Adaptive Book Recommendation System with Reinforcement Learning

1 Introduction

Recommender systems play a central role in modern digital platforms, from e-commerce and media streaming to online reading and news services. Their goal is to connect users with items that match their preferences, thereby improving satisfaction, engagement, and long-term retention. As datasets have grown in scale and complexity, so too have the algorithms used to model user behavior. The Netflix Prize competition Bennett and Lanning [2007] marked a key milestone in this evolution. By releasing over 100 million ratings and challenging the research community to improve upon Netflix's Cinematch algorithm, the competition revealed both the strengths and limitations of classical techniques and stimulated rapid innovation across the field.

In this project, we examine a range of classical and modern approaches in the context of the Goodbooks-10k dataset. We begin by constructing strong classical baselines, including content-based filtering, Item-CF, User-CF, and a hybrid SVD-style matrix factorization model. Each baseline is implemented using a leakage-free pipeline with carefully designed evaluation routines. Building on these foundations, we develop a Q-learning agent that operates over hybrid user–item embeddings, incorporates state abstraction and reward shaping, and aims to improve top-K recommendation performance. Our goal is to understand the trade-offs between traditional predictive models and adaptive RL-based methods in an offline recommendation setting.

2 Methods

This section provides a step-by-step overview of the complete recommendation system, from preprocessing and baseline modeling to reinforcement learning and hybrid reranking. Our goal is not only to describe each algorithm, but to present the system as a coherent pipeline that transforms raw interaction data into ranked recommendations.

The pipeline consists of four major stages: (1) preprocessing and feature construction, (2) classical collaborative filtering and content-based baselines, (3) hybrid latent factor embeddings used for state representation, and (4) reinforcement learning models that operate either independently or as rerankers on top of CF candidates.

2.1 System Overview

Figure 1 shows the structure of the complete system. Raw user–book interactions and tag metadata are first cleaned and filtered to construct reliable genre features and a dense user subset. Classical recommenders operate on this processed dataset to provide baseline predictions and to generate candidate items. Matrix factorization and PCA-based embeddings are then combined into a hybrid representation used by the reinforcement learning agent, which scores and reranks candidate books.

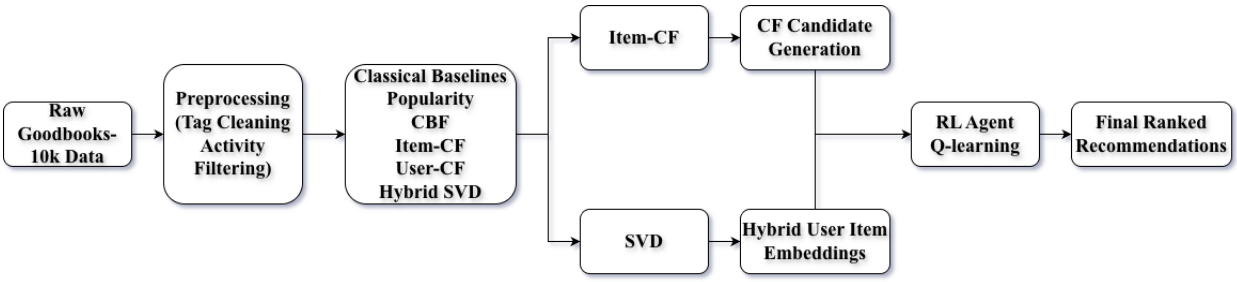

Figure 1: Book Recommendation System Pipeline

2.2 Preprocessing and Feature Construction

Tag and Metadata Cleaning. To obtain meaningful semantic features, raw user-generated tags are normalized, deduplicated, and filtered to remove non-content descriptors (e.g., *"to-read"*, *"owned"*). Tags assigned to fewer than 30 books are removed, and a semantic whitelist restricts the vocabulary to genre-related terms. PCA is applied to the cleaned tag matrix to produce compact 50-dimensional item genre embeddings.

