Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Adaptive Implicit Feedback Weighting: Addressing Signal Saturation and User Heterogeneity in Collaborative Filtering

A. E. Shyraliiev

anarshyraliiev@gmail.com

National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

I. O. Pyshnograiev

National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Research Article

Keywords: Recommender Systems, Collaborative Filtering, Matrix Factorization, Implicit Feedback, Adaptive Weighting

Posted Date: March 31st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9216103/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Adaptive Implicit Feedback Weighting: Addressing Signal Saturation and User Heterogeneity in Collaborative Filtering

A. E. Shyraliiev[1], I. O. Pyshnograiev[2]

[1,2]Department of Artificial Intelligence, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

E-mails: [1]anarshyraliiev@gmail.com ORCID 0009-0007-4140-5476

[2]pyshnograiev@gmail.com ORCID 0000-0002-3346-8318

Abstract

Recommender systems increasingly rely on implicit feedback (e.g., clicks, play counts, watch times) due to the scarcity of explicit ratings. However, standard collaborative filtering models, such as Weighted Matrix Factorization, traditionally assume that a user's preference grows continuously and unboundedly with raw interaction counts. This "one-size-fits-all" assumption ignores the severe distributional skew caused by signal saturation and user heterogeneity, where identical interaction volumes can represent vastly different preference intensities. In this paper, we challenge static scaling assumptions by proposing a suite of dynamic, adaptive confidence weighting strategies designed to stabilize latent representations. We introduce the Power-Lift strategy, a numerically stable generalization of Pointwise Mutual Information that captures the probabilistic surprise of interactions, alongside Robust User-Centric weighting, which normalizes frequencies utilizing local, non-parametric user statistics. Furthermore, we formalize the Saturation Hypothesis, demonstrating that sigmoid-based limits on confidence successfully prevent extreme behavioral outliers from distorting latent space geometries. Through rigorous offline evaluation across five diverse datasets, we show that our adaptive weighting approaches consistently outperform traditional unweighted baselines and standard Information Retrieval scaling methods. Notably, our Power-Lift strategy achieves up to a 54.3% relative improvement in Normalized Discounted Cumulative Gain at rank 20 on the highly skewed Steam dataset. Ultimately, these methods provide a highly scalable, computationally efficient mechanism to boost top-N recommendation accuracy and reduce required model capacity without altering downstream architectural inference latency.

Keywords: Recommender Systems, Collaborative Filtering, Matrix Factorization, Implicit Feedback, Adaptive Weighting

1 Introduction

Today, digital Recommender Systems rarely rely on explicit feedback, like five-star ratings or thumbs-up buttons [1]–[4]. Instead, they use implicit feedback, such as clicks, purchase history, and watch times. This data is easy to collect but much harder to understand. Unlike explicit ratings that clearly show what a user likes or dislikes, implicit feedback only shows that an interaction happened. It does not prove the user actually liked the item, and a lack of interaction just means they might not have seen it. This creates a basic problem we call the "Implicit Gap": a large difference between how much a user interacts with an item and how much they actually like it.

To solve this problem, earlier research established the Weighted Matrix Factorization (WMF) model [5]–[7]. This method separates the binary fact that an interaction happened from the model's confidence in that interaction. It assigns a continuous confidence weight, w_{ui}, to every observed interaction r_{ui} between a user u and an item i. Standard WMF models assume that confidence grows steadily as the interaction count goes up. Usually, this is calculated using simple linear multipliers (e.g., $1 + \alpha \cdot r_{ui}$) or logarithmic scaling.

However, these standard weighting strategies rely on a flawed assumption. They assume that a specific number of interactions, like "ten plays" of a song or "twenty hours" in a video game, means the exact same thing for every user. This "one-size-fits-all" approach ignores two important factors: User Heterogeneity and Signal Saturation. User Heterogeneity means that ten interactions from a "power user" might just be background noise, while ten interactions from a "casual user" might mean they love the item. Standard models treat both situations exactly the same, which drowns out casual users and overfits to the power users. Furthermore, standard linear and log formulas fail to capture the "Saturation Point." We introduce the Saturation Hypothesis: there is a point where more interactions do not mean more preference. After a certain threshold, the user isn't showing active interest anymore; they are just consuming the item out of habit or leaving it on in the background.

In this paper, we argue that confidence weighting must be dynamic and adaptive, rather than using a single global formula. In practical terms, adaptive weighting is a very fast and effective way to boost the retrieval and collaborative filtering components of modern recommender systems without slowing them down.

Our main contributions are as follows:

- The Power-Lift Weighting Strategy: we introduce a generalized Pointwise Mutual Information (PMI) framework that measures the probabilistic "surprise" of an interaction. This approach avoids the mathematical errors and artificial cutoffs that typically cause standard PMI models to fail.
- Robust User-Centric Strategy: we develop a method that uses local user statistics, specifically the median and interquartile range (IQR), to scale confidence weights. This strategy normalizes data based on a user's personal habits, effectively ignoring global outliers and extreme shifts in behavior.
- Evidence for Saturation Hypothesis: we formalize the concept of signal saturation through contextual dampening methods like the Log-IDF strategy. Furthermore, we mathematically demonstrate that applying a sigmoid-based limit on confidence helps stabilize models when processing noisy data, outperforming unbounded logarithmic scaling.
- Architectural Efficiency: we demonstrate that applying appropriate mathematical weighting can effectively reduce the required capacity of a model. By filtering out noisy data prior to training, lightweight models can achieve competitive or superior predictive performance compared to unweighted baseline models that use significantly more parameters.
- Empirical Validation: we demonstrate that these adaptive weighting methods yield substantial improvements in ranking metrics across five diverse datasets. Notably, the Power-Lift strategy achieves up to a 54.3% relative improvement in ranking accuracy on the highly skewed Steam dataset.

2 Theoretical framework and related work

2.1 The Foundation of weighted matrix factorization

The math behind implicit recommendations is rooted in WMF [5]. Because most of the implicit interaction matrix is empty, WMF breaks the raw data down into two parts: a binary preference variable p_{ui} and a confidence weight w_{ui}. Let r_{ui} be the raw interaction count (like total clicks or hours played) between user $u \in U$ and item $i \in I$. The preference variable p_{ui} is strictly binary: it equals 1 if $r_{ui} > 0$ and 0 if $r_{ui} = 0$.

