Movie Recommendation Systems: An Empirical Comparison of Collaborative, Matrix Factorisation, and Hybrid Methods

Tan Yongjun, Kuan Kai Xuan Keefe, Sui Wei Xian Ryan, Darrell Goh Rui Jie

Abstract

Recommender systems play a crucial role in guiding user engagement on content platforms. This paper explores various recommendation approaches using the MovieLens 1M dataset enriched with DBpedia metadata. We implement and compare a popularity baseline, user-based and item-based collaborative filtering, matrix factorisation with implicit Alternating Least Squares (ALS), and a hybrid LightFM model. Models are evaluated using Precision@K, Recall@K, and NDCG@K. Our findings demonstrate the value of personalised approaches over non-personalised baselines, and highlight the trade-offs between pure collaborative, matrix factorisation, and hybrid techniques, particularly regarding cold-start handling and diversity. Code and data can be found here: https://github.com/Keefekkx/Movie-Recommender

1. Introduction

1.1 Background

Recommender systems have become a vital component of many online platforms, helping users discover relevant content in the face of information overload. In the movie domain, recommendation engines drive user engagement on streaming services and e-commerce sites by personalizing content to individual tastes. Collaborative filtering (CF) and content-based filtering are the two classic approaches to recommendation. Collaborative filtering leverages patterns of user behavior, operating on the principle that users with similar past preferences will enjoy similar items. In contrast, content-based filtering relies on item features, recommending items that are similar in content (e.g. genre, actors, etc.) to those a user liked before. Each approach has its limitations: collaborative methods struggle with new users or items (the cold-start problem) since they have no prior interactions, while content-based methods can only recommend items that share obvious attributes with a user's history and may not capture subtle taste similarities. To address these limitations, hybrid recommender systems combine collaborative and content-based techniques. In fact, many production systems (such as Netflix's prize-winning algorithm) use a hybrid approach to achieve both accuracy and novelty in recommendations.

1.2 Objective

This project is an exploratory study of different recommendation algorithms, with an emphasis on understanding their mechanics and evaluating their performance on a real-world dataset. We implement five models of increasing complexity: (1) a Popularity baseline that recommends top-rated movies to everyone, (2) User-based CF and (3) Item-based CF using neighborhood similarity, (4) an implicit feedback ALS matrix factorization model, and (5) a LightFM hybrid model that incorporates both collaborative signals and content features. We evaluate these models on a held-out test set using ranking metrics that reflect the quality of the top recommendations. Rather than trying to crown a single "best" model, we analyze the pros and cons of each, considering factors like accuracy, coverage, personalization, and real-world deployment considerations. Through this comparative analysis, we aim to derive insights into how recommendation algorithms work and how they can be applied or combined in practical applications.

2. Dataset and Data Preprocessing

2.1 Data Overview

We use the MovieLens 1M dataset, which consists of 1,000,209 anonymous ratings (on a 5-star scale) from 6,040 users for 3,883 movies. Each user has rated at least 20 movies, and each rating is accompanied by a timestamp. The dataset also provides basic user attributes (age group, gender, occupation, and zipcode) and movie attributes (title and genres). The user rating matrix is very sparse – out of the ~22 million possible user-movie combinations, only about 4.5% are observed as ratings, meaning over 95% of the matrix is empty. This level of sparsity is typical in recommender system data and reflects the reality that each user only interacts with a tiny subset of all items.

To enable content-based and hybrid recommendations, we augmented the MovieLens data with additional movie metadata from DBpedia. DBpedia is a project that extracts structured information from Wikipedia. For each movie in the dataset, we used a pre-existing mapping from MovieLens to DBpedia to retrieve attributes such as the movie's director(s), top cast (actors), country of origin, and original language. These attributes provide a richer description of each movie. We collected such information via SPARQL queries to the DBpedia knowledge base and merged it with the MovieLens movie list. This resulted in a consolidated movies metadata table, where each movie is described not only by its title and genre (from MovieLens) but also by a set of content features from DBpedia. These features are later used in the LightFM hybrid model to represent items.

