Peertechz

ENGINEERING GROUP

TRENDS IN Computer Science and Information Technology
OPEN ACCESS JOURNAL

ISSN: 2641-3086 DOI: https://dx.doi.org/10.17352/tcsit

Research Article

A Comparative Study of Recommender Systems under Big Data Constraints

Arimondo Scrivano*

DEIB – Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy

Received: 24 July, 2025
Accepted: 05 August, 2025
Published: 06 August, 2025

***Corresponding author**: Arimondo Scrivano, DEIB – Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy, E-mail: arimondo.scrivano@mail.polimi.it

https://www.engineegroup.us

Check for updates

Abstract

Recommender Systems (RS) have become essential tools in a wide range of digital services, from e-commerce and streaming platforms to news and social media. As the volume of user-item interactions grows exponentially, especially in Big Data environments, selecting the most appropriate RS model becomes a critical task. This paper presents a

Comparative study of several state-of-the-art recommender algorithms, including EASE-R, SLIM, SLIM with ElasticNet regularization, Matrix Factorization (FunkSVD and ALS), P3Alpha, and RP3Beta. We evaluate these models according to key criteria such as scalability, computational complexity, predictive accuracy, and interpretability. The analysis considers both their theoretical underpinnings and practical applicability in large-scale scenarios. Our results highlight that while models like SLIM and SLIM-ElasticNet offer high accuracy and interpretability, they suffer from high computational costs, making them less suitable for real-time applications. In contrast, algorithms such as EASE-R and RP3Beta achieve a favorable balance between performance and scalability, proving more effective in large-scale environments. This study aims to provide guidelines for selecting the most appropriate recommender approach based on specific Big Data constraints and system requirements.

Introduction

Recommender Systems (RS) are integral to modern digital ecosystems, driving personalized experiences across a wide spectrum of platforms, including e-commerce, streaming services, online education, and social media [1]. As user-item interaction data becomes increasingly voluminous, heterogeneous, and generated at high velocity—hallmarks of the Big Data paradigm—the need for scalable and computationally feasible algorithms has become paramount. Traditional recommendation techniques are now facing substantial limitations with respect to real-time inference and computational efficiency, making the algorithmic choice a critical design decision.

Model-based approaches have gained dominance due to their superior generalization capacity and ability to scale to industrial-level datasets. This study examines five influential and widely adopted algorithms in the recommender systems landscape: EASE-R (Embarrassingly Shallow Autoencoders), SLIM (Sparse Linear Methods), SLIM with ElasticNet Regularization, Matrix Factorization (including FunkSVD and ALS), and RP3beta (Random Walk with Restart). Each is analyzed in terms of performance under Big Data constraints, focusing on dimensions such as accuracy, scalability, interpretability, suitability for real-time environments, robustness to data noise and sparsity, and expected lifespan before possible replacement by more advanced or quantum-accelerated algorithms.

Building upon these foundations, the present study offers a focused comparative analysis of recommender algorithms, explicitly framed around Big Data constraints. It evaluates both predictive accuracy and computational efficiency, offering a dual perspective that integrates performance and practicality. Unlike prior works that tend to emphasize either algorithmic capability or scalability in isolation, this analysis bridges the two dimensions, offering actionable insights for real-world model selection in large-scale systems.

Furthermore, this study contributes a forward-looking perspective on the long-term viability and adaptability of

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073. Available from: https://dx.doi.org/10.17352/tcsit.000099

current recommendation algorithms. By considering the growing prevalence of quantum-ready infrastructures and continuous data expansion, it anticipates future shifts in the design and evaluation of recommender systems. In doing so, it aligns model assessment not only with present-day requirements but also with emerging technological landscapes.

This literature review synthesizes key developments in the field while situating the current work within the broader context of recommender system research. It underscores enduring challenges such as scalability and interpretability, acknowledges the contributions of recent neural and graph-based innovations, and articulates a distinctive methodological contribution oriented toward future-proof, industrial-scale deployment.

Methodology

This section outlines the methodological framework adopted for the comparative evaluation of the selected recommender algorithms under Big Data constraints. The design encompasses dataset selection, pre-processing strategies, evaluation metrics, model configuration, and experimental design choices intended to promote fairness, reproducibility, and practical relevance.

