applied sciences

MDPI

Article

A Hybrid Recommendation System Based on Similar-Price Content in a Large-Scale E-Commerce Environment

Youngoh Kwon [1], Gwiman Bak [1] and Youngchul Bae [2,*]

[1] Department Electrical and Semiconductor Engineering, Chonnam National University, Yeosu 59626, Republic of Korea; iahalu@nate.com (Y.K.); qkrrlend@naver.com (G.B.)
[2] Division Electrical and Computing Engineering, Chonnam National University, Yeosu 59626, Republic of Korea
* Correspondence: ycbae@jnu.ac.kr

Abstract

In large-scale e-commerce, recommendation systems must overcome the shortcomings of conventional models, which often struggle to convert user interest into purchases. This study proposes a revenue-driven recommendation approach that explicitly incorporates user price sensitivity. This study introduces a hybrid recommendation engine that combines collaborative filtering (CF), best match 25 (BM25) for textual relevance, and a price-similarity algorithm. The system is deployed within a scalable three-tier architecture using Elasticsearch and Redis to maintain stability under high-traffic conditions. The system's performance was evaluated through a large-scale A/B test against both a CF-only model and a popular-item baseline. Results showed that while the CF-only model reduced revenue by 5.10%, our hybrid system increased revenue by 5.55% and improved click-through rate (CTR) by 2.55%. These findings demonstrate that integrating price similarity is an effective strategy for developing commercially viable recommendation systems that enhance both user engagement and revenue growth on large online platforms.

Keywords: recommendation system; hybrid recommendation; collaborative filtering; price similarity; BM25; A/B testing; E-commerce; revenue growth; large-scale platforms

check for updates

Academic Editors: Yue Kou, Xiangmin Zhou and Chaochao Chen

Received: 3 September 2025
Revised: 2 October 2025
Accepted: 2 October 2025
Published: 6 October 2025

Citation: Kwon, Y.; Bak, G.; Bae, Y. A Hybrid Recommendation System Based on Similar-Price Content in a Large-Scale E-Commerce Environment. *Appl. Sci.* **2025**, *15*, 10758. https://doi.org/10.3390/app151910758

1. Introduction

With the rapid expansion of the e-commerce sector, recommender systems have become indispensable tools [1] for identifying diverse user preferences and delivering personalized product suggestions at the right moment. Their importance extends beyond product discovery to enhancing the overall user experience, which in turn improves conversion rates and drives revenue [2–4].

Previous research has shown that recommendation strategies leveraging real-time transaction data and user activity logs, while simultaneously considering value perception and profit optimization, can generate sustainable long-term revenue growth for online platforms [5]. Moreover, evidence from a case study of a digital media service has revealed that a personalized recommendation algorithm not only reduced customer churn but also substantially increased content consumption, underscoring the transformative role of personalization in digital commerce [6].

With the rapid growth of big data and artificial intelligence, numerous deep learning-based models, such as those that have revolutionized image recognition [7], have been developed. In recent years, graph-based methods such as Knowledge Graph-based models [8] and Graph Neural Networks (GNNs) have become a mainstream approach, effectively modeling the complex relational structure of user-item interaction graphs [9,10]. Advanced techniques such as self-supervised learning have further enhanced their predictive power [11]. More recently, the advent of Large Language Models (LLMs) has introduced a new paradigm by leveraging their vast semantic understanding and world knowledge for recommendation [12]. While these state-of-the-art models excel at capturing latent user preferences, their operational complexity and the challenge of integrating explicit, business-critical factors like price sensitivity remain significant hurdles in large-scale commercial deployments.

Among these, the attention mechanism has been introduced to enhance traditional matrix factorization by dynamically assessing the relative importance of neighboring items, thereby improving both prediction accuracy and computational efficiency [13]. In addition, distributed architectures have been proposed to manage the heavy traffic generated by e-commerce platforms, ensuring faster response times and alleviating computational bottlenecks, compared to centralized systems. These advances suggest that real-time performance can be sustained even in large-scale applications [14].

Nevertheless, studies have noted that as models grow more complex, maintaining responsiveness during inference becomes increasingly challenging. Research has examined the trade-off between accuracy and response speed of recommendation systems, showing that the resource demands of distributed architectures increase as deep learning algorithms scale up [15]. In large-scale e-commerce contexts, delays in processing diverse user requests across multiple access points can lead to customer churn, reinforcing the need for strategies that optimize the use of limited computational resources [16–18].

Meta-learning has been explored as a means of mitigating bias and noise in recommendation models. Evidence indicates that incorporating consumer price sensitivity into recommendation strategies can improve conversion rates by aligning product suggestions with users' budget constraints [19,20]. Accordingly, there is growing recognition of the need for models that integrate diverse recommendation techniques, price factors, and real user logs.

An advanced personalized recommendation framework is proposed, built on a hybrid engine that combines collaborative filtering with content-based information, augmented by a price similarity algorithm. Drawing on prior findings, the hybrid collaborative filtering enhances both accuracy and stability relative to single algorithms [21]. Performance changes in click-through rate (CTR), conversion rate, and sales are assessed using log data from a large-scale e-commerce platform. The evaluation confirms that the proposed model delivers substantial academic and practical value: in high-traffic real-world conditions, it improves CTR, purchase volume, and revenue.

2. Literature Review on Recommendation Systems and Price-Based Approaches

The Materials and Methods section should provide sufficient detail to allow for the replication and extension of the results. Authors are expected to make all relevant materials, datasets, source code, and protocols associated with the study available to readers. Any restrictions on data or material availability should be disclosed at the time of submission. Newly developed methods and protocols must be described in full detail, while well-established methods may be summarized and appropriately cited.

Manuscripts reporting large datasets deposited in public repositories must specify the database location and include accession numbers that are not available at the time of submission. Authors should indicate that they will be provided during the review process; they must be supplied before publication.

Studies involving human or animal subjects, or any research requiring ethical clearance, must specify the approving authority and provide the corresponding approval code. Where applicable, authors should also disclose the role of generative artificial intelligence (GenAI) in their study (e.g., text generation, data creation, graphic production, study design, data collection, analysis, or interpretation). However, the use of GenAI for superficial editing tasks such as grammar correction, spelling, punctuation, or formatting does not require disclosure.

2.1. Recent Advances in Deep Learning-Based Recommendation

The landscape of recommender systems has been profoundly shaped by deep learning. A dominant trend in recent years is the application of Graph Neural Networks (GNNs), which treat user-item interactions as a graph and excel at capturing complex, high-order dependencies using techniques like representation learning [22]. Comprehensive surveys have detailed the challenges, methods, and future directions of GNNs in this domain [9,10]. Techniques such as self-supervised graph learning have been proposed to tackle issues like data sparsity and further improve the robustness of GNN-based recommendations [11].

More recently, Large Language Models (LLMs) have emerged as a powerful new frontier. By leveraging their pre-trained knowledge, LLMs can provide recommendations with a deep semantic understanding of both user queries and item descriptions, offering a new avenue for personalization, especially in cold-start scenarios [12].

Despite the remarkable accuracy of these state-of-the-art models, their direct application in high-traffic, revenue-driven e-commerce environments presents practical challenges. These include high computational costs for training and inference, the "black box" nature of their predictions, and the difficulty of explicitly enforcing practical business constraints, such as aligning recommendations with user price sensitivity. Therefore, a significant need remains for hybrid models that balance advanced algorithmic techniques with scalable, interpretable, and commercially pragmatic factors, which is the primary focus of our study.

2.2. Collaborative Filtering

Collaborative filtering (CF) is a recommendation algorithm that identifies "similar users" or "similar items" based on historical user-item interactions, including product views, items added to the cart, and purchases [23].

- Advantages:
  a. Relies solely on user behavior data, without requiring domain-specific knowledge.
  b. Effective captures and reflects emerging trends.
- Disadvantages:
  a. Performance declines when interaction data are sparse, a limitation known as the "cold-start" problem.
  b. Susceptible to recommendation bias, in which a few popular items dominate exposure [24].

Representative CF approaches include user-based CF, Item-based CF, and matrix factorization (MF) [25]. While each method infers user-item relationships differently, they all require additional scaling strategies, such as distributed computing, caching, and index optimization, to handle large-scale traffic and data sparsity. In practice, techniques such as in-memory caching and Elasticsearch indexing are commonly employed [26,27] to reduce latency and deliver real-time, personalized recommendations.

2.3. Content-Based Filtering with BM25

Content-based recommendation systems evaluate product-related textual information, such as titles, descriptions, and user reviews, to identify items that align with a user's profile [28]. In e-commerce applications, text-mining techniques frequently improve recommendation accuracy by extracting relevant keywords from product descriptions [29].

$$score(D, Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{1}$$

The BM25 algorithm, shown in Equation (1), is a widely adopted text similarity ranking function in information retrieval (IR). It calculates a score for each product by considering term frequency (TF), inverse document frequency (IDF), and document length normalization [30,31]. In e-commerce, a higher BM25 score for a product title or description indicates stronger alignment with a user's query, thereby enhancing search accuracy and user satisfaction [32].

- Example: For query "wireless earphone", products containing the term more frequently, combined with a high IDF value, receive higher BM25 scores and appear more prominently in search rankings.

By integrating CF with content-based methods such as BM25, hybrid recommendation models can be developed. These models leverage both behavioral data and product text features, thereby mitigating the weakness of individual algorithms. The effectiveness of hybrid approaches has been consistently demonstrated in recent research [23,28,33,34].

2.4. The Concept and Necessity of Price Similarity Recommendation

In e-commerce, price is a critical determinant of consumer purchasing behavior. When recommended products fall outside a user's expected price range, strong psychological resistance may arise, often reducing the likelihood of purchase [35–37]. To address this, the concept of "price similarity" is introduced, which estimates a user's preferred price range from recent purchases or cart history and prioritizes recommendations within that range. For example, a customer who frequently purchases items for ₩30,000 or less is unlikely to engage with products priced above ₩100,000 and may even be at a greater risk of churn if such recommendations dominate.

The "price similarity" framework proposed in this study extends beyond a simple "±n% range." It integrates price sensitivity into recommendations by first selecting products based on textual similarity (via BM25) and then merging this pool with outputs from collaborative filtering. A price-adjustment algorithm then applies dynamic weights to control ranking order.

- If a product's price exceeds the user's baseline, its score is reduced to prevent overexposure of high-cost items.
- If a product's price falls below baseline, the adjustment depends on user tendencies; for instance, some may avoid excessively cheap items, ensuring balanced placement in the ranking.

This weighting mechanism can be modeled using a logarithmic function, where adjustments intensify as the price deviation increases. Specifically, the algorithm modifies a product's score by adding or subtracting coefficients relative to the baseline. Higher-priced items incur deductions, whereas different coefficients adjust lower-priced items to maintain relevance. This approach ensures that products within the preferred range receive priority while preventing excessive recommendations of extreme outliers.

By combining BM25-based textual similarity, price-sensitivity weighting, and CF within a hybrid model, the system simultaneously captures user-interest keywords and maintains alignment with individual spending preferences. Relevance and quality are further validated using interaction data from other users.

This methodology is particularly effective in large-scale, high-traffic environments. Because price weighting can be integrated directly into real-time log analysis and large-scale index querying, the system improves efficiency while delivering personalized price-aware recommendations in near real-time [38]. Additionally, it mitigates the popularity bias inherent in conventional CF and increases visibility for mid-priced or newly introduced products, thereby promoting sales diversity [24,39,40].

Moreover, the principle of considering user price sensitivity is not limited to single-item recommendations. Recent research has also explored this concept within the context of bundle recommendations, developing methods to suggest a package of items tailored to budget-conscious consumers [41]. This highlights the broad academic and practical interest in price-aware strategies, though our study maintains its focus on applying price similarity to enhance individual product recommendations in a large-scale e-commerce environment.

2.5. Hybrid Recommendation Architecture in High-Traffic Environments

Modern e-commerce platforms manage tens to hundreds of millions of product records, while real-time user activity logs may generate thousands to tens of thousands of events per second [38,42]. To ensure stable operation under such large-scale traffic conditions, the architectural elements outlined in Table 1 are essential.

