Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.3, June 2026, pp. 2145~2154
e-ISSN: 2548-6861

From Web Extraction to Collaborative Filtering: An End-to-End Architecture for Reliable Recommendation Systems

Lamanabwe Epus Hervé[1]*, Blaise Muhala Luhepa[2*,], Matondo Mananga Herman[3]*, Dieuleveut Nianga Kaya-Kaya[3*,**], Benjamin Consolant Majegeza[4]*****
* Department of Mathematics, Statistics and Computer Sciences, P.O Box 190, Kinshasa, University of Kinshasa, D.R. Congo
** Department of Didactics, Research Center for Mathematics Education, DRCongo
*** Department of Multidisciplinary, Research Center for Mathematics Education, DRCongo
**** Department of observatory, One Health Institute for Africa, DRCongo
lamanabweepus@gmail.com [1], blaise.muhala@unikin.ac.cd [2], herman.matondo@unikin.ac.cd[3], niangadieuleveut@gmail.com [4], consobenmaj@gmail.com [5]

Article Info

Article history:

Received 2026-02-27
Revised 2026-04-21
Accepted 2026-05-11

Keyword:

Recommender Systems,
Data Pipeline Quality,
Web Data Extraction,
Collaborative Filtering,
Hybrid Recommendation.

ABSTRACT

The growth of digital platforms has generated large volumes of Web-derived interaction data, but these data are often noisy, duplicated, incomplete, and temporally unstable. Recommendation quality therefore depends not only on the ranking model, but also on how extraction, validation, and temporal control are integrated upstream. This paper presents an end-to-end architecture in which Web extraction, schema normalization, cleaning, deduplication, anomaly quarantine, recency-aware processing, and recommendation generation are treated as a single operational pipeline. The contribution is not the use of hybrid recommendation alone, which is already common, but the explicit integration of these quality-control stages with temporally valid offline evaluation and system-level monitoring. Four recommendation strategies are studied within the same pipeline: global popularity, recency-weighted popularity, implicit matrix factorization, and a hybrid method that combines collaborative filtering with a recency-based fallback for sparse-user cold-start situations. Experiments are conducted on a realistic e-commerce dataset comprising approximately 50,000 users, 18,000 items, and 1.2 million interactions under a strict chronological 80/20 split. Evaluation includes Precision@K, Recall@K, NDCG@K, Coverage@K, sparse-user cold-start analysis, and system indicators. Results indicate that the hybrid approach achieves the best observed aggregate ranking performance under the present protocol, improves sparse-user robustness (Recall@10 = 0.158), maintains broad catalog coverage (38.9%), and remains operationally stable under the tested evaluation conditions (p95 latency = 48 ms; uptime = 99.7%). These findings support assessing recommendation quality as a property of the full data-to-recommendation pipeline rather than of the ranking algorithm alone.

I. INTRODUCTION

The explosive growth of digital platforms has reshaped how information is produced, accessed, and monetized. In e-commerce in particular, users continuously generate heterogeneous traces—clicks, searches, baskets, purchases, ratings, and textual feedback creating a rich but messy substrate for personalization. Recommender systems have therefore become core decision engines that reduce choice overload while supporting engagement and revenue objectives [1,20]. Yet, as recommendation pipelines scale, the dominant bottleneck is increasingly less about modeling capacity and more about the reliability of the end-to-end data-to-ranking process [31,12,14]. A persistent issue is that Web-derived data are inherently noisy and unstable. At ingestion time, pipelines may face duplicated events, inconsistent identifiers, missing fields, shifting schemas, bots, and

abnormal interactions, all of which distort learning signals and degrade ranking stability [12,7,29]. More broadly, the software engineering literature has shown that data pipeline quality problems concentrate in integration, ingestion, and cleaning stages, with root causes often tied to data typing and compatibility mismatches—precisely the kinds of failures that silently propagate into downstream analytics and learning systems [12]. In recommendation, such upstream imperfections can be amplified: collaborative filtering models may overfit spurious co-occurrence patterns, while popularity-based baselines can lock into feedback loops that reinforce exposure and concentrate attention [18,5,33]. These effects matter operationally because they reduce catalog exploration and can degrade user experience over time [18, 8]. Another widely observed challenge concerns temporal dynamics. User intent evolves, item attractiveness shifts, and external factors (seasonality, campaigns, stock availability) introduce non-stationarity. If evaluation ignores time and relies on random splits, offline estimates can become optimistic and misaligned with production reality [27,15]. Conversely, strictly time-aware evaluation better reflects the real constraint that future interactions are unavailable at training time [27]. In production-facing settings, this temporal mismatch is often accompanied by drift phenomena: both user preferences and algorithmic exposure can co-evolve, leading to performance decay and distributional shifts that must be monitored [22,21]. These dynamics become especially visible in cold-start contexts, where new or sparsely active users lack sufficient collaborative evidence, making purely CF-based ranking brittle [9,26].

In parallel, the community has placed growing emphasis on "trustworthy" recommendation, moving beyond accuracy to include robustness, reliability, bias awareness, and predictable behavior under noise and adversarial conditions [14,7,23]. Recent works highlight that robustness failures arise not only from explicit attacks but also from natural noise in interaction histories, which can be structural (e.g., rating flips, corrupted feedback) and substantially harmful at scale [29,2]. As a result, algorithmic progress alone is insufficient: trustworthy performance requires a pipeline view in which data quality control, temporal validation, and ranking models are designed and evaluated jointly [12,14,27].

Hybrid strategies are frequently proposed as a pragmatic way to improve resilience. By combining popularity and collaborative signals, hybrids can provide reasonable recommendations when personalization evidence is limited, while leveraging CF where it is reliable [20,11,19]. However, many hybrid approaches treat upstream data processing as a black box. In practice, without explicit mechanisms for cleaning, deduplication, recency handling, and quarantine of suspicious records, hybridization cannot guarantee stable behavior: it may simply blend two imperfect signals, each affected by data artifacts [31,12,7]. This gap is particularly problematic when systems are built from Web extraction stages, where input variability is the norm rather than the exception [10,17].

