Data Science Lab - Assignement 1

Barthélémy Charlier Benjamin Séguin Marius Henry

PSL University – ENS PSL

Abstract

In the context of the Collaborative Filtering assignment of the Data Science Lab course of PSL, this work aims to investigate lightweight yet effective approaches. It begins with presentation and mathematical improvement of the baseline followed by a selection of novel method to incorporate side information (user preferences genres, year of release) in the matrix factorization framework.

1 Introduction

Matrix Factorization (MF) remains a cornerstone methodology for rating prediction in recommender systems due to its effectiveness and computational efficiency. Despite the recent dominance of transformer-based architectures in machine learning, MF approaches demonstrate that well-designed, interpretable models can achieve competitive performance without the computational overhead of complex deep learning methods. This makes MF particularly valuable for large-scale recommendation tasks where efficiency and scalability are paramount.

This paper addresses two primary objectives:

- First, we implement and optimize **Alternating Least Squares (ALS)** as a baseline, then demonstrate that the **UserReg** framework—which incorporates user consistency regularization—achieves superior performance through its principled handling of user rating patterns.
- Second, we evaluate **Heterogeneous Information Network Embedding for Recommendation (HERec)**, an advanced approach that extends matrix factorization by leveraging meta-path-based embeddings to incorporate side information (movie genres and release years) into the rating prediction model, thereby capturing richer semantic relationships between users and items.

This paper is structured as follows: Section 2 presents the data, Section 3 describes the baseline ALS, Section 4 shows the UserReg model, Section 6 explains the HIN Embedding methodology, Section 7 presents the results, and Section 8 concludes.

Note to reader : This report might include inconsistent notation on the MF framework across the different section as we might have lacked the time to transpose different papers's notation into a uniform one.

2 Dataset Analysis

This dataset, characteristic of real-world recommendation scenarios is primarily defined by its extreme sparsity of more than 98% (data_exploration.ipynb). This poses a significant challenge for predictive modeling. The ratings are skewed positively, with an average score of approximately 3.5, suggesting users tend to rate content they enjoy. We also observe that integer ratings (e.g., 3.0, 4.0) are submitted far more frequently than half-point scores. While this suggests a potential for ordinal regression models, we chose to treat ratings as continuous values to align with the standard implementation of established collaborative filtering algorithms like ALS. Furthermore, the mean user has over 50 ratings, but the median is only 23, with one user reaching a maximum of 815. We observe the same pattern among movie ratings with a median of 6 ratings per movie while the most rated movie has been rated 219 times. This sparsity necessitates a content-based compensation model, for which the 20 available genres provide some necessary feature set.

3 Baseline Model and Fine-Tuning

In this section, we present the ALS implementation that serves as a baseline for the evaluation of our other models. We also show some improvements of the ALS regarding the most basic constraints enforced by our problem definition, namely the non-negativity and range of ratings constraints.

3.1 Vanilla ALS

We begin with a baseline ALS implementation (`tests/test_vanilla_als.py`). Movie and user factor matrices are initialized with uniform random values in $[0, 0.1]$. Missing ratings (`NaN` values) are replaced with zeros for computational purposes; this introduces no bias as the loss function and gradient updates are computed only on observed ratings.

The standard ALS procedure alternates between fixing the user matrix and updating movie factors, then fixing the movie matrix and updating user factors. After convergence, predictions are constrained to $[0, 5]$ using `np.clip` and rounded to the nearest 0.5 increment to match the rating scale.

3.2 Alternating Non-Negative Least Squares (ANNLS)

Since movie ratings are inherently non-negative, we enforce non-negativity constraints on the factor matrices. We modify the ALS framework to ANNLS using `scipy.nnls` (`tests/test_annls.py`, transforming the unconstrained optimization problem yields the following objective function:

$$\min_{I\geq 0, U\geq 0} \quad C(I,U) = \sum_{(i,j)\in S} \left(r_{ij} - \sum_{s=1}^{k} i_{is} \cdot u_{js} \right)^2 + \lambda \|I\|_F^2 + \mu \|U\|_F^2, \quad (1)$$

where $I \in \mathbb{R}^{n_{\text{movies}} \times k}$ and $U \in \mathbb{R}^{n_{\text{users}} \times k}$ are the movie and user factor matrices, with i_{is} and u_{js} representing the s-th latent feature. The set S contains all observed (user, movie) rating pairs, r_{ij} denotes the rating given by user j to movie i, and λ, μ are regularization parameters.