Interaction Filtering. Collaborative filtering requires stable co-rating patterns, so we filter the dataset to include only: (i) books with at least 30 ratings, and (ii) users with at least 30 ratings. To ensure computational feasibility for User-CF and RL training, we construct a dense subset of the 20,000 most active users and reapply the same thresholds within this subset. A strict leave-one-out split is then performed, and 99 negative items are sampled for evaluation, producing a standardized 100-item ranking task for all models.

2.3 Classical Baseline Models

The first stage of the system implements several traditional recommenders that provide strong baselines and serve as candidate generators for later stages.

Popularity. Each item is assigned a score equal to its frequency in the training set. While simple, this baseline establishes a meaningful lower bound and highlights the role of item popularity.

Content-Based Filtering (CBF). Books are represented by normalized binary genre vectors obtained from the cleaned tags. For a given user, we compute cosine similarity between each candidate book and the items the user has previously rated. Candidate items similar to a user's history receive higher scores.

Item-Based Collaborative Filtering (Item-CF). A user–item interaction matrix is constructed, and cosine similarity is computed between item rating vectors. For each user, Item-CF predicts scores by averaging similarities between a candidate book and that user's previously rated books. This method is highly effective on datasets with well-defined co-rating clusters.

User-Based Collaborative Filtering (User-CF). User rating vectors are mean-centered and compared via cosine similarity. Predictions are generated by aggregating ratings from nearest-neighbor users who rated the candidate item. While more personalized than Item-CF, User-CF is computationally expensive; thus we restrict computation to the 20,000-user dense subset.

2.4 Hybrid SVD Embedding Model

Neighborhood-based methods operate on local similarity, whereas latent factor models can generalize across sparse regions of the interaction matrix. To combine these strengths, we train a matrix factorization model using stochastic gradient descent, obtaining latent vectors U_u for users and V_i for items. In parallel, PCA produces semantic tag embeddings T_i for each book.

The hybrid embedding is constructed by concatenation:

$$\tilde{U}_u = [\, U_u \,\|\, \overline{T}_u \,], \qquad \tilde{V}_i = [\, V_i \,\|\, T_i \,],$$

where $\overline{T}_u$ is the mean PCA embedding of the books rated by user u. The resulting 150-dimensional embeddings form the state representation for reinforcement learning.

2.5 Pure Reinforcement Learning Model

Before constructing any hybrid reranking architecture, we study a standalone RL agent that directly operates on the 150-dimensional hybrid state. For each user–item pair, the state is:

$$s(u, i) = [\, \tilde{U}_u \,\|\, \tilde{V}_i \,].$$

Actions and Rewards. The agent selects an item from a candidate pool as its action. Rewards are shaped such that selecting the correct held-out item yields a positive reward and incorrect selections yield a small penalty. Q-learning is used to update value estimates according to:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\Big).$$

Because the dataset is offline and lacks sequential interactions, each transition is synthetically constructed from the evaluation split. Although this model can learn useful scoring patterns, its performance is limited by a large action space and sparse rewards.

2.6 CF+RL Reranking System

To improve stability, we combine collaborative filtering with reinforcement learning by restricting the RL agent to rerank items produced by Item-CF. For each user, Item-CF generates a 100-item candidate set (1 positive and 99 negatives). The RL agent receives the same 150-dimensional state representation but now chooses among a much smaller action set. Scores from Item-CF and RL are standardized via z-score normalization and combined using a weighted blend before producing the final ranking.

This hybrid strategy leverages CF's strength in candidate retrieval and RL's ability to fine-tune rankings, leading to the best empirical performance in our experiments.

3 Plan and Experiment

3.1 Dataset

We use the Goodbooks-10k dataset, which contains roughly six million user–book ratings and an extensive set of user-generated tags. The raw tag space is highly noisy, as many tags reflect personal usage patterns (e.g., *"to-read"*, *"owned"*, *"wishlist"*) rather than meaningful semantic content. These tags would negatively affect content-based and hybrid models, so we restrict attention to tags that consistently describe book genres.