Motivated by these limitations, this paper argues that high-quality recommendation in real environments requires an integrated architecture spanning the entire chain from Web data extraction to recommendation serving while explicitly controlling data quality and operational reliability. We propose a unified pipeline that includes (i) Web extraction and structured ingestion, (ii) automated cleaning, deduplication and consistency checks, (iii) recency-aware processing and temporal alignment, and (iv) recommendation modules covering popularity baselines, implicit matrix factorization for collaborative filtering, and a hybrid strategy that combines collaborative and temporal popularity signals with a cold-start fallback [31,12,20,9]. Our experimental protocol adopts a strict temporal split to avoid leakage and reflect deployment constraints, reporting standard top-K ranking metrics (Precision@K, Recall@K, NDCG@K) as well as coverage indicators to quantify catalog exploration under realistic conditions [27,33,15].

The contributions of this study should be understood at the architectural and evaluation levels rather than as the introduction of a new hybrid formula. First, we formalize an end-to-end pipeline that explicitly connects Web extraction, normalization, deduplication, quarantine, temporal weighting, collaborative filtering, and recommendation serving. Second, we show how these components can be evaluated jointly through temporally valid offline metrics, sparse-user cold-start analysis, coverage, and system-level indicators. Third, we position the hybrid module as one component inside a broader reliability-oriented workflow, thereby clarifying that the novelty lies in the explicit integration and assessment of the pipeline rather than in hybrid recommendation alone [31,12,14].

II. METHODOLOGY

This section describes the end-to-end methodology used to build the proposed recommender pipeline, from Web data extraction to collaborative filtering. The methodological originality of the study does not rest on hybrid recommendation alone, but on the explicit integration of extraction, validation, temporal alignment, recommendation, and monitoring within a single workflow. Our key stance is that recommendation quality in production is not only a modeling problem, but also a pipeline problem: ingestion, data quality control, temporal validity, and serving constraints can strongly shape the final ranking behavior [12,28]. Consequently, data quality control is treated here as a first-class component rather than as a preliminary cleanup step [12,14].

A. Integrated pipeline overview

Each stage of the pipeline produces inputs that are consumed by the next stage. Web extraction yields raw interaction events; normalization harmonizes their structure; cleaning, deduplication, and quarantine reduce the propagation of unreliable signals; recency-aware weighting

converts validated events into recommendation-ready inputs; and the serving layer exposes both ranking outputs and operational indicators. The study therefore evaluates an integrated chain rather than a standalone ranking module.

Figure 1 presents the global workflow. The pipeline consists of four layers: (i) Web extraction and ingestion, (ii) quality control and normalization, (iii) interaction modeling and recommendation, and (iv) evaluation at both model and system levels. Such a pipeline-oriented view is aligned with recent work emphasizing trustworthy and operational recommender systems, where robustness, reliability, and bias control complement classical accuracy objectives [28,18].

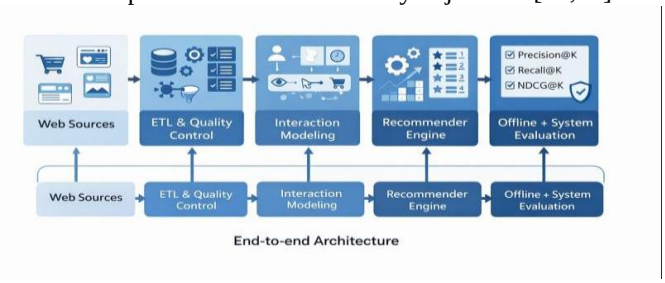

Figure 1. End-to-end architecture: Web sources → ETL & Quality Control → Interaction Modeling → Recommender Engine → Offline + System Evaluation.

B. Web data extraction and event representation

We assume a typical large-scale e-commerce scenario where user behavior is captured through heterogeneous events (view, click, add-to-cart, purchase, rating). Each raw event is represented as:

$$e = (u, i, t, a),$$

where u is a user, i an item, t the timestamp and a the action type. The dataset is then:

$$D = \{e_1, \cdots, e_N\},$$

with N large ($\approx$ 1.2M interactions), consistent with real-world RS workloads [39]. This representation allows us to keep temporal information explicit-crucial for realistic evaluation and for recency-aware signals [19].

TABLE I
DATASET STATISTICS

Statistic	Value
Domain	E-commerce
Number of users	≈ 50,000
Number of items	≈ 18,000
Number of interactions	≈ 1.2 million
Interaction types	view, click, add_to_cart, purchase, rating
View ratio	62 %
Click ratio	21 %
Add-to-cart ratio	9 %
Purchase ratio	6 %
Rating ratio	2 %
Temporal split	80 % train / 20 % test (chronological)

C. Data quality control and reliability mechanisms

A core contribution of our methodology is a structured quality-control layer applied before model training. In this paper, reliability is defined in a restricted and operational sense: data reliability refers to the consistency of validated interaction logs, ranking reliability refers to the stability of recommendation outputs under a temporal protocol, and operational stability refers to service behavior under the tested evaluation conditions. Data pipeline issues such as duplication, schema drift, abnormal frequency, and inconsistent identifiers can silently propagate into downstream learning performance [31]. We therefore operationalize quality control through deduplication, anomaly filtering with quarantine, and temporal validation [12,28].