3.3 Bounded Non-Negative Least Squares (BNNLS)

To enforce the rating scale $[0, 5]$, we apply box constraints to all factor entries using `scipy.lsq_linear` (`tests/test_bounded_Annls.py`). Each entry is constrained to $[0, q]$ where $q = \sqrt{5/k}$, ensuring that predicted ratings satisfy:

$$\hat{r}_{ij} = \sum_{s=1}^{k} i_{is} \cdot u_{js} \leq k \cdot q^2 = 5, \quad (2)$$

since each product $i_{is} \cdot u_{js} \leq q^2$ and the sum contains k terms. Predictions are then rounded to the nearest 0.5 increment to match the discrete rating scale.

3.4 Hyperparameter Tuning

We tune BNNLS hyperparameters using grid search over the following ranges:

- Latent dimension $k \in \{1, 2, 5, 10, 15\}$
- Movie regularization $\lambda \in \{0.001, 0.01, 0.1, 1.0\}$
- User regularization $\mu \in \{0.001, 0.01, 0.1, 1.0\}$

The optimal configuration identified via cross-validation is $k = 1$, $\lambda = 0.01$, and $\mu = 1.0$. Grid search results are visualized in Figure 1.

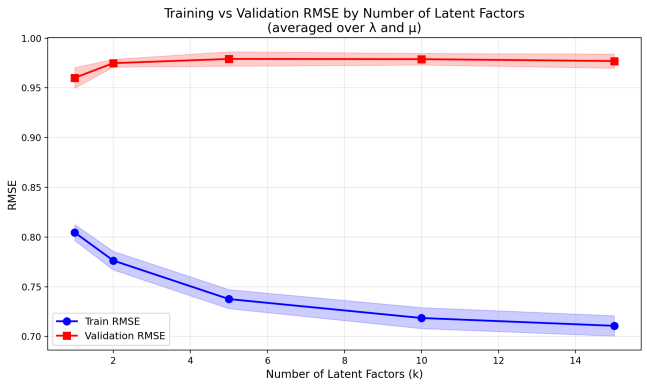

Figure 1: Training and validation RMSE versus number of latent factors k for the BNNLS model. Results are averaged over regularization parameters λ and μ, with shaded regions indicating variance.

4 UserReg

In this section, we describe our implementation of the UserReg model from [5].

4.1 Model Approach

The basic form of MF assumes that the rating $r_{u,i}$ depends only on the latent representations of user u and item i. However, [5] introduce a more nuanced assumption: a user's rating for an item is influenced not only by that item's features, but also by the user's historical preferences across all items they have interacted with, denoted as $I(u)$.

Building on this intuition, the authors formalize a user consistency constraint: the latent feature vector P_u of user u should be similar to the average latent representation of items in $I(u)$. This constraint is incorporated as a regularization term in the objective function:

$$\left\| P_u - \frac{1}{|I(u)|} \sum_{j \in I(u)} Q_j \right\|_F^2 \quad (3)$$

where Q_j represents the latent features of item j. This term penalizes deviations between a user's learned representation and their actual interaction history.

The complete UserReg objective function combines three components: a prediction error term based on observed ratings, the user consistency regularization introduced above, and standard Frobenius norm regularization on all parameters. The objective is minimized using gradient descent:

$$O_{UR} = \frac{1}{2} \sum_{u \in U} \sum_{i \in R_u} (r_{ui} - \tilde{r}_{ui})^2 + \frac{\beta}{2} \sum_{u \in U} \left\| P_u - \frac{1}{|I(u)|} \sum_{j \in I(u)} Q_j \right\|_F^2 + \frac{\lambda}{2} \left(\|P\|_F^2 + \|Q\|_F^2 + \|b_u\|^2 + \|b_i\|^2 \right) \quad (4)$$

where $\|P\|_F^2 = \sum_{u \in U} \|P_u\|_F^2$ denotes the sum of squared Frobenius norms across all user latent vectors, $\|Q\|_F^2 = \sum_{i \in I} \|Q_i\|_F^2$ denotes the same for item latent vectors, and $\|b_u\|^2 = \sum_{u \in U} \|b_u\|^2$ and $\|b_i\|^2 = \sum_{i \in I} \|b_i\|^2$ represent the regularization of user and item biases, respectively.

