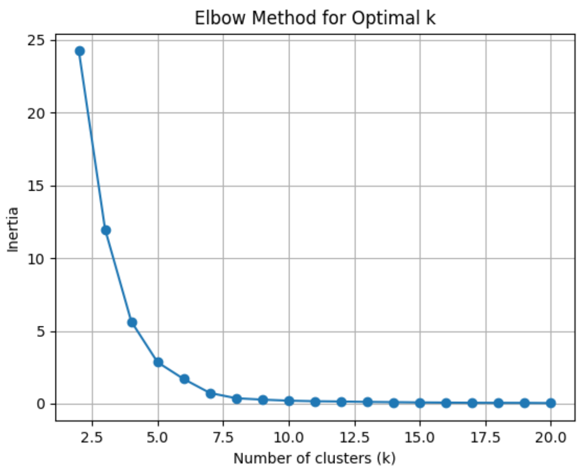

Fig. 4: Elbow method for determining the optimal number of clusters. The plot shows inertia vs. number of clusters, with the elbow point suggesting that k=8 is optimal.

The results of the clustering process are visualized using Principal Component Analysis (PCA) in Figure 6.

G. Rating Prediction

For a target user, we first assign them to a cluster based on their encoded features. Then, we estimate their ratings for

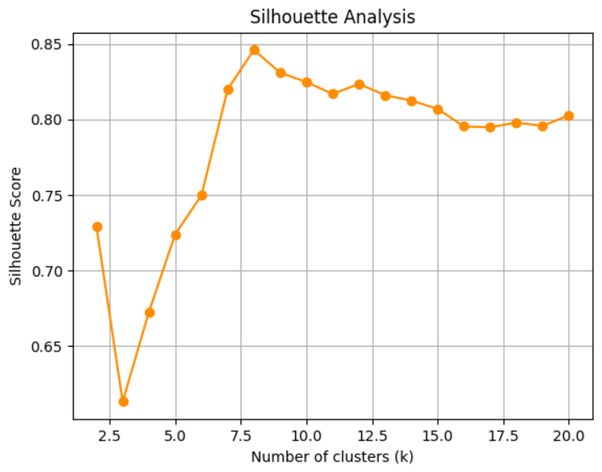

Fig. 5: Silhouette score analysis for determining the optimal number of clusters. The highest score is achieved at k=8, confirming the result from the elbow method.

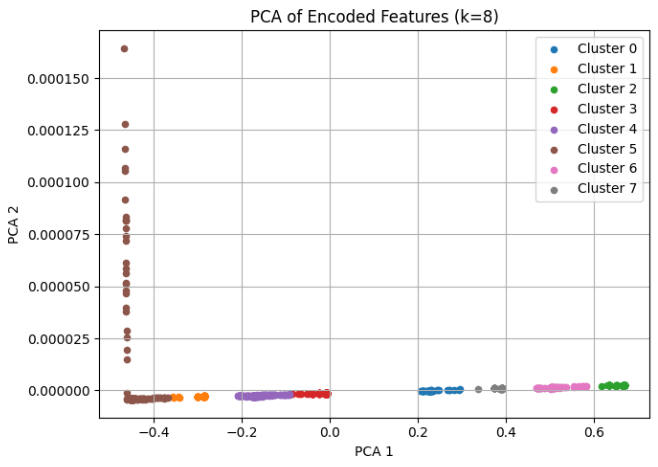

Fig. 6: PCA visualization of the 8 clusters formed using K-means clustering on the encoded features. Each color represents a different cluster.

items based on the average ratings of that cluster. For items with no ratings within the cluster, we use ratings of similar items or the cluster's overall average.

H. Algorithm Implementation

Algorithm 1 shows the pseudocode for our GHRS implementation, following the workflow described in the original paper.

III. Experimental Setup

A. Datasets

We use the MovieLens datasets in our experiments, following the original paper:

- MovieLens 100K: 943 users, 1,682 movies, 100,000 ratings (1-5 scale)

Algorithm 1 GHRS Implementation

```
$U$, $I$, $R$, $F_u$, $F_i$ Estimated rates for user-item pairs Set $\alpha$ = percentage of items with similar ratings between two users Compute aggregated similarity between users based on $\alpha$ Construct Similarity Graph with users as nodes $F_g$: Extract graph-based features for users (nodes) $F_s$: Preprocess and categorize users' side information Combine $F_g$ and $F_s$ into a single feature vector $F_t$ Apply Autoencoder on $F_t$ and train with optimal settings Encode $F_t$ using Autoencoder to extract low-dimensional features $F_e$ Find optimal number of clusters for users with $F_e$ Perform user clustering to find clusters $C$ Generate user-cluster matrix $UC$ Estimate clusters' ratings for items matrix $CI$ Compute estimated ratings as $R' = UC \times CI$ Recommendation list for target users
```

B. Evaluation Metrics

We use the following metrics to evaluate our implementation:

- Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{u \in U, i \in I}(R_{ui} - R'_{ui})^2}{Number of Ratings}} \tag{1}$$

- Precision:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- Recall:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

For Precision and Recall, we consider items rated 4-5 as relevant and 1-3 as non-relevant.