Table 1. Large-scale traffic action items.

Component	Key Takeaways
Distributed search engine utilization	Uses distributed search engines such as Elasticsearch or Solr for product searches that integrate BM25 and price weighting, allowing horizontal scaling of query performance under heavy traffic. Well-suited for scale-out architectures.
Memory efficiency and caching	Reduces the computational cost of collaborative filtering by caching predicted results for frequently queried or computed user-item pairs. Ensure rapid response times by minimizing redundant computations.
Log ingestion and real-time analysis	Collects click, cart, and purchase logs through an event streaming platform (e.g., Kafka) and performs batch, parallel, and real-time analysis. Rapidly integrates analysis outcomes into the recommendation model, improving freshness and accuracy.
Hybrid ranking adjustment	Determines final recommendation order using a ranking function that combines BM25 scores, CF scores, and price weights Built with a modular structure that supports scale-out in large-scale environments

In summary, the proposed hybrid architecture (CF + BM25 + price similarity) is designed for large-scale traffic scenarios. Its strength lies in overcoming the limitations of individual algorithms, such as cold start and popularity bias, by jointly incorporating product text features, CF signals, and user-specific price ranges.

3. Design and Implementation of a Very Large-Scale Hybrid Recommendation System

This section details the research process for developing a large-scale hybrid recommendation system and a three-tier architecture capable of reliably collecting, analyzing, and learning from user event streams while generating recommendations. The system provides personalized results by integrating CF with BM25-based search.

3.1. Research Process Design

This research process begins with the collection and preprocessing of large-scale e-commerce logs and event data. It then advances to the design and implementation of a hybrid recommendation algorithm that integrates CF, BM25, and price similarity (Figure 1).

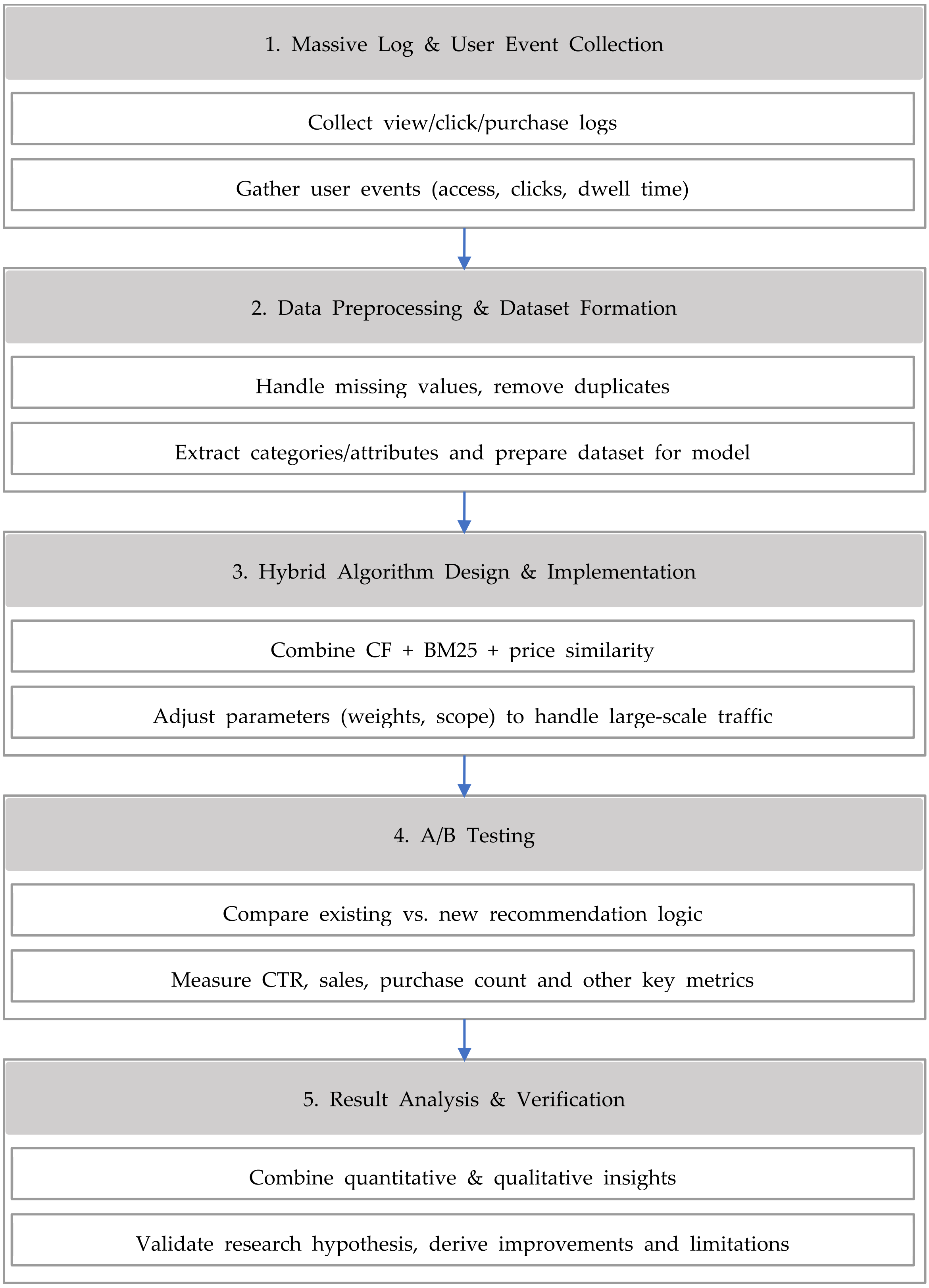

1. Massive Log & User Event Collection
Collect view/click/purchase logs
Gather user events (access, clicks, dwell time)
2. Data Preprocessing & Dataset Formation
Handle missing values, remove duplicates
Extract categories/attributes and prepare dataset for model
3. Hybrid Algorithm Design & Implementation
Combine CF + BM25 + price similarity
Adjust parameters (weights, scope) to handle large-scale traffic
4. A/B Testing
Compare existing vs. new recommendation logic
Measure CTR, sales, purchase count and other key metrics
5. Result Analysis & Verification
Combine quantitative & qualitative insights
Validate research hypothesis, derive improvements and limitations

Figure 1. Research process.

In this study, we developed a five-step process for constructing a recommendation system tailored to large-scale e-commerce applications. The research process commenced with the collection of diverse log data from real shopping mall operations—including product views, clicks, and purchases—along with user event data such as access records and clickstreams, providing a foundational dataset for in-depth analysis of user-specific behavioral patterns. Subsequently, the collected data were systematically classified and refined through processes such as handling missing values, removing duplicates, and extracting relevant attributes to create a structured dataset suitable for analysis. The core of the process was the design and implementation of a hybrid architecture integrating CF, BM25 text matching, and a price similarity algorithm, which concluded with the optimization of each algorithm's scope and weighting to ensure robust and stable performance under high-traffic conditions.

To evaluate the performance of the implemented algorithm, an A/B test was conducted. The new recommendation logic was applied to a subset of the system, and its impact on key metrics, such as CTR, purchase volume, and revenue, was compared with the existing method. Statistical tests were performed to ensure the observed improvements were significant, thereby objectively validating the superiority of the proposed approach. Both quantitative and qualitative outcomes were then examined, with particular attention to increases in CTR and revenue. This process also highlighted potential limitations and areas for improvement, providing insights for future research.

A large-scale, log-based dataset was constructed using transaction records from a leading domestic e-commerce platform spanning more than one month. The dataset comprises information on over 80 million products and contains between 9.6 million and 12 million daily user interaction logs, including product views, cart additions, and purchases. These logs incorporate product attributes, including names, categories, prices, and descriptions, alongside user-level histories of browsing, cart activity, and purchases.

Leveraging this extensive dataset, which reflects real-world large-scale e-commerce activity, enabled the evaluation of the proposed hybrid recommendation system under realistic conditions. By analyzing product- and user-level information from multiple perspectives, the study demonstrates the system's ability to operate reliably in high-traffic environments.

To assess performance, the evaluation employed the metrics summarized in Table 2. Each metric serves as a key diagnostic indicator for understanding the current state of the recommendation system and identifying opportunities for improvement. Collectively, they provide a comprehensive assessment of both algorithmic efficiency and business outcomes.

Table 2. Performance evaluation metrics.

Metric	Definition & Characteristics	Application & Significance
CTR (Click-Through Rate)	Ratio of clicks to impressions for a product or recommendation slot. Quantifies user engagement with recommendation results. Serves as an indirect indicator of system effectiveness.	Used to refine recommendation strategies and gauge user interest. Optimization results are validated through CTR changes.
Purchase Growth Rate	Measures the extent to which purchasing activity is stimulated by comparing the number of purchases before and after the implementation of the recommender system. Indicates whether higher CTR translates into actual purchase growth.	Assesses whether CTR improvements lead to conversions. Demonstrates the business impact of the recommendation strategy over time

Table 2. *Cont.*

Metric	Definition & Characteristics	Application & Significance
Revenue	Compares total sales before and after the implementation of the recommender system. Captures the algorithm's contribution to revenue growth.	Confirms financial outcomes (ROI) in relation to CTR and purchase trends. Provide persuasive evidence for management or resource allocation.
TPS (Transaction Per Second)	Indicates the number of transactions processed per second. Serves as an indicator of system stability and scalability under high-traffic conditions. Confirms whether operational requirements, such as zero-downtime index updates, are satisfied.	Evaluates the system's capacity to manage a sudden surge in concurrent users or traffic when the recommender system is deployed. Detects performance bottlenecks and validates scalability.

CTR, purchase growth rate, revenue, and TPS together provide valuable insights for both research and practice by assessing the recommender system from multiple perspectives and highlighting areas for improvement. These metrics enable a layered evaluation encompassing algorithmic sophistication, practical effectiveness, and operational stability under heavy traffic, thereby establishing themselves as essential criteria for designing and validating recommender systems.

By employing this framework, the present study seeks to assess not only the accuracy of recommendations but also system robustness and economic outcomes (e.g., revenue growth) in a real-world, large-scale e-commerce environment.

3.2. Algorithm Proposal and Performance Verification Strategy for Analyzing Practical Effect

Figure 2 illustrates the design of the hybrid recommendation algorithm proposed in this study. To capture realistic user preferences in an e-commerce environment, this study proposes a hybrid recommendation algorithm that integrates CF, BM25-based document search, price similarity modeling, and a combined scoring logic.

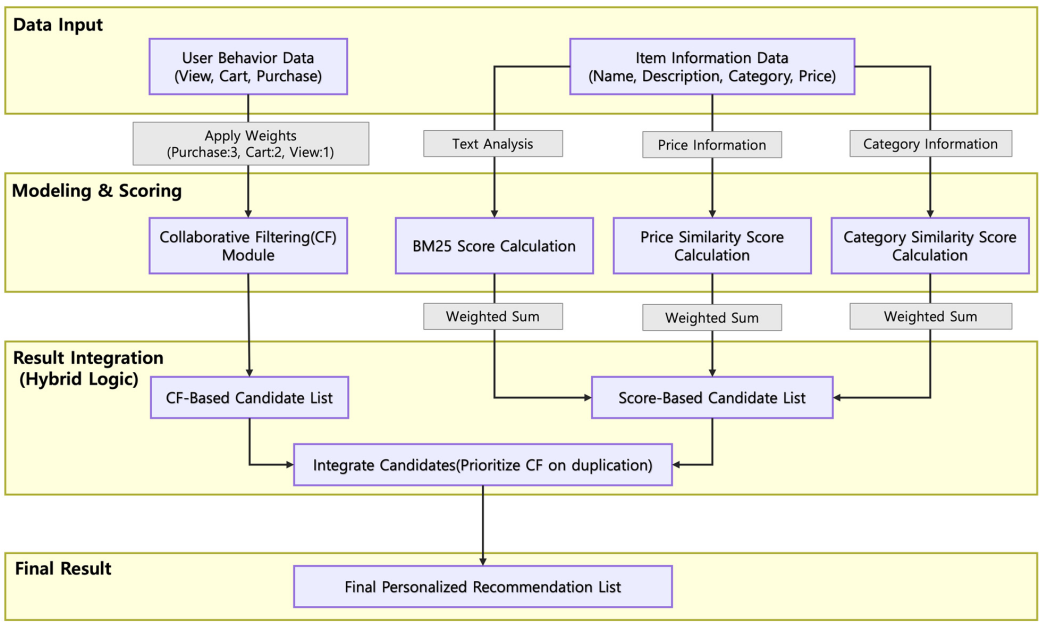

Data Input
User Behavior Data (View, Cart, Purchase)
Item Information Data (Name, Description, Category, Price)
Apply Weights (Purchase:3, Cart:2, View:1)
Text Analysis
Price Information
Category Information
Modeling & Scoring
Collaborative Filtering(CF) Module
BM25 Score Calculation
Price Similarity Score Calculation
Category Similarity Score Calculation
Weighted Sum
Weighted Sum
Weighted Sum
Result Integration (Hybrid Logic)
CF-Based Candidate List
Score-Based Candidate List
Integrate Candidates(Prioritize CF on duplication)
Final Result
Final Personalized Recommendation List

Figure 2. Hybrid recommendation algorithm architecture.