We detect duplicates using a composite key *(u, i, t, a)* within a tolerance window **Δt**. Duplicate bursts are merged into a single canonical event. Deduplication is necessary to prevent inflated confidence weights and popularity artefacts [12, 18]**.**

1). Anomaly filtering with quarantine

Rather than deleting suspicious events, we quarantine them to preserve auditability (important for trustworthy RS pipelines) [28, 12]. Let f_u be the interaction rate for user u within a window. We flag events when:

$$f_u > \mu_u + \lambda\sigma_u,$$

where μ_u and σ_u are user-level historical statistics and λ is a sensitivity coefficient. This is motivated by robustness studies showing that noisy or adversarial feedback-whether intentional or natural-can significantly affect ranking stability [28, 9].

2). Temporal validation and recency handling

To capture temporal dynamics, we assign an exponential decay weight to event:

$$\omega(t) = \exp\big(-\gamma(T - t)\big),$$

where T is the reference time and γ controls decay. Recency is widely recognized as a strong signal in e-commerce and help reduce staleness while supporting realistic deployment behavior [35]. At the same time, recency can intensify popularity concentration; therefore, our evaluation also includes coverage metrics to quantity catalog exploration [18, 5].

D. Interaction modeling (implicit feedback)

After quality control, we build an implicit feedback r_{ui} by aggregating weighted events:

$$r_{ui} = \sum_{e \in D_{ui}} \alpha_{a(e)} \cdot \omega(t_e),$$

where α_a is an action-specific coefficient (e.g., purchase > add-to-cart > click > view). This is a standard and scalable way to model multi-event behavior while preserving intensity and recency information [27]. The resulting matrix $R \in \mathbb{R}^{|U| \times |I|}$ is then used for collaborative filtering.

E. Recommendation strategies

We compare four methods: POP, POP-REC, CF-MF, and HYBRID. This selection is deliberate because it allows a partial component-wise reading of the architecture: POP

versus POP-REC isolates the effect of recency-aware popularity, CF-MF captures collaborative personalization, and HYBRID shows the effect of combining collaborative inference with an explicit fallback for sparse-user cases. Although this is not a full factorial ablation of every pipeline component, it provides a first decomposition of the contribution of the main recommendation modules under a common preprocessing pipeline [28,18,1].

The different recommendation strategies evaluated in this work are summarized in Table 2.

TABLE II
COMPARED RECOMMENDATION METHODS USED IN THE EXPERIMENTATION EVALUATION

Method	Description	Purpose
POP	Global item popularity based on interaction frequency	Strong non-personalized baseline
POP-REC	Popularity weighted by temporal recency	Captures trending items
CF-MF	Implicit matrix factorization	Personalized collaborative filtering
HYBRID	CF with recency-based popularity fallback	Cold-start robustness

1). POP (global popularity)

$$\text{score}_{\text{POP}}(i) = \sum_{u} r_{ui}.$$

POP is robust in sparse regimes but suffers from popularity bias and low diversity [18, 5].

2). POP-REC (recency-weighted popularity)

$$\text{score}_{\text{POP}}(i) = \sum_{u} r_{ui} \cdot \omega(t_{ui}).$$

This captures trending items and aligns with temporal dynamics [19]

3). CF-MF (implicit matrix factorization)

We instantiate CF-MF as a standard implicit-feedback matrix factorization model in which each user and each item are represented by latent vectors. The model is trained on the weighted interaction matrix produced after cleaning, deduplication, quarantine, and recency processing, so that stronger validated interactions contribute more confidence than weak or missing ones.

$$\min_{P,Q} \sum_{u,i} c_{ui}\,(r_{ui} - p_u q_i)^2 + \lambda\left(||p_u||^2 + ||q_i||^2\right).$$

Training is performed on the training split only through a regularized optimization of the user and item factors, with alternating updates until convergence. The same latent dimensionality, regularization regime, and stopping criterion are kept fixed across comparisons to ensure fairness, and no information from the test period is used during model fitting. In this sense, CF-MF serves as a reproducible collaborative baseline rather than as a novel modeling contribution.

4). Hybrid (CF with cold-start fallback)

Because pure CF degrades when user history is too sparse, the hybrid module uses an explicit gating rule rather than an opaque score blend:

$$\text{score\_HYB}\,(u,i) = \begin{cases} p_u^T q_i, & |I_u| \geq \tau \\ \text{score}_{\text{POP.REC}}(i), & otherwise \end{cases}$$

If a user has at least a minimum number of historical interactions in the training data, ranking is produced by CF-MF; otherwise the system falls back to POP-REC. This makes the user-side cold-start behavior explicit and auditable. The present evaluation focuses on new or sparse users; item cold-start is not directly benchmarked in the current protocol, so our robustness claims are limited to user-side cold-start conditions [9,1].

F. Experimental protocol

1). Strict temporal split

To avoid leakage and simulate deployment, we use a global chronological split in which the oldest 80% of interactions form the training set and the most recent 20% form the test set. All model parameters, popularity statistics, and temporal weights are derived from training-period information only. This choice better reflects deployment constraints than random splitting and reduces the risk of optimistic offline estimates [19].

2). Offline ranking metrics

We report standard top-K metrics for $K \in \{5,10\}$:

$$\text{Precision@}K = \frac{1}{|U|}\sum_{u}\frac{|R_u^K \cap T_u|}{K}, \quad \text{Recall@}K = \frac{1}{|U|}\sum_{u}\frac{|R_u^K \cap T_u|}{|T_u|},$$

$$\text{NDCG@}K = \frac{1}{|U|}\sum_{u}\frac{DCG_u@K}{IDCG_u@K}.$$

We also compute coverage to quality catalog exploration:

$$\text{Coverage@}K = \frac{1}{|U|}\sum_{u}\frac{|\bigcup_u R_u^K|}{I}.$$

$$\alpha_{purchase} > \alpha_{add\_to\_cart} > \alpha_{click} > \alpha_{view}.$$

Coverage is important to diagnose popularity concentration effects and to evaluate diversity indirectly [18,5]. Because the present manuscript reports aggregate results, the metric comparisons should be interpreted as descriptive trends under a common protocol rather than as formal statistical significance claims.