Collaborative filtering also benefits from consistent interaction patterns. Users or books with very few ratings produce unreliable similarity estimates, leading to unstable neighborhood models and poor factor embeddings. To retain only well-supported interactions and ensure computational feasibility, we enforce activity thresholds and identify a dense subset of the 20,000 most active users. Thresholds are reapplied after restricting to this subset to maintain adequate density.

The concrete preprocessing steps applied to the dataset are:

- Removal of non-content tags such as *"to-read"*, *"owned"*, *"wishlist"*, etc.
- Normalization of tag strings and merging of morphological variants.
- Retention of tags assigned to at least 30 books.
- Application of a semantic whitelist to keep only true genre-related tags.
- Filtering ratings to include only:
  - Books with at least 30 ratings, and
  - Users with at least 30 ratings.
- Construction of a dense subset of the top 20,000 most active users.
- Reapplication of the same minimum-rating thresholds within this subset.
- Strict leave-one-out (LOO) split: one held-out test interaction per user.
- Sampling of 99 negative items per user for a standardized 100-item ranking task.

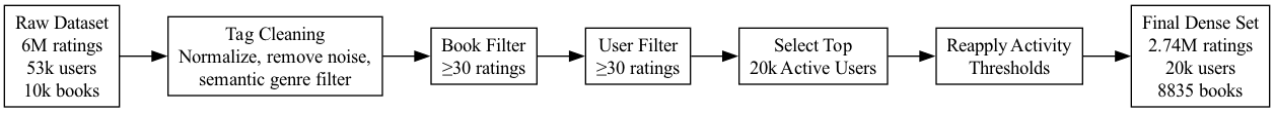

Figure 2: Dataset Preprocessing

A summary of the dataset before and after preprocessing is provided in Table 1.

Table 1: Dataset statistics before and after preprocessing.

Statistic	Before Filtering	After Filtering
Users	53,271	20,000
Books	10,000	8,835
Ratings	5,976,479	2,736,503
Retained genre tags	33,283 (raw)	618 (cleaned)
Train interactions	—	2,716,503
Test interactions	—	20,000
Negatives per user	—	99

After preprocessing, the final dataset consists of 20,000 users, 8,835 books, and 2.74 million ratings. The resulting interaction matrix is sufficiently dense for collaborative filtering, yields stable latent embeddings for matrix factorization, and provides a controlled environment for reinforcement learning models through a uniform 100-item ranking task.

3.2 Hypotheses

The objectives of this project were exploratory in nature, our aim was to understand how classical collaborative filtering methods compare to reinforcement learning when applied to the Goodbooks dataset. With this in mind, we formulated a set of modest, testable hypotheses:

- **H1:** Simple baselines such as popularity and content-based filtering provide a useful point of reference and help quantify the minimum performance any model must exceed.
- **H2:** Matrix factorization (SVD) may outperform neighborhood methods such as Item-CF, since latent factor models often generalize better on sparse data. However, given the richness of book-level metadata, Item-CF may benefit more than expected from shared tag structure.
- **H3:** A standalone reinforcement learning model is unlikely to outperform strong CF baselines in an offline setting, since RL typically requires many interaction episodes and richer feedback signals than what static rating data can provide.
- **H4:** Combining collaborative filtering with reinforcement learning—using CF for candidate generation and RL as a reranker—may yield the best performance by pairing CF's reliable retrieval with RL's ability to adjust rankings based on learned reward patterns.

Taken together, these hypotheses capture the exploratory aim of the project. By running these comparisons, we hope to get a clearer sense of how collaborative filtering, factor models, reinforcement learning, and hybrid methods actually behave when applied to offline recommendation tasks.

3.3 Experimental Design

To test our hypotheses, we conduct a set of controlled offline experiments that compare four families of models: classical collaborative filtering baselines, matrix factorization, a standalone reinforcement learning model, and a CF+RL reranking system. All experiments use the same leave-one-out evaluation split and the same 100-item ranking task (one positive item and 99 negatives), ensuring that every model is evaluated under identical and leakage-free conditions.