The CF module constructs a user-item interaction matrix and is trained by assigning different weights to user behaviors such as viewing, adding to cart, and purchasing (Table 3). Purchases are assigned the highest weight to more accurately reflect actual preferences, while views and cart additions receive lower weights. This weighting scheme enables the CF algorithm to produce recommendation scores (or candidate lists) that better capture user interests.

Table 3. User event type and weights.

Event Type	Description	Weight
View	Product viewed by clicking	1
Cart	Product added to cart	2
Purchase	Purchased product	3

The BM25-based document search module evaluates text relevance by calculating BM25 scores for product names, descriptions, and category texts. By accounting for document length and term frequency, this module enhances the alignment between user query keywords and product information, thereby filtering products that match the user's likely interests.

To further capture purchasing behavior, a price similarity model is applied. This method defines a $\pm\alpha\%$ range around either the user's most recent payment amount or the average price of previously viewed items. Products with this range receive additional scoring, while those outside the range are penalized. Since price strongly influences purchasing decisions, this mechanism prioritizes products with a higher likelihood of being purchased.

The hybrid logic then integrates these modules using a weighted score aggregation. First, the BM25 score, price similarity score, and category similarity score are computed and standardized. Predefined weights are applied to obtain a final score, and items are ranked accordingly. The candidate list based on the CF is then merged with the score-based results. When the product appears in both lists, the CF recommendation is prioritized. If necessary, recommendation diversity is enhanced by retrieving similar products through the search engine or by selecting complementary items from different categories using CF. This composite strategy leverages the strengths of each component to deliver personalized recommendations that reflect both user interests and purchasing probability.

Through this multi-factor integration, the proposed method combines user behavior (CF) with textual and price-based features (BM25, price similarity, category similarity), enabling the generation of more accurate personalized recommendations for large-scale e-commerce platforms.

Figure 3 illustrates the A/B testing strategy and evaluation procedure used to validate the system's performance.

The control group, composed of popular products updated every four hours, represents items promoted through various campaigns, discounts, and high-visibility placements on the e-commerce platform.

- A/B Test design
    - Subject assignment: Users are randomly selected from the overall user base.
        - Experimental Group 1 (Hybrid): Receives hybrid recommendations integrating CF, BM25, and price similarity.
        - Experimental Group 2 (CF-only): Receives recommendations based solely on CF.
        - Control Group (popular products): Receives a list of popular products refreshed every four hours, reflecting the existing operational approach.

- Experiment period and data collection
  - The experiment runs for approximately 8–16 days, during which recommendation outputs and user interactions are recorded.
  - An adequate sample size (in terms of sessions and users) is ensured to support statistically valid analysis.
- Evaluation procedure
  - Recommendation logging: Each session's recommendations and their ranking positions (e.g., Top-N) are stored in logs.
  - Click and purchase tracking: User clicks and completed purchases are tracked at the session and item levels.
  - Key Metrics:
    - CTR: Ratio of clicks to impressions.
    - Purchases: Number of transactions completed during a test period.
    - Revenue: Calculated from total payment amounts.
  - Statistical testing: CTR, purchase counts, and revenue are compared across groups to assess whether differences are statistically significant. Performance is evaluated to determine whether the hybrid method outperforms the CF-only and popular-products baselines.

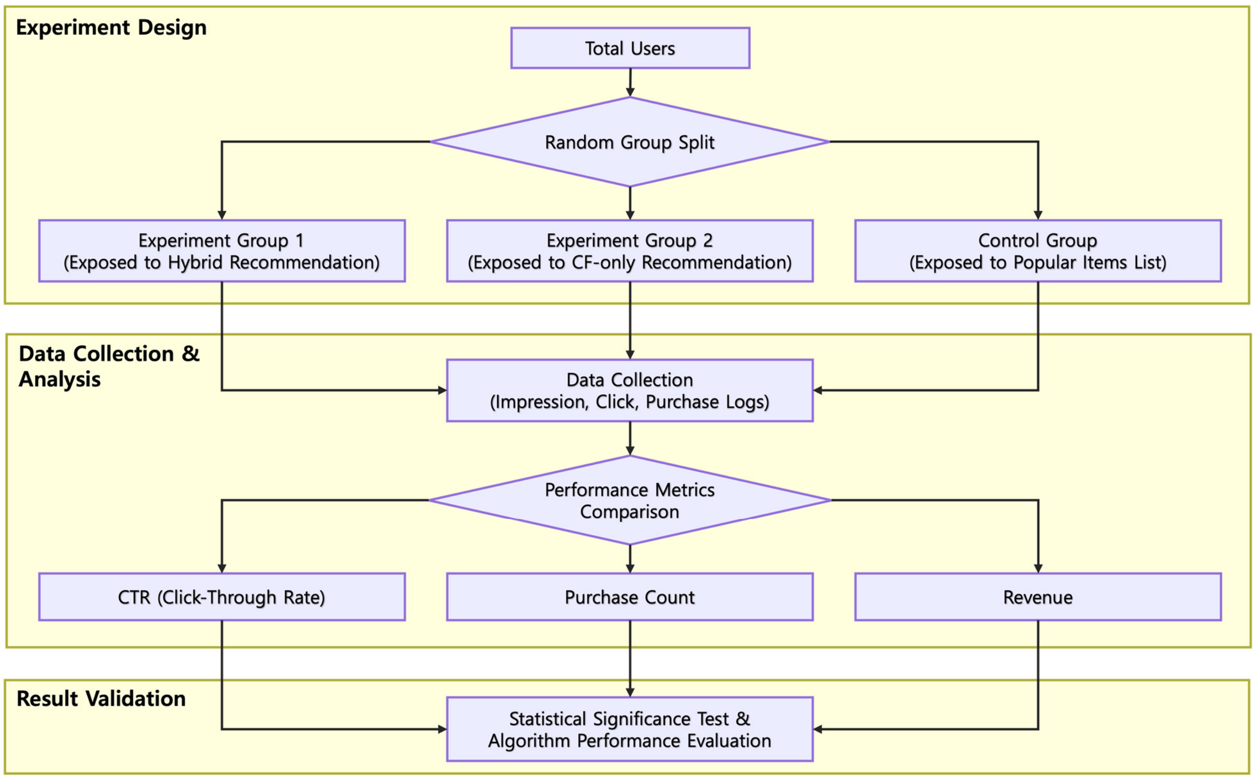

Experiment Design
Total Users
Random Group Split
Experiment Group 1 (Exposed to Hybrid Recommendation)
Experiment Group 2 (Exposed to CF-only Recommendation)
Control Group (Exposed to Popular Items List)
Data Collection & Analysis
Data Collection (Impression, Click, Purchase Logs)
Performance Metrics Comparison
CTR (Click-Through Rate)
Purchase Count
Revenue
Result Validation
Statistical Significance Test & Algorithm Performance Evaluation

Figure 3. A/B Test design diagram.

This A/B testing framework, a widely-used methodology for objective evaluation in large-scale online services [43], provides an objective basis among recommendation algorithms. It validates their impact in a real-world commercial environment, including improvements in CTR, purchase volume, and revenue.

3.3. Designing the Architecture for a Large-Scale Hybrid Recommendation System

This study demonstrates that incorporating techniques such as price similarity functions, zero-downtime index updates, and distributed caching can simultaneously improve search accuracy and system availability. Additionally, by systematically applying large-scale traffic management strategies, such as scale-out mechanisms and hash optimization, the system achieves high recommendation accuracy and scalability in real-time environments.

The proposed architecture, as illustrated in Figure 4, is designed to efficiently process large volumes of user events, including product views, add-to-cart actions, and purchases, while enabling advanced recommendation services. It consists of three primary layers: the Data collection layer, the analysis and learning layer, and the real-time recommendation layer. By clearly defining the roles and interactions of these layers, the architecture ensures both scalability and maintainability.

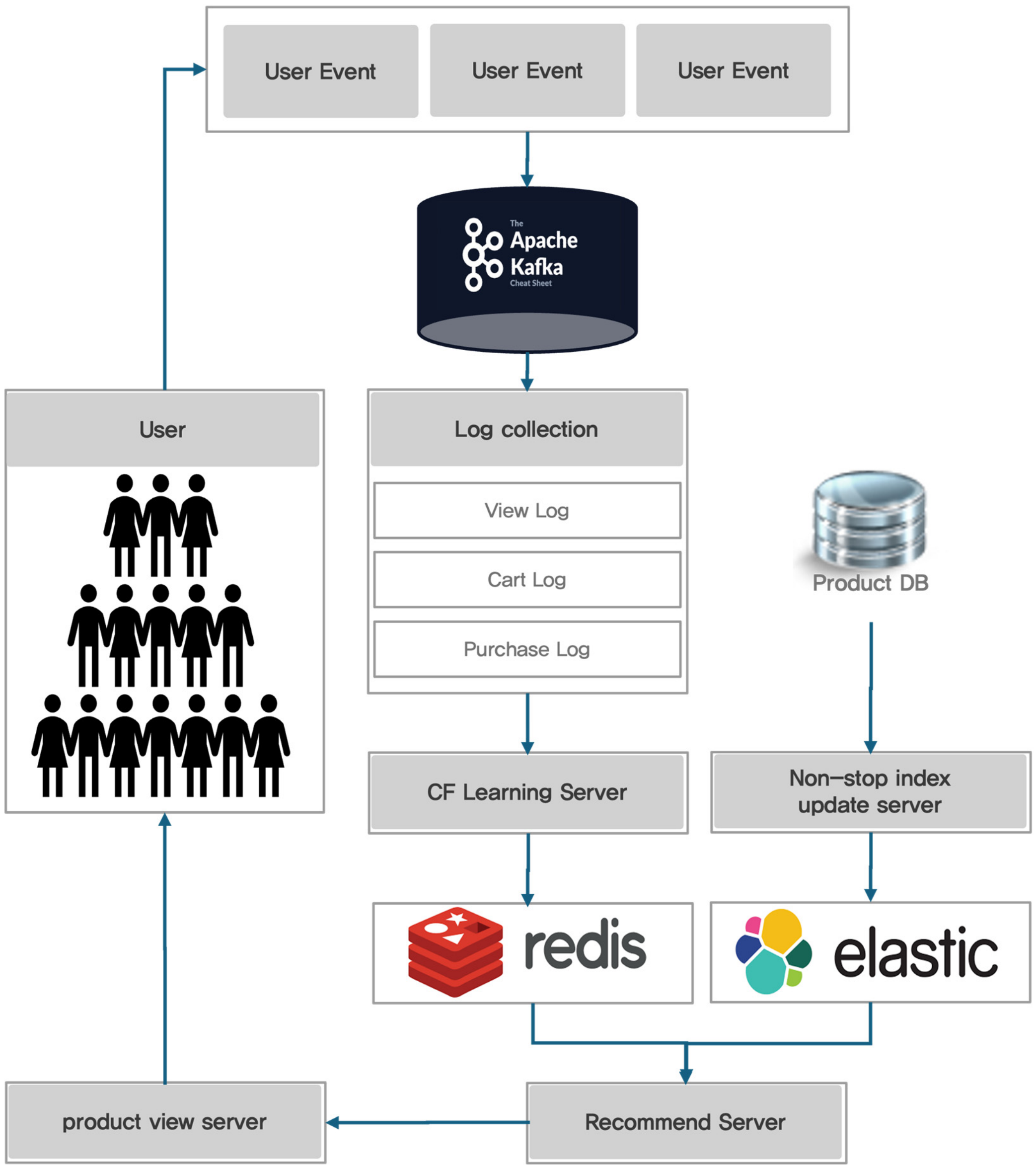


Figure 4. System architecture. The arrows indicate the flow of data and requests between the components.