3). Cold-start protocol

Cold-start users are defined as those with fewer than three interactions in training. This threshold captures typical sparsity regimes in which collaborative filtering lacks sufficient evidence and a fallback strategy becomes necessary [29,27]. The current protocol evaluates user-side cold-start only; unseen-item cold-start remains outside the scope of the reported experiments.

The evaluation metrics used in our experimental protocol are summarized in Table 3.

TABLE III
EVALUATION METRICS USED TO ASSESS RECOMMENDATION PERFORMANCE

Metric	Definition	Objective
Precision@K	Proportion of relevant items among top-K	Ranking accuracy
Recall@K	Proportion of retrieved relevant items	Coverage of user interest
NDCG@K	Position-aware ranking quality	Importance of item order
Coverage@K	Ratio of recommended items over catalog	Diversity analysis
Cold-start Recall@K	Recall for users with < 3 interactions	Robustness evaluation

G. System-level reliability evaluation

Beyond offline ranking, we log operational indicators: quarantine rate, duplicate ratio, API p95 latency, cache hit rate, and uptime. This aligns with the "trustworthy RS" perspective: recommendation quality should not be achieved at the expense of system reliability and service constraints [14, 24, 4].

TABLE IV
SYSTEM-LEVEL INDICATORS EVALUATING THE RELIABILITY AND OPERATIONAL STABILITY OF THE PROPOSED ARCHITECTURE

Indicator	Description	Value
Quarantine rate	Proportion of interactions flagged and isolated during quality control	1.8 %
Duplicate detection rate	Ratio of duplicated events detected during ingestion	3.2 %
API latency (p95)	95th percentile response time	48 ms
Cache hit rate	Ratio of requests served from cache	71 %
Service uptime	System availability during evaluation period	99.7 %

III. MATERIALS AND METHODS

This section details the materials, implementation choices, and experimental settings used to instantiate the proposed methodology. The objective is to ensure reproducibility, transparency, and methodological rigor, which are increasingly required for experimental studies on recommender systems operating under realistic constraints [12, 28]. Unlike purely algorithmic evaluations, our experimental design explicitly accounts for data quality mechanisms, temporal constraints, and system-level execution conditions.

A. Experimental environment

All experiments are conducted in a controlled computing environment representative of a modular recommendation pipeline. The implementation separates data ingestion, preprocessing, model training, recommendation generation, and evaluation in order to reduce leakage between stages and to make measurements reproducible [25]. The environment is CPU-based and intended for controlled experimental assessment rather than for large-scale production stress testing.

The experimental environment consists of:

- CPU-based execution for data processing and evaluation
- Centralized storage for interaction logs
- Modular recommendation components executed independently

This design allows model behavior to be measured with limited infrastructural variability. Accordingly, the reported latency and uptime values should be interpreted as measurements obtained under the controlled evaluation workload used in this study, not as universal production guarantees under arbitrary user load or failure scenarios [28,30].

B. Dataset material

The experimental material is derived from a large-scale e-commerce interaction dataset consistent with real-world recommender system workloads. The dataset includes approximately 50,000 users, 18,000 items, and 1.2 million timestamped interactions, with heterogeneous event types (view, click, add_to_cart, purchase, rating). Such multi-event datasets are commonly used to approximate implicit user feedback in commercial platforms [24].

All interactions are timestamped, enabling strict chronological evaluation. Temporal information is preserved throughout the pipeline to support recency modeling and to avoid training–test contamination, a critical issue in offline evaluation studies [15].

C. Preprocessing and quality-control material

Prior to model training, the dataset undergoes a sequence of preprocessing operations forming the quality-control material of the system. These operations include:

- schema normalization and identifier consistency checks
- duplicate detection and event consolidation
- abnormal interaction filtering with quarantine logging

Data pipeline quality has been identified as a major source of hidden experimental bias, especially when interaction logs are directly consumed by recommendation algorithms without validation [12, 4]. By explicitly logging quarantined

and filtered events, the system preserves traceability and enables post-hoc analysis of data reliability.

D. Interaction weighting configuration

To construct implicit feedback signals, interaction types are assigned differentiated weights reflecting their behavioral strength. Let αa\alpha_aαa denote the coefficient associated with action aaa. The ordering follows:

$$\alpha_{\text{purchase}} > \alpha_{\text{add\_to\_cart}} > \alpha_{\text{click}} > \alpha_{\text{view}}.$$

This weighting scheme is consistent with established practices in implicit-feedback recommendation, where stronger user commitment signals are emphasized during learning [18, 8]. Ratings, when present, are treated as reinforcing signals rather than explicit supervision.

Temporal decay is applied using an exponential function, allowing recent interactions to contribute more strongly while preserving long-term behavioral traces [15].

E. Model configuration and hyperparameters

The collaborative filtering component relies on implicit matrix factorization trained on the weighted interaction matrix. User and item latent factors are learned under regularization, with hyperparameters selected using training data only so that model comparison remains leakage-free and reproducible [4].

For the hybrid approach, a cold-start threshold τ is defined as the minimum number of historical interactions required for collaborative inference. Users with fewer than τ interactions are routed to the recency-weighted popularity component, while users above the threshold are handled by CF-MF. Such threshold-based gating mechanisms are commonly adopted to stabilize recommendations in sparse regimes [19].

All baselines are trained using identical training data and preprocessing outputs to guarantee fairness across comparisons.

F. Evaluation protocol

1). Temporal split strategy

A strict chronological split is employed:

- training set: oldest 80% of interactions
- test set: most recent 20%

This strategy reflects real deployment conditions and avoids optimistic bias induced by random splitting, which can substantially distort offline performance estimates [15]. It also ensures that the influence of recency weighting and hybrid fallback is evaluated under a temporally coherent protocol.