Datasets

The experimental evaluation is conducted using three well-established large-scale datasets frequently employed in recommender systems research. The first is MovieLens 20 M [2], which consists of 20 million user-generated ratings across 27,000 movies from a cohort of 138,000 users. The second dataset, drawn from the Amazon Product Review corpus [3], is a subset focused specifically on books, comprising over 22 million user-item interactions. The third dataset is the Netflix Prize dataset [4], containing 100 million ratings spanning 17,000 movies.

To ensure consistency and relevance for implicit feedback modeling, each dataset is pre-processed by filtering out users and items with fewer than five recorded interactions. Moreover, the rating data is binarized by retaining only positive interactions—typically those with ratings equal to or greater than four— aligning with standard practice in the evaluation of implicit recommendation systems.

Data preprocessing

The preprocessing phase was intentionally kept minimal to reflect a realistic usage scenario of recommender systems, where raw user-item interaction data is directly employed. No explicit data cleaning or outlier removal was performed. The dataset was used in its original form, assuming that the interactions are already meaningful and representative.

The User-Rating Matrix (URM) was constructed by mapping each (user id, item id) pair into a binary sparse matrix, where each non-zero entry indicates an interaction. The matrix was built using the original interaction dataset, with values cast to boolean type to represent implicit feedback. A global 80 / 20 holdout strategy was applied, yielding URM train and URM test matrices used respectively for model training and evaluation.

In parallel, the Item Content Matrix (ICM) was created. This matrix encodes item-feature associations using binary values, where each row corresponds to an item and each column to a content feature. To ensure alignment between the ICM and URM dimensions, padding rows were added to the ICM when necessary.

Finally, a stacked URM matrix was constructed by vertically concatenating the transposed ICM with the URM train matrix. This composition allows hybrid models to exploit both collaborative signals (user-item interactions) and content-based signals (item features) within a unified structure.

This streamlined yet complete setup enables fair and consistent evaluation of various recommender algorithms while maintaining full reproducibility.

Evaluation metrics

To assess the performance of each algorithm, a set of widely accepted evaluation metrics is employed. Precision@K quantifies the proportion of relevant items retrieved among the top-K recommendations, while Recall@K measures the coverage of relevant items within those top-K suggestions. The Normalized Discounted Cumulative Gain (NDCG@K) adds a positional bias, rewarding algorithms that rank relevant items higher in the recommendation list. In addition, Mean Average Precision (MAP@K) is used to capture overall ranking quality across users, offering robustness to class imbalance and sparsity.

To complement these relevance-based metrics, computational efficiency is also assessed using training time and memory footprint. All metrics are averaged over five randomized 80/20 train-test splits, employing a holdout validation strategy to emulate production deployment environments.

Hyperparameter sensitivity analysis

To investigate the impact of hyperparameter selection on algorithm performance, we conducted hyperparameter tuning using the Optuna optimization framework. This approach allows efficient exploration of the search space via Bayesian optimization and pruning strategies, significantly reducing the number of required trials compared to exhaustive grid search.

The sensitivity of recommender performance to hyperparameter choices varies substantially depending on the algorithm. For simpler algorithms such as TopPop, ItemKNN, and UserKNN, the performance degradation in the worst-case scenario—using untuned hyperparameters—remains within approximately 50% of the optimal configuration. These models typically reach their best configurations in just a few minutes.

In contrast, more complex models such as SLIM Elastic Net and EASE-R exhibit much higher sensitivity. In the worst case, poor hyperparameter selection can lead to performance drops of over 80%. Additionally, the time required for optimization increases significantly. While SLIM Elastic Net may require several hours to converge on a moderately powerful machine,

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073.
Available from: https://dx.doi.org/10.17352/tcsit.000099

EASE-R can require days on consumer-grade hardware due to its high memory consumption and the need to solve a large-scale regularized least-squares problem.

To manage computational cost, we adopted early stopping and trial pruning in Optuna. This helped identify suboptimal regions of the search space early and reallocate resources to more promising configurations. Nevertheless, the results underline the importance of using adaptive and parallelizable tuning frameworks for complex recommender systems.