2.2 Positive Feedback & Cold-Start Filtering

In recommender systems, clearly distinguishing positive user preferences from negative or neutral ones is essential for accurately capturing tastes and providing meaningful recommendations. In this project, ratings of 4 and 5 stars were explicitly defined as positive interactions, reflecting strong user preferences and genuine satisfaction. Ratings below 4 stars (1–3 stars) were excluded, as they typically indicate neutral or negative sentiment—suggesting weaker enthusiasm, ambiguity, or explicit dissatisfaction. By isolating these high-confidence positive ratings, we enable the recommender models to learn effectively from interactions where the user's interest is clearly signaled. Additionally, each positive interaction was assigned a confidence weight: interactions rated 5 stars received a weight of 1.0, indicating strong certainty of preference, while 4-star interactions received a slightly lower weight of 0.5. These weighted interactions form critical inputs for the subsequent modeling stages.

Furthermore, we addressed the cold-start problem, which often affects the reliability of collaborative filtering algorithms due to insufficient interaction data for new or inactive users and unpopular items. Specifically, we removed users and items with very few positive interactions—the bottom 10% of least active users and least popular movies (based on the count of positive ratings). By filtering out these sparse interactions, we improved data quality and ensured that our recommendation algorithms had adequate data points to generate reliable predictions. After this cold-start filtering process, our dataset contained 5,390 users and 3,125 movies, encompassing 565,817 high-confidence ratings. This preprocessing step provided a robust foundation for meaningful comparative analysis across different recommender models.

2.3 Train-Test Split

To rigorously evaluate our recommender models, we partitioned the data into distinct training and test sets. Specifically, we performed an 80/20 chronological split per user to closely simulate real-world recommendation scenarios and prevent temporal leakage. For each user, interactions (ratings) were sorted by timestamp, and the most recent 20% were designated as the test set, while the earlier 80% formed the training set. This approach mimics the practical task of recommending new movies based on previously observed interactions, thereby ensuring the models are evaluated on their ability to predict future user preferences. After applying this splitting strategy, our final training set contained 450,543 interactions, and the test set comprised 115,274 interactions. All interactions in the test set were treated as relevant items, representing movies that users actively chose and rated in the future. During evaluation, a recommendation was considered successful if it retrieved a movie from the user's test set, aligning with standard practice in implicit feedback settings where observed future interactions are assumed to reflect user preferences.

2.4 Feature Engineering for Hybrid Modeling

To construct feature representations for hybrid recommendation modeling, we engineered item features and user features based on available metadata. For item features, we utilized attributes such as genres, actors, languages, and countries from the processed movie metadata. Each categorical field was multi-hot encoded, while movie titles were transformed into TF-IDF vectors (limited to the top 500 terms) to capture textual information. Additionally, release years were discretized into decade bins and encoded using one-hot encoding to model temporal characteristics. The resulting item feature matrix was a sparse concatenation of all encoded components.

Similarly, for user features, we leveraged demographic information from the MovieLens dataset, including gender and occupation, both of which were one-hot encoded. Age was grouped into discrete bins (e.g., 18–25, 25–35, etc.) and likewise one-hot encoded to handle age as a categorical feature. Both the item and user feature matrices were sparsely constructed to ensure memory efficiency and compatibility with hybrid models like LightFM, which can exploit these rich side features during training.

3. Exploratory Data Analysis (EDA)

Before building models, we conducted an exploratory data analysis (EDA) to understand the characteristics of

the MovieLens 1M dataset and the enriched metadata from DBpedia. This analysis provided essential context for the challenges a recommender system must address.

3.1 Rating Distribution

The distribution of movie ratings is positively skewed (Figure 1). Users are much more likely to give favourable ratings: 4-star ratings are the most frequent, followed by 3-star and 5-star ratings. The mean rating across the dataset is approximately 3.5 out of 5. This strong positive bias suggests that many movies in the dataset are regarded favourably, and that a random selection of high-rated movies might still yield moderate user satisfaction. However, this also introduces risk: models optimising purely for predicting high ratings may not necessarily offer meaningful personalisation.