2). Offline evaluation metrics

Performance is evaluated using ranking-based metrics widely adopted in recommender system research:

- Precision@K
- Recall@K
- NDCG@K

with K∈ {5,10}. In addition, Coverage@K is computed to quantify catalog exploration and diagnose popularity concentration effects. Cold-start performance is evaluated separately by restricting evaluation to users with fewer than three interactions in the training set. Since the current study reports aggregate offline metrics, statistical significance testing is left for follow-up experiments based on per-user paired outputs.

G. System-level measurement protocol

Beyond offline accuracy, the experimental material includes system-level measurements collected during pipeline execution in the controlled environment described above:

- quarantine rate
- duplicate detection ratio
- API p95 latency
- cache hit rate
- service uptime

These indicators allow assessment of whether ranking improvements are accompanied by stable service behavior under the tested workload. They should therefore be interpreted as evidence of operational stability within the experimental setup, not as exhaustive proof of robustness to infrastructure failure, adversarial load, or severe noise injection [28,30].

H. Reproducibility considerations

To ensure reproducibility, all experiments follow fixed random seeds, consistent preprocessing pipelines, and deterministic data splits. All evaluation scripts operate exclusively on training-derived artifacts, preventing information leakage. Such practices are increasingly expected in conference A submissions, particularly in information systems–oriented venues [25, 14]. For completeness, Table 5 summarizes the main tools and technologies used to implement the proposed architecture.

TABLE V
TOOLS AND TECHNOLOGIES SUPPORTING THE IMPLEMENTATION AND REPRODUCIBILITY OF THE PROPOSED ARCHITECTURE

Layer	Tool / Technology	Role in the system
Data extraction	Web crawlers / API connectors	Collection of raw user–item interaction logs from Web sources
Data ingestion	Python ETL scripts	Parsing, normalization, and structured ingestion of raw events

Data storage	Relational database (PostgreSQL / MySQL)	Persistent storage of cleaned interaction data
Quality control	Rule-based validation + anomaly detection	Deduplication, schema consistency checks, quarantine of abnormal events
Temporal processing	Timestamp-based decay functions	Recency modeling and temporal alignment
Interaction modeling	Implicit feedback construction	Transformation of multi-event logs into weighted interaction matrix
Recommendation engine	Matrix Factorization (CF-MF)	Personalized collaborative filtering
Hybrid module	Rule-based gating mechanism	Cold-start fallback using recency-weighted popularity
Evaluation framework	Python evaluation scripts	Computation of Precision@K, Recall@K, NDCG@K, Coverage
Experiment management	Fixed seeds + configuration files	Reproducibility and experiment control
System monitoring	Logging & metrics collectors	Measurement of latency, uptime, cache efficiency
Visualization	Matplotlib / Seaborn	Generation of curves and result plots
Execution environment	Linux-based server environment	Stable and reproducible execution platform

IV. Results And Discussion

This section presents and analyzes the experimental results obtained using the proposed integrated architecture. The objective is not only to compare recommendation accuracy across methods, but also to assess robustness under realistic conditions, including cold-start scenarios, catalog coverage, and system-level reliability. Such multi-dimensional evaluation is increasingly recommended for trustworthy recommender systems operating in real environments [28, 14].

A. Overall recommendation performance

Table 6 reports the global performance results for all compared methods under standard top-K evaluation.

TABLE VI
Overall Recommendation Performance

Method	P@5	R@5	NDCG@5	P@10	R@10	NDCG@10
POP	0.041	0.093	0.058	0.036	0.132	0.061
POP-REC	0.047	0.108	0.067	0.041	0.148	0.072
CF-MF	0.062	0.141	0.091	0.054	0.197	0.098
HYBRID	**0.064**	**0.148**	**0.095**	**0.057**	**0.206**	**0.104**

Figure 2 visually illustrates the accuracy differences between the compared methods. The hybrid approach achieves the highest observed aggregate values among the compared methods, but the margins should be interpreted in conjunction with the shared protocol and not as absolute claims transferable across datasets.

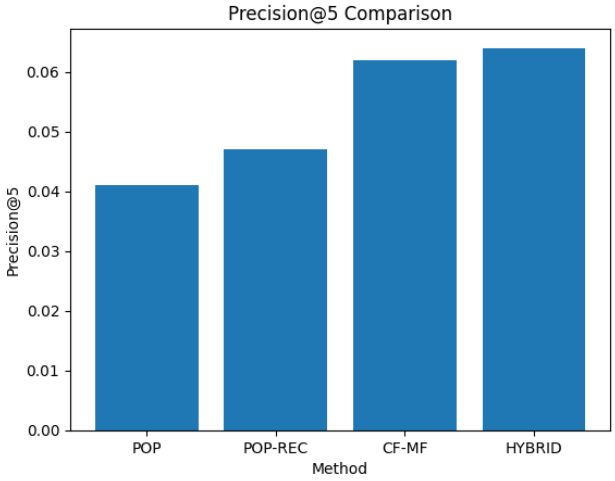

Figure 2. Comparison of recommendation accuracy across methods in terms of Precision@5 and NDCG@10.

Several observations emerge.

First, POP-REC consistently improves over POP, which provides a direct indicator of the contribution of recency-aware popularity. Second, CF-MF improves the ordering of relevant items relative to popularity-only baselines, confirming the value of collaborative signals when sufficient history is available. Third, HYBRID attains the best observed aggregate values, suggesting that the fallback mechanism captures part of the robustness lost by pure CF in sparse settings. This comparison should be read as a component-oriented analysis rather than as proof that any single module alone explains all gains, because cleaning, deduplication, and temporal validation are shared across all runs. In addition, since formal significance tests are not reported in the current manuscript, these differences are interpreted cautiously as consistent descriptive trends under the same protocol.