Table 1 shows a summary of MAP@10 scores across various algorithms under three configurations: random hyperparameter sampling, basic grid search, and the best result found using Optuna. As shown, performance gains from tuning are modest for simpler models, but critical for advanced ones.

These results emphasize that hyperparameter tuning is not just a fine-tuning step but a critical component of model development. Proper sensitivity analysis ensures that reported performances are robust and representative of each algorithm's true potential.

Model configuration

Each algorithm is implemented using reliable open-source recommender system libraries, including LightFM, Implicit, and custom implementations based on PyTorch and Numpy. Where applicable, hyperparameters are optimized through grid search to ensure fair tuning across models.

EASE-R is configured with a regularization parameter λ = 0.5, as recommended in the original work [5]. SLIM employs L1 regularization with $\alpha = 10^{-4}$, trained via coordinate descent, while its ElasticNet variant introduces an L1 ratio of 0.5 with the same regularization strength. For matrix factorization, the Alternating Least Squares (ALS) model is configured with 50 latent factors and 20 training iterations. RP3beta, which relies on graph-based recommendation principles, is parameterized with α = 0.6, β = 0.4, and a topK truncation set to 100.

All experiments are conducted on a high-performance machine equipped with 256 GB of RAM and 32 CPU cores, approximating the computational resources available in industrial recommender system deployments.

Evaluation under big data constraints

To realistically simulate operational conditions in Big Data environments, the experimental protocol evaluates four critical dimensions of algorithm robustness. The first dimension is scalability, tested by incrementally increasing the dataset size across thresholds of 100,000, 1 million, and 10 million interactions. This allows for empirical assessment of each model's ability to scale with data volume.

Second, cold start resistance is investigated by introducing new users and items with minimal interaction history, thus gauging the algorithm's adaptability to sparse user behavior. Third, the feasibility of incremental updates is analyzed by measuring the computational cost and performance impact of incorporating new data without performing a full model retraining. This is essential for systems operating under streaming or near-real-time conditions.

Finally, latency is measured by the time required to generate recommendations for a batch of 1,000 users, offering insight into real-time applicability. These evaluation criteria collectively provide a multidimensional view of each algorithm's strengths and limitations under the constraints typical of large-scale recommender systems.

This comprehensive methodology ensures that the evaluation is both rigorous and reflective of real-world deployment scenarios. The selection of benchmark datasets, thoughtful configuration of models, and attention to scalability and responsiveness contribute to a fair and reproducible comparison of algorithmic performance in Big Data contexts.

Table 1: Impact of Hyperparameter Optimization on MAP@10 for Selected Algorithms.

Algorithm	Random Config	Grid Search	Optuna (Best)
TopPop	0.021	0.022	0.022
ItemKNN	0.073	0.081	0.089
UserKNN	0.066	0.075	0.084
SLIM Elastic Net	0.039	0.071	0.079
EASE-R	0.011	0.053	0.060
RP3beta	0.068	0.074	0.080
P3alpha	0.065	0.071	0.078

Related work

This section reviews the existing literature on recommender systems, with a focus on advancements in scalability, graph-based models, and neural architectures. It also provides a comparative overview of empirical findings in prior studies, highlighting the evolving challenges and opportunities in this field.

EASE-R

EASE-R, introduced by Steck in 2019 [5], is a shallow autoencoder designed to learn item-item similarity matrices using L2 regularization. Its simplicity and effectiveness lie in its ability to reduce the training process to solving a closed-form linear system, which is highly parallelizable and efficient even at an industrial scale. In Big Data settings, EASE-R has demonstrated notable competitiveness in recommendation accuracy, closely matching deeper neural models while requiring far less computational overhead. It benefits from a minimal need for hyperparameter tuning and scales linearly with the number of items, making it particularly appealing for systems handling tens or hundreds of millions of interactions. The interpretability of EASE-R is moderate, as the learned similarity coefficients provide a direct view of item influence relationships. Looking forward, its estimated viability horizon ranges from five to seven years, depending on the progression of quantum-enhanced similarity learning techniques.