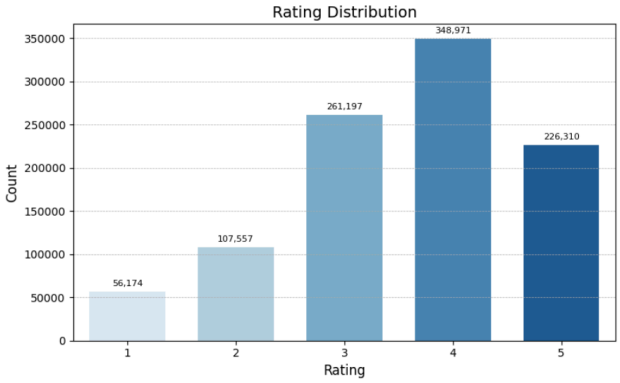

Figure 1: Rating distribution showing counts of each rating from 1 to 5.

Additionally, because there are significantly more positive than negative ratings, any supervised model training must account for this imbalance to avoid trivial solutions (e.g., always predicting 4 stars).

3.2 User-Item Interaction Sparsity

The user-item interaction matrix is extremely sparse. Only around 3.3% of all possible user-movie pairs contain a positive rating (≥4 stars).

To better visualise engagement patterns, we plot the distributions of rating counts per user and per item on a log scale (Figure 2). Most users have rated relatively few movies, while a few highly active users have rated hundreds. Similarly, most movies are rated by only a handful of users, whereas blockbuster films accumulate thousands of ratings. This long-tail behaviour is a fundamental characteristic of recommendation datasets.

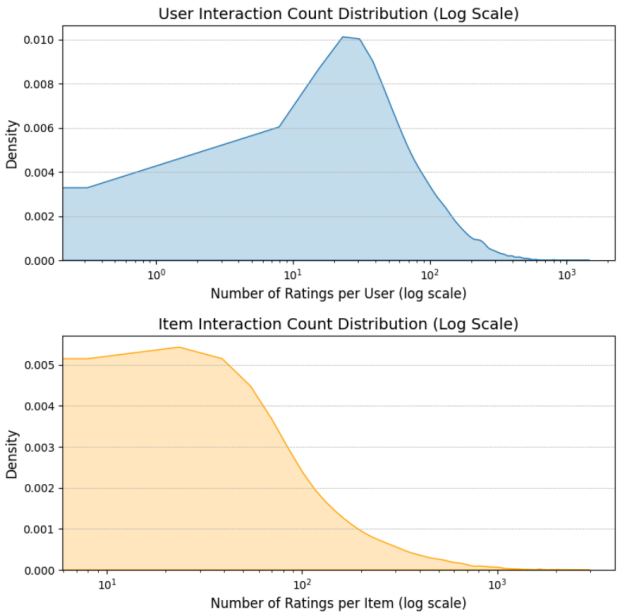

Figure 2: (Top) Density distribution of number of ratings per user

(Bottom) Density distribution of number of ratings per item, both shown in log-log scale

These patterns indicate that methods must generalise from sparse data. Popularity-based methods risk focusing only on a small subset of blockbuster movies, while collaborative filtering models must infer user preferences from limited interactions.

3.3 Genre Distribution

We analysed the distribution of genres across movies (Figure 3). Drama, Comedy, and Action dominate the dataset, with Drama appearing in more than 1,600 movies. Less frequent genres include Adventure, Sci-Fi, and Crime.

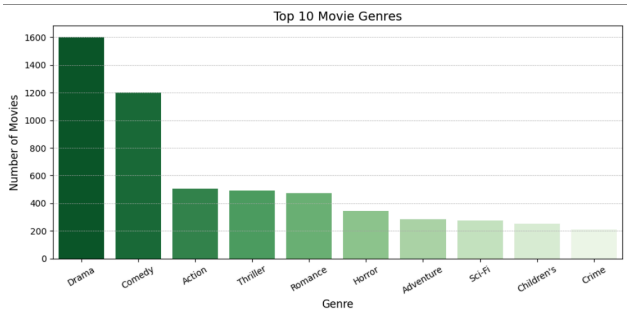

Figure 3: Top 10 genres by number of movies.

The genre imbalance implies that content-based recommendation approaches might unintentionally favour

more common genres unless corrective mechanisms are applied.