The key feature of WMF is its loss function, which minimizes the confidence-weighted squared error across all observed and unobserved interactions:

$$\min_{x_u, y_i} \sum_{u,i} w_{ui}(p_{ui} - {x_u}^T y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2), \tag{1}$$

where $x_u, y_i \in \mathbb{R}^k$ are the latent feature vectors for the user and item in a k-dimensional space, and λ is the L2 regularization parameter used to prevent overfitting. In this setup, w_{ui} decides how much each interaction influences the model during training. The original WMF used linear global weighting, written as $w_{ui} = 1 + \alpha \cdot r_{ui}$, where α controls how fast confidence grows. While this is easy to compute, it assumes confidence can grow forever. Under a linear model, a user who interacts with an item 5,000 times will pull the item's vector 5,000 times harder, completely distorting the model to fit extreme outliers.

2.2 Baseline heuristics

Beyond basic linear scaling, several standard mathematical transformations are frequently used as baselines to manually control interaction frequency:

- Logarithmic Smoothing (Log): to dampen the influence of extreme outliers, a simple log transformation is often applied [6]:

$$w_{ui} = \log(1 + r_{ui}). \tag{2}$$

While this slows the growth of confidence, the function remains mathematically unbounded and never truly plateaus.

- Power Weighting (Power): this strategy uses a fractional exponent p to scale the raw count:

$$w_{ui} = (r_{ui})^p. \tag{3}$$

It acts as a tunable middle ground between linear scaling ($p = 1$) and extreme dampening. It is inspired by similar subsampling strategies in neural embedding learning [8], where dampening high-frequency tokens via a fractional exponent (e.g., $p = 0.75$) has been shown to improve representation quality by reducing the dominance of global stop-words.

- Unit Normalization (Normalization): adapted from standard vector space models [9], this strategy applies L2 row normalization to the interaction matrix:

$$w_{ui} = \frac{r_{ui}}{\sqrt{\sum_j (r_{uj})^2}}. \tag{4}$$

By forcing every user's interaction vector to a length of 1, it completely removes the magnitude of overall interaction volume, testing if relative distribution alone is enough to capture user preference.

2.3 Information retrieval baselines

To fix the problems caused by linear scaling, researchers adapted techniques from Information Retrieval (IR), treating users like "documents" and items like "terms". The most common of these is Term Frequency-Inverse Document Frequency (TF-IDF) [10], [11]. Using the following formula, the system keeps the interaction frequency but heavily penalizes items that are universally popular:

$$w_{ui} = r_{ui} \cdot IDF_i \,. \tag{5}$$

The Inverse Document Frequency is calculated as:

$$IDF_i = \log\left(\frac{|U|}{|U_i|}\right), \tag{6}$$

where $|U|$ is the total number of users and $|U_i|$ represents the number of users who interacted with item i. This helps stop common items from dominating everyone's recommendations. However, TF-IDF still uses a linear term frequency (r_{ui}), so it still suffers from signal saturation. A massive r_{ui} value will easily overwhelm the IDF_i penalty.

A more advanced IR method is Best Matching 25 (BM25) [12], [13]. In recommender systems, BM25 scales the raw interaction count based on how much a specific user interacts compared to the average user. This uses saturation parameters (k_1) and length normalization (b) via the following confidence weight calculation:

$$w_{ui} = \frac{r_{ui}(k_1 + 1)}{r_{ui} + k_1(1 - b + b \cdot \frac{L_u}{L_{avg}})} \cdot IDF_i \,, \tag{7}$$

where $L_u = \sum_j r_{uj}$ represents the total interaction volume of user u, and L_{avg} is the average activity level across all users. In theory, BM25 solves User Heterogeneity by penalizing super-active users so their massive data doesn't erase the preferences of quieter users. However, BM25 requires careful tuning, and it often struggles with behavioral datasets where "document length" translates to thousands of continuous hours rather than a simple word count.

2.4 Pointwise mutual information

Looking for an alternative to raw frequency, researchers also adapted PMI from natural language processing [14]–[17]. PMI measures how much the interaction between a user and an item deviates from random chance. The confidence weight is the log ratio of the expected joint probability of the interaction to their independent probabilities, which equals the logarithm of the statistical Lift:

$$\mathrm{PMI}_{u,i} = \log\big(\mathrm{Lift}(u,i)\big) = \log\left(\frac{P(u,i)}{P(u)P(i)}\right). \tag{8}$$

To calculate this directly from the raw interactions r_{ui}, the probabilities are estimated using the interaction frequencies relative to the total volume of all interactions in the dataset, $N = \sum_u \sum_i r_{ui}$. Specifically, the joint probability is $P(u,i) = r_{ui}/N$, the user probability is $P(u) = \sum_j r_{uj}\,/N$, and the item probability is $P(i) = \sum_v r_{vi}\,/N$. Substituting these into the formula yields:

$$w_{ui} = \log\left(r_{ui} \cdot \frac{\sum_u \sum_i r_{ui}}{(\sum_j r_{uj})(\sum_v r_{vi})}\right). \tag{9}$$

This represents the statistical "surprise" of the interaction, a concept that has been successfully connected to implicit matrix factorization. If a highly inactive user interacts heavily with an obscure item, the probability of that happening by chance is tiny. Therefore, the formula outputs a disproportionately high confidence weight, signaling a very strong preference. While PMI is great at finding true signals, it is mathematically unstable in matrix factorization. Because logarithms can produce negative numbers when an interaction happens less often than random chance, standard algorithms require strict zero-clipping (turning negative values to zero) to establish the final confidence weight: $w_{ui} = \max(0, \mathrm{PMI}_{u,i})$. This creates

artificial cutoffs and introduces mathematical discontinuities into the gradient updates required by the model. This harsh cutoff motivates the need for better, continuous lift-based formulas.

3 Proposed weighting strategies

To fix the limitations of static formulas and unstable probabilistic methods, we propose several advanced strategies. We group these into three categories: Global Saturation, Local Contextualization, and Statistical Lift.

3.1 Saturation-based strategies (global)

This group fixes the linear scaling problem by forcing saturation, matching the natural limits of human attention.

The Log-IDF Weighting strategy combines logarithmic dampening with a penalty for popular items:

$$w_{ui} = 1 + \alpha \cdot \log(1 + r_{ui}) \cdot IDF_i \,. \tag{10}$$

It assumes that while confidence grows with volume, each new interaction matters a little less than the last. However, standard log functions never truly stop growing, so they fail to capture an absolute psychological limit.

To enforce a hard limit, we propose Sigmoid Propensity Weighting. This strategy uses an asymptotic limit, creating a "soft cap" that filters out noise from background consumption. The formula is:

$$w_{ui} = \frac{1}{1 + e^{-\gamma(\log r_{ui} - \mu)}} \cdot (1 + \beta \cdot IDF_i) \,, \tag{11}$$

where μ sets the exact point of saturation. In our implementation, μ is dynamically calculated as the mean of the log-transformed non-zero interaction counts across the entire dataset ($\mu = \text{mean}(\log r_{ui})$). Furthermore, γ controls how fast the transition happens, and β decides how much to penalize popular items. By setting a strict maximum weight, the model is completely protected from extreme high-frequency outliers.