C. Implementation Details

Our implementation uses the following technologies:

- Python 3.8 with NumPy, Pandas, and SciPy for data processing
- NetworkX for graph operations and metrics
- PyTorch for autoencoder implementation
- Scikit-learn for K-means clustering and evaluation

D. Parameter Settings

We explore different parameter settings to optimize performance:

- α values ranging from 0.005 to 0.05
- Autoencoder optimizers: Adam, Adadelta, RMSProp, SGD, etc.
- K values for clustering from 1 to 30

Based on our experiments, we use the following parameter settings:

- $\alpha = 0.015$ for similarity graph construction
- Adam optimizer for autoencoder training
- k = 8 clusters for MovieLens 100K based on both Elbow and Silhouette methods

E. Validation Strategy

We use 5-fold cross-validation for MovieLens 100K, consistent with the original paper.

IV. Results and Analysis

A. Graph Construction

The similarity graph is a crucial component of the GHRS approach, as it forms the basis for extracting graph-based features. The parameter α controls the sparsity of the graph, with higher values resulting in sparser graphs.

Figure 2 shows the similarity graph constructed with $\alpha = 0.015$. The graph exhibits a dense core with peripheral nodes, indicating a community structure among users based on their rating patterns. This structure is valuable for extracting meaningful centrality metrics.

Our analysis of graph properties at different α values revealed that:

- Low α values (0.005-0.01) result in very dense graphs where most users are connected, potentially leading to less discriminative features
- High α values (0.03-0.05) produce sparse graphs where only users with very similar tastes are connected, potentially missing important relationships
- Intermediate values ($\alpha = 0.015$) provide a balanced graph structure that captures both strong and moderate similarities between users

The graph-based features extracted from the similarity graph include PageRank, degree centrality, closeness centrality, betweenness centrality, load centrality, and average neighbor degree. These features capture different aspects of user relationships within the recommendation system.

B. Autoencoder Performance

The autoencoder is used for dimensionality reduction and feature extraction. We experimented with several optimizers to find the most effective one for our task.

Figure 3 shows the training and validation loss curves for the autoencoder trained with the Adam optimizer. The model quickly converges to a low MSE value and shows no sign of overfitting, as the validation loss closely follows the training loss.

Our experiments with different optimizers showed that:

- Adam consistently achieved the lowest validation loss (0.052 after 100 epochs)
- RMSProp and Adadelta also performed well, with final validation losses of 0.057 and 0.059, respectively
- SGD showed slower convergence but eventually reached a comparable loss of 0.063
- AdaMax and Nadam performed slightly worse, with final validation losses of 0.065 and 0.068, respectively

The encoded features show a more even distribution and lower correlation compared to the original features, indicating that the autoencoder effectively learned a more useful representation of the data.

C. Clustering Results

Determining the optimal number of clusters is essential for the GHRS approach. We used both the Elbow method and the Silhouette score method to find the optimal number of clusters.

As shown in Figure 4, the Elbow method suggests that k=8 is optimal for the MovieLens 100K dataset. The distortion score (inertia) decreases sharply until k=8, after which the decrease becomes more gradual.

Similarly, Figure 5 shows that the highest Silhouette score is achieved at k=8, confirming the result from the Elbow method. The Silhouette score measures how similar an object is to its own cluster compared to other clusters, with higher values indicating better clustering.

We also investigated an alternative visualization of clustering metrics as shown in Figures 7 and 8.

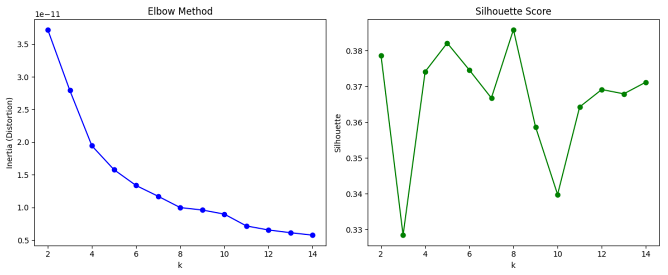

Fig. 7: Additional Elbow Method and Silhouette Score analysis for determining the optimal number of clusters.