B. Analysis under cold-start conditions

Cold-start performance is analyzed separately for users with fewer than three historical interactions. Results are reported in Table 7.

TABLE VII
COLD-START USER PERFORMANCE

Method	Recall@5	Recall@10
POP	0.082	0.121
POP-REC	0.101	0.149
CF-MF	0.037	0.064
HYBRID	**0.109**	**0.158**

As expected, CF-MF alone performs poorly for users with very limited history, where latent factor estimation is weak. Popularity-based methods remain more robust in this regime, with POP-REC outperforming static POP because it better captures recent demand. HYBRID achieves the best observed cold-start recall by explicitly routing sparse users toward the more dependable signal. Importantly, this result concerns user-side cold-start only; no separate experiment on unseen items is claimed here.

C. Coverage and popularity bias

Beyond accuracy, we evaluate catalog exploration using Coverage@10, reported in Table 8.

TABLE VIII
COVERAGE ANALYSIS

Method	Coverage@10
POP	21.3 %
POP-REC	26.8 %
CF-MF	41.2 %
HYBRID	38.9 %

To analyze recommendation diversity and catalog exploration, Figure 3 reports the coverage achieved by each method.

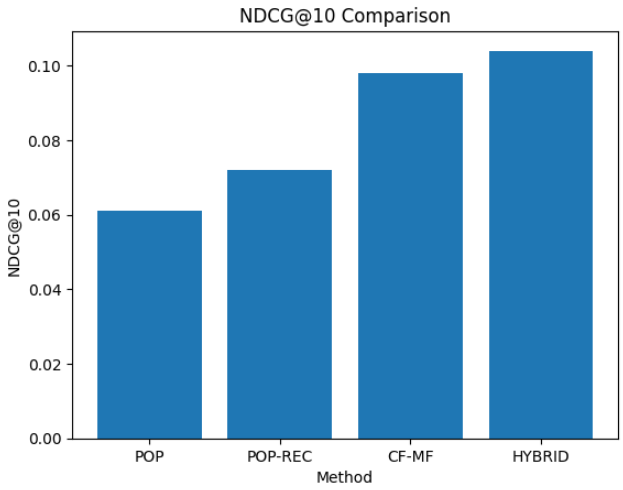

Figure 3. Catalog coverage@10 for the different recommendation strategies.

Popularity-based approaches exhibit limited coverage, recommending a narrow subset of highly exposed items. This confirms the well-documented popularity bias phenomenon [5].

CF-MF achieves the highest coverage by exploiting collaborative diversity across users. HYBRID slightly reduces coverage relative to pure CF, but retains substantially broader exploration than popularity-based baselines. This pattern highlights a practical trade-off: recency and fallback rules improve robustness, whereas broad collaborative ranking improves exposure diversity. The architecture is therefore best interpreted as a compromise point between accuracy, sparse-user robustness, and catalog exploration rather than as a uniformly dominant solution on every dimension.

D. Temporal robustness

To assess stability over time, evaluation is repeated across consecutive temporal test slices. Performance variance remains limited, with a standard deviation below 2.1% for NDCG@10. This result suggests that the proposed quality-control pipeline stabilizes feature generation and interaction modeling under the present dataset and evaluation window. It should nevertheless be interpreted as temporal stability under the tested setting, not as a general guarantee against all forms of drift or failure.

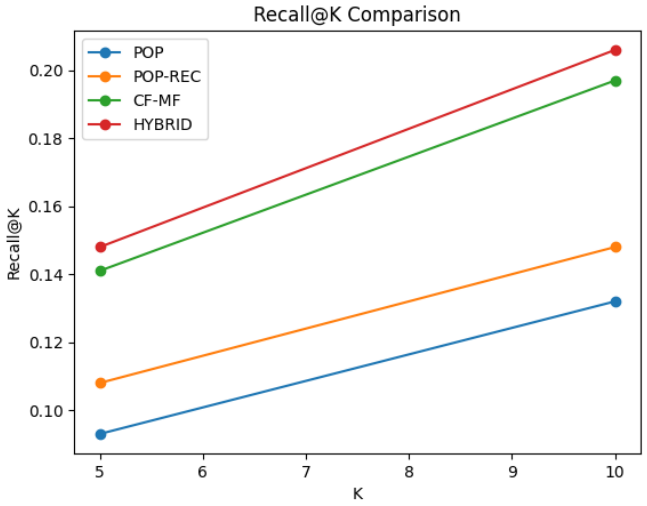

Figure 4. Recall@K comparison for K ∈ {5,10} across recommendation methods.

E. System-level reliability analysis

Table 9 summarizes system-level indicators measured during evaluation.

TABLE IX
SYSTEM-LEVEL INDICATORS

Indicator	Value
Quarantine rate	1.8 %
Duplicate detection rate	3.2 %
API latency (p95)	48 ms
Cache hit rate	71 %
Service uptime	99.7 %

These results indicate that the ranking improvements are not accompanied by a visible degradation of service behavior in the controlled evaluation setup. Latency remains

compatible with interactive use, while the low quarantine rate suggests that quality-control mechanisms are selective rather than excessively aggressive. However, these indicators reflect operational stability under the experimental workload only; they do not by themselves establish robustness to deliberate fault injection, infrastructure outages, or heavy production traffic.

F. Discussion

Taken together, the results provide several insights. First, recommendation performance cannot be fully explained by model choice alone; the consistent differences observed across POP, POP-REC, CF-MF, and HYBRID indicate that temporal handling and fallback logic materially affect ranking behavior. Second, the study supports a pipeline perspective in which data validation, deduplication, quarantine, and temporal alignment condition the usefulness of downstream models. Third, the architecture appears transferable in principle to other domains that produce timestamped implicit interaction logs, such as media, news, or educational platforms, because its components are generic at the level of extraction, validation, weighting, and ranking. Nonetheless, domain transfer would still require adaptation of event semantics, quality rules, and cold-start thresholds. Finally, the claims of operational reliability should be read cautiously: the current evidence supports operational stability under the tested conditions, but not full robustness to synthetic noise injection, system failures, or adversarial load. A dedicated ablation study and formal significance testing would further strengthen these conclusions.