3.4 Top Directors and Actors

We explored metadata on directors and actors. Alfred Hitchcock, Woody Allen, and Steven Spielberg are among the most prolific directors in the dataset (Figure 4). For actors, frequent appearances include ensembles such as those involved in Star Wars and Monty Python films (Figure 5).

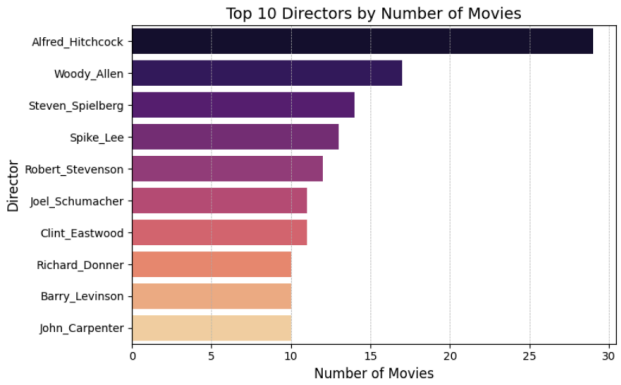

Figure 4: Top 10 directors by number of unique movies directed.

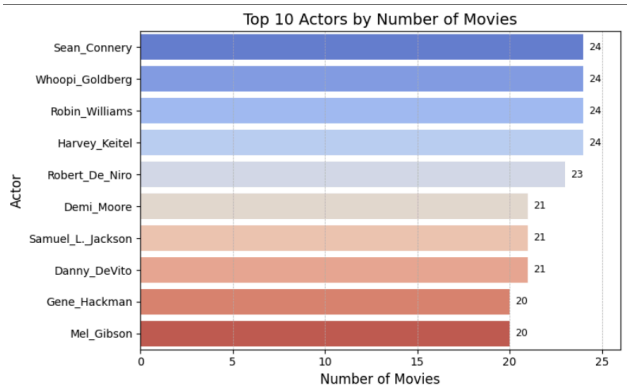

Figure 5: Top 10 actors by number of unique movies acted in.

Incorporating information about directors and actors can enhance content-based and hybrid models by capturing implicit user preferences for certain filmmakers or performers.

3.5 Genre Co-occurrence

To understand how different genres are related within the MovieLens 1M dataset, we constructed a genre co-occurrence matrix. Each movie may belong to multiple genres (e.g., Action and Sci-Fi), and we analysed the frequency with which genres appear together.

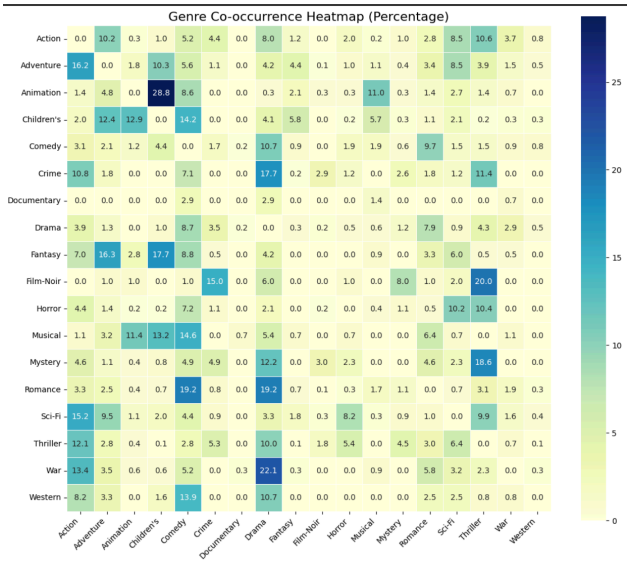

Figure 6: Genre Co-occurrence

Figure 6 shows the resulting heatmap. Each row of the heatmap is normalised relative to the total number of movies in the row's genre. Therefore, the percentages represent, for a given originating genre, the proportion of its movies that also belong to each column genre. Brighter colours indicate stronger associations between genres. We observe several notable patterns:

Action movies frequently co-occur with Science Fiction (15.2%), and also with Adventure (10.2%), suggesting that Action/Sci-Fi-Adventure is a common thematic cluster prevalent in blockbuster films.