4.2 Hyperparameter Tuning

The UserReg model demonstrates strong performance with default parameters. We tune the hyperparameters β (user consistency weight), λ (regularization strength), latent dimension k, learning rate, and number of epochs using a two-stage approach: first, random search to identify promising regions of the hyperparameter space, followed by grid search with cross-validation to refine the optimal configuration.

4.3 Computational Complexity Analysis

UserReg provides a strong yet simple framework which efficiently beats the ALS with favorable computational complexity. The computational cost depends on evaluating O_{UR} and computing its gradients. Let m denote the total number of users and $\bar{r}$ the average number of ratings per user. Then $|R| \approx m \times \bar{r}$, and the computational complexity of the objective function is $\mathcal{O}(m\bar{r}d + m\bar{r}ld)$, where l represents the average number of items a user has interacted with and d is the latent dimension. Since $m \gg \bar{r}, l, d$, the complexity is linear with respect to the number of users.

Note: UserReg performs well on both our simplified MovieLens dataset and the testing platform. However, as discussed in Section 2, there exists a correlation between the ratings users assign to movies and the movie's genre and/or release year. This observation motivates our next approach: incorporating item and user metadata into the matrix factorization framework to capture these relationships explicitly.

5 Experiments on ALS and UserReg

Following the last two sections, we describe the attempts of tuning the ALS and UserReg models.

5.1 Initialization of Factor Matrices and Biases

We explore initialization strategies for the factor matrices (P_u, Q_i) and potential bias vectors (b_u, b_i) in BNNLS and UserReg. An attempt to initialize the factor matrices using a heuristic based on the global average rating does not yield better results than a standard initialization with small random noise. However, a similar heuristic for initializing the bias terms in the UserReg model is successful (`userreg.py`); this approach centers the biases on the global average rating and then adjusts them based on each user's or item's specific average rating.
Initializing the factor matrices with a pre-computed mean fails, most likely because it imposes a strong, premature structure on the complex latent interactions, whereas initializing the simpler bias terms with this information works well because it directly encodes known, separable rating tendencies, providing the model with a better starting point for convergence.

5.2 Normalization of the Rating Matrix

In our experiments, we also implement a global normalization scheme on the rating matrix (`tests/normalization.py`). The intuition is that transforming the ratings to have a mean of zero and unit variance before training would improve model performance due to the resulting data symmetry.
However, this pre-processing step does not yield a significant improvement in final predictive accuracy for either model, even though it was empirically observed to slightly accelerate the convergence for the UserReg model (`userreg.ipynb`). The lack of performance gain is likely because models are already inherently designed to capture and isolate the same rating scale and offset information that global normalization removes, making this normalization step redundant for final accuracy.

5.3 Weighting Down the Ubiquitous Protagonists

We implement a weighting scheme in both ANNLS (`test/test_wannls.ipynb`) and UserReg (`userreg.ipynb`) to mitigate potential popularity bias. The idea is to make each rating's contribution to the loss function inversely proportional to that popular users and movies has a too big impact on the predictions, likely leading to over-fitting.
We modify the objective function's squared error term to $\sum w_{ui}(r_{ui} - \hat{r}_{ui})^2$, which subsequently alters the update rules derived from the new gradients. Despite using a stabilized and normalized weighting approach, this method does not improve predictive accuracy for either model. The reason it fails to improve performance might be because both models already include biais or other regularization terms, which are specifically designed to capture and isolate that kind of popularity-driven deviations..

5.4 Alternative Models

We test alternative MF models, including a standard non-binary MF [4] (`test/test_NNBMF.py`) and a sigmoid gradient descent model. The rationale is that a sigmoid function would naturally constrain predictions to a valid rating scale. Both models perform worse than a simpler BNNLS baseline. These methods likely underperform because sigmoid functions can suffer from vanishing gradients during optimization, while BNNLS benefits from robust least-squares training and uses a simple, effective clipping step to enforce valid outputs. Binary matrix factorization trades model expressiveness for interpretability (binary = "feature present/absent"). For rating prediction with continuous scales, this trade-off hurts performance significantly.

6 Matrix Factorization With Fused HIN Embedding

We adopt the approach of [3], which presents a computationally efficient method for incorporating side information into matrix factorization. The authors propose HERec, a novel framework that leverages HINs to enhance collaborative filtering. HERec operates in two stages: first, it extracts latent representations from the HIN structure using a dedicated embedding method; second, it incorporates these learned embeddings into an extended matrix factorization model for rating prediction.

In this section, we present each of these two steps.