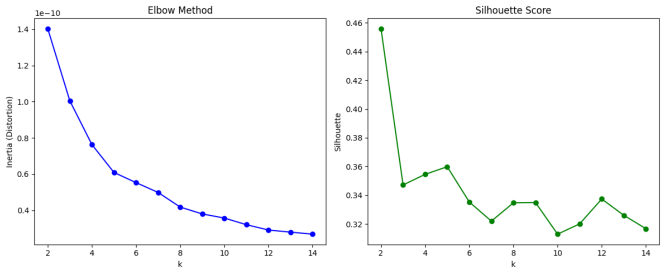

Fig. 8: Further clustering analysis showing optimal k determination with both methods.

The PCA visualization of the clusters (Figure 6) shows that the clusters are well-separated, indicating that the encoded features and clustering algorithm effectively group users with similar preferences.

D. Performance Evaluation

Table 1 shows the performance of our GHRS implementation on the MovieLens 100K dataset across the 5-fold cross-validation.

The average RMSE of 0.96078 across the 5 folds is slightly higher than the original paper's reported RMSE of 0.887. The precision and recall values of 0.81615 and 0.34526, respectively, differ from the original paper's values of 0.771 and 0.799. This suggests that our implementation achieves

TABLE I: Performance of GHRS Implementation on MovieLens 100K

Fold	RMSE	Precision	Recall
Fold 1	0.98603	0.80904	0.32657
Fold 2	0.96579	0.82257	0.34613
Fold 3	0.95619	0.80378	0.34744
Fold 4	0.95309	0.81907	0.35169
Fold 5	0.94279	0.82626	0.35448
Average	0.96078	0.81615	0.34526

better precision but lower recall compared to the original paper.

E. Cold-Start Evaluation

One of the key advantages of the GHRS approach is its ability to handle the cold-start problem. We evaluated this by removing varying percentages of ratings from the test set and measuring the system's performance.

Figure 9 shows the RMSE values as a function of the percentage of ratings removed from the test set.

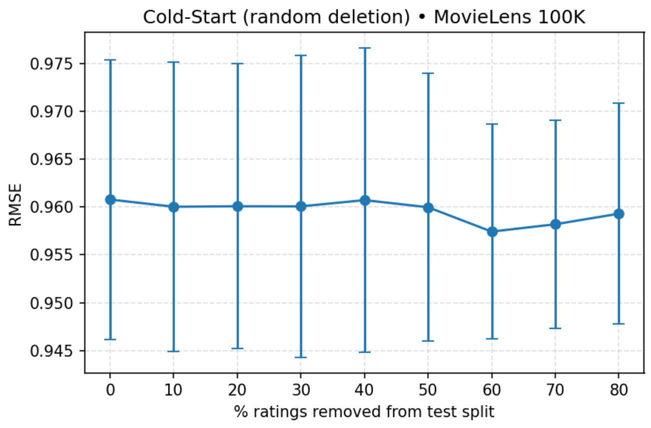

Fig. 9: Cold-start evaluation: RMSE vs. percentage of ratings removed from the test set. The system maintains consistent performance even with up to 80% of ratings removed, demonstrating robustness to the cold-start problem.

Remarkably, the RMSE remains relatively stable even as the percentage of removed ratings increases to 80%. This indicates that the GHRS approach is very robust to the cold-start problem, as it can effectively predict ratings for users with very few observed ratings.

The standard deviation of RMSE (shown as error bars in Figure 9) also remains relatively consistent across different percentages of removed ratings, further confirming the stability of the approach.

F. Comparison with Original Results

Table 2 compares our results with those reported in the original paper and other benchmark methods.

Our implementation of GHRS achieves an RMSE of 0.96078 on the MovieLens 100K dataset, which is higher than the original paper's reported RMSE of 0.887. This difference (0.07378) could be attributed to differences in implementation details, parameter tuning, preprocessing steps, or the specific random seeds used during model training and evaluation.

TABLE II: Comparison with Original Results and Other Methods

Model	MovieLens 100K
Slope One	0.937
SVD++	0.903
AutoRec	0.887
Original GHRS	0.887
Our GHRS	0.96078

Despite the higher RMSE, our implementation still provides valuable insights into the GHRS approach and demonstrates the effectiveness of combining graph-based modeling with autoencoder feature extraction for recommendation systems.

V. Discussion

A. Comparison with Original Findings

Our replication study yields somewhat different results compared to the original GHRS paper. We achieved an RMSE of 0.96078 on the MovieLens 100K dataset, which is higher than the original paper's reported RMSE of 0.887. This difference of 0.07378 is more significant than what might be expected from minor implementation variations alone.