The proposed system is structured into three layers that collectively support scalable, accurate, and real-time recommendation services.

Data collection layer. This layer employs a message queue or streaming platform (e.g., Kafka) to process large-scale events in a distributed environment. User interaction logs, including product views, cart additions, and purchases, are captured and stored in real-time, ensuring accurate and reliable data for subsequent analysis.

Analysis and learning layer. This layer performs offline or batch-based machine learning and text indexing tasks. Its core functions include relationship modeling through CF, text matching with BM25, and the development of price-range classification models. This process generates candidate recommendations from multiple perspectives. High-performance training servers, combined with a search engine (Elasticsearch), enable efficient large-scale data processing and indexing.

Real-time recommendation layer. When requests are received from web or mobile applications, this layer computes a hybrid score by integrating pre-computed results from the analysis and learning layer with cache server outputs (Redis) and search engine results. The system then immediately returns the top-N recommendations. REST APIs and distributed caching ensure ultra-low latency (ULL), enabling real-time responsiveness and high user satisfaction.

The rationale for this three-layer architecture is the separation of concerns, which facilitates both scalability and maintainability. Each layer can be upgraded and deployed independently, ensuring system flexibility.

Technology stack for large-scale traffic handling.

1. Distributed caching. To accelerate retrieval, similarity scores, particularly cosine similarity, are pre-computed and stored in a multi-node cache system (e.g., Redis). During a request, the cached similarity values are directly integrated with BM25 search results, minimizing additional computation. This considerably reduces response time by offloading the most resource-intensive operations. Because the cache operates in a distributed structure, workload surges on individual nodes can be balanced across the cluster, ensuring both stability and scalability.
2. Horizontal scaling. The architecture is designed for scale-out deployment, where the search engine, cache servers, and analysis servers (responsible for statistical processing and model computation) are deployed as independent nodes or containers. Load balancing ensures that traffic spikes are distributed across the cluster, preventing bottlenecks. Since nodes can be dynamically added or removed, the system can flexibly adapt to unpredictable traffic patterns while balancing operational costs and performance.
3. Memory optimization through hashing. To improve memory efficiency, hashing techniques such as CRC32 and simplified similarity tables are applied. For example, in CF computations, hash-based compression reduces the memory required for storing similarity scores and indexing tables while maintaining recommendation accuracy. This approach is particularly essential for large-scale user-item datasets, where memory access costs directly impact processing speed. By minimizing collisions and enabling rapid lookups, the hashing strategy ensures stable system performance, even in the face of exponential data growth. By appropriately integrating these optimization techniques, data operations in a distributed environment can be managed with flexibility, while system performance remains stable even as the data volume grows exponentially.

For the implementation of the proposed three-layer architecture using a technology stack, this study selects, configures, and applies tools for database, management, search engines, distributed caching, and data analysis pipelines.

- Database
  - Distributed RDB: A relational database distributed across multiple nodes is employed to ensure reliable large-scale transaction processing.
  - NoSQL (MongoDB): MongoDB is used as a complementary solution for storing semi-structured and unstructured data and for supporting high-speed read/write operations.
- Search engine
  - Elasticsearch: Responsible for searching text fields such as product names, categories, and descriptions, and for computing BM25 scores. It is particularly suitable for large-scale document processing and real-time tasks.
- Distributed cache
  - Redis: Enhances query speed by caching frequently accessed metadata (e.g., session information, recommendation results). Scalability and fault tolerance are achieved through data partitioning and replication.
- Analysis tools and pipeline
  - Java-based statistical analysis: In a Java environment for large-scale processing, logs are aggregated and analyzed to support model refinement. Large-scale traffic monitoring is conducted using Scouter.
  - Log processing pipeline: Collects and stores event-based logs in both real-time and batch modes, supporting the computation and visualization of statistical metrics.

By deploying this environment, large-scale traffic and data processing can be managed efficiently, while recommendation performance can be continuously enhanced through rapid search and analysis.

3.4. Building a Hybrid Recommendation System Based on CF and a Search Engine

This study proposes a recommendation module that delivers personalized product suggestions by applying CF. The CF module is built on a data structure consisting of (User ID, product ID, event type, timestamp) and generates more accurate results by assigning weights to different event types. Importantly, the system is designed for low-cost similarity computation without reliance on GPUs, achieved by reducing memory usage through CRC32-based hashing of IDs for both user differentiation and cosine similarity calculations.

Additionally, recommendation accuracy is improved by applying higher weights to products within the same category.

The implementation process, as outlined in the flowchart in Figure 5, consists of the following steps:

1. Preprocessing and threshold setting. To maximize the efficiency of processing large-scale user-product interaction logs, a preprocessing step was conducted in the offline stage to retain only data that satisfied specific criteria. The distributions of products and users were first analyzed, and thresholds were established to exclude cases with extremely low or disproportionately high view frequencies. In addition, thresholds were dynamically adjusted using metrics that varied according to the system's application environment (e.g., user-to-product ratios). This approach prevented an excessive number of candidate pairs and improved computational efficiency.
2. Co-occurrence extraction. From the filtered interaction history, only products that met the criteria were selected, and the co-occurrence count between product pairs was calculated based on the set of users who had viewed each product. Specifically, the method tracked the frequency with which users who viewed one product also viewed

another. This co-occurrence information was accumulated in a parallel and distributed manner, ensuring efficient computation for subsequent similarity estimation.

3. Similarity calculation (cosine similarity) Using the extracted co-occurrence data, cosine similarity values were computed by combining the co-occurrence counts with exposure frequencies (e.g., the number of users who viewed the clicked each product). Noise was reduced by applying the standard cosine normalization, placing the co-occurrence count in the numerator and the square root of the product of the individual view count counts in the denominator, and excluding cases below a defined threshold. The resulting similarity values provided a quantitative measure of correlations among products.
4. Category-based correction. Product classification information was then incorporated to refine the similarity estimates. Additional weights were applied to product pairs within the same or similar categories, allowing the system to prioritize candidates with stronger domain-relevant similarity rather than relying solely on view-based measures. This adjustment integrated domain knowledge into recommendation process and captured category-specific characteristics more accurately.
5. Recommendation list composition and sorting. For the extracted product pairs, a recommendation list was generated by ranking them in descending order of similarity. The products were then grouped by category or closely related categories, facilitating use in specific scenarios (e.g., shopping cart recommendations or set configurations). Items that did not meet the minimum occurrence threshold were filtered out to improve the reliability of the final list.
6. Result storage and merging. The final recommendation candidates were stored in fast-access storage system (e.g., caching servers or external file storage) to enable real-time retrieval. These results were later combined with other offline analyses, such as user purchase history, and aggregated into a format immediately usable at critical interaction points (e.g., when a user opens a product detail page). This strategy reduces response time in high traffic online environments and supports the implementation of a multi-factor recommendation system that integrates multiple metrics.
7. Expansion and optimization strategy. The algorithm can adapt to varying scales of user traffic and product diversity by dynamically adjusting parameters such as occurrence thresholds and category weights (Appendix A). In large-scale environments, caching and streaming techniques distribute the computational load and accelerate processing. Moreover, incorporating additional metadata, such as brand or price range, enables the system to deliver more refined recommendation functions beyond basic category-based weighting.

Through this process, the collaborative filtering module delivers accurate, efficient, and low-cost recommendations by applying event-type weights and category-based weighting.

BM25 (Best Match 25) is a probabilistic retrieval model that extends the traditional Term Frequency–Inverse Document Frequency (TF-IDF) approach. It calculates document-query similarity by considering term frequency (TF), inverse document frequency (IDF), and document-length normalization parameters (e.g., as k1 and b). This approach downweights rare terms, and reduces the bias toward excessively long documents. As a result, BM25 is widely adopted as a core ranking function in applications such as e-commerce search engines and news retrieval, where it effectively captures fine-grained relevance between user queries and documents.

As summarized in Table 4, three specialized dictionaries were developed to enhance product search accuracy: a stopword dictionary, a weighted-word dictionary, and a noun dictionary. The stopword dictionary contained terms unrelated to product attributes, which were excluded from queries to reduce noise and improve retrieval precision. The weighted-word dictionary emphasized key descriptors relevant in shopping contexts, such as age or gender, which were extracted through morphological analysis and assigned higher weights in scoring search queries. Additionally, a noun dictionary was constructed to enhance the accuracy of morphological analysis, particularly in extracting and recognizing compound nouns from lengthy queries. Together, these dictionaries enable more sophisticated and precise product search results in an Elasticsearch-based system utilizing the BM25 algorithm.

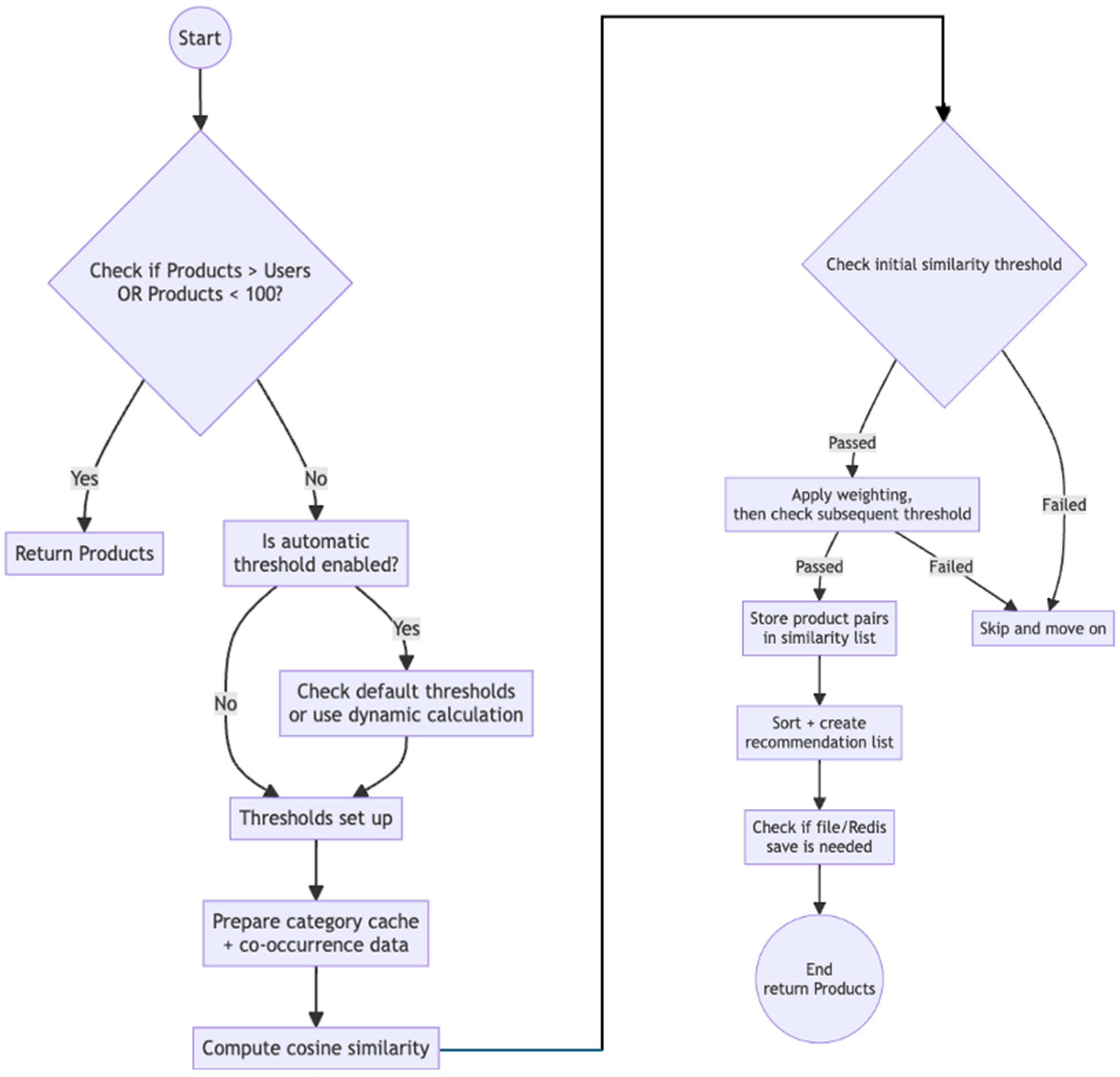

Start
Check if Products > Users OR Products < 100?
Yes
No
Return Products
Is automatic threshold enabled?
Yes
No
Check default thresholds or use dynamic calculation
Thresholds set up
Prepare category cache + co-occurrence data
Compute cosine similarity
Check initial similarity threshold
Passed
Failed
Apply weighting, then check subsequent threshold
Passed
Failed
Store product pairs in similarity list
Skip and move on
Sort + create recommendation list
Check if file/Redis save is needed
End return Products

Figure 5. CF learning flow chart.