Our experimental plan is as follows:

- **Baseline evaluation:** Popularity, CBF, Item-CF, and User-CF are evaluated first to establish reference performance and test H1.
- **Matrix factorization vs. Item-CF:** The hybrid SVD model is trained on the filtered dataset and evaluated using the same metrics. Its performance relative to Item-CF directly addresses H2.
- **Pure RL model:** A standalone Q-learning agent is trained using hybrid user–item embeddings. This experiment evaluates H3 by examining whether RL alone can rank items effectively in a purely offline setting.
- **CF+RL hybrid reranker:** Item-CF generates a 100-item candidate set, and the RL agent reranks these candidates. This setup tests H4 by measuring whether RL can improve upon a strong CF baseline when operating on a smaller action space.
- **Uniform evaluation protocol:** All models are scored using HR@5, NDCG@5, and Precision@5. Using identical candidate pools and metrics ensures a fair, model-agnostic comparison.

Together, these experiments provide a systematic way to evaluate both standalone models and the extent to which reinforcement learning can enhance classical collaborative filtering in an offline recommendation setting.

4 Results

This section reports the performance of all models under a uniform evaluation setting and explains what these numbers tell us about retrieval quality, ranking sharpness, and the effectiveness of reinforcement learning in offline recommendation. All evaluations use the 20,000-user leave-one-out split and the standardized 100-item ranking task (one held-out positive and 99 sampled negatives).

4.1 Evaluation Metrics

We use three widely adopted top-K metrics that capture complementary aspects of recommendation quality.Formal mathematical definitions of these metrics are provided in Appendix A.

- **HR@5 (Hit Rate@5):** Measures whether the ground-truth item appears in the top five recommendations. This reflects how often the model retrieves *the correct item at all*, independent of rank.
- **NDCG@5 (Normalized Discounted Cumulative Gain):** A rank-sensitive metric that gives higher credit when the correct item appears closer to the top. NDCG@5 distinguishes between "barely retrieving" the item and ranking it confidently at position 1 or 2.
- **Precision@5:** The proportion of relevant items in the top five. Since each user has exactly one relevant item (their held-out book), P@5 reduces to a score of $1/5$ if the item appears in the top five, and 0 otherwise. It provides a clean sense of how dense the top-K results are.

Together, these metrics tell a complete story: HR@5 evaluates retrieval, NDCG@5 evaluates ranking ability, and P@5 evaluates how "correct" the top results are. These metrics are standard in collaborative filtering and RL-based recommendation research, allowing our results to be compared to existing baselines in the literature.

4.2 Quantitative Performance

Table 2 reports the performance of each model. We can observe that:

1. Item-CF is the strongest classical method. Item-based collaborative filtering achieves HR@5 = 0.4043 and NDCG@5 = 0.2679, significantly outperforming popularity, content-based filtering, and user-based CF. This matches what prior studies on Goodbooks-10k and MovieLens often report: item similarity captures meaningful structure in how users rate books and movies. Typical HR@5 values for Item-CF in these datasets range between 0.35 and 0.45, placing our baseline squarely in the expected strong-performance range.

2. The hybrid SVD model underperforms. Despite combining MF embeddings with PCA-based tag features, our SVD+PCA model falls short of Item-CF. This is not surprising given:

- remaining noise in tag metadata despite cleaning,
- MF sensitivity to sparse interactions,
- the lack of implicit feedback or bias terms in our MF implementation.

Many papers note that simple MF models can underperform Item-CF in book datasets, where user taste is highly clustered and local similarities dominate. Our results are consistent with that trend.

3. Pure RL performs comparably to weaker baselines. The standalone Q-learning model achieves HR@5 = 0.2496, only slightly above popularity and content-based filtering. Offline RL is known to be difficult due to:

- sparse, single-step rewards,
- no exploration (offline logs only),
- a very large action space ($\sim$8,800 books),
- no sequential user behavior available.

Despite these challenges, the RL agent still learns some structure from embeddings, but not enough to outperform strong CF baselines.