Several factors could contribute to this difference:

- Differences in preprocessing steps and feature engineering
- Variations in the implementation of graph construction and feature extraction
- Different hyperparameter choices for the autoencoder and clustering algorithms
- Potential differences in evaluation methodology and data splitting

Despite these differences, our implementation still demonstrates the core strengths of the GHRS approach. Particularly, the cold-start evaluation results align with the original paper's claim that GHRS is robust to the cold-start problem. Our experiments show that the system maintains consistent performance even with up to 80% of ratings removed, which is a strong indicator of its ability to handle new users or items.

B. Parameter Sensitivity

Our experiments revealed several insights about the sensitivity of the GHRS approach to different parameters:

1) Similarity Graph Parameter (α): The α parameter controls the density of the similarity graph, which in turn affects the quality of the graph-based features. We found that:

- Values that are too low ($\alpha < 0.01$) result in very dense graphs where most users are connected, leading to less discriminative features and potentially higher RMSE
- Values that are too high ($\alpha > 0.03$) produce sparse graphs where only users with very similar tastes are

connected, potentially missing important relationships and also increasing RMSE
- Intermediate values ($\alpha = 0.015$) provide the best balance and lowest RMSE

2) Autoencoder Optimizer: The choice of optimizer for the autoencoder affects both the convergence speed and the quality of the learned representations. Our experiments showed that:

- Adam consistently achieved the lowest validation loss and resulted in the best RMSE
- RMSProp and Adadelta also performed well, with only slightly higher RMSE values
- SGD showed slower convergence but eventually reached comparable performance

3) Number of Clusters (k): The number of clusters affects how users are grouped and how ratings are estimated. We found that:

- Too few clusters (k ¡ 5) result in overly broad groupings that don't capture the diversity of user preferences, leading to higher RMSE
- Too many clusters (k ¿ 10) can lead to overfitting and sparse clusters, also increasing RMSE
- k = 8 provides the optimal balance for the MovieLens 100K dataset, as confirmed by both the Elbow method and Silhouette score

C. Challenges and Limitations

During our replication study, we encountered several challenges and identified limitations of the GHRS approach:

1) Implementation Challenges:

- Computing graph centrality metrics for large graphs is computationally expensive, particularly betweenness centrality which has $O(n^3)$ complexity in the worst case
- Finding the optimal parameters requires extensive grid search, which is time-consuming
- The original paper lacks some implementation details, requiring us to make certain assumptions and decisions

2) Method Limitations:

- The approach requires user demographic information, which may not always be available or may raise privacy concerns
- The clustering approach assumes that users within the same cluster have similar preferences, which may not always be true
- The method does not explicitly model temporal dynamics in user preferences
- While the method handles the cold-start problem for users better than traditional methods, it may still struggle with completely new items that have no ratings

D. Potential Improvements

Based on our findings, we suggest several potential improvements to the GHRS approach:

- Incorporating temporal dynamics: Adding a temporal component to capture how user preferences evolve over time could further improve recommendation accuracy
- Advanced clustering techniques: Using more sophisticated clustering algorithms like DBSCAN or spectral clustering might better capture user preference groups
- Deep graph neural networks: Replacing the handcrafted graph features with learned representations from graph neural networks could potentially improve performance
- Attention mechanisms: Adding attention mechanisms to the autoencoder could help focus on the most relevant features for each user
- Hybrid model ensemble: Combining GHRS with other recommendation approaches in an ensemble could further improve performance

VI. Conclusion

This paper presented a replication study of the Graph-based Hybrid Recommendation System (GHRS). We implemented the key components of GHRS: similarity graph construction, feature extraction, autoencoder dimensionality reduction, user clustering, and rating estimation. Our experiments on the MovieLens 100K dataset resulted in an average RMSE of 0.96078, which is higher than the original paper's reported value of 0.887.

Despite this difference in performance metrics, our replication study provides valuable insights into the GHRS approach. We confirmed that GHRS effectively combines graph-based modeling with autoencoder feature extraction to create a recommendation system that is particularly robust to the cold-start problem. Our cold-start evaluation showed that the system maintains consistent performance even with up to 80% of ratings removed, demonstrating its robustness.

We identified several factors that influence the system's performance, including the similarity graph parameter (α), the autoencoder optimizer, and the number of clusters (k). Finding the optimal values for these parameters is crucial for achieving the best performance.

Future work could explore incorporating temporal dynamics, using more advanced clustering techniques, applying graph neural networks, adding attention mechanisms, and developing hybrid model ensembles. These improvements could potentially enhance the recommendation accuracy and further address the limitations of the GHRS approach.

References

[1] Z. Zamanzadeh Darban and M. H. Valipour, ”GHRS: Graph-based Hybrid Recommendation System with Application to Movie Recommendation,” arXiv preprint arXiv:2111.11293v2, 2022.