Table 4. Word dictionary.

Dictionary Type	Word List
Stopword dictionary	control, reheating, stop button, at time of purchase, when adding, bestseller, free gift random giveaway, series, multiple, choice product, special price event, multi-purpose, rush of orders, owner's recommendation, limited quantity, discount, coupon, official importer, option, single item sale, free shipping, same-day shipping, time-sensitive, points, or more, choose 1, way-drawer, launch, super high quality, of material, BEST, recommendation, launching, collection, coming soon, Namyang, more than X sheets, same-day shipping, fashion beauty week, free shipping, time-sensitive, self-produced, special price in progress, store price, domestic invention patent, product, direct shipping, use, additional, limited, limited quantity, discount, new launching, points, coupon discount, wonder coupon, mat excluded, official importer, option, single item sale, additional choice, included, free round-trip shipping, round-trip shipping included, free round-trip shipping, available, product sale, partner card
Weighted word dictionary	men, women, male, female, girl, boy, child, infant, for infants, kids, stage 1–4, jacket, pants, belt, hat, gloves, ball cap, cap, sun cap, heavy down, golf shoes, scrub holder, extended padding, loafer, sneakers, bloafer, knit cap, cardigan, bikini, shirt, jumpsuit, v-neck top, lace, culottes, skirt, cooler bag, colored pencil, swimsuit, card, stiletto heels, socks, headset, clutch, dinosaur, glue stick, sandals, note, toothpaste, wet wipes, beauty tissues, tteokbokki, shorts, string pants, liner socks, slippers, long one-piece dress, turtleneck, culottes, spikes, fleece-lined, neck cushion, newborn baby clothes, trash can, long skirt, gauze, airway maintainer, Velpeau band, cloth plaster, non-sterile gauze, sterile gauze, roll gauze, absorbent cotton, cotton bandage, cotton ball, compressed cotton, InBody, height scale, cast, mugwort moxibustion, heating mat, electric mat, hot water mat, rhinitis, nasal wash, moxibustion tool, mini moxibustion, fire cupping set, cupping, cupping device, moxibustion device, bathroom slippers, Black Friday, hot pack, neck traction, turtle neck traction device, disc care, vision test chart, toothbrush, bath thermometer, digital thermometer, colored jeans, sweater, padded jumper, suspenders, apron
Noun Dictionary	tight, female, male, spring, summer, autumn, winter, timing, leg, value, British, stripe, girl, boy, pants, button one-piece dress, halter one-piece dress, bikini, girl, short sleeve, melody, halter, culottes pants, Kanuda, I am Mother, fresh, arugula, for in-flight, sweetim, dream, bear, beanie, Pororo, heavy equipment, basketball, basketball (ball), desk, chair, set, complete set, random color shipping, pang-pang, C'est Vsi Bon, houndstooth, Novalac, pillow stuffing, Superga, bed, captain, u-line, sweet pumpkin, commercial, living room cabinet, cross-body bag, Unika, smart, hanger, Saekom-Dalkom, lace, point, one-piece dress, simple, ring, desk, clay bed, Harim, rabbit, Bioderma, clothing, slingback, solid wood, baby high chair, dried fruit, coffee, waterproof, name sticker, party, card, flower, materials, French Cafe, Cafe Mix, Coffee Mix, Chamgreen, one-touch, kimchi refrigerator, drawer container, stainless steel, freezer compartment, refrigerator compartment, flatty, flat, tall small, basic set, tongs, funnel, storage container, no-collar, wine, red wine, white wine, color scheme, one-button, jacket, zipper, fine dust, yellow dust, mask, navigation, handmade

The zero-downtime index update feature implemented in this study is a core module that enables switching the search index to a new version without service interruption, consisting of six main stages, illustrated in Figure 6.

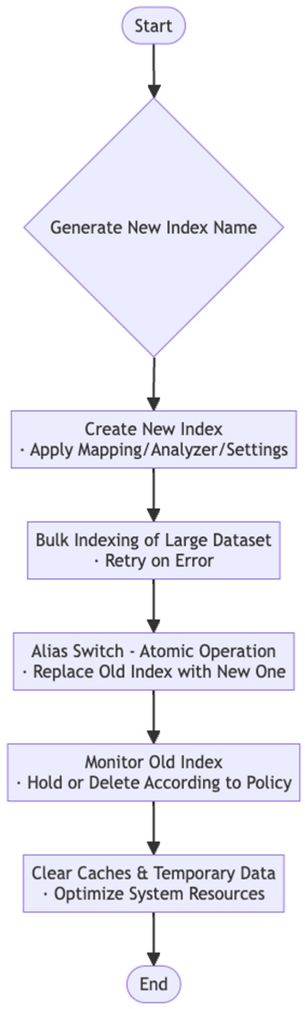

Start
Generate New Index Name
Create New Index
· Apply Mapping/Analyzer/Settings
Bulk Indexing of Large Dataset
· Retry on Error
Alias Switch - Atomic Operation
· Replace Old Index with New One
Monitor Old Index
· Hold or Delete According to Policy
Clear Caches & Temporary Data
· Optimize System Resources
End

Figure 6. Non-stop index update flow chart. The returning arrow indicates the iterative process of similarity calculation and threshold checking applied to each product pair.

In the first stage, a new index name is dynamically generated in accordance with the update schedule. This name incorporates both date and time information, ensuring precise identification of the moment the index was created and its version. Such differentiation prevents conflicts with existing indices and enhances long-term traceability.

In the second stage, a new index is created in the Elasticsearch cluster by applying predefined mappings, morphological analyzers, and shard/replica configurations. This process runs independently of the existing index, thereby avoiding service disruption. Before deployment, the index structure is thoroughly validated to guarantee indexing quality.

In the third stage, large-scale data insertion is executed on the new index. Data are ingested in batches using the Elasticsearch's standard Bulk API for efficient batch processing, which maximizes efficiency and minimizes communication overhead. For very large datasets, processing speed is further optimized through parallelization. A recovery mechanism is also incorporated to detect and retry failed processing units, thereby improving fault tolerance.

In the fourth stage, once all data have been successfully indexed, the alias is atomically switched to the new index to preserve uninterrupted search availability. This is an accomplished transaction using Elasticsearch's alias switch API. As a result, users continue querying the old index during the transition, while search results are immediately served from the new index after the switch.

In the fifth stage, the previous index is temporarily retained to monitor system stability, advertisement delivery, and search logic. A rollback policy is established to enable a rapid reversion in the event of unexpected errors or quality degradation. After the retention period, the old index is safely deleted to reclaim system resources.

In the sixth and final stage, post-processing is conducted to clear caches and temporary data. Logs and metric data are collected for future optimization, and accumulated caches within Elasticsearch are reset to ensure consistent system performance.

This mechanism ensures high availability while seamlessly replacing the index with updated data, even in high-traffic e-commerce environments. The use of aliases enables index replacement through a single atomic request, thereby minimizing downtime and allowing for a clear separation between the old and new indices. Consequently, this staged approach shortens the indexing cycle while sustaining search accuracy, quality, and usability in real-time (or near-real-time) systems.

3.5. Applying Price-Similarity Weighting in the Hybrid Recommendation Engine

A custom algorithm was implemented as a function script to integrate price similarity with the BM25 scoring model for product searches in Elasticsearch. Specifically, for products priced higher than the currently selected item, similarity was adjusted using the negative logarithm of the price difference. The same negative logarithmic adjustment was applied when comparing with lower-priced items.

A distinctive feature of the algorithm is the assignment of additional weight to higher-priced products. This weighting was strategically designed to guide users toward premium options within comparable price ranges, based on the hypothesis that higher prices often correlate with better quality. The overall goal is to increase purchase conversion rates and sales by promoting higher-value items while still aligning with user preferences. This relationship between price difference and product similarity is illustrated in Figure 7.

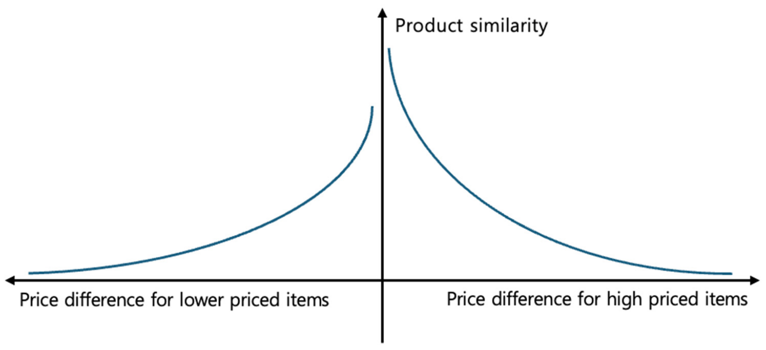


Figure 7. Graph of price similarity.

To enhance both search accuracy and user-oriented recommendations, additional weight is assigned to the BM25 score for products with similar price ranges.

Equation (2) applies when the searched product is more expensive than the product previously clicked by the user. This equation quantifies the difference between the current product price and the reference price, and integrates this difference into the ranking. A logarithmic transformation is applied to the price gap to prevent disproportionately large penalties when differences are extreme. The results are normalized by dividing them by the product's actual price, ensuring that the relative ratio of the price gap is considered. Finally, the value is multiplied by a negative weight of -2, so that higher-priced products receive a stronger penalty, thereby lowering their final ranking scores. This mechanism reduces the likelihood of products priced above the user's preferred range appearing at the top of the search results.

$$Rank = BM25\ score + \frac{Log(product\ price -\ clicked\ product\ price + 1)}{Product\ price} \times -2 \tag{2}$$

Equation (3) applies when the searched product is less expensive than the clicked product. Like Equation (2), it penalizes based on price difference, but with two key distinctions. First, the denominator used for normalization is the price of the clicked product rather than the product's own price. This adjustment reflects the user's price sensitivity. Second, the penalty weight is smaller (−1.2 compared to −2), indicating a design choice to penalize cheaper products less severely than more expensive ones. In other words, while the system restricts exposure of products priced considerably below the user's reference point, the penalty is intentionally weaker than that applied to higher-priced products.

$$Rank = BM25\ score + \frac{Log(clicked\ product\ price -\ product\ price + 1)}{clicked\ product\ price} \times -1.2 \tag{3}$$

The Elasticsearch function script, implemented as Algorithm 1, operationalizes Equations (2) and (3) to incorporate price similarity into the ranking process. The script dynamically adjusts scores based on whether the document's product price exceeds or falls below the reference price. In both cases, the baseline BM25 score is boosted by a factor of 1.3, and an additional adjustment is applied using the logarithmic transformation of the price difference. To ensure numerical stability, ±1 is added before the logarithm, and the denominator is set to the corresponding price value.