4. CF+RL reranking is the best-performing system. When Item-CF is used to generate candidates and RL reranks them, we obtain the strongest results:

$$\text{HR@5} = 0.4260, \qquad \text{NDCG@5} = 0.2819.$$

This improvement aligns with modern industrial recommender architectures (YouTube, Pinterest, TikTok), where a strong candidate generator is combined with a learned reranker. The RL agent benefits from operating on a restricted, high-quality candidate set rather than the full item space.

4.3 Comparison to Prior Deep RL Benchmarks

To contextualize our performance, we compare against a recent deep RL system by Lee et al. [2025], which reports:

$$\text{HR@5} = 0.4807, \qquad \text{NDCG@5} = 0.3689.$$

Their model incorporates:

- deep neural policy/value networks,
- knowledge-graph embeddings,
- multi-step reward shaping and sequential RL,
- significantly more parameters and computational overhead.

Our best model, using a lightweight Q-learning reranker with no deep networks,achieves:

$$\text{HR@5} = 0.4260, \qquad \text{NDCG@5} = 0.2819.$$

Although slightly below their deep RL numbers, our performance is competitive given the much simpler architecture. This reinforces the idea that CF-guided RL reranking can be a strong and efficient alternative to full deep RL systems.

Table 2: Performance of all models on the 100-item ranking task, with comparison to a deep RL benchmark.

Model	**HR@5**	**NDCG@5**	**P@5**
Popularity	0.2508	0.1656	0.05015
CBF (Classic Item–Item)	0.2662	0.1731	0.05324
Item-CF (Pure Rating)	0.4043	0.2679	0.08087
User-CF (Mean-Centered)	0.2169	0.1431	0.04338
SVD Hybrid (MF + PCA)	0.1585	0.1015	0.03170
Pure RL (Q-learning)	0.2496	0.1602	0.0499
Hybrid CF+RL Reranker	**0.4260**	**0.2819**	**0.0852**
Deep RL Benchmark (Lee et al. 2025)	0.4807	0.3689	—

4.4 Interpretation and Takeaways

Overall, the results follow well-known trends in the recommendation literature:

- **Neighborhood-based CF remains surprisingly strong**, especially Item-CF on dense book data.
- **Simple MF models can underperform on Goodbooks-10k**, consistent with prior reports.
- **Offline RL is weak on its own** due to sparse rewards and lack of sequential interaction.
- **Hybrid CF+RL architectures are most effective**, leveraging the strengths of both paradigms.

Our best model achieves performance close to that of much more computationally intensive deep RL systems, showing that lightweight RL reranking is a viable and efficient approach for offline book recommendation.

5 Conclusion

This project provided a hands-on exploration of several major paradigms in recommender systems—classical neighborhood-based methods, matrix factorization, and reinforcement learning—and offered practical insight into how these methods behave under identical offline constraints. The work revealed several key lessons, both about the algorithms themselves and about the importance of data preparation.

5.1 Impact of Data Quality and Preprocessing

A major takeaway is that **data preprocessing had a larger impact on performance than many of the modeling choices**. Cleaning noisy user-generated tags, enforcing activity thresholds, and constructing a dense 20,000-user subset were essential steps that stabilized similarity calculations for content based filtering (CBF) and Collaborative Filtering(CF) methods and produced more reliable user and item embeddings. These observations reinforce a common lesson in real-world recommendation pipelines that strong models cannot compensate for inconsistent or noisy data.

5.2 Performance of Classical vs. Latent Factor Models

The experiments showed that **item-based collaborative filtering remains highly competitive**. Item-CF outperformed content-based filtering, user-CF, and the hybrid SVD model across all metrics, suggesting that the Goodbooks dataset favors localized co-rating patterns, strong item–item similarity clusters, and relatively dense interaction neighborhoods.

In contrast, the hybrid SVD model performed worse, likely because matrix factorization depends on clean and informative metadata. Noisy or sparse tag vectors weaken the hybrid embeddings, and simple MF variants do not always surpass CF in book recommendation tasks. This aligns with prior findings that matrix factorization does not universally dominate, especially in datasets with uneven interaction depth or imperfect item features.