3.2 Adaptive and contextual strategies (local)

Global models forget that user psychology is relative. This category shifts the focus from the whole dataset strictly to the individual user.

The Robust User-Centric Weighting strategy ignores global averages entirely. Instead, it uses robust non-parametric statistics, the median and the IQR, calculated only on the specific user's non-zero interactions, $R_u = \{r_{ui} | r_{ui} > 0\}$ [18]. Standard methods (like Z-scores) break if a user has one crazy weekend of activity. By using medians, we propose this formula:

$$w_{ui} = \text{Sigmoid}\left(\frac{r_{ui} - \text{Median}(R_u)}{\text{IQR}(R_u)}\right). \tag{12}$$

This normalizes the user's specific action against their own normal behavior. A casual user who engages 10 times gets a massive weight if their personal median is 2. A power user who engages 10 times gets a low weight if their median is 50. This turns absolute volume into accurate relative preference, protecting the model from user differences.

3.3 Statistical surprise and lift strategies

To capture the "surprise" of an event without introducing mathematical discontinuities like standard PMI does, we introduce Power-Lift Weighting. Instead of using logarithms, Power-Lift uses a continuous power-law expansion:

$$w_{ui} = \left(r_{ui} \cdot \frac{\sum_u \sum_i r_{ui}}{(\sum_j r_{uj})(\sum_v r_{vi})}\right)^p . \tag{13}$$

Notice that the inner term of this equation mathematically reduces to $\text{Lift}(u, i)$, which measures the ratio of the observed joint probability to the expected independent probabilities. Here, the exponent $p > 0$ acts as a tunable dial that controls the lift amplification. As p increases, the strategy heavily punishes common, noisy interactions while exponentially rewarding rare, surprising interactions. Because the base of the function is strictly positive, it avoids computational singularities, ensuring stable model training.

4 Methods

To prove our adaptive weighting schemas work and to test the Saturation Hypothesis, we built a highly reproducible testing pipeline. A key part of this was choosing the right baseline algorithms and explicitly rejecting popular neural networks that handle input data differently.

4.1 Evaluated models and the disqualification of neural baselines

Our experiments use Alternating Least Squares (ALS) [9], [19] and Memory-based K-Nearest Neighbors (KNN) using cosine similarity [20]–[22]. For highly optimized execution, we utilize the fast, multi-threaded implicit library [23].

We deliberately exclude modern autoencoders like Embarrassingly Shallow Autoencoders ($EASE^R$) [24] and deep generative models like Variational Autoencoders for Collaborative Filtering (VaeCF) [25] because they are designed strictly for binary data (1s and 0s). If you feed $EASE^R$ highly skewed continuous counts (e.g., 5,000 hours played), it ruins its L2 regularization, causing catastrophic overfitting. Similarly, VaeCF flattens input layers to binary occurrences; forcing it to ingest massive variance destroys its loss function. In contrast, ALS handles continuous numbers perfectly as bounded confidence parameters (w_{ui}) separated from the binary preference (p_{ui}). Likewise, KNN can process our weighted matrices before calculating distances, ensuring the cosine similarity accurately reflects the scaled preferences. Therefore, ALS and KNN are structurally well-suited for evaluating continuous weighting strategies.

Furthermore, the exclusion of complex deep learning and graph-based architectures is driven by practical compute limitations in large-scale candidate retrieval. Recent comprehensive benchmarking of collaborative filtering methods [26] has demonstrated that while advanced neural architectures, such as Neural Graph Collaborative Filtering [27] and Light Graph Convolution Network [28], offer strong performance in dense scenarios, they face severe scalability constraints on larger datasets. Specifically, the training of these computationally intensive methods becomes entirely infeasible due to prohibitively long runtimes when applied to massive or highly sparse datasets [26]. Given these fundamental computational bottlenecks, enhancing the data ingestion of highly efficient, traditional algorithms like ALS and KNN via adaptive weighting provides a scalable and practical alternative to adopting increasingly complex and computationally expensive architectures.

4.2 Datasets and density mechanics

To ensure our models work across different environments, we tested them on five real-world datasets [29]–[32] with varying levels of noise and density. An overview of the dataset statistics is provided in Table 1.

Dataset Justification: we use the 1-5 ratings in the MovieLens datasets as continuous, “capped implicit” feedback. This gives us a baseline where maximum interaction is biologically limited (you can’t rate a movie higher than a 5). We contrast this sharply with the Last.fm 360k, Taste Profile, and Steam datasets. In these domains, interactions are continuous and technically endless (audio streams and gameplay hours). Because users can leave a game or playlist running idle for thousands of hours, these datasets require robust outlier suppression.

Table 1 Experimental data statistics

Dataset	Users #	Items #	Interactions #	Density	Feedback Type
MovieLens 100k	943	1,682	100,000	6.30%	Capped Implicit (1-5)
MovieLens 20M	138,493	26,744	20,000,263	0.54%	Capped Implicit (1-5)
Last.fm 360k	359,347	294,015	17,559,530	0.0166%	Play Counts
Taste Profile	1,019,318	384,546	48,373,586	0.0123%	Play Counts
Steam	13,781,059	37,610	41,154,773	0.0079%	Hours Played

4.3 Data partitioning and evaluation metrics

Preventing data leaks is critical for testing recommender systems. We use a strict nested splitting method with a fixed random seed to guarantee the results are perfectly reproducible.

The original matrix is split, holding exactly 10% of the data as a final test set. The remaining 90% is split again (90/10). This leaves us with exactly 81% of the data for training, 9% for hyperparameter validation, and 10% for final testing. Importantly, the matrix shape remains exactly the same across all three sets so the indexes always match.

We measure success using Normalized Discounted Cumulative Gain at rank 20 (NDCG@20) [33]. This metric rewards the model for finding relevant items and placing them high on the recommendation list. We also report Precision@20 to show how stable the recommendation lists are [34].

4.4 Hyperparameter optimization and parameter freezing

Because weighting models rely heavily on their specific tuning variables, testing them with random static parameters is unfair. To fix this, we tuned all tunable strategies using the Optuna optimization framework [35], leveraging its Tree-structured Parzen Estimator algorithm [36].

Optuna ran 20 trials for each strategy to find the exact parameters that maximized NDCG@20 on the validation set. Simple models without parameters, like TF-IDF or basic Log, were run once as baselines. Only the best parameters discovered are reported in the results.