Differential weighting is then applied: −2 when the product is more expensive, and −1.2 when it is less expensive or equal. As a result, the ranking algorithm prioritizes products that align more closely with the user's preferred price range, thereby improving both search quality and recommendation effectiveness. where 'Log' denotes the natural logarithm (ln).

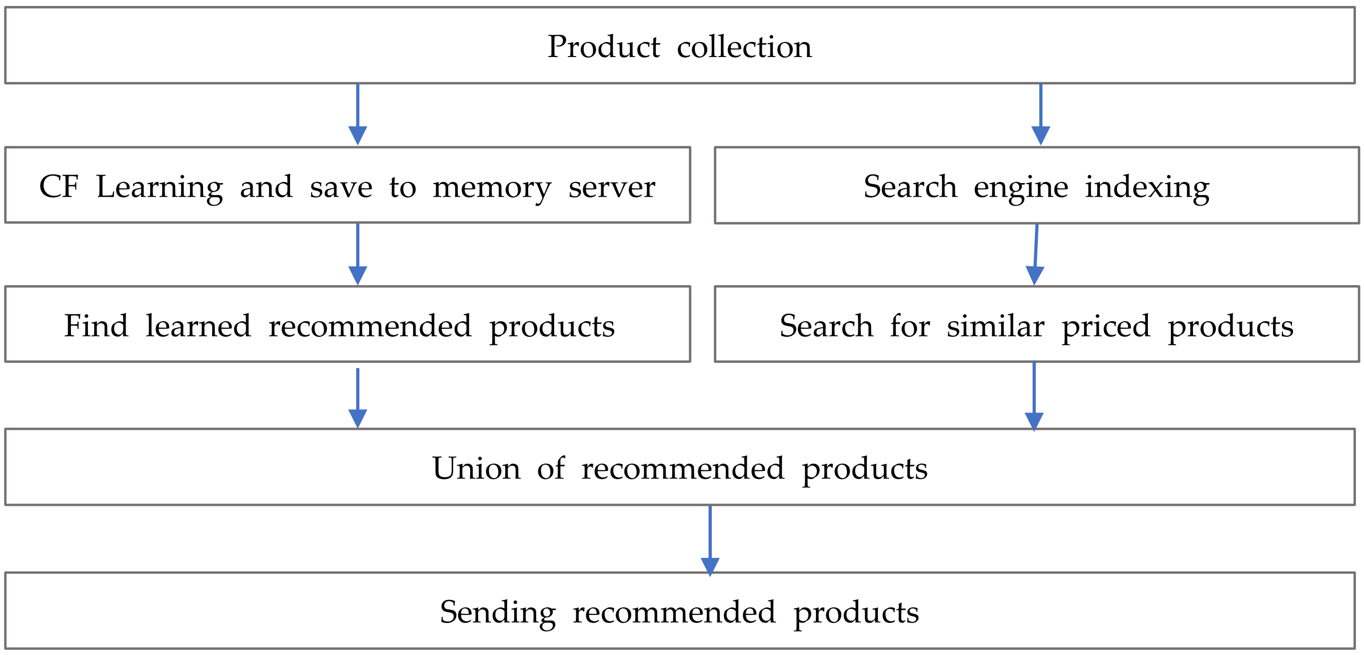
Algorithm 1. Price-adjusted scoring function

```
Input: document price, product price
Output: Adjusted score reflecting the document's price relative to the product price

1.  for each document price in the document set:
2.      If the document price is greater than the product price:
3.          Compute the price difference: document price—product price.
4.          Increment the difference by 1.
5.          Compute the relative price: difference/document price.
6.          Take the natural logarithm: Log (relative price)
7.          Compute the adjustment factor: Log relative price * −2.
8.          Update the adjusted score: (_score) + price adjustment
9.          Return the adjusted score.
10.     Else if the document price is less than the product price:
11.         Calculate the price difference: product price—document price
12.         Increment the difference by
13.         Compute the relative price: price difference/product price
14.         Take the natural logarithm: Log (relative price).
15.         Compute the adjustment factor: Log relative price * −1.2.
16.         Updated the adjusted score: (_score) + price adjustment.
17.         Return adjusted score.
18.     end if
19. end for
```

Note: _score denotes the document's initial score before price adjustment. The logarithmic transformation ensures that the influence diminishes as the price difference increases.

The hybrid algorithm incorporates price-adjusted weighting through the steps illustrated in Figure 8. First, it initializes a storage space for recommendation results and a data structure to prevent duplicate entries. Next, for each base item, the algorithm retrieves related information, primarily from cache data or a database. At this stage, it generates two distinct lists: items within the same category and items from other categories. The retrieved items are then checked for duplication before being stored in the recommendation set.



Product collection
CF Learning and save to memory server
Search engine indexing
Find learned recommended products
Search for similar priced products
Union of recommended products
Sending recommended products

Figure 8. Weight optimization method.

After verifying that the total number of acquired items is sufficient, the search function is applied when the results do not meet a predefined threshold. The search is restricted to a limited scope (e.g., a single page), and any additional items retrieved are incorporated into the final dataset after duplicate verification. The extracted lists are then progressively integrated to match the output format. Once this process is repeated for all base items, the final, consolidated recommendation result is generated.

4. Experiments and Analysis

A comparative evaluation was conducted to assess the performance of the hybrid recommendation model against popular-item and CF-based methods. The analysis focuses on quantitative performance metrics, including CTR, purchase conversion rate, and revenue growth. Results demonstrate that the hybrid model achieved statistically significant improvements while maintaining rapid response times under high-traffic conditions.

4.1. Experimental Method

The experiments were conducted in two phases. The first comparison was between experimental group 1 (popular items) and experimental group 2 (CF-only). Group 1 recommended products based on category-level popularity over four-hour intervals, while Group 2 applied a CF model. Performance was measured using CTR, purchase count, and total revenue.

In the second phase, Group 1 (Popular Items) was compared with Group 2 (Hybrid model). The Hybrid system generated personalized recommendations by combining CF with BM25 scores and a price-similarity factor. As in the first phase, CTR, purchase count, and revenue were used to assess whether the hybrid model outperformed simple popularity-based recommendations.

Data preprocessing and sampling followed a structured procedure. The user group consisted of individuals who had previously viewed at least one product in the shopping mall, ensuring reliable response data. The recommendation target set encompassed the full product database of more than 80 million items, enabling evaluation across diverse categories and ensuring the model's capacity for general performance.

For CF, interaction logs, such as product views, cart additions, and ad purchases, were collected and transformed into a user-product mapping format. This allowed the model to learn user interests and preferences. Metadata (e.g., product names, categories, and prices) was also indexed to enable efficient product retrieval via a search engine.

Users were randomly assigned to groups, with each group assigned to a specific algorithm. This ensured independent datasets without overlap, minimized cross-group contamination, and strengthened the reliability of the comparative performance evaluation.

4.2. Test of CF-Only Recommendation Vs. Popular Items

Compared to the baseline method of recommending category-specific popular items every four hours, the experiment using CF without price weighting produced a higher CTR. Still, it resulted in a 5.1% decrease in decline over the two weeks. This outcome indicates that CF-based recommendations exposed items with relatively lower unit prices or weaker purchase conversion rates, suggesting that optimizing CTR alone does not guarantee revenue growth.

Among 80 million products, only about 200,000 received user interactions (e.g., views) on a daily basis. As detailed in Table 5, because CF alone cannot address the cold-start problem, the CTR rose modestly to 3.06% and the purchase rate to 1.98%, yet overall revenue decreased by 5.10%. Over time, revenue continued to fall with CF-only recommendations, as visualized in Figure 9. This decline can be attributed to user fatigue, as repeated exposure to a limited product set initially attracted clicks but reduced purchasing interest.

Table 5. A/B test comparing CF-only recommendations with popular-item recommendations.

Day	View A	View B	CTR Diff.	Purchase Diff.	Revenue
1	10,242,358	10,244,362	3.15%	−5.11%	15.88%
2	10,794,068	10,793,921	2.02%	−1.90%	−2.02%
3	10,865,519	10,858,848	3.64%	−3.44%	4.41%
4	11,105,431	11,106,800	3.24%	6.19%	20.32%
5	8,463,556	8,461,972	2.44%	0.85%	1.18%
6	8,680,147	8,685,480	2.81%	5.68%	34.88%
7	10,965,151	10,966,954	3.70%	2.14%	−8.39%
8	11,068,590	11,068,059	3.29%	−3.06%	−28.63%
9	11,226,542	11,225,323	2.39%	−4.41%	−23.36%
10	10,796,796	10,799,107	3.55%	3.49%	−15.06%
11	9,927,224	9,928,889	3.08%	8.47%	−5.55%
12	8,022,208	8,016,256	2.90%	5.04%	−13.52%
13	8,106,215	8,108,953	3.80%	9.02%	−5.56%
14	10,497,534	10,495,112	2.86%	13.52%	−12.09%
Total	24,378,054	24,381,210	3.06%	1.98%	−5.10%

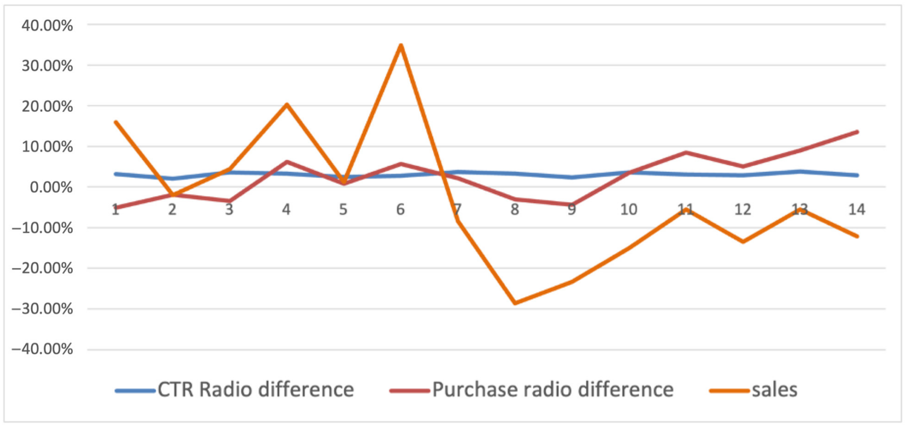


Figure 9. Daily performance comparison between popular product and CF-only recommendations over the 14-day A/B test period.

4.3. Test of the Hybrid Recommendation Method Versus Popular Items

An A/B test was conducted to compare a traditional popular-item-based recommendation system (control group) with a hybrid method that integrates CF and price similarity (experimental group). The objective was to evaluate improvements in user engagement, measured by CTR, and revenue performance. Results indicated that the hybrid approach achieved a 2.55% increase in CTR and a 5.55% increase in revenue compared with the control group (Table 6), with daily performance trends detailed in Figure 10. These findings suggest that incorporating price weighting into CF is more effective in promoting purchase conversions than conventional popularity-driven recommendations.

Table 6. A/B test results: hybrid recommendation method versus popular products.