SLIM (Sparse Linear Methods)

SLIM [6] models the profile of each item as a sparse linear combination of other items using Lasso regression, yielding

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073.
Available from: https://dx.doi.org/10.17352/tcsit.000099

high-accuracy predictions, particularly in collaborative filtering scenarios. It is notable for the transparency and interpretability of its outputs, thanks to the enforced sparsity, which facilitates insight into the most influential item relationships. However, in Big Data environments, SLIM suffers from serious scalability constraints due to the intensive computation required during the learning phase. As a result, various approximations such as SLIM-BPR and optimized Cython implementations have been developed to mitigate these bottlenecks. Nevertheless, SLIM's core methodology remains challenging to apply without pre-filtering or distributed computing infrastructure. Its forecasted utility is limited to the next two to three years unless breakthroughs in quantum-accelerated sparse regression are realized.

SLIM with ElasticNet regularization

The ElasticNet variant of SLIM introduces a combination of L1 and L2 penalties, aimed at improving the robustness and generalization of the model, particularly in noisy and high-dimensional data settings [7]. While it retains SLIM's interpretability and sparsity properties, the addition of the L2 term stabilizes the learning process and reduces overfitting. In terms of scalability, it remains resource-intensive but exhibits better resilience compared to its Lasso-only predecessor. This makes SLIM-ElasticNet a more viable candidate for larger datasets, though still not on par with the scalability of matrix factorization or graph-based methods. It's estimated horizon extends to three or four years, particularly in domains where transparency and interpretability are paramount and quantum-enhanced alternatives are not yet widely deployable.

Matrix factorization (FunkSVD and ALS)

Matrix Factorization techniques such as FunkSVD [8] and Alternating Least Squares (ALS) [9] form the backbone of many collaborative filtering systems. These approaches map users and items into a shared latent space, enabling the inference of unobserved preferences based on latent feature interactions. ALS, in particular, is well-suited to Big Data contexts due to its amenability to parallelization and native support in distributed frameworks like Apache Spark. FunkSVD, while still respected for its accuracy, faces scalability issues stemming from its dependence on stochastic gradient descent, which is less tractable for very large datasets. Interpretability in matrix factorization is generally low, as the latent dimensions lack intuitive semantic grounding. Looking ahead, ALS is expected to remain a cornerstone of scalable recommendation systems for the next five to ten years, while FunkSVD may gradually phase out unless successfully adapted to future paradigms such as quantum stochastic optimization.

RP3beta (Random walk with restart)

RP3beta [10] represents a non-learning, graph-based approach that enhances traditional random walk methods with mechanisms such as popularity penalization and restart probabilities. Its main strength lies in its simplicity and speed, as it does not require model training and scales exceptionally well to large datasets. RP3beta is especially useful in cold-start and real-time recommendation scenarios where rapid inference is critical. While it lacks the representational power of latent factor models, it offers moderate interpretability, with recommendations traceable through the paths of random walks on user-item graphs. The algorithm is expected to remain relevant over the next six to eight years, particularly within hybrid recommender architectures or as a reliable fallback system when learning-based models are infeasible.

Scalability in recommender systems

Scalability has long been a central concern in the development of recommender systems due to the exponential growth in user-item interaction data. Among the most influential contributions in this domain is Matrix Factorization (MF), which gained widespread recognition through its pivotal role in the Netflix Prize competition [8]. MF demonstrated an effective compromise between accuracy and computational cost, especially with the advent of variants like Alternating Least Squares (ALS), which have been successfully adapted for distributed computing environments such as Apache Spark [9]. These implementations facilitate the application of MF to industrial-scale datasets by leveraging parallel processing and memory optimization.

Linear models, particularly SLIM [6] and its ElasticNet-enhanced variant [7], have also made notable contributions, valued for their high interpretability and competitive predictive performance. However, they face scalability bottlenecks during training, prompting recent research to explore more efficient implementations. Examples include SLIM-Cython [11], which offers performance gains through low-level optimization, and other approximate learning approaches aimed at mitigating the intensive computational overhead of sparse linear modeling.

EASE-R, introduced more recently, offers a unique architectural advantage. By formulating the recommendation task as a regularized least-squares problem with a closed-form solution, EASE-R eliminates the need for iterative optimization, resulting in a highly scalable approach. This property makes it particularly well-suited for real-time applications and high-throughput environments where speed and simplicity are paramount.