6.1 Heterogeneous Network Embeddings

The first step constructs a HIN to extract meaningful representations from the data. Using the metadata described in Section 2, we build a graph with users, movies, release years, and genres as nodes, connected by edges representing user ratings, movie release dates, and genre associations, as illustrated in Figure 2.

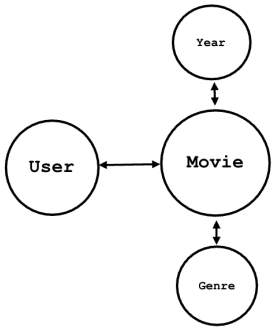

Figure 2: Network schemas of heterogeneous information network

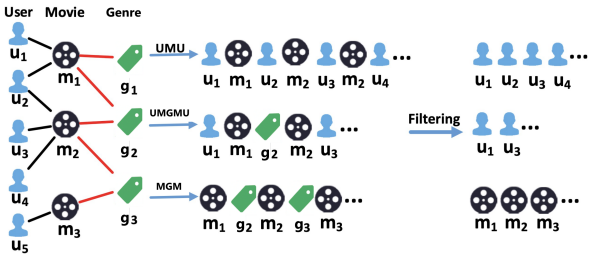

Figure 3: An illustrative example of the proposed meta-path based random walk. We first perform random walks guided by selected meta-paths, and then filter the node sequences to retain only those ending with user or item nodes.

6.1.1 Meta-Path Based Random Walk

The paper presents the meta-path random walk method with mathematical rigor, but the underlying intuition is straightforward. To generate meaningful node sequences, we design paths that follow specific meta-path schemas—templates that describe composite relations among different node types. A meta-path guides the random walk by determining not only which nodes can be visited next, but also whether such transitions are semantically valid.

A meta-path defines a sequence of node types, such as User–Movie–User (UMU) or User–Movie–Year–Movie–User (UMYMU), each encoding a distinct semantic relation. For example, the UMU path represents "users who have watched the same movies," while UMYMU captures "users connected through movies released in the same year." By constraining random walks to follow these structured patterns, the approach captures higher-order, semantically meaningful relationships beyond simple graph connectivity.

During the walk, the next node is sampled uniformly from neighbors matching the target type specified by the meta-path. The walker iteratively follows this pattern until reaching a predefined sequence length.

6.1.2 Type Constraint and Filtering

Since our goal is to generate embeddings specifically for movies and users, we constrain meta-paths to begin and end with these node types. After completing each meta-path traversal, we filter the resulting sequence to retain only nodes matching the starting node's type, as illustrated in Figure 3. This type-filtering strategy offers two key advantages.

First, although node sequences are constructed via meta-paths traversing heterogeneous node types, the final embeddings are learned from homogeneous neighborhoods. By embedding nodes of the same type in a unified space, we avoid the challenging task of representing all heterogeneous objects in a single shared space.

Second, within a fixed-length context window, a node can leverage more same-type neighbors, which are more likely to be relevant for the prediction task than neighbors of different types. This filtering thus improves both the computational efficiency and the semantic quality of the learned representations.

6.1.3 Learning The Embeddings

Given a meta-path, we construct the neighborhood $\mathcal{N}_u$ for node u based on co-occurrence in a fixed-length window. Using node2vec from [1], we can learn the representations of nodes to optimize the following objective:

$$\max_f \sum_u \log Pr(\mathcal{N}_u | f(u)), \tag{5}$$

where f is a function (that we aim to learn) mapping each node to a d-dimensional feature space, and $\mathcal{N}_u \subset \mathcal{V}$ represents the neighborhood of node u, with respect to a specific meta-path. We can learn the embedding mapping function $f(\cdot)$ by optimizing the objective using stochastic gradient descent (SGD).

The complete algorithmic framework is presented in the following algorithm.

6.2 Integrating Matrix Factorization with Fused HIN Embeddings

6.2.1 Fused HIN Embedding

Existing approaches often employ linear weighting mechanisms to combine information mined from HINs (e.g., meta-path-based similarities). While simple and interpretable, such linear fusion methods may fail to capture complex, non-linear relationships among different patterns, limiting their effectiveness for recommendation.

Following [3], we adopt a **personalized non-linear fusion function** $g(\cdot)$ to integrate multiple meta-path-based embeddings into a unified representation for each user and item. This design enables the model to automatically learn user-specific preferences across different meta-paths, yielding more expressive and adaptive embeddings.