Comedy and Romance genres often appear together, with Comedy co-occurring with Romance at 10.7%, and Romance reciprocating with Comedy at 19.2%, reflecting the prevalence of romantic comedies.

Drama shows moderate co-occurrence with Thriller (10.0%) and Crime (8.7%), consistent with narratives that blend emotional depth with suspenseful or criminal elements.

We zeroed the self-co-occurrence diagonals to focus attention on cross-genre relationships. These observations are important for content-based and hybrid recommender systems, where understanding genre correlations can enhance the quality and diversity of recommendations.

4. Methodology

We implemented five recommendation algorithms, each representing a different approach: a popularity-based non-personalized model, two memory-based collaborative filtering models (user-user and item-item), a model-based collaborative filtering technique (ALS matrix factorization), and a hybrid collaborative-content model (LightFM). In this section, we describe each method, explaining how it works and what assumptions it makes about user behavior.

4.1 Popularity Baseline

The Popularity baseline recommender serves as a straightforward evaluation benchmark. It suggests the same set of top-N movies to all users, ranking movies by the number of 4- or 5-star ratings received in the training data. This approach captures overall user preference, recommending widely liked films such as *Star Wars*, *The Godfather*, or *Titanic* to any user who has not already rated them. Although simplistic, popularity-based recommendations often perform reasonably well, as popular content tends to appeal broadly. The model is immune to the user cold-start problem, making it immediately applicable even for new users with no prior interactions. However, it lacks personalization, failing to serve users with niche interests, and tends to reduce recommendation diversity by amplifying already popular items. We include the popularity baseline in our evaluation to demonstrate the performance gains achieved by more sophisticated, personalized algorithms Nonetheless, popularity-based strategies often remain a critical fallback component in hybrid recommendation systems, especially for handling new users.

4.2 User-Based Collaborative Filtering (UserCF)

UserCF is a classic memory-based recommendation approach that generates suggestions for a user based on the preferences of "similar" users. The core idea is that like-minded users tend to enjoy similar items. In our implementation, we constructed a user–user similarity matrix from the training data, where each user was represented by a vector of positively interacted movies (rated 4 or 5 stars). Pairwise similarities between users were computed using cosine similarity, weighted according to the user's confidence (assigning higher weight to 5-star ratings). For a given target user, we identified the top-K most similar users (K = 10) and aggregated their preferences to generate recommendations. Specifically, candidate movies were scored by summing the neighbors' weighted interactions, adjusted by similarity to the target user. Movies highly liked by similar users but not yet rated by the target user received higher scores and were ranked accordingly.

This method operates under the assumption that users with similar historical preferences will continue to like similar content. For instance, if two users share a love for classic sci-fi films (Star Wars, Alien), and one user has enjoyed Blade Runner, the model would recommend Blade Runner to the other user even without direct knowledge of the movie's content. While user-based CF offers personalization and transparency, as recommendations are easily explained through similar users, it faces scalability challenges as the user base grows. Similarity calculations can also be noisy when users have rated only a few items, potentially leading to unreliable recommendations. We mitigated this by applying cold-start filtering and selecting a reasonably large neighborhood size (K=10). Overall, user-based CF is expected to significantly outperform a popularity-based baseline, particularly for users with niche or distinctive preferences.

4.3 Item-Based Collaborative Filtering (ItemCF)

ItemCF inverts the user-based approach by focusing on similarities between items rather than users. Popularized by systems like Amazon's "customers who bought this item also bought…" feature, the method assumes that items liked by similar groups of users are likely to be similar. Specifically, we computed pairwise cosine similarities between movies, treating each movie as a vector of users who positively rated it. For a target user, recommendations are generated by identifying movies similar to those the user has previously liked. Each candidate movie is scored by aggregating its similarity to the user's liked movies—using a weighted sum of similarities—and the top-N candidates are ranked accordingly. For example, if a user enjoyed Toy Story and Finding Nemo, the model would recommend The Lion King if it shares high similarity with those favorites.