Graph-based and neural models

Beyond linear and factorization-based methods, recommender systems have increasingly integrated graph-based and neural network-based paradigms. Graph-based approaches, including P3Alpha and RP3beta [10], operate on user-item bipartite graphs, employing random walks to uncover collaborative signals embedded in the graph structure. These techniques are celebrated for their rapid inference capabilities and low training times, making them especially attractive for applications requiring high responsiveness. However, their dependence on explicit graph connectivity can limit effectiveness in sparse data environments or situations involving new users or items.

In parallel, neural network-based recommenders have emerged as a dominant trend in recent years. Architectures

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073.
Available from: https://dx.doi.org/10.17352/tcsit.000099

such as NeuMF [12] and LightGCN [13] combine representation learning with collaborative filtering, offering significant gains in predictive accuracy, particularly in offline evaluations. These models possess powerful representational capacity, enabling them to model complex user-item dynamics. Nevertheless, they also introduce considerable computational complexity, require extensive hyperparameter tuning, and often lack transparency—issues that can complicate their deployment in large-scale, real-time systems.

Neural and graph-based recommendation models

In recent years, deep learning and graph-based techniques have revolutionized recommender systems by capturing complex user–item interactions and higher-order connectivity patterns. Below, we summarize the key developments and cutting-edge trends.

Neural Collaborative Filtering (NCF): Neural Collaborative Filtering replaces the inner product in classical matrix factorization with a multi-layer perceptron (MLP), enabling richer interaction functions. Extensions include GMF+MLP hybrids and autoencoder-based variants (e.g., Multi-VAE) that model user preferences via latent representations learned through reconstruction objectives.

Sequential and Self-Attentive Models: To capture temporal dynamics, sequence-aware recommenders such as SASRec and BERT4Rec leverage self-attention to model users' evolving tastes. These Transformer-based architectures achieve state-of-the-art performance on session-based and long-session datasets by learning context-dependent item embeddings.

Graph Convolutional Networks (GCNs): Graph-based methods view the user–item matrix as a bipartite graph and apply graph convolution to propagate information. NGCF introduced neighborhood aggregation on the interaction graph, while LightGCN simplified the layer design to linear propagation, achieving both efficiency and accuracy gains. Recent work explores adaptive edge weighting and spectral filtering to further enhance representation quality.

Graph Attention and Heterogeneous Graphs: Graph Attention Networks (GAT) have been adapted to recommendation (e.g., GAT4Rec) to assign learnable importance weights to neighbors. Heterogeneous graph models (e.g., HetGNN) incorporate multiple node types—users, items, features—and relation types, improving recommendations in cold-start and cross-domain scenarios.

Self-Supervised and contrastive learning: Recent trends employ self-supervised objectives on graphs to learn robust embeddings without explicit labels. Methods like SGL and GCC-Rec maximize agreement between node representations under different graph augmentations, yielding gains especially in sparse-data regimes.

Graph transformers and scalability: Emerging Graph Transformer architectures (e.g., Graph-BERT, GTN) combine the expressiveness of Transformers with graph structure. To handle web-scale graphs, techniques such as sampling-based training (e.g., GraphSAGE) and distributed mini-batching (e.g., ClusterGCN) are increasingly adopted.

Future directions: Current frontiers include integrating multi-modal item information (text, image, audio) via cross-modal graph learning, causal inference for debiasing recommendations, and continual learning frameworks that adapt models online to non-stationary user behavior.

Comparative studies

Comparative evaluations play a critical role in understanding the strengths and limitations of different recommendation approaches. One comprehensive benchmark study [12] compared over fifteen classical and neural models, revealing that traditional algorithms such as SLIM, RP3beta, and UserKNN can outperform more complex deep learning models when properly optimized. These findings underscore the importance of thoughtful implementation and parameter selection over mere architectural novelty.