Crucially, to make sure we were only measuring the effect of the weighting strategies, the core settings of the ALS and KNN algorithms were frozen. For ALS, regularization was locked at $\lambda = 10$, and we tested latent factors at exactly $k = 10$ and $k = 100$. For KNN, neighborhoods were locked at $K = 20$ and $K = 100$. This guarantees that any change in accuracy is completely due to the weighting math.

5 Results and analysis

To keep the analysis focused, Sections 5.1 through 5.5 evaluate the high-capacity models: ALS with $k = 100$ and KNN with $K = 100$. Section 5.6 looks at how smaller, lightweight models perform.

5.1 Last.fm 360k: combating ambient consumption

In the Last.fm dataset, users often leave music running in the background. Because of this noise, bounded and normalized weights performed best, as detailed in Table 2. The raw, unweighted baselines performed terribly for both ALS and KNN, sitting near the bottom of the charts. For ALS, the BM25 model achieved the highest NDCG@20 (0.1931), proving

Table 2 Performance comparison on the Last.fm 360k dataset

Model	Strategy	NDCG@20	Precision@20	Weighting (s)	Training (s)	Best Parameters
ALS	BM25	**0.1931**	<u>0.2508</u>	0.66	10.66	k_1: 2.06, b: 0.32
ALS	Log-IDF	<u>0.1917</u>	**0.2515**	0.53	10.57	α: 2.06
ALS	PMI	0.1852	0.2403	0.73	10.56	-
ALS	Power-Lift	0.1818	0.2374	0.33	10.57	p: 0.33
ALS	TF-IDF	0.1801	0.2433	0.50	10.62	-
ALS	Power	0.1785	0.2354	0.37	10.65	p: 0.57
ALS	Sigmoid Propensity	0.1726	0.2244	0.34	10.59	γ: 1.95, β: 0.99
ALS	Robust User-Centric	0.1725	0.2243	2.60	10.55	-
ALS	Log	0.1666	0.2170	0.33	10.58	-
ALS	*No Weighting*	*0.1454*	*0.2075*	-	*10.58*	-
ALS	Normalization	0.0926	0.1336	0.50	10.58	-
KNN	BM25	**0.1728**	<u>0.2203</u>	0.66	5.14	k_1: 334.86, b: 0.99
KNN	PMI	**0.1728**	0.2197	0.73	5.09	-
KNN	Robust User-Centric	<u>0.1725</u>	**0.2204**	2.62	5.08	-
KNN	Power-Lift	0.1718	0.2194	0.32	5.11	p: 0.34
KNN	Normalization	0.1690	0.2155	0.50	5.06	-
KNN	Power	0.1687	0.2164	0.37	5.08	p: 0.15
KNN	Sigmoid Propensity	0.1684	0.2161	0.34	5.12	γ: 0.25, β: 0.94
KNN	Log	0.1683	0.2160	0.33	5.11	-
KNN	Log-IDF	0.1683	0.2160	0.54	5.13	α: 3.47
KNN	TF-IDF	0.1667	0.2132	0.50	5.12	-
KNN	*No Weighting*	*0.0966*	*0.1287*	-	*5.18*	-

Note: For each model and metric, bold indicates the best-performing weighting, and underlined indicates the second-best.

that scaling down a user's total listening time relative to the global average is a great way to ignore background listening noise. However, the analysis of the KNN model reveals a more nuanced result: BM25 and PMI actually tied for the highest NDCG@20 at 0.1728. While BM25 won overall in NDCG for ALS, our proposed Log-IDF strategy actually secured the highest Precision@20 (0.2515) for the ALS model. Furthermore, while our Robust User-Centric strategy was highly

competitive in NDCG for KNN, it actually outperformed BM25 slightly in Precision@20 (0.2204 compared to 0.2203). Conversely, strict unit normalization ruined ALS performance completely (0.0926 NDCG), showing that if you erase the magnitude of interaction entirely, you destroy the model's ability to learn.

5.2 MovieLens 100k: stabilizing dense domains

Because MovieLens relies on discrete 1-5 ratings, it acts as a capped baseline. Unsurprisingly, and as shown in Table 3, strategies that enforce an absolute limit won out here. For ALS, Sigmoid Propensity was the dominant winner (0.3623 NDCG), closely followed by Robust User-Centric (0.3594 NDCG). Capping the influence of users perfectly stabilizes dense matrices. On the other hand, traditional global methods like Log-IDF (0.3102 NDCG) and TF-IDF (0.3138 NDCG) actually performed worse than the raw unweighted baseline (0.3188 NDCG) for ALS.

Table 3 Performance comparison on the MovieLens 100k dataset

Model	Strategy	NDCG@20	Precision@20	Weighting (s)	Training (s)	Best Parameters
ALS	Sigmoid Propensity	**0.3623**	**0.3964**	< 0.01	0.21	γ: 2.21, β: 0.29
ALS	Robust User-Centric	<u>0.3594</u>	<u>0.3935</u>	< 0.01	0.21	-
ALS	Power	0.3566	0.3855	< 0.01	0.24	p: 0.10
ALS	Power-Lift	0.3548	0.3859	< 0.01	0.21	p: 0.11
ALS	Log	0.3473	0.3774	< 0.01	0.21	-
ALS	PMI	0.3261	0.3416	< 0.01	0.22	-
ALS	*No Weighting*	*0.3188*	*0.3411*	-	*0.25*	-
ALS	TF-IDF	0.3138	0.3352	< 0.01	0.22	-
ALS	Log-IDF	0.3102	0.3298	< 0.01	0.25	α: 1.27
ALS	BM25	0.2988	0.3177	< 0.01	0.21	k_1: 984.84, b: 0.01
ALS	Normalization	0.1543	0.1941	< 0.01	0.18	-
KNN	Power-Lift	**0.3142**	<u>0.3343</u>	< 0.01	0.02	p: 1.16
KNN	BM25	<u>0.3098</u>	0.3308	< 0.01	0.02	k_1: 338.75, b: 0.99
KNN	PMI	0.3097	**0.3355**	< 0.01	0.02	-
KNN	Normalization	0.2950	0.3132	< 0.01	0.02	-
KNN	Robust User-Centric	0.2674	0.2869	< 0.01	0.04	-
KNN	Log-IDF	0.2653	0.2882	< 0.01	0.02	α:17.62
KNN	Log	0.2651	0.2881	< 0.01	0.02	-
KNN	*No Weighting*	*0.2650*	*0.2880*	-	*0.03*	-
KNN	TF-IDF	0.2649	0.2873	< 0.01	0.04	-
KNN	Power	0.2649	0.2877	< 0.01	0.04	p: 0.63
KNN	Sigmoid Propensity	0.2618	0.2819	< 0.01	0.02	γ: 0.28, β: 0.58

Note: For each model and metric, bold indicates the best-performing weighting, and underlined indicates the second-best.