Item-based CF operates under the assumption that user preferences are stable across similar items. It offers practical advantages: the item-item similarity matrix can be precomputed offline, and item similarities tend to be more statistically robust than user similarities, as popular items attract ratings from many users. Consequently, ItemCF is often more scalable and stable than user-based CF, especially in larger systems. In our dataset, where both user and item counts are moderate, we expect item-based CF to slightly outperform user-based CF, consistent with literature findings. Moreover, it handles the user cold-start problem relatively well (as long as the user has rated at least one movie), though it still suffers from item cold-start limitations, where new movies with no ratings cannot be recommended until they accumulate interactions.

4.4 ALS Matrix Factorisation

For a more model-driven approach, we implemented Alternating Least Squares (ALS) matrix factorization for implicit feedback. Matrix factorization is a model-based collaborative filtering method that embeds both users and items in a low-dimensional latent space. The idea is that each user is represented by a vector of latent factors and each item by a vector of latent factors, such that the dot product of a user's vector and an item's vector predicts the user's preference for that item. In our case, we have implicit feedback (user likes/dislikes), so we use the approach of Hu, Koren, and Volinsky (2008) which adapts ALS to implicit data with confidence weights.

We constructed a binary preference matrix P, where $P[u, i] = 1$ if user u positively interacted with item i, and 0 otherwise. A corresponding confidence matrix C was also built, assigning higher weights to interactions with greater certainty (scaled by rating confidence). ALS minimizes a weighted regularized loss function, alternating between solving for user and item latent vectors. Intuitively, it learns user and item embeddings such that their dot product approximates the observed interaction strength. We used an existing library implementation of ALS (from the *implicit* Python library) and set the number of latent factors to 64 with regularization to prevent overfitting, and ran for 30 iterations.

After training, recommendations are generated by ranking the dot product scores between a user's latent vector and all unseen item vectors. ALS is particularly effective at capturing subtle, latent relationships that might not be visible from direct co-occurrence, enabling it to generalize well beyond exact neighbor matches. The advantages of ALS include its ability to scale to large datasets through parallelized updates and its capacity to infer missing preferences by exploiting latent structure. However, its limitations include behaving as a black-box model, making interpretation challenging and it also struggles with cold-start users or items, since factorization requires observed interactions to learn meaningful embeddings.

4.5 LightFM Hybrid Model

LightFM is a hybrid recommendation model that combines collaborative filtering with content-based information by jointly learning embeddings for users, items, and their associated features. Unlike pure collaborative models that rely solely on interaction patterns, LightFM incorporates metadata (such as genres, actors, directors, country, and language for movies, and demographics for users) to enhance recommendations, particularly in cold-start scenarios.

In our implementation, each user was represented by their ID and features such as gender, age group, and occupation. Each movie was represented by its ID along with enriched features retrieved from DBpedia, including genres, director, actors, country, and language. LightFM learns latent vectors for users, items, and features simultaneously, and predicts the affinity between a user and an item as the sum of their interactions in the latent space. The model was trained using the WARP (Weighted Approximate-Rank Pairwise) loss, which directly optimizes ranking quality by prioritizing correct ordering of positive over negative examples. We also trained the LightFM model for 10 epochs with 128 latent components (factors) and a moderate learning rate.

This hybrid approach offers several advantages. By leveraging content features, LightFM can alleviate cold-start issues for new users and items and increase recommendation diversity beyond what pure collaborative signals allow. It can recommend new movies based on shared metadata even if they have few or no historical ratings. However, the inclusion of many features can also introduce noise if irrelevant attributes dilute the strength of collaborative signals. Moreover, LightFM's effectiveness heavily depends on the quality and informativeness of the feature engineering.

5. Results & Discussion

5.1 Evaluation Metrics

To comprehensively assess the performance of our recommendation models, we computed multiple ranking-based metrics. These metrics evaluate both the ability of the models to retrieve relevant items and the quality of the ranking order. Their definitions are summarised in Table 1 below.