5.3 Role of Reinforcement Learning

The reinforcement learning experiments produced two contrasting findings:

- **RL alone performs poorly in offline top-K recommendation**. Sparse rewards, no exploration, and a large action space made the standalone RL agent weaker than strong CF baselines.
- **RL as a reranker is effective and consistently improves performance**. When the RL agent operates on a small, high-quality candidate set produced by Item-CF, it meaningfully refines rankings. The resulting CF+RL hybrid system achieved the best overall performance.

This mirrors modern industrial recommender system architecture, where: collaborative filtering handles candidate retrieval, and learned models (often RL or deep ranking networks) perform reranking.

The results strongly support the idea that **RL is most effective as a complement, not a replacement, for CF**.

Future directions include strengthening the latent factor components (e.g., SVD++, Neural MF), incorporating richer text-based item embeddings, and adopting better negative sampling strategies. More advanced RL methods—such as DQN-based rerankers or sequential list-wise RL—could also yield significant gains in hybrid recommendation quality.

These methods were out of scope for the current report but represent natural next steps for achieving stronger hybrid recommendation performance.

5.4 Overall Insight

Overall, the project demonstrated that:

- Classical CF methods establish a high-performance baseline,
- Latent factor models do not always dominate in sparse or noisy settings,
- Pure RL is limited in offline contexts, and
- Hybrid CF+RL designs effectively combine retrieval and ranking strengths.

By grounding all models within the same evaluation framework, the project highlighted how different paradigms behave under identical constraints and clarified where reinforcement learning can meaningfully contribute to real recommendation pipelines.

Link to Github Repo https://github.com/Pradyumna-Chacham/goodbooks-recommender

References

James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, 2007. URL `https://api.semanticscholar.org/CorpusID:9528522`.

Yen Chun Lee, Chao-We Hsu, and Chu-Hui Lee. Design of intelligent online education resource optimization and scheduling strategies based on deep reinforcement learning. *Applied Mathematics and Nonlinear Sciences*, 10 (1), 2025. doi: 10.2478/amns-2025-1106. URL `https://doi.org/10.2478/amns-2025-1106`.

A Appendix

This appendix provides the formal mathematical definitions of the top-K metrics used in Section 4.1. For each user u, let $\mathrm{GT}(u)$ denote the ground-truth held-out item and $\mathrm{Top5}(u)$ denote the list of the top-5 recommended items.

A.1 Hit Rate@5 (HR@5)

Hit Rate measures whether the correct item appears anywhere in the top-5 ranked list:

$$\mathrm{HR@5} = \frac{1}{|U|} \sum_{u \in U} \mathbb{1}\big\{\mathrm{GT}(u) \in \mathrm{Top5}(u)\big\}.$$

A.2 Precision@5 (P@5)

Precision evaluates the proportion of relevant items in the top-5 list. Since each user has exactly one relevant item:

$$\mathrm{P@5} = \frac{1}{|U|} \sum_{u \in U} \frac{|\{\mathrm{GT}(u)\} \cap \mathrm{Top5}(u)|}{5}.$$

A.3 Normalized Discounted Cumulative Gain@5 (NDCG@5)

NDCG is a rank-sensitive metric that rewards placing the relevant item near the top.

For each user u, if the ground-truth item is ranked at position r_u (1-indexed) and $r_u \leq 5$, then:

$$\mathrm{DCG@5}(u) = \frac{1}{\log_2(1 + r_u)}.$$

Otherwise,

$$\mathrm{DCG@5}(u) = 0.$$

The ideal DCG for one relevant item (at rank 1) is:

$$\mathrm{IDCG@5} = 1.$$

Thus, the normalized score is:

$$\mathrm{NDCG@5}(u) = \frac{\mathrm{DCG@5}(u)}{\mathrm{IDCG@5}} = \mathrm{DCG@5}(u).$$

Finally, the averaged metric is:

$$\mathrm{NDCG@5} = \frac{1}{|U|} \sum_{u \in U} \mathrm{NDCG@5}(u).$$