For the KNN model, Power-Lift delivered the strongest overall ranking performance (0.3142 NDCG), followed closely by BM25 (0.3098 NDCG) and PMI (0.3097 NDCG). The p exponent effectively amplified surprising movie views, helping the model match users with niche tastes. Best of all, applying these weights did not slow down training; the weighted models often finished faster than the unweighted ones.

5.3 MovieLens 20M: scaling latent geometries

On the massive 20-million interaction dataset, the advanced probability and saturation models significantly beat the unweighted inputs. According to Table 4, Power-Lift captured the best results for ALS (0.3375 NDCG and 0.3693 Precision), slightly ahead of Sigmoid Propensity (0.3338 NDCG). Just like the smaller MovieLens dataset, applying rigid global transformations, such as the universal popularity penalty of Log-IDF (0.2940 NDCG@20) or the strict geometric constraints of unit Normalization (0.2287 NDCG@20), actually made the ALS model perform substantially worse than the unweighted baseline (0.3239 NDCG@20).

For the KNN model, the parameter-free PMI formula performed the best (0.2620 NDCG), while Power-Lift remained strong in second place (0.2588 NDCG). Even though we processed 20 million records, the vectorized implementations ran very fast, taking just 0.4 seconds to weigh the input data and 10.8 seconds to train, which is perfect for production-scale

systems.

Table 4 Performance comparison on the MovieLens 20M dataset

Model	Strategy	NDCG@20	Precision@20	Weighting (s)	Training (s)	Best Parameters
ALS	Power-Lift	**0.3375**	**0.3693**	0.37	10.77	p: 0.34
ALS	Sigmoid Propensity	<u>0.3338</u>	<u>0.3662</u>	0.34	10.77	γ: 2.10, β: 0.99
ALS	PMI	0.3263	0.3482	0.75	10.84	-
ALS	*No Weighting*	*0.3239*	*0.3573*	-	*10.71*	-
ALS	Robust User-Centric	0.3225	0.3551	1.83	10.78	-
ALS	BM25	0.3221	0.3573	0.68	10.87	k_1: 4.03, b: 0.74
ALS	Power	0.3214	0.3544	0.41	10.92	p: 1.29
ALS	TF-IDF	0.3191	0.3507	0.51	10.85	-
ALS	Log	0.3067	0.3426	0.37	10.87	-
ALS	Log-IDF	0.2940	0.3218	0.56	10.88	α: 54.38
ALS	Normalization	0.2287	0.2766	0.57	10.86	-
KNN	PMI	**0.2620**	**0.2824**	0.75	3.13	-
KNN	Power-Lift	<u>0.2588</u>	<u>0.2694</u>	0.37	3.11	p: 0.87
KNN	BM25	0.2582	0.2681	0.68	3.12	k_1: 779.05, b: 0.95
KNN	Normalization	0.2522	0.2734	0.57	3.12	-
KNN	Power	0.2269	0.2558	0.40	3.19	p: 1.44
KNN	*No Weighting*	*0.2256*	*0.2548*	-	*5.25*	-
KNN	Robust User-Centric	0.2253	0.2542	1.83	3.12	-
KNN	Sigmoid Propensity	0.2245	0.2528	0.34	3.13	γ: 3.37, β: 0.01
KNN	TF-IDF	0.2240	0.2528	0.51	3.14	-
KNN	Log-IDF	0.2238	0.2528	0.56	3.14	α: 147.15
KNN	Log	0.2238	0.2528	0.37	3.17	-

Note: For each model and metric, bold indicates the best-performing weighting, and underlined indicates the second-best.

5.4 Taste Profile: managing large-scale audio interactions

The Taste Profile dataset represents a massive collection of over 48 million audio streaming interactions. Similar to Last.fm, play counts can grow endlessly, requiring robust mathematical scaling. As highlighted in Table 5, probabilistic lift metrics proved to be the most effective approach. For the ALS model, standard PMI secured the top position (0.1074 NDCG), closely followed by our tunable Power-Lift strategy (0.1068 NDCG). Both easily beat the unweighted baseline, which scored a low 0.0882 NDCG.

The KNN model showed even more dramatic improvements. Power-Lift reached 0.1692 NDCG, leaving the unweighted KNN baseline (0.1197 NDCG) far behind. However, while Power-Lift outperformed BM25 in NDCG (0.1692 vs. 0.1672), BM25 actually achieved a slightly higher Precision@20 for the KNN model (0.1981 compared to Power-Lift's 0.1971). Overall, these results strongly confirm that identifying "surprising" interactions works much better than just trusting raw play counts for continuous audio consumption.

Table 5 Performance comparison on the Taste Profile dataset

Model	Strategy	NDCG@20	Precision@20	Weighting (s)	Training (s)	Best Parameters
ALS	PMI	**0.1074**	**0.1326**	2.09	29.45	-
ALS	Power-Lift	<u>0.1068</u>	<u>0.1320</u>	0.97	29.40	p: 0.28
ALS	Log-IDF	0.1061	0.1285	1.51	29.45	α: 3.66
ALS	TF-IDF	0.1059	0.1300	1.40	29.61	-
ALS	Sigmoid Propensity	0.1028	0.1270	0.98	29.48	γ: 1.27, β: 0.95
ALS	Robust User-Centric	0.1013	0.1251	3.55	29.46	-
ALS	BM25	0.0976	0.1185	1.91	29.47	k_1: 315.91, b: 0.98
ALS	*No Weighting*	*0.0882*	*0.1074*	-	*29.40*	-
ALS	Power	0.0874	0.1058	1.00	29.39	p: 1.12
ALS	Log	0.0742	0.0914	0.90	29.45	-
ALS	Normalization	0.0540	0.0671	1.37	29.49	-
KNN	Power-Lift	**0.1692**	<u>0.1971</u>	0.98	15.25	p: 0.37
KNN	BM25	<u>0.1672</u>	**0.1981**	1.90	15.32	k_1: 20.68, b: 0.96
KNN	PMI	0.1671	0.1932	2.09	15.31	-
KNN	Normalization	0.1641	0.1943	1.36	15.32	-
KNN	Power	0.1638	0.1872	0.99	15.25	p: 0.18
KNN	Sigmoid Propensity	0.1637	0.1871	0.96	15.28	γ: 0.38, β: 0.86
KNN	Robust User-Centric	0.1624	0.1857	3.53	15.35	-
KNN	Log-IDF	0.1612	0.1850	1.52	15.28	α: 9.65
KNN	Log	0.1611	0.1850	0.90	15.27	-
KNN	TF-IDF	0.1600	0.1841	1.40	15.35	-
KNN	*No Weighting*	*0.1197*	*0.1441*	-	*15.41*	-

Note: For each model and metric, bold indicates the best-performing weighting, and underlined indicates the second-best.