Table 1. Definitions of different evaluation metrics

Metric	Definition
Precision@K	Measures the proportion of recommended items in the top-K list that are relevant. It emphasizes accuracy at the top of the recommendation list, which is critical for user engagement in real-world applications.
Recall@K	Calculates the proportion of a user's relevant items that are successfully retrieved in the top-K recommendations. Recall is important for evaluating the

	comprehensiveness of a model's recommendations.
NDCG@K	Assesses not only whether relevant items are recommended but also their ranking positions, with higher relevance assigned to items appearing earlier in the list. NDCG balances both ranking quality and relevance retrieval.
Hit Rate@K	Measures the fraction of users for whom at least one relevant item appears in the top-K recommendations. It provides a simple, binary measure of success but does not account for how many relevant items were retrieved or their ranking.
Average Precision @K	Computes the mean of the precision scores at the ranks where relevant items are found. It rewards models that retrieve relevant items earlier and more consistently across the ranked list. However, since it is highly correlated with NDCG and less interpretable for a general audience, we treat it as a secondary metric.

Given the wide range of evaluation criteria, we categorize them into primary and secondary metrics based on their importance for our analysis. We designate precision, recall and NDCG as primary because they jointly capture the key dimensions of recommendation system quality: accuracy, coverage, and ranking effectiveness. Hit rate and average precision will be used as secondary metrics to offer supplementary insights.

For all metrics, we computed results at $K = 5, 10, 20$ to capture recommendation performance across different list lengths. However, for consistency and clarity in this report, we primarily present results at $K = 10$ as it strikes a balance between short-list precision and broad coverage. A top-10 recommendation list is also commonly used in real-world platforms, making it a practical and interpretable standard.

5.2 Model Performance

Table 1. Primary Metrics (Precision@10 and Recall@10 as percentages, NDCG@10 scaled 0-1)

Data Set	Precision (%)	Recall (%)	NDCG
Popularity	3.80	2.67	0.0410
User-based CF	9.40	6.29	0.1058
Item-based CF	9.81	6.40	0.1108
ALS (Implicit)	7.64	5.92	0.0867
LightFM Hybrid	2.95	1.74	0.0325

Table 2. Secondary Metrics (Hit Rate @10 and Average Precision@10 as percentages)

Data Set	Hit Rate (%)	Average Precision (%)
Popularity	26.72	1.57
User-based CF	50.74	4.79
Item-based CF	50.24	5.21
ALS (Implicit)	48.78	3.57
LightFM Hybrid	22.88	1.20

Several observations can be made from these results.

5.2.1 Popularity vs Personalized Methods

All three pure collaborative filtering models (UserCF, ItemCF, ALS) significantly outperformed the popularity baseline. This confirms that personalization adds substantial value: recommending movies tailored to a user's own history is much more effective than simply recommending the globally popular titles. The baseline, while recommending well-liked movies, often suggests films a user has already seen or is not interested in (perhaps due to genre preferences), hence the low precision. In contrast, the personalized methods filter out movies the user has rated and focus on ones that align with their profile, yielding more hits.

5.2.2 UserCF vs ItemCF

Item-based collaborative filtering slightly outperformed user-based collaborative filtering across all measured metrics. Both models used the same neighborhood size (K=10) and cosine similarity, but ItemCF had an edge likely because item similarities are more robust: popular movies accumulate richer user interaction data, while user profiles are often sparser. Additionally, ItemCF can recommend items similar to what a user has liked, even if their immediate neighbors have not rated them, offering a broader and more reliable candidate pool. In practice, while both methods often overlapped on popular

recommendations, ItemCF showed better ranking of niche items. For instance, in one case, UserCF recommended a widely known action movie, whereas ItemCF successfully surfaced a slightly less popular film closely aligned to the user's past preferences, which was present in their test set. This highlights ItemCF's strength in capturing fine-grained item associations.

5.2.3 ALS (Implicit MF) performance

The ALS matrix factorization model showed reasonable performance, outperforming the popularity baseline but falling slightly behind both user-based and item-based collaborative filtering on key metrics. At $K = 10$, ALS achieved a lower performance than the neighborhood-based methods but still substantially higher than the non-personalized baseline. Its NDCG@10 score of 0.0867 indicates that relevant items were ranked reasonably well, although not as highly prioritized as in UserCF or ItemCF. ALS performed particularly well in terms of Recall@20 (not shown here), suggesting its strength in retrieving a broader set of relevant items. This reflects the model's ability to uncover latent user preferences beyond direct co-occurrence. However, because ALS distributes preference scores across all items, it can sometimes prioritize less relevant items compared to memory-based methods that rely on direct similarity. Overall, ALS provided a good balance between coverage and personalization, though it was slightly less precise in surfacing top-ranked recommendations in this dataset.