Day	View A	View B	CTR Diff.	Purchase Diff.	Revenue
1	10,706,774	10,709,954	0.17%	5.70%	78.92%
2	8,611,818	8,620,872	6.58%	3.46%	2.98%
3	10,788,443	10,781,719	5.56%	−1.24%	−21.40%
4	10,760,489	10,763,836	−1.11%	3.38%	−5.20%
5	11,036,644	11,032,724	2.79%	−2.22%	−2.98%
6	8,868,809	8,876,145	1.01%	−9.32%	−5.37%
7	9,483,432	9,479,642	2.97%	−11.01%	23.77%
8	10,383,113	10,392,445	2.00%	−4.63%	−1.11%
9	10,996,398	10,993,813	2.90%	−6.48%	11.13%
10	10,768,384	10,768,389	4.10%	4.46%	5.31%
11	10,498,714	10,499,502	3.44%	−7.77%	−1.07%
12	9,387,142	9,394,486	4.18%	8.46%	−2.49%
13	7,051,044	7,051,973	1.91%	15.20%	65.74%
14	7,714,353	7,707,807	1.52%	−3.99%	2.73%
15	9,694,423	9,690,520	3.33%	−1.88%	−8.62%
16	9,074,197	9,075,797	1.55%	4.59%	5.38%
17	9,987,808	9,992,299	3.16%	−5.65%	−13.06%
18	8,551,118	8,549,539	0.86%	−3.31%	21.13%
19	10,204,142	10,205,193	1.62%	5.05%	7.60%
20	8,212,553	8,204,835	1.88%	−6.26%	−3.48%
21	8,451,097	8,454,349	2.02%	−3.05%	−13.06%
22	10,261,191	10,257,224	2.35%	0.52%	−11.75%
23	10,242,358	10,244,362	3.15%	−5.11%	15.88%
24	10,794,068	10,793,921	2.02%	−1.90%	−2.02%
25	10,865,519	10,858,848	3.64%	−3.44%	4.41%
26	11,105,431	11,106,800	3.24%	6.19%	20.32%
27	8,463,556	8,461,972	2.44%	0.85%	1.18%
28	8,680,147	8,685,480	2.81%	5.68%	34.88%
Total	271,643,165	271,654,446	2.55%	−0.53%	5.55%

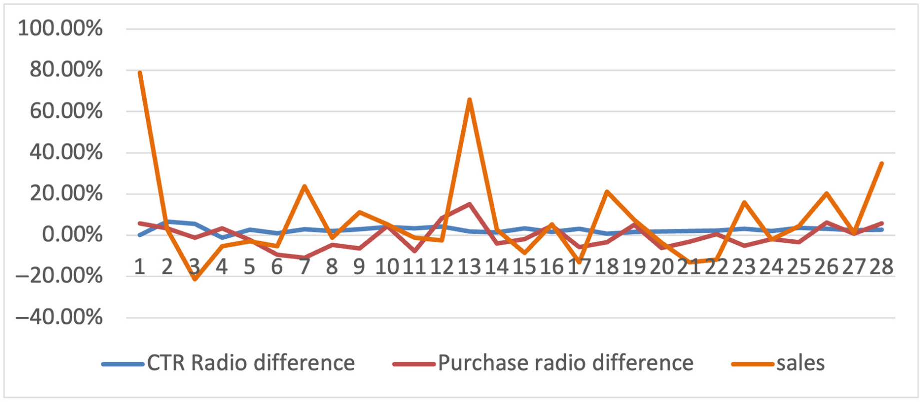


Figure 10. Daily performance comparison between popular product and hybrid recommendations over the 28-day A/B test period.

In contrast to the CF-only recommendation method, which previously exhibited declining revenue over time, the hybrid approach consistently demonstrated more stable and effective performance in enhancing both purchase conversions and overall revenue relative to popular-item-based recommendations.

As presented in Table 7, an A/B test was conducted during the same period, focusing only on product categories that showed sales growth attributable to the recommendation system. For these categories, which represented 56% of total exposure, sales increased by 13.3%. Figure 11 shows the daily fluctuations in CTR difference, purchase difference, and revenue for these specific categories.

Table 7. A/B test results for categories with improved sales performance.

Day	View A	View B	CTR Diff.	Purchase Diff.	Revenue
1	6,120,323	6,127,115	0.64%	9.05%	136.00%
2	4,703,805	4,712,818	11.01%	−2.57%	−7.67%
3	6,214,008	6,206,601	8.33%	1.16%	−28.20%
4	6,055,943	6,057,083	−2.15%	10.06%	−4.67%
5	6,290,783	6,290,295	5.06%	−1.27%	−6.17%
6	4,853,853	4,860,400	2.54%	−5.30%	13.58%
7	5,251,918	5,251,540	4.12%	−4.79%	22.13%
8	5,829,956	5,832,374	3.64%	−7.10%	−5.24%
9	6,187,362	6,187,827	5.01%	−11.55%	14.21%
10	6,161,072	6,162,085	7.00%	7.96%	14.86%
11	5,891,823	5,892,350	5.53%	−8.66%	7.05%
12	5,412,457	5,416,814	6.86%	17.55%	−3.65%
13	3,966,932	3,966,488	2.43%	7.43%	92.17%
14	4,327,500	4,324,016	2.46%	1.84%	24.00%
15	5,480,579	5,478,136	4.42%	0.35%	−8.72%
16	5,074,602	5,069,520	3.18%	−0.55%	15.54%
17	5,595,537	5,598,759	5.66%	−2.36%	−5.04%
18	4,689,294	4,688,116	2.06%	−8.31%	53.31%
19	5,737,617	5,735,085	3.22%	3.70%	3.46%
20	4,515,367	4,510,117	2.25%	−5.78%	0.27%
21	4,717,987	4,722,724	3.59%	−1.19%	−21.30%
22	5,947,914	5,945,721	1.86%	12.80%	−14.02%
23	5,882,857	5,886,064	4.81%	5.37%	42.67%
24	6,184,598	6,182,748	3.69%	1.82%	14.03%
25	6,228,829	6,225,887	5.59%	0.85%	18.87%
26	6,407,521	6,408,274	4.99%	17.25%	33.71%
27	4,696,162	4,697,371	4.35%	0.29%	−1.97%
28	4,792,173	4,797,007	5.02%	25.00%	83.71%
Total	153,218,772	153,233,335	4.13%	2.20%	13.30%

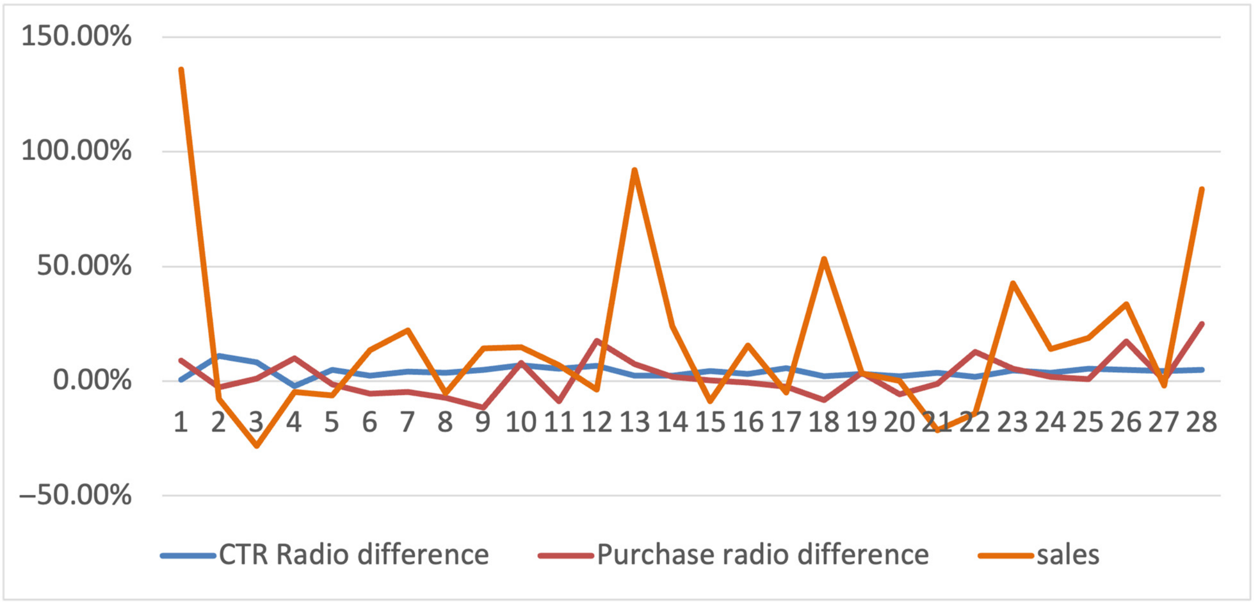


Figure 11. Daily performance comparison for product categories with sales growth, comparing popular product and hybrid recommendations.

When the revenue contribution of the effective product categories was evaluated against the baseline sales of Group A in the A/B test, an overall sales increase of approximately 7.99% was observed as presented in Figure 12. This finding suggests that implementing the hybrid recommendation system exclusively for effective product categories, which account for 56% of total exposures, can drive more efficient revenue growth.

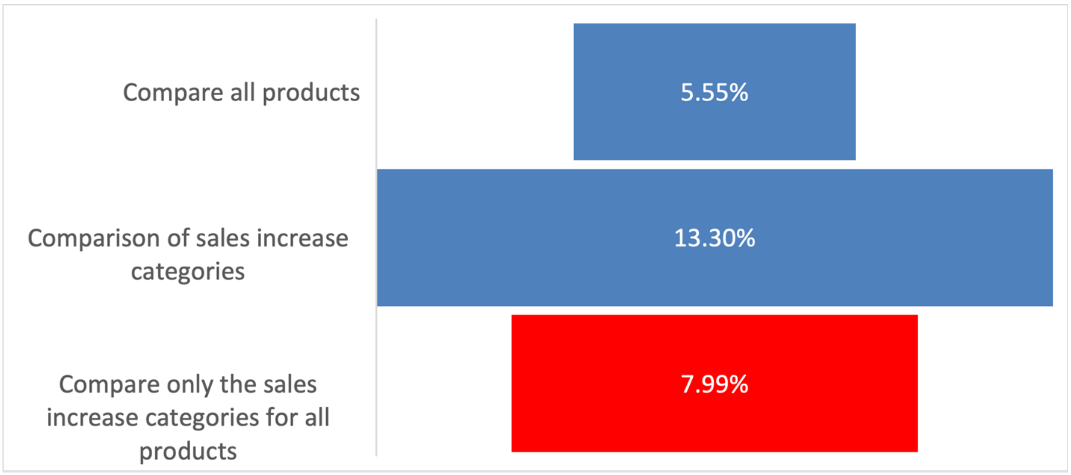


Figure 12. Summary of overall revenue increase. The chart compares the total revenue lift for all products, the lift within sales-increase categories only, and the overall contribution of those categories to the total revenue lift (highlighted in red).

5. Discussion

The proposed hybrid recommendation system was empirically validated through A/B testing, demonstrating its effectiveness in a large-scale e-commerce environment. The experimental results indicated statistically significant improvements across key performance indicators, including CTR, purchase conversion, and total revenue. This discussion provides an in-depth interpretation of these results and examines their theoretical and practical implications in the context of existing literature.

1. Effectiveness of Price-similarity recommendations. Prioritizing items within a user's preferred price range led to notable gains in CTR and purchase conversion. These results highlight that within the same product category, price sensitivity significantly influences purchasing behavior. Thus, incorporating personalized price-range filtering effectively enhances user engagement and increases the likelihood of purchase.
2. Improvement in recommendation diversity. The proposed method reduced exposure bias toward popular items, increasing visibility for new and less popular products. This outcome demonstrates that the hybrid approach successfully fosters diversity in product recommendations, thereby broadening user options and reducing algorithmic homogenization.

To understand the underlying causes of this success, it is essential to analyze the contrasting outcomes of the CF-only and hybrid models. The CF-only model, while successful at predicting user interest and increasing CTR by 3.06%, failed to account for the practical constraint of price, which paradoxically resulted in a 5.10% decrease in total revenue. This suggests a clear gap between generating user interest and inducing actual purchase behavior. The success of the hybrid model stems from its ability to effectively bridge this gap by integrating the user's 'purchasing power' through the price similarity factor. The system decisively reduced 'purchase friction' by prioritizing 'viable' options, a conclusion supported by the concurrent growth in both CTR (2.55%) and revenue (5.55%).

These findings offer a pragmatic perspective on recent research trends dominated by complex deep learning models such as GNNs and LLMs [9,12]. The proposed hybrid approach demonstrates that, rather than replacing these complex models, significant business outcomes can be achieved by integrating an interpretable and commercially critical factor like 'price' into a scalable architecture. This provides empirical evidence for the necessity of a hybrid approach that balances algorithmic sophistication with tangible business value.

While the concept of price-similarity weighting may be intuitive, the core contribution of this work lies in its successful implementation and rigorous, quantitative validation in a real-world, large-scale traffic environment through A/B testing. Whereas many theoretical models are not tested for their scalability or business impact in commercial settings, the proposed system, by integrating engineering solutions like distributed caching and zero-downtime indexing, presents a practical framework for how an academic idea can be translated into tangible commercial value. Therefore, the innovation of this study should be viewed not in the novelty of a single algorithm, but in the successful construction and empirical validation of an entire system that bridges the gap between theory and practice.