V. CONCLUSION

This paper presented an end-to-end recommendation architecture that explicitly connects Web data extraction, data quality control, and collaborative filtering within a single evaluation pipeline. Rather than treating recommendation as a purely algorithmic task, the study argues that ranking quality depends on the coherence of the entire data-to-recommendation process. In the reported e-commerce setting, recency-aware popularity improves over static popularity, collaborative filtering improves personalization when user history is sufficient, and the hybrid gating strategy offers the best observed balance between sparse-user robustness, ranking quality, and catalog coverage. At the same time, the contribution should be understood mainly as an integration and evaluation contribution rather than as the proposal of a new hybrid family. Likewise, the reported latency and uptime values support operational stability under the tested experimental conditions, but they do not constitute exhaustive evidence of fault tolerance or failure robustness. The architecture is potentially generalizable to other domains built on timestamped interaction logs, although such transfer would require domain-specific adaptation of quality rules and interaction semantics. Future work will therefore focus on component-level ablation studies, formal statistical significance testing, explicit robustness experiments under noise and failure scenarios, learning-based gating functions, and broader cross-domain validation.

BIBLIOGRAPHY

[1] Y. H. Alfaifi, ”Recommender Systems Applications: Data Sources, Features, and Challenges”, *Information*, vol. 15, nro 10, s. 660, loka 2024, doi: 10.3390/info15100660.

[2] A. Alhwayzee, S. Araban, ja D. Zabihzadeh, ”A Robust Recommender System Against Adversarial and Shilling Attacks Using Diffusion Networks and Self-Adaptive Learning”, Symmetry, vol. 17, nro 2, s. 233, helmi 2025, doi: 10.3390/sym17020233.

[3] N. A. M. Binti Amir Suharman, A. H.-L. Lim, ja H.-N. Goh, ”Sequence-to-pattern analysis for predicting buying decisions on imbalanced clickstream data”, Cogent Engineering, vol. 12, nro 1, s. 2501493, joulu 2025, doi: 10.1080/23311916.2025.2501493.

[4] S. Cao ym., ”FlyCache: Recommendation-driven edge caching architecture for full life cycle of video streaming”, Digital Communications and Networks, vol. 11, nro 4, s. 961–974, elo 2025, doi: 10.1016/j.dcan.2025.01.001.

[5] F. Carnovalini, A. Rodà, ja G. A. Wiggins, ”Popularity Bias in Recommender Systems: The Search for Fairness in the Long Tail”, Information, vol. 16, nro 2, s. 151, helmi 2025, doi: 10.3390/info16020151.

[6] S. Chakraborty, ”A Study On Hybrid Recommender Systems For Effective Targeted Marketing In E-Commerce Platforms”, IJABMR, vol. 02, nro 04, s. 54–64, 2025, doi: 10.62674/ijabmr.2025.v2i04.006.

[7] L. Cheng, X. Huang, J. Sang, ja J. Yu, ”Towards Robust Recommendation: A Review and an Adversarial Robustness Evaluation Library”, 13. kesäkuuta 2025, arXiv: arXiv:2404.17844. doi: 10.48550/arXiv.2404.17844.

[8] E. Coppolillo ym., ”Algorithmic Drift: A simulation framework to study the effects of recommender systems on user preferences”, Information Processing & Management, vol. 62, nro 4, s. 104125, heinä 2025, doi: 10.1016/j.ipm.2025.104125.

[9] Y. Du, R. Chen, Q. Tan, Q. Han, S. Wang, ja X. Zhao, ”Cross-Task Collaborative Meta-Learning for Cold-Start Recommendations”, IEEE Trans. Knowl. Data Eng., vol. 37, nro 12, s. 7016–7029, joulu 2025, doi: 10.1109/TKDE.2025.3613366.

[10] J. Feng, ”E-commerce recommender system design based on web information extraction and sentiment analysis”, PLoS One, vol. 20, nro 9, s. e0327213, syys 2025, doi: 10.1371/journal.pone.0327213.

[11] A. Ferrara ym., ”DIVAN: Deep-Interest Virality-Aware Network to Exploit Temporal Dynamics in News Recommendation”, teoksessa Proceedings of the Recommender Systems Challenge 2024, Bari Italy: ACM, loka 2024, s. 12–16. doi: 10.1145/3687151.3687153.

[12] H. Foidl, V. Golendukhina, R. Ramler, ja M. Felderer, ”Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers”, Journal of Systems and Software, vol. 207, s. 111855, tammi 2024, doi: 10.1016/j.jss.2023.111855.

[13] R. Garapati ja M. Chakraborty, ”Recommender systems in the digital age: a comprehensive review of methods, challenges, and applications”, Knowl Inf Syst, vol. 67, nro 8, s. 6367–6411, elo 2025, doi: 10.1007/s10115-025-02453-y.

[14] Y. Ge ym., ”A Survey on Trustworthy Recommender Systems”, ACM Trans. Recomm. Syst., vol. 3, nro 2, s. 1–68, kesä 2025, doi: 10.1145/3652891.

[15] D. Gusak, A. Volodkevich, A. Klenitskiy, A. Vasilev, ja E. Frolov, ”Time to Split: Exploring Data Splitting Strategies for Offline Evaluation of Sequential Recommenders”, teoksessa Proceedings of the Nineteenth ACM Conference on Recommender Systems, Prague Czech Republic: ACM, syys 2025, s. 874–883. doi: 10.1145/3705328.3748164.