5.2.4 LightFM Hybrid performance

The LightFM hybrid model, while designed to leverage both collaborative signals and content-based metadata, underperformed relative to the other models across all evaluation metrics. At $K = 10$, it achieved a lower performance than even the non-personalized popularity baseline. Its NDCG@10 score of 0.0325 further reflects that relevant items were not well prioritized in the recommendation rankings. This performance suggests that, in this setting, the hybrid model struggled to effectively integrate the available metadata into meaningful recommendations. Several factors could explain this outcome: the side information from DBpedia and user demographics may have been too sparse or noisy, or the model may have been under-trained due to limited epochs or suboptimal hyperparameter settings. Despite its lower precision, LightFM still holds practical advantages, particularly its ability to recommend new or less-interacted items based on content features, which traditional collaborative filtering models cannot handle. However, in this project's offline evaluation focused on known-user known-item interactions, LightFM was less effective compared to memory-based and matrix factorization approaches.

6. Limitations and Future Work

Although this project provides valuable insights into various recommender system methods, it is subject to several limitations. Firstly, our evaluation relied exclusively on offline metrics computed from historical interaction data, which may not fully capture user satisfaction or real-world recommendation quality. Metrics like precision and recall do not explicitly measure user-perceived novelty, diversity, or serendipity, all of which significantly influence user experience in practical scenarios.

Secondly, the performance of hybrid models, specifically LightFM, heavily depends on the quality, completeness, and relevance of metadata. In this project, the metadata sourced from DBpedia was limited and potentially noisy, which may have adversely impacted the hybrid model's effectiveness. Additionally, hyperparameter tuning and extensive experimentation were constrained by available resources and time, potentially leaving room for performance improvements across all models.

Future work could address these limitations by integrating more robust evaluation frameworks, such as online A/B testing or user studies, to measure real-world user engagement directly. Further enrichment of metadata—perhaps leveraging richer or domain-specific data sources—could also enhance the performance of hybrid models. Additionally, exploring advanced recommendation techniques like deep-learning-based models (e.g., Neural Collaborative Filtering, graph neural networks, or transformers) may provide stronger performance, particularly at scale. Finally, incorporating techniques specifically targeting recommendation diversity and novelty, such as diversification algorithms or re-ranking strategies, would align recommendation outcomes more closely with realistic user preferences and expectations.

7. Conclusion

In this project, we systematically explored and compared several recommender system approaches using the MovieLens dataset enriched with external metadata. Our analysis demonstrated that personalized collaborative filtering methods, notably item-based CF, consistently achieved strong performance, substantially outperforming the popularity baseline in key metrics such as Precision@10 and NDCG@10. The ALS matrix factorization method offered a robust alternative,

particularly effective in uncovering latent user preferences, although with slightly lower top-ranked accuracy compared to memory-based approaches. Meanwhile, the hybrid model (LightFM), despite its lower numerical performance in this setting, showed potential value in addressing cold-start problems and enhancing diversity when richer and more comprehensive metadata is available.

The findings underline the importance of selecting recommendation approaches aligned to specific real-world contexts—considering factors such as data availability, scalability requirements, explainability, and cold-start scenarios. While our evaluation provided valuable insights through offline metrics, future studies incorporating user studies or online A/B testing would offer deeper understanding into real-world user engagement and satisfaction. Overall, this exploration significantly deepened our understanding of recommender systems, highlighting both their practical applicability and areas for future improvement.

References

Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TIIS), 5(4), Article 19. https://doi.org/10.1145/2827872

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76–80. https://doi.org/10.1109/MIC.2003.1167344

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (pp. 263–272). IEEE. https://doi.org/10.1109/ICDM.2008.22

Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. In Proceedings of the 2nd Workshop on New Trends in Content-Based Recommender Systems (CBRecSys) (pp. 14–21). CEUR Workshop Proceedings. https://arxiv.org/abs/1507.08439

IBM Cloud Education. (2024). What is collaborative filtering? IBM Knowledge Center. Retrieved from https://www.ibm.com/think/topics/collaborative-filtering