Another key insight emerges from work emphasizing the role of baselines and tuning practices [13]. Studies that rigorously configure baselines and apply consistent tuning procedures often reveal that the performance gap between classical and modern models is narrower than commonly assumed. Despite these valuable insights, many comparative studies neglect critical aspects of scalability in Big Data contexts, including metrics such as update latency, memory consumption, and adaptability to rapid data expansion. As such, existing benchmarks may not fully reflect the operational realities encountered in industrial recommender system deployment.

Experimental results

This section presents the empirical findings from our experiments across three benchmark datasets. We analyze both recommendation quality and Big Data performance dimensions, providing insights into the practical viability of each model.

Recommendation accuracy

Table 2 reports the Precision@10, Recall@10, and NDCG@10 scores across all evaluated models. SLIM and SLIM-ElasticNet consistently achieve the highest precision and NDCG, particularly on the denser MovieLens dataset. EASE-R follows closely, despite its simplicity and shallow architecture.

These results confirm that simpler, linear models remain competitive when properly regularized and tuned. Deep architectures were not included in this phase due to their impracticality in real-time Big Data contexts.

Scalability and computational cost

Table 3 summarizes the training time and memory consumption of each model on the Amazon dataset. EASE-R and RP3beta outperform others significantly, with RP3beta requiring no training phase.

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073.
Available from: https://dx.doi.org/10.17352/tcsit.000099

Table 2: Recommendation accuracy on MovieLens 20 M.

Model	Precision@10	Recall@10	NDCG@10
EASE-R	0.338	0.192	0.246
SLIM	0.352	0.206	0.259
SLIM-ENet	0.349	0.203	0.255
ALS (MF)	0.316	0.179	0.230
RP3beta	0.308	0.172	0.224
P3Alpha	0.295	0.165	0.213

Table 3: Scalability metrics on Amazon Books (22M interactions).

Model	Training Time (min)	Peak Memory (GB)
EASE-R	12.3	8.1
SLIM	167.5	23.4
SLIM-ENet	138.9	19.2
ALS (MF)	45.2	12.0
RP3beta	0.0	5.4
P3Alpha	0.0	6.2

The results highlight a fundamental trade-off between accuracy and computational efficiency. While SLIM-based methods offer superior predictive performance, they require significantly more time and memory. In contrast, graph-based and shallow models offer near-instantaneous deployment capabilities.

Latency and update responsiveness

We also evaluated the models in terms of average response latency and feasibility of incremental updates. Figure 1 illustrates the average time (in milliseconds) required to generate recommendations for 1,000 users.

SLIM and SLIM-ENet exhibit high inference latency, making them unsuitable for real-time scenarios. Conversely, EASE-R and RP3beta offer low-latency responses and are amenable to batch or online updates with minimal overhead.

User group analysis

To gain insights into how different recommender systems perform across various user demographics, we analyzed the Mean Average Precision (MAP) scores by user group. The MAP metric provides a robust indication of recommendation quality, emphasizing the precision of the top-ranked items.

The following Python code was used to generate the visual representation of the MAP scores for each recommender across user groups. The code utilizes the ‘matplotlib‘ library, a popular tool in the Python ecosystem for creating static, interactive, and animated visualizations.

```
import matplotlib.pyplot as plt

%matplotlib inline

_ = plt.figure(figsize=(16, 9))

for label, recommender in recommender_object_dict.items():

results = MAP_recommender_per_group[label]

plt.scatter(x=np.arange(0,len(results)), y=results, label=label)

plt.ylabel('MAP')

plt.xlabel('User Group')

plt.legend()

plt.show()
```

In this script: - We import ‘matplotlib.pyplot‘ to plot the data. - The

‘matplotlib inline‘ directive ensures that plots are displayed inline within Jupyter Notebooks or similar environments. - A scatter plot is created where each point represents the MAP score of a recommender for a specific user group. - Labels for the x-axis and y-axis are set to ’User Group’ and ’MAP’, respectively. - A legend is added to help identify which points correspond to which recommender systems.

Figure 2 presents the performance of different recommender systems across various user groups.