Formally, for a given user u, the fused HIN embedding is defined as:

$$g(\{\mathbf{e}_u^{(l)}\}) = \sigma\left(\sum_{l=1}^{|\mathcal{P}|} w_u^{(l)} \left(\mathbf{M}^{(l)}\mathbf{e}_u^{(l)} + \mathbf{b}^{(l)}\right)\right), \tag{6}$$

where $\mathbf{e}_u^{(l)} \in \mathbb{R}^d$ denotes the embedding of user u derived from the l-th meta-path in $\mathcal{P}$, $w_u^{(l)} \in \mathbb{R}$ represents the personalized weight for that meta-path, $\mathbf{M}^{(l)} \in \mathbb{R}^{D \times d}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^D$ are transformation parameters, and $\sigma(\cdot)$ is a non-linear activation function (e.g., sigmoid). The resulting fused embedding $\mathbf{e}_u^{(U)} = g(\{\mathbf{e}_u^{(l)}\}) \in \mathbb{R}^D$ captures user-specific semantics across multiple meta-paths.

The same fusion function is applied to items:

$$\mathbf{e}_i^{(I)} = g(\{\mathbf{e}_i^{(l)}\}), \quad \text{where } \mathbf{e}_i^{(l)} \in \mathbb{R}^d \text{ and } \mathbf{e}_i^{(I)} \in \mathbb{R}^D. \quad (7)$$

These fused embeddings are then incorporated into the matrix factorization framework for rating prediction:

$$\hat{r}_{ui} = \mathbf{x}_u^\top \mathbf{y}_i + \alpha \, (\mathbf{e}_u^{(U)})^\top \boldsymbol{\gamma}_i^{(I)} + \beta \, (\boldsymbol{\gamma}_u^{(U)})^\top \mathbf{e}_i^{(I)}, \quad (8)$$

where $\mathbf{x}_u, \mathbf{y}_i \in \mathbb{R}^D$ are standard latent factors, $\boldsymbol{\gamma}_u^{(U)}, \boldsymbol{\gamma}_i^{(I)} \in \mathbb{R}^D$ are projection vectors, and α, β are scalar coefficients that balance the contribution of HIN-derived information against traditional collaborative filtering.

The complete framework is summarized in Algorithm 1 in Section A of the Appendix.

6.2.2 Learning Objective

Having defined the personalized non-linear fusion function and the rating prediction model, we now describe the optimization framework. Given the observed user-item interaction set $\mathcal{R}$, the model minimizes the discrepancy between true and predicted ratings while regularizing all latent parameters to prevent overfitting.

The overall objective function is:

$$\begin{aligned} \mathcal{L} = & \sum_{(u,i,r_{u,i}) \in \mathcal{R}} (r_{u,i} - \hat{r}_{u,i})^2 \\ & + \lambda \sum_{u,i} \Big(\|\mathbf{x}_u\|_2^2 + \|\mathbf{y}_i\|_2^2 + \|\boldsymbol{\gamma}_u^{(U)}\|_2^2 + \|\boldsymbol{\gamma}_i^{(I)}\|_2^2 \quad (9) \\ & + \|\boldsymbol{\Theta}^{(U)}\|_2^2 + \|\boldsymbol{\Theta}^{(I)}\|_2^2 \Big), \end{aligned}$$

where $\hat{r}_{u,i}$ is computed as in Equation 8, λ controls regularization strength, and $\boldsymbol{\Theta}^{(U)}$, $\boldsymbol{\Theta}^{(I)}$ denote the fusion function parameters for users and items respectively (i.e., $\{w_u^{(l)}, \mathbf{M}^{(l)}, \mathbf{b}^{(l)}\}$).

The first term measures prediction error on observed ratings, while the second term regularizes all learned parameters. We optimize this objective using SGD. The gradient computations follow standard backpropagation and are omitted for brevity; full derivations are provided in the original work [3].

6.3 Hyperparameter Tuning

6.3.1 Embeddings learning

To first generate the node sequences to embed, we kept the values of the paper for the walk lenght , $wl = 40$, the *fixed-lenght window size* to 5 and the *walks per node*, 10, as we had little time to question the authors choices. We also kept the embeddings dimension given by the paper, d= 64

This will also be adressed in the Future Work section but also due to lack of time we choose to tune the hyperparameter on only a single simple meta-path MUM and used the hyperparameter of node2vec to embed the other meta-paths. We choose to set the *negative samples* to 5, the *lr* to 0.001 over 20 *epochs*. Since we had a problem with using an already existing implementation of node2vec, we implemented our own using *Pytorch Lighting* which allowed us to monitor that the training loss had plateau with *tensorboard*.