5.5 Steam video game dataset: the breakthrough on behavioral extremes

The most important proof for our weighting models came from the Steam dataset. Because Steam tracks continuous "hours played," the maximum values are huge and completely ruin standard linear models. As demonstrated in Table 6, the raw baseline for the ALS model scored a terrible 0.0462 NDCG. But applying the Power-Lift strategy pushed the score all the way to 0.0713. This is an incredible 54.3% relative improvement.

The KNN model showed a similar 52.6% improvement when using the Log-IDF strategy (0.0586 NDCG) compared to the unweighted baseline (0.0384 NDCG), with Robust User-Centric performing nearly identically well (0.0582 NDCG). The data highlights a crucial point: while intelligent relative scaling (like Power-Lift) creates massive gains, applying rigid limits (like Sigmoid Propensity or strict Normalization) to the KNN model caused it to completely fail on Steam, resulting in NDCG scores of 0.0000 and 0.0004 respectively. Extreme behavioral data requires relative probabilistic dampening, not hard cutoffs that accidentally erase important engagement signals.

Table 6 Performance comparison on the Steam dataset.

Model	Strategy	NDCG@20	Precision@20	Weighting (s)	Training (s)	Best Parameters
ALS	Power-Lift	**0.0713**	**0.1498**	1.29	49.50	p: 0.37
ALS	BM25	<u>0.0656</u>	<u>0.1365</u>	1.67	49.60	k_1: 3.46, b: 0.97
ALS	PMI	0.0626	0.1302	1.88	49.52	-
ALS	Log-IDF	0.0616	0.1292	1.32	49.55	α: 1.43
ALS	TF-IDF	0.0607	0.1280	1.22	52.58	-
ALS	Power	0.0562	0.1179	1.00	49.61	p: 0.71
ALS	Robust User-Centric	0.0558	0.1168	6.83	49.53	-
ALS	*No Weighting*	*0.0462*	*0.1006*	-	*52.19*	-
ALS	Log	0.0401	0.0860	0.93	53.06	-
ALS	Normalization	0.0240	0.0409	1.37	49.57	-
ALS	Sigmoid Propensity	0.0124	0.0266	0.72	50.78	γ: 1.58, β: 0.84
KNN	Log-IDF	**0.0586**	**0.1240**	1.32	3.17	α: 54.95
KNN	Robust User-Centric	<u>0.0582</u>	<u>0.1231</u>	6.83	3.14	-
KNN	Power-Lift	0.0560	0.1204	1.29	3.23	p: 0.25
KNN	BM25	0.0551	0.1185	1.67	2.87	k_1: 47.35, b: 0.32
KNN	Power	0.0544	0.1165	1.00	2.84	p: 0.25
KNN	Log	0.0543	0.1161	0.93	2.95	-
KNN	PMI	0.0533	0.1118	1.89	2.81	-
KNN	TF-IDF	0.0510	0.1089	1.22	2.88	-
KNN	*No Weighting*	*0.0384*	*0.0786*	-	*2.84*	-
KNN	Normalization	0.0004	0.0011	1.37	3.11	-
KNN	Sigmoid Propensity	0.0000	0.0002	0.70	3.07	γ: 3.15, β: 0.76

Note: For each model and metric, bold indicates the best-performing weighting, and underlined indicates the second-best.

5.6 Dimensionality reduction and parameter sensitivity

One important secondary finding is how weighting affects the size of the model itself. Usually, increasing a model's capacity (using 100 latent factors instead of 10) allows it to learn more complex patterns, but this takes up huge amounts of memory and slows down predictions.

However, experimental findings demonstrate that in certain cases, proper data weighting can effectively substitute for brute-force parameter expansion. When we restricted the ALS model to just $k = 10$ factors on the MovieLens 100k data, the Sigmoid weighted version scored 0.3313 NDCG. This easily beats the unweighted, massive $k = 100$ model, which capped out at 0.3188 NDCG. We saw the same architectural efficiency in the Last.fm dataset. A small KNN model restricted to only $K = 20$ neighbors and paired with BM25 weighting scored 0.1587 NDCG. This crushed the large unweighted KNN $K = 100$ model (0.0966 NDCG).

These results suggest a valuable architectural benefit: appropriate mathematical weighting can effectively reduce the required capacity of the model. By cleaning up noisy data before training, lightweight models are often able to achieve competitive or even superior predictive performance compared to unweighted baseline models operating with an order of magnitude more parameters.

6 Discussion

The experimental results across five diverse datasets provide robust empirical validation for the proposed Saturation Hypothesis, fundamentally challenging the static scaling assumptions inherent in traditional WMF frameworks [5]–[7]. Historically, recommender systems relying on implicit feedback [1], [2] have operated under the assumption that a user's preference grows continuously and unboundedly with raw interaction volume. Consequently, standard baselines frequently apply unbounded linear scaling or simple logarithmic smoothing [6] to user-item interactions. Our findings directly contradict the efficacy of this "one-size-fits-all" approach. In environments characterized by technically endless consumption, such as the Steam and Last.fm datasets, unbounded models are highly susceptible to signal distortion caused by ambient noise and power users. Conversely, enforcing an asymptotic limit via our Sigmoid Propensity Weighting effectively functions as a soft cap that prevents extreme behavioral outliers from disrupting latent space geometries. This saturation approach proved

completely dominant in dense, capped environments like the MovieLens 100k dataset, demonstrating that capping the influence of users perfectly stabilizes matrices without collapsing the underlying signal.

Furthermore, our results demonstrate that measuring the probabilistic "surprise" of an interaction yields significantly better representations of user preference than raw frequency counts or standard IR baselines like TF-IDF [10], [11]. While previous research has successfully adapted PMI [14]–[17] to matrix factorization to capture this statistical surprise, standard PMI implementations suffer from mathematical instability, relying on artificial zero-clipping to handle negative logarithms. Our proposed Power-Lift strategy mathematically resolves these computational singularities by utilizing a continuous power-law expansion that remains strictly positive. The empirical superiority of Power-Lift is most evident in the highly skewed Steam dataset, where it achieved a massive 54.3% relative improvement in ranking accuracy over unweighted baselines, heavily outperforming traditional BM25 [12], [13] models. Interestingly, the success of Power-Lift's fractional exponent in dampening high-frequency noise while amplifying rare interactions strongly aligns with analogous subsampling strategies [8] that have been proven to improve representation quality in neural embedding learning.