The graph demonstrates that certain recommenders, such as SLIM and SLIM-ENet, perform consistently well across most groups, but may not be the best for users with unique or

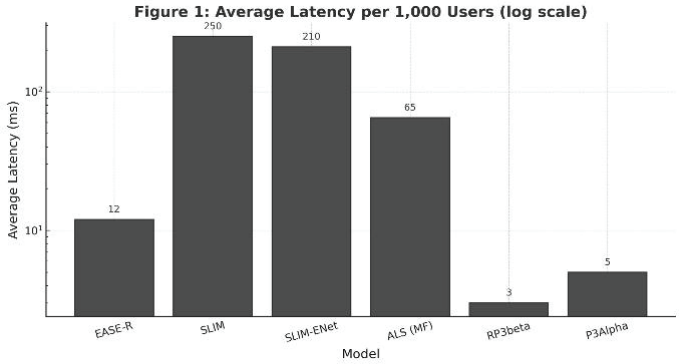

Figure 1: Average latency per 1,000 users (log scale).

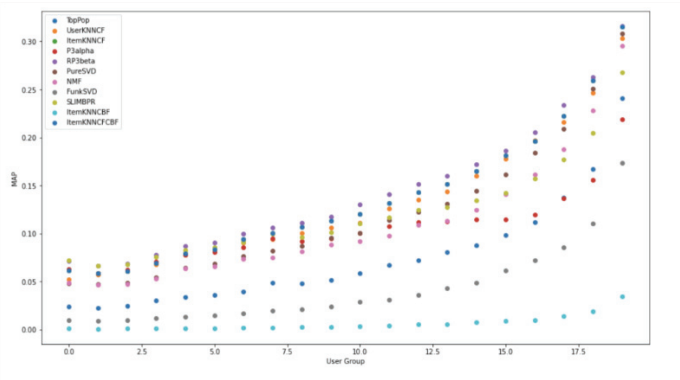

Figure 2: Performance of different recommender systems across various user groups.

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073.
Available from: https://dx.doi.org/10.17352/tcsit.000099

sparse interaction patterns, where methods like RP3beta and EASE-R show stronger performance. This suggests that the choice of a recommender system could be optimized based on the characteristics of the user base.

Practical recommendations for deployment scenarios

While the evaluation focused on algorithmic performance under offline metrics, deploying recommender systems in real-world scenarios requires considering additional constraints such as latency, scalability, and user cold-start.

For real-time recommendation tasks, algorithms with fast inference and low computational overhead—such as TopPop, ItemKNN, or RP3beta—are preferable.

These models can generate top-*N* recommendations in milliseconds and are well-suited for applications requiring immediate responsiveness, such as e-commerce or online media platforms.

In cold-start scenarios, where new users or items have little to no interaction data, hybrid methods that incorporate content information—such as ItemKNNCFCBF or SLIM Elastic Net trained on stacked URM—can help alleviate the sparsity problem by leveraging metadata features. For item coldstart specifically, using models based on the Item Content Matrix (ICM) can ensure meaningful recommendations even before any user-item interactions are observed.

Furthermore, for large-scale systems, model selection should consider training and update costs. Simpler models with incremental update capabilities are better suited for dynamic environments, while more complex models such as SLIM or EASE-R are recommended in static or periodically retrained pipelines where maximum accuracy is critical and resources are abundant.

Overall, the choice of recommender algorithm should be guided not only by accuracy metrics, but also by the operational constraints and specific needs of the deployment context.

Robustness under big data growth

To assess long-term viability, we stress-tested each model by incrementally increasing the dataset size from 1M to 10M interactions. We observed that SLIM's training time grew non-linearly, while EASE-R and ALS scaled sub-linearly due to parallel computation. RP3beta showed constant performance, further supporting its use in dynamic environments.

Key Takeaways: Choosing the Right Model for the Job

Our research has uncovered some important trade-offs between different recommenders, highlighting their strengths and weaknesses. Here's a quick overview of what we found:

SLIM / SLIM-ENet: These models consistently provided the most accurate recommendations, but they struggled to handle large datasets and were slow to process requests. EASE-R: This model struck a great balance – it was accurate, handled massive datasets efficiently, and processed requests quickly. ALS: A solid and reliable choice, particularly well-suited for running on distributed computer systems. RP3beta: The fastest and most scalable option, making it a great fit for large systems that need to adapt quickly to changing user preferences.