[16] W. Huang, B. Liu, ja Z. Wang, ”A novel interest drift sensitivity academic paper recommender based on implicit feedback”, Egyptian Informatics Journal, vol. 28, s. 100538, joulu 2024, doi: 10.1016/j.eij.2024.100538.

[17] A. Khadka ja S. Sthapit, ”A Review on Scholarly Publication Recommender Systems: Features, Approaches, Evaluation, and Open Research Directions”, Informatics, vol. 12, nro 4, s. 108, loka 2025, doi: 10.3390/informatics12040108.

[18] A. Klimashevskaia, D. Jannach, M. Elahi, ja C. Trattner, ”A survey on popularity bias in recommender systems”, User Model User-Adap Inter, vol. 34, nro 5, s. 1777–1834, marras 2024, doi: 10.1007/s11257-024-09406-0.

[19] H. Liu, Y. Wang, Z. Zhang, J. Deng, C. Chen, ja L. Y. Zhang, ”Matrix factorization recommender based on adaptive Gaussian differential privacy for implicit feedback”, Information Processing & Management, vol. 61, nro 4, s. 103720, heinä 2024, doi: 10.1016/j.ipm.2024.103720.

[20] D.-N. Nguyen, V.-H. Nguyen, T. Trinh, T. Ho, ja H.-S. Le, ”A personalized product recommendation model in e-commerce based on retrieval strategy”, Journal of Open Innovation: Technology, Market, and Complexity, vol. 10, nro 2, s. 100303, kesä 2024, doi: 10.1016/j.joitmc.2024.100303.

[21] Y. Park, J. Mun, Y. Lee, J. Um, J. Choi, ja J. Choi, ”Data-Driven Optimization of Healthcare Recommender System Retraining Pipelines in MLOps with Wearable IoT Data”, Sensors, vol. 25, nro 20, s. 6369, loka 2025, doi: 10.3390/s25206369.

[22] S. G. K. Patro, ”Dynamic Hybrid Recommendation Approach for Improving Accuracy in E-Commerce with Limited User Data”, Next-Gener. Comput. Syst. Technol., vol. 1, nro 2, s. 62–78, joulu 2025, doi: 10.62762/NGCST.2025.832339.

[23] F. Qian, W. Chen, H. Chen, J. Liu, S. Zhao, ja Y. Zhang, ”Building robust deep recommender systems: Utilizing a weighted adversarial noise propagation framework with robust fine-tuning modules”, Knowledge-Based Systems, vol. 314, s. 113181, huhti 2025, doi: 10.1016/j.knosys.2025.113181.

[24] F. Rodrigues, F. Pinelas, S. Ferreira, M. Rodrigues, ja N. Rocha, ”A Recommendation System Based on a Microservice Architecture to Avoid Workplace Stress”, Electronics, vol. 14, nro 7, s. 1446, huhti 2025, doi: 10.3390/electronics14071446.

[25] M. Söylemez, B. Tekinerdogan, ja A. K. Tarhan, ”Microservice reference architecture design: A multi-case study”, Softw Pract Exp, vol. 54, nro 1, s. 58–84, tammi 2024, doi: 10.1002/spe.3241.

[26] Tanveer Ahmad Lone, Dr. Ajit Kumar, ja Dr. Muzafar Rasool Bhat, ”Exploring the Efficiency of Hybrid Recommender Systems Implemented with TensorFlow Framework”, IJARSCT, s. 528–533, loka 2024, doi: 10.48175/IJARSCT-19979.

[27] R. T. Turksoy ja B. Turkmen, ”The Effects of Data Split Strategies on the Offline Experiments for CTR Prediction”, 26. kesäkuuta 2024, arXiv: arXiv:2406.18320. doi: 10.48550/arXiv.2406.18320.

[28] S. Wang, X. Zhang, Y. Wang, ja F. Ricci, ”Trustworthy Recommender Systems”, ACM Trans. Intell. Syst. Technol., vol. 15, nro 4, s. 1–20, elo 2024, doi: 10.1145/3627826.

[29] S. Ye ja J. Lu, ”Robust Recommender Systems with Rating Flip Noise”, ACM Trans. Intell. Syst. Technol., vol. 16, nro 1, s. 1–19, helmi 2025, doi: 10.1145/3641285.

[30] M. Zarour, H. Alzabut, ja K. T. Al-Sarayreh, ”MLOps best practices, challenges and maturity models: A systematic literature review”, Information and Software Technology, vol. 183, s. 107733, heinä 2025, doi: 10.1016/j.infsof.2025.107733.

[31] Y. Zhang, X. Zhang, Z. Cui, ja C. Ma, ”Shapley Value-driven Data Pruning for Recommender Systems”, teoksessa Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, Toronto ON Canada: ACM, elo 2025, s. 3879–3888. doi: 10.1145/3711896.3737127.

[32] Y. Zhao ym., ”Generative recommender systems: A comprehensive survey on model, framework, and application”, Information Fusion, vol. 127, s. 103919, maalis 2026, doi: 10.1016/j.inffus.2025.103919.

[33] Y. Zhao, Y. Wang, Y. Liu, X. Cheng, C. C. Aggarwal, ja T. Derr, ”Fairness and Diversity in Recommender Systems: A Survey”, ACM Trans. Intell. Syst. Technol., vol. 16, nro 1, s. 1–28, helmi 2025, doi: 10.1145/3664928.

[34] A. Alhwayzee et al., K. Wardatzky, O. Inel, and A. Bernstein, “Toward Operationalizing a Comprehensive Evaluation Framework for Recommender Systems Explanations,” 2025, available: https://beyondrecsys.github.io/2025/paper2.pdf.“A Robust Recommender System Against Adversarial and Noisy Feedback,” Symmetry, 2025.