We only select short meta-paths of at most four steps, since long meta-paths are likely to introduce noisy semantics.

Meta-Paths
MYM,MUM,UMU,UMYMU,UMGMU

Table 1: The Meta-Paths selected in our work

We also tried MGMYM but noticed it was adding noise, we also made the design choice to have more user based embeddings as we have no information on them.

6.3.2 HERec hyperparameter tuning

We tuned our hyperparameters using first a random search followed by a grid search on :

- Latent dimension $k \in \{5, 10, 20, 30\}$
- user coefficient $\alpha \in \{0.1, 0.5, 1, 5\}$
- item coefficient $\beta \in \{0.1, 0.5, 1, 1.5, 5\}$

that yielded, has verified in Figure 4, $\alpha = 1$ and $\beta = 1.5$. The SGD is then performed over 60 epochs with a learning rate of 0.001.

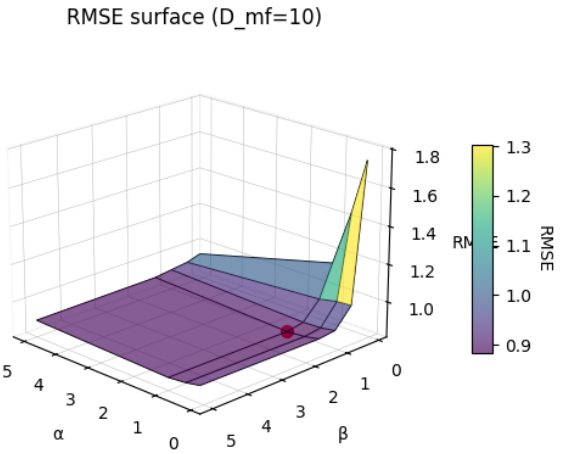

Figure 4: α and β variation for a fixed latent factor space dimension (here 10)

7 Results

In this section, we present the results obtained with both UserReg and HERec models and compare them to the BNNLS, that serves as a robust baseline. We use three key metrics for comparison, namely the RMSE, the accuracy, and the computing time.

Cross-Validation Strategy. Following standard evaluation practices for collaborative filtering systems [2], we employ a stratified per-user holdout validation approach to evaluate model performance. For each user, we randomly partition their rated items into training and test sets while maintaining the specified training ratio.

This stratification ensures that each user contributes proportionally to both sets, avoiding cold-start scenarios in testing and providing a realistic evaluation of the models' ability to predict preferences for existing users. We test four training ratios (20%, 40%, 60%, 80%) and repeat each experiment with three random seeds to assess result stability. All models are trained for 50 iterations, and model performance is measured using RMSE on the held-out test ratings, with final results reported as mean ± standard deviation across seeds. The cross-validation results are presented in Figure 5.

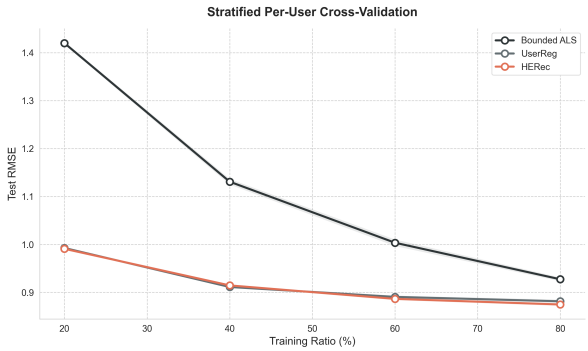

Figure 5: Cross-validation performance comparison across training ratios. HERec and UserReg consistently outperform Bounded ALS across all training ratios, with HERec achieving the lowest RMSE of 0.874 at 80% training data. Error bands represent ±1 standard deviation across seeds.

Performance results using the best hyperparameters per model are reported in Table 2.

Model	RMSE	Accuracy (%)	Time (s)
BNNLS	0.9146	24.14	11.36
UserReg			
HERec	0.8740	24.74	–

Table 2: Model Performance Comparison

7.1 Embedding Visualization

The year-based (MYM) and genre-based (MGM) embeddings show clear clustering by temporal and semantic similarity, confirming that meta-paths capture meaningful movie relations.