Beyond global transformations, our experiments underscore the critical importance of addressing User Heterogeneity through local contextualization. Standard unit normalization strategies [9] explicitly test if relative distribution alone captures preference by erasing the magnitude of interaction volume entirely. Our results completely contradict this methodology; strict normalization catastrophically degraded performance on datasets like Steam and Last.fm, proving that erasing magnitude destroys the model's ability to learn. Instead, our Robust User-Centric weighting strategy acts as a true psychological equalizer by utilizing local non-parametric statistics, specifically the median and interquartile range [18]. This ensures that a user's behavior is normalized strictly against their own baseline habits, effectively managing abrupt shifts in engagement without relying on vulnerable global averages.

Finally, a significant architectural implication of our study is the observation that intelligent mathematical weighting can effectively substitute for brute-force parameter expansion. We achieved results contrary to the common assumption that larger capacity guarantees better performance; for example, a lightweight ALS model restricted to just $k = 10$ latent factors utilizing Sigmoid weighting outperformed a massive, unweighted $k = 100$ factor model. This confirms that filtering noisy data prior to training provides a computationally efficient mechanism to boost top-N recommendation accuracy without increasing downstream inference latency.

Despite the clear empirical advantages of adaptive weighting, we acknowledge certain limitations in the proposed methodologies. The reliance on hyperparameter optimization frameworks, such as Optuna [35], to identify optimal configuration values, such as the tuning exponent p or the sigmoid bounds γ and β, introduces additional computational overhead during the initial model tuning phase. Furthermore, the Robust User-Centric strategy, which relies on the calculation of a user's local median and IQR [18], faces inherent challenges in extreme cold-start scenarios. For new users with fewer than three distinct interactions, non-parametric dispersion metrics become statistically uninformative, necessitating a mathematical fallback to global baseline weights until sufficient personal behavioral data is accumulated.

7 Conclusion and future work

Moving from explicit ratings to implicit tracking means we cannot rely on raw, unscaled interaction counts. This research clearly proves that using raw frequencies degrades ranking accuracy because it fails to account for background noise, power users, and signal saturation. By separating the binary preference from a continuous confidence weight using ALS and KNN, we created a clear path for better normalization. Our results show that dampening mechanisms, especially Power-Lift, Sigmoid Propensity, and our hybrid Log-IDF, create massive improvements over standard baselines. By scaling down popular items and controlling interaction counts, Log-IDF achieved peak precision in ambient listening domains like Last.fm, while Power-Lift drove up to a 54.3% improvement in ranking accuracy on extreme datasets like Steam.

Ultimately, intelligent mathematical weighting provides a highly scalable and computationally efficient mechanism to upgrade the retrieval and collaborative filtering components of modern recommender systems. It acts as an operational multiplier by cleaning up noisy data before training. Our findings prove that a fast, lightweight model paired with adaptive weighting can effectively substitute for brute-force parameter expansion. By optimizing the input signal, these streamlined models achieve superior predictive performance compared to their unweighted, high-capacity counterparts operating with an order of magnitude more parameters. This demonstrates that intelligently bounding user data offers a highly competitive, low-latency alternative to adopting increasingly complex and computationally expensive architectures.

Future work will explore integrating these weighting schemas directly as preprocessing layers or custom loss function components within advanced neural architectures, such as Neural Collaborative Filtering [37], Deep Neural Networks [38], and Graph Neural Networks [27], [28]. Among these, application to graph-based models is particularly promising given recent advancements in sequential recommendation. Recent research [39] has successfully demonstrated that utilizing statistical data mining algorithms, specifically PMI and TF-IDF, to automatically construct edge-weighted knowledge graphs significantly improves the ranking accuracy of Graph Neural Networks. Because our proposed Power-Lift and Log-IDF

strategies resolve the mathematical discontinuities and signal saturation vulnerabilities inherent in standard PMI and TF-IDF formulas, applying our adaptive continuous weighting as the foundational edge-weighting mechanism for these synthesized knowledge graphs presents a highly effective direction for future architectural scaling.

Notably, autoencoders like EASE^R [24] and VaeCF [25] are traditionally restricted to binary input matrices, as massive variance in continuous behavioral data critically destabilizes their loss functions and regularization frameworks. Because strategies like Sigmoid Propensity successfully compress and bound extreme behavioral variance into stable, continuous parameters, they present a highly promising mechanism to finally allow deep generative frameworks to safely ingest and learn from unclipped, continuous implicit feedback.

Declarations

Funding The authors declare that no funds, grants, or other financial support were received during the preparation of this manuscript.

Competing interests The authors declare no competing interests.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Data availability The datasets analyzed during the current study (MovieLens 100k, MovieLens 20M, Last.fm 360k, Taste Profile, and Steam) are strictly derived from publicly available sources.

Materials availability Not applicable.

Code availability The custom code, evaluation pipeline, and implementations of the adaptive weighting strategies supporting the findings of this study are openly available in the GitHub repository at https://github.com/insanewarrior/rec-sys-research/tree/main/weighting_is_fun.

Author contribution A.E.S. and I.O.P. contributed to the study conception, design, and methodology. Data preprocessing, partitioning, implementation of the weighting strategies, and development of the experimental pipeline were performed by A.E.S. and I.O.P. The first draft of the manuscript was written by A.E.S., and both authors reviewed, edited, and commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

1. D. W. Oard and J. Kim, “Implicit feedback for recommender systems,” in *Proc. AAAI Workshop Recommender Syst.*, Madison, WI, USA, 1998, pp. 81–83. https://www.semanticscholar.org/paper/Normalized-(pointwise)-mutual-information-in-Bouma/15218d9c029cbb903ae7c729b2c644c24994c201. Accessed 21 March 2026

2. D. Kelly and J. Teevan, “Implicit feedback for inferring user preference: A bibliography,” *SIGIR Forum*, vol. 37, no. 2, pp. 18–28, Sept. 2003. https://doi.org/10.1145/959258.959260

3. P. Resnick and H. R. Varian, “Recommender systems,” *Commun. ACM*, vol. 40, no. 3, pp. 56–58, Mar. 1997. https://doi.org/10.1145/245108.245121

4. G. Jawaheer, P. Weller, and P. Kostkova, “Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback,” *ACM Trans. Interact. Intell. Syst.*, vol. 4, no. 2, pp. 1–26, May 2014. https://doi.org/10.1145/2512208

5. Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 263–272. https://doi.org/10.1109/ICDM.2008.22

6. R. Pan et al., “One-class collaborative filtering,” in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 502–511. https://doi.org/10.1109/ICDM.2008.16

7. Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. https://doi.org/10.1109/MC.2009.263

8. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 26, 2013, pp. 3111–3119. https://www.semanticscholar.org/paper/Distributed-Representations-of-Words-and-Phrases-Mikolov-Sutskever/87f40e6f3022adbc1f1905e3e506abad05a9964f. Accessed 21 March 2026

9. I. Pilászy, D. Zibriczky, and D. Tikk, “Fast ALS-based matrix factorization for explicit and implicit feedback datasets,” in *Proc. 4th ACM Conf. Recommender Syst. (RecSys)*, Barcelona, Spain, 2010, pp. 71–78. https://doi.org/10.1145/1864708.1864726

10. G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, Jan. 1988. https://doi.org/10.1016/0306-4573(88)90021-0

11. K. Spärck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *J. Doc.*, vol. 28, no. 1, pp. 11–21, Jan. 1972. https://doi.org/10.1108/eb026526

12. S. E. Robertson and S. Walker, “Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval,” in *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, Dublin, Ireland, 1994, pp. 232–241. https://doi.org/10.1007/978-1-4471-2099-5_24

13. S. Robertson and H. Zaragoza, “The probabilistic relevance framework: BM25 and beyond,” *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, Apr. 2009. https://doi.org/10.1561/1500000019

14. K. W. Church and P. Hanks, “Word association norms, mutual information, and lexicography,” *Comput. Linguistics*, vol. 16, no. 1, pp. 22–29, 1990. https://www.semanticscholar.org/paper/Word-Association-Norms%2C-Mutual-Information%2C-and-Church-Hanks/9e2caa39ac534744a180972a30a320ad0ae41ea3. Accessed 21 March 2026

15. P. D. Turney, “Mining the web for synonyms: PMI-IR versus LSA on TOEFL,” in *Proc. 12th Eur. Conf. Mach. Learn. (ECML)*, Freiburg, Germany, 2001, pp. 491–502. https://doi.org/10.1007/3-540-44795-4_42

16. O. Levy and Y. Goldberg, “Neural word embedding as implicit matrix factorization,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 27, Montreal, QC, Canada, 2014, pp. 2177–2185. https://doi.org/10.5555/2969033.2969070

17. G. Bouma, “Normalized (pointwise) mutual information in collocation extraction,” in *Proc. Biennial GSCL Conf.*, Potsdam, Germany, 2009, pp. 31–40. https://www.semanticscholar.org/paper/Normalized-(pointwise)-mutual-information-in-Bouma/15218d9c029cbb903ae7c729b2c644c24994c201. Accessed 21 March 2026

18. P. J. Huber, *Robust Statistics*. New York, NY, USA: John Wiley & Sons, 1981. https://doi.org/10.1002/0471725250

19. X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, “Fast matrix factorization for online recommendation with implicit feedback,” in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, Pisa, Italy, 2016, pp. 549–558. https://doi.org/10.1145/2911451.2911489

20. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. 10th Int. Conf. World Wide Web (WWW)*, Hong Kong, 2001, pp. 285–295. https://doi.org/10.1145/371920.372071

21. G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003. https://doi.org/10.1109/MIC.2003.1167344

22. M. Deshpande and G. Karypis, “Item-based top-n recommendation algorithms,” *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, Jan. 2004. https://doi.org/10.1145/963770.963776

23. B. Frederickson, “Implicit: Fast Python collaborative filtering for implicit datasets,” GitHub repository, 2016. https://github.com/benfred/implicit. Accessed 21 March 2026

24. H. Steck, “Embarrassingly shallow autoencoders for sparse data,” in *Proc. World Wide Web Conf. (WWW)*, San Francisco, CA, USA, 2019, pp. 3251–3257. https://doi.org/10.1145/3308558.3313710

25. D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” in *Proc. World Wide Web Conf. (WWW)*, Lyon, France, 2018, pp. 689–698. https://doi.org/10.1145/3178876.3186150

26. I. Pyshnograiev and A. Shyraliiev, “A Comprehensive Benchmark of Collaborative Filtering Methods on Implicit Feedback Datasets”, *Electronics and Control Systems*, vol. 2, no. 84, pp. 34–45, Jun. 2025. https://doi.org/10.18372/1990-5548.84.20192

27. X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, “Neural graph collaborative filtering,” in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, Paris, France, 2019, pp. 165-174. https://doi.org/10.1145/3331184.3331267

28. X. He et al., “LightGCN: Simplifying and powering graph convolution network for recommendation,” in *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. (SIGIR)*, 2020, pp. 639–648, https://doi.org/10.1145/3397271.3401063

29. F. M. Harper and J. A. Konstan, “The MovieLens datasets: History and context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, Dec. 2015. https://doi.org/10.1145/2827872

30. O. Celma, *Music Recommendation and Discovery in the Long Tail*. Berlin, Germany: Springer, 2010. https://doi.org/10.1007/978-3-642-13287-2

31. T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proc. 12th Int. Soc. Music Inf. Retr. Conf. (ISMIR)*, Miami, FL, USA, 2011, pp. 591–596. https://doi.org/10.7916/D8NZ8J07

32. A. Kozyriev, “Game recommendations on Steam,” Kaggle, 2023. https://www.kaggle.com/datasets/antonkozyriev/game-recommendations-on-steam. Accessed 21 March 2026

33. K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of IR techniques,” *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002. https://doi.org/10.1145/582415.582418

34. P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-n recommendation tasks,” in *Proc. 4th ACM Conf. Recommender Syst. (RecSys)*, Barcelona, Spain, 2010, pp. 39–46. https://doi.org/10.1145/1864708.1864721

35. T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, 2019, pp. 2623–2631. https://doi.org/10.1145/3292500.3330701

36. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Proc. 24th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Granada, Spain, 2011, pp. 2546–2554. https://doi.org/10.5555/2986459.2986743

37. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proc. 26th Int. Conf. World Wide Web (WWW)*, Perth, Australia, 2017, pp. 173–182. https://doi.org/10.1145/3038912.3052569

38. P. Covington, J. Adams, and E. Sargin, “Deep neural networks for YouTube recommendations,” in *Proc. 10th ACM Conf. Recommender Syst. (RecSys)*, Boston, MA, USA, 2016, pp. 191–198. https://doi.org/10.1145/2959100.2959190

39. D. V. Androsov and N. I. Nedashkovskaya, “The hybrid sequential recommender system synthesis method based on attention mechanism with automatic knowledge graph construction,” *System Research & Information Technologies*, no. 1, pp. 7–18, Mar. 2025. https://doi.org/10.20535/SRIT.2308-8893.2025.1.01