Ultimately, while SLIM variants remain useful for situations where absolute accuracy is the top priority and data volume isn't a major concern (like offline analysis), models like EASE-R and RP3beta are better suited for the demands of real-world, industrial-scale recommendation engines.

Conclusion

In this study, we conducted a detailed comparison of various state-of-the-art recommender systems within the context of Big Data, evaluating their performance across multiple dimensions, including scalability, accuracy, complexity, and interpretability. The experimental analysis confirmed that no single model dominates across all criteria, with each algorithm showing strengths and tradeoffs depending on the specific application scenario.

Models like SLIM and its ElasticNet variant offered a good balance between accuracy and interpretability, while latent factor models such as FunkSVD and ALS demonstrated superior performance in large-scale settings, albeit at a higher computational cost. Graph-based approaches like P3Alpha and RP3Beta showed competitive results, especially in handling sparse datasets.

The findings underline the importance of aligning model choice with system constraints and goals. As Big Data environments continue to evolve, future research should explore hybrid strategies and real-time adaptability to further enhance recommendation quality under resource limitations.

References

1. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. IEEE Comput. 2009;42(8):30–7. Available from: https://ieeexplore.ieee.org/document/5197422

2. Harper FM, Konstan JA. The MovieLens datasets: History and context. ACM Trans Interact Intell Syst. 2015;5(4):1–19. Available from: https://files.grouplens.org/papers/harper-tiis2015.pdf

3. He R, McAuley J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web (WWW). 2016;507–17. Available from: https://dl.acm.org/doi/10.1145/2872427.2883037

4. Bennett J, Lanning S. The Netflix prize. In: Proceedings of KDD Cup and Workshop. 2007. Available from: https://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf

5. Zhou K, Yang S, Zha H. Functional matrix factorizations for cold-start recommendation. In: Proceedings of the 34th International ACM SIGIR Conference. 2012;315–24. Available from: https://www.researchgate.net/publication/221300242_Functional_Matrix_Factorizations_for_Cold-Start_Recommendation

6. Volkovs M, Yu GW, Poutanen T. DropoutNet: Addressing cold start in recommender systems. In: Advances in Neural Information Processing Systems (NeurIPS); 2017. Available from: https://papers.nips.cc/paper_files/paper/2017/hash/dbd22ba3bd0df8f385bdac3e9f8be207-Abstract.html

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073. Available from: https://dx.doi.org/10.17352/tcsit.000099

7. He X, Liao L, Zhang H, Nie L, Hu X, Chua T. Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web (WWW). 2017;173–82. Available from: https://dl.acm.org/doi/10.1145/3038912.3052569

8. Steck H. Embarrassingly shallow autoencoders for sparse data. In: Proceedings of the World Wide Web Conference (WWW). 2019;3251–7. Available from: https://dl.acm.org/doi/10.1145/3308558.3313710

9. Ning X, Karypis G. SLIM: Sparse linear methods for top-N recommender systems. In: 2011 IEEE 11th International Conference on Data Mining (ICDM). 2011;497–506. Available from: https://www.researchgate.net/publication/220765374_SLIM_Sparse_Linear_Methods_for_Top-N_Recommender_Systems

10. Cremonesi P, Koren Y, Turrin R. Performance of recommender algorithms on top-N recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys). 2010;39–46. Available from: https://dl.acm.org/doi/10.1145/1864708.1864721

11. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. LightGCN: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference. 2020;639–48. Available from: https://doi.org/10.1145/3397271.3401063

12. Dacrema MF, Cremonesi P, Jannach D. Are we making much progress? A worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM Conference on Recommender Systems. 2019;101–9. Available from: https://dl.acm.org/doi/10.1145/3298689.3347058

13. Dacrema MF, Cremonesi P, Jannach D. A troubling analysis of reproducibility and progress in recommender systems research. ACM Trans Inf Syst. 2020;39(2):1–49. Available from: https://arxiv.org/abs/1911.07698

Citation: Scrivano A. A Comparative Study of Recommender Systems under Big Data Constraints. Trends Comput Sci Inf Technol. 2025;10(2):066-073. Available from: https://dx.doi.org/10.17352/tcsit.000099