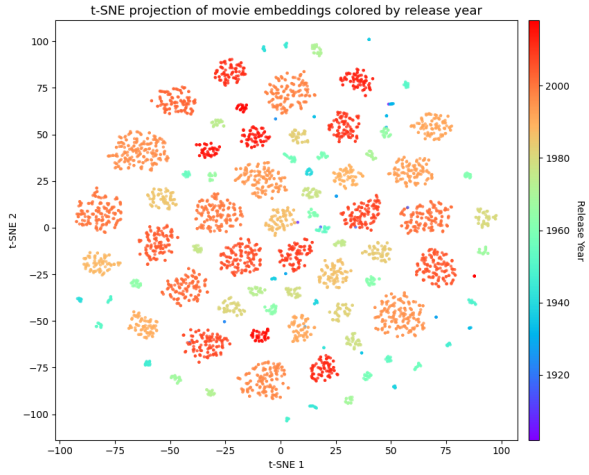

Figure 6

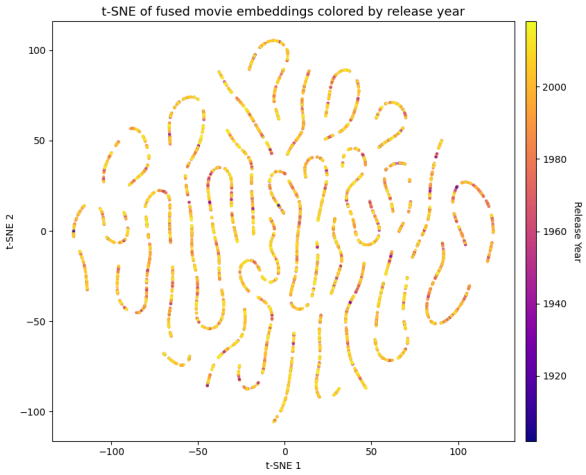

Figure 7

After fusion, both item and user embeddings form smoother, continuous manifolds, indicating successful integration of heterogeneous information into unified latent spaces suitable for recommendation.

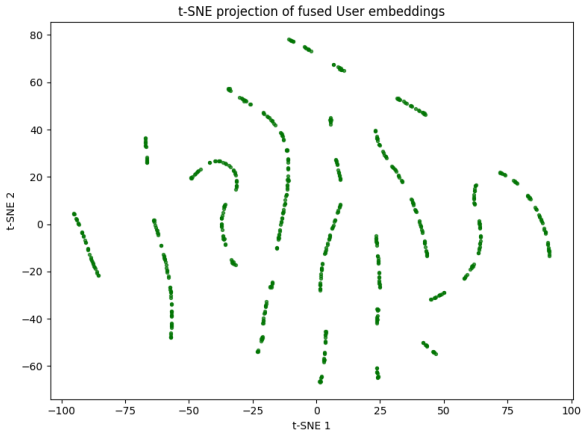

Figure 8

8 Conclusion

This work systematically evaluates matrix factorization approaches for collaborative filtering, progressing from vanilla ALS to constrained variants (ANNLS, BNNLS), the user consistency-based UserReg framework, and finally HERec, which incorporates side information through heterogeneous information networks. Our results demonstrate that well-designed MF models can achieve competitive performance with significantly lower computational complexity than deep learning alternatives. While UserReg provides efficiency gains through principled regularization, HERec shows that leveraging structured metadata via meta-path-based embeddings yields further improvements at moderate computational cost.

Future work could explore hybrid approaches combining MF efficiency with selective side information integration, as well as alternative meta-path schemas for different recommendation domains.

References

[1] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754. URL: https://doi.org/10.1145/2939672.2939754.

[2] Jonathan L. Herlocker et al. "Evaluating Collaborative Filtering Recommender Systems". In: *ACM Transactions on Information Systems* 22.1 (2004), pp. 5–53. DOI: 10.1145/963770.963772.

[3] Chuan Shi et al. *Heterogeneous Information Network Embedding for Recommendation*. 2017. DOI: 10.48550/arXiv.1711.10730. URL: https://arxiv.org/abs/1711.10730.

[4] Yukino Terui et al. "Collaborative Filtering Based on Nonnegative/Binary Matrix Factorization". In: *Frontiers in Big Data* 8 (2025). DOI: 10.3389/fdata.2025.1599704. URL: https://doi.org/10.3389/fdata.2025.1599704.

[5] Haiyang Zhang et al. *UserReg: A Simple but Strong Model for Rating Prediction*. 2021. DOI: 10.48550/arXiv.2102.07601. URL: https://arxiv.org/abs/2102.07601.

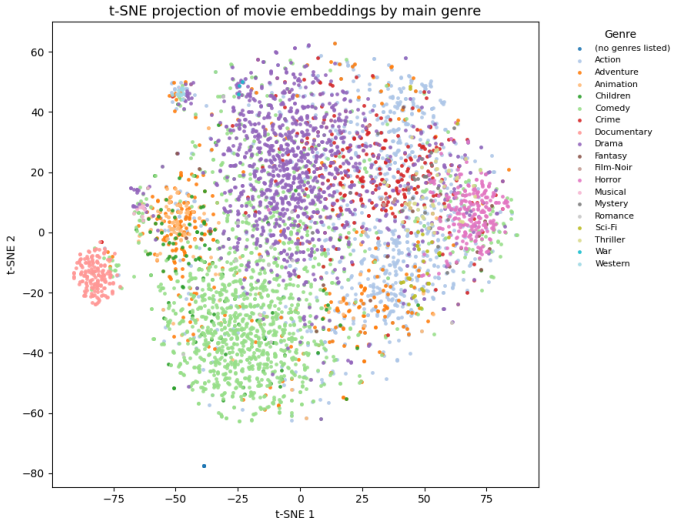

Figure 9: Enter Caption

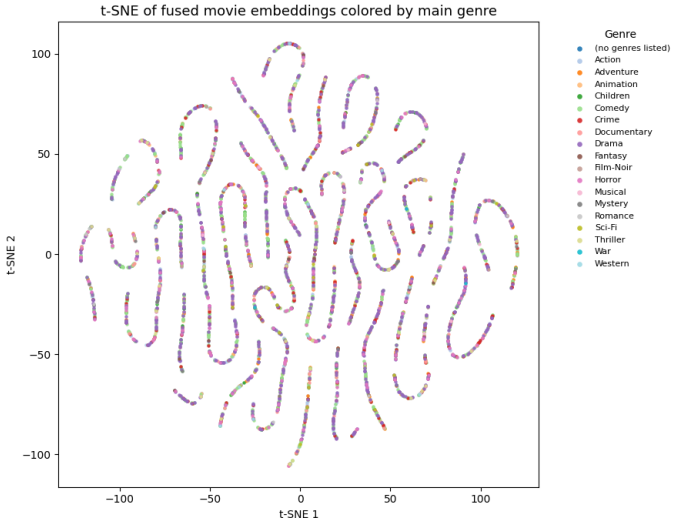

Figure 10: Enter Caption

A Algorithm Details

Algorithm 1 HERec: Heterogeneous Information Network Embedding for Recommendation

Require: Rating matrix $\mathbf{R}$, HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, meta-path set $\mathcal{P}$, embedding dimension d
Ensure: Predicted ratings $\hat{\mathbf{R}}$

Phase 1: Meta-path Based Random Walk
for each meta-path $p \in \mathcal{P}$ **do**
  for each node $v \in \mathcal{V}$ **do**
    Generate random walks from v following p
    Filter sequences to retain only v's node type
  end for
end for
Phase 2: Node Embedding
for each meta-path $p \in \mathcal{P}$ **do**
  Train skip-gram on filtered sequences $\rightarrow \{\mathbf{e}_v^{(p)}\}$
end for
Phase 3: Non-linear Fusion
Compute $\mathbf{e}_u^{(U)} = g(\{\mathbf{e}_u^{(l)}\})$ for all users
Compute $\mathbf{e}_i^{(I)} = g(\{\mathbf{e}_i^{(l)}\})$ for all items
Phase 4: Extended Matrix Factorization
Initialize latent factors $\mathbf{x}_u, \mathbf{y}_i, \boldsymbol{\gamma}_u^{(U)}, \boldsymbol{\gamma}_i^{(I)}$
repeat
  for each rating $r_{ui} \in \mathbf{R}$ **do**
    $\hat{r}_{ui} \leftarrow \mathbf{x}_u^\top \mathbf{y}_i + \alpha(\mathbf{e}_u^{(U)})^\top \boldsymbol{\gamma}_i^{(I)} + \beta(\boldsymbol{\gamma}_u^{(U)})^\top \mathbf{e}_i^{(I)}$
    Update parameters via gradient descent
  end for
until convergence
return $\hat{\mathbf{R}}$