The potential risk of a 'filter bubble' induced by price similarity was mitigated by the system's hybrid design. Price was not used as a filter during candidate generation but rather as a re-ranking factor applied to a diverse list already generated by CF and BM25. This mechanism directly contributed to the aforementioned improvement in recommendation diversity, which in the long term can positively impact customer satisfaction and revenue diversification.

Finally, the variance in performance across different product categories, as evidenced by the results in Table 7, suggests that price sensitivity is not uniform. The substantial revenue increase of 13.30% in specific categories, such as fashion and electronics, where similarly featured products exist at various price points, supports the inference that consumers engage in more active price comparisons in these domains.

The findings of this study confirm the effectiveness of price-similarity weighting for single-item recommendations. A logical extension of this work is to investigate its applicability to bundle recommendations, where a hybrid framework could recommend a curated set of products. Future research could focus on optimizing the total price of the bundle to align with a user's inferred price sensitivity, a challenge already being explored for budget-conscious consumers [41]. Such an approach could further enhance customer satisfaction and drive revenue by offering comprehensive, budget-aligned solutions.

6. Conclusions

Overall, the hybrid system demonstrated scalability in high-traffic environments through zero-downtime index updates while effectively incorporating price considerations. These findings provide a practical framework for large-scale e-commerce platforms, contributing to both revenue growth and enhanced user satisfaction through greater recommendation diversity. This study further establishes the link between diversity in recommendations and positive revenue impact, offering a generalizable model for commercial services.

- Practical applications.
    - In high-traffic domains such as online marketplaces or streaming platforms, the combination of CF, BM25, and price similarity can be flexibly turned by adjusting their respective weights.
    - A practical implementation strategy is to integrate CF and BM25 scores as a baseline, followed by incorporating price similarity as a weighting factor to generate the final recommendation list.
- Limitations
    - Implementation complexity: Deploying the system requires advanced search engine configuration, caching strategies, and large-scale data pipelines, which demand substantial development and operational expertise.
    - Variability in price sensitivity: For branded, luxury, or niche products, price exerts limited influence on user decisions. In such cases, alternative or supplementary recommendation strategies should be considered to address varying levels of price elasticity across product categories.
    - Generalizability of Findings: A primary limitation of this study is its reliance on data from a single, large-scale domestic e-commerce platform. Consequently, the findings regarding the impact of price sensitivity may not be directly generalizable to other platforms that differ in their product categories, primary customer demographics, or overall market environment.

In deploying a hybrid recommendation system, it is essential to carefully assess compatibility with the existing operational environment and account category-specific price sensitivity. Optimal performance can be achieved by fine-tuning weight parameters and algorithmic configurations to reflect these contextual factors. However, as the present experiment was conducted over a limited time frame and relied on data from a single e-commerce platform, its generalizability is constrained. The lack of long-term and diverse user-group data underscores the need for future studies that expand and validate these findings under broader conditions.

In personalized recommendation systems, excluding products that fail to generate purchases after repeated exposures can enhance both CTR and conversion rates. In addition, refining the category classification engine to distinguish more subtle similarities among products can enhance the system's ability to recommend items that users are genuinely interested in purchasing.

The specific strategies are as follows:

Exclusion of repeated ignored items: When an item receives multiple exposures without resulting in a purchase, it can be excluded from the recommendation pool or assigned a lower priority. This approach minimizes unnecessary impressions and reallocates recommendation capacity toward items with higher conversion potential.

Enhanced category classification: By systematically categorizing similar products based on user preferences and purchase history, the recommendation algorithm can prioritize items that are more aligned with user interests. This refinement increases recommendation accuracy and improves overall user satisfaction.

Continuous feedback loop: By tracking post-recommendation behaviors such as clicks and purchases and periodically updating the model, the system can adapt to evolving user preferences and market conditions. This iterative process increases both the efficiency and long-term accuracy of recommendations.

Collectively, integrating repeated-exposure filtering, advanced category classification, and a feedback loop is expected to deliver substantial improvements in recommendation accuracy and conversion rates.

In conclusion, this study proposes and validates a practical recommendation framework that integrates collaborative filtering, BM25-based text matching, and price similarity within a large-scale e-commerce environment. The findings empirically demonstrate the synergistic effect of combining traditional recommendation techniques with price-based modeling, offering contributions to both academic research and industrial applications. Moreover, this study provides practical guidelines for implementing high-traffic systems, including zero-downtime index updates, search engine integration, caching strategies, and memory management. Finally, it highlights opportunities for future research in areas such as deep neural network-based or reinforcement learning-based recommendation systems, as well as their integration with marketing strategies.

Author Contributions: Formal analysis, Y.K.; Resources, Y.K.; Conceptualization, Y.B. and G.B.; Validation, G.B.; Writing—original draft, Y.K.; Visualization, Y.B.; Supervision, Y.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by "Chonnam National University (Smart Plant Reliability Center)" grant funded by the Ministry of Education (2020R1A6C101B197).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1 Implementation Details and Parameters

To ensure the reproducibility of the experiments and provide a clear understanding of the system's behavior, key implementation details and parameter settings are described below. The parameters used were optimized through empirical testing, and the experimental environment is based on the architecture detailed in the system architecture section. The primary configurations are summarized in Table A1.

BM25 Hyperparameters: The BM25 algorithm built into Elasticsearch was utilized for text similarity calculations. For the parameters k1 and b, the standard default values of 1.2 and 0.75 were used, respectively, as they provide robust performance in most text retrieval contexts.

Hybrid Scoring Weights: The proposed hybrid model integrates candidates from multiple sources. As described in the algorithm proposal section, the final score-based candidate list is generated from a weighted sum of BM25, price, and category similarity scores. This list is then merged with the CF-based candidate list, with CF recommendations given higher priority for items present in both. The specific weights for each score were dynamically tuned in the experimental environment.

Caching Strategy: To ensure real-time recommendation response speeds, computationally expensive results were stored in a Redis cache. The results from the CF model were cached with a Time-To-Live (TTL) of 48 h to balance freshness and system load. Additionally, to reduce the load on the search engine, results for similar-priced products were also cached; this data was assigned a shorter TTL of 4 h to account for more frequent changes in metadata such as product names, descriptions, and prices.

Table A1. Key Parameters and Implementation Details.

Component	Parameter	Value/Description
BM25 Algorithm	k1 (Term frequency saturation)	1.2 (Elasticsearch default)
	b (Document length normalization)	0.75 (Elasticsearch default)
Hybrid Scoring	Score-Based Candidate List Generation	As illustrated in Figure 2, the final score was calculated as a weighted sum of the BM25 score, price similarity score, and category similarity score. Weights were empirically tuned.
	Final List Integration	When integrating the CF-based and score-based candidate lists, CF-based recommendations were prioritized for items appearing in both lists.
Redis Caching Strategy	Time-To-Live (TTL) for CF recommendations	178,800 s (48 h)
	Time-To-Live (TTL) for price-similarity search results	14,400 s (4 h)
	Cache Update Policy	The CF cache was refreshed every 24 h to reflect the latest results from the offline model.
Elasticsearch indexing Strategy	Bulk Index Policy	The Elasticsearch product information is refreshed every 4 h to reflect the latest results from the offline model.

References

1. Fayyaz, Z.; Afzal, A.; Mirza, H.T. Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Appl. Sci.* **2020**, *10*, 7748. [CrossRef]
2. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* **2019**, *52*, 5. [CrossRef]
3. Krishna, E.S.P.; Ramu, T.B.; Chaitanya, R.K.; Ram, M.S.; Balayesu, N.; Gandikota, H.P.; Jagadesh, B.N. Enhancing E-commerce recommendations with sentiment analysis using MLA-EDTCNet and collaborative filtering. *Sci. Rep.* **2025**, *15*, 6739. [CrossRef] [PubMed]
4. Ou, T.Y.; Chen, C.H.; Tsai, W.L. Establishing a Dynamic Recommendation System for E-commerce by Integrating Online Reviews, Product Feature Expansion, and Deep Learning. *Appl. Artif. Intell.* **2025**, *39*, 2463723. [CrossRef]
5. Pei, C.; Yang, X.; Cui, Q.; Lin, X.; Sun, F.; Jiang, P.; Ou, W.; Zhang, Y. Value-aware Recommendation based on Reinforcement Profit Maximization. In Proceedings of the World Wide Web Conf 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 3123–3129. [CrossRef]
6. Gomez-Uribe, C.A.; Hunt, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manag. Inf. Syst.* **2016**, *6*, 13. [CrossRef]
7. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
8. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 3549–3568. [CrossRef]
9. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 97. [CrossRef]
10. Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J.; Jin, D.; He, X.; et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *arXiv* **2023**, arXiv:2109.12843. [CrossRef]

11. Wu, J.; Wang, X.; Feng, F.; He, X.; Chen, L.; Lian, J.; Xie, X. Self-supervised Graph Learning for Recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event Canada, 11–15 July 2021. [CrossRef]
12. Zhao, Z.; Fan, W.; Li, J.; Liu, Y.; Mei, X.; Wang, Y. Recommender Systems in the Era of Large Language Models (LLMs). *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 6889–6907. [CrossRef]
13. Min, X.; Sun, Z.; Liu, F.; Li, Q.; Liu, Y.; Zhang, D. Matrix Factorization Recommendation Algorithm Based on Attention Interaction. *Symmetry* **2024**, *16*, 267. [CrossRef]
14. Covington, P.; Adams, J.; Sargin, E. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 191–198. [CrossRef]
15. Bell, R.M.; Koren, Y. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007), Omaha, NE, USA, 28–31 October 2007; pp. 43–52. [CrossRef]
16. Jiang, W. Enhancing Operational Efficiency in E-Commerce Through Artificial Intelligence and Information Management Integration. *Rev. d'Intell. Artif.* **2023**, *37*, 1441–1449. [CrossRef]
17. Asfar, M.; Crump, T.; Far, B. Reinforcement Learning based Recommender Systems: A Survey. *ACM Comput. Surv.* **2022**, *55*, 38. [CrossRef]
18. Lu, J.; Wu, D.; Mao, M.; Wang, W.; Zhang, G. Recommender System Application Developments: A Survey. *Decis. Support Syst.* **2015**, *74*, 12–32. [CrossRef]
19. Lu, S.; Liu, Z.; Yang, X.; Ding, Y.; Gao, Y.; Yuan, Y. Meta-Learning for Debiasing Recommendation using Simulated Uniform Data. In Proceedings of the 2024 IEEE International Conference on Big Data (BigData), Washington, DC, USA, 15–18 December 2024. [CrossRef]
20. Zheng, Y.; Gao, C.; He, X.; Li, Y.; Jin, D. Price-aware Recommendation with Graph Convolutional Networks. In Proceedings of the 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1–12. [CrossRef]
21. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
22. Dong, Y.; Chawla, N.V.; Swami, A. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 135–144. [CrossRef]
23. Su, X.; Khoshgoftaar, T.M. A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**, *2009*, 421425. [CrossRef]
24. Fleder, D.; Hosanagar, K. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Manag. Sci.* **2009**, *55*, 697–712. [CrossRef]
25. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182. [CrossRef]
26. Agarwal, S.; Mehta, R.; Singh, V. A Healthy Food Recommendation System Using KNN Model and Elasticsearch with Quantum Computing. In *Handbook of Research on Predictive Analysis Using AI Tools and Applications*; IGI Global: Hershey, PA, USA, 2024; pp. 1–18. [CrossRef]
27. Liu, H.; Wang, Y.; Zhang, T. Optimizing Real Estate Recommendations with Elasticsearch and Collaborative Filtering. In *Advanced Computational Intelligence, Proceedings of the ICACI 2024, Singapore, 5–7 January 2024*; Springer: Singapore, 2024; pp. 187–201. [CrossRef]
28. Burke, R. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.* **2002**, *12*, 331–370. [CrossRef]
29. Lian, J.; Zhang, F.; Chen, X.; Xie, X.; Sun, G. XDeepFM: Combining explicit and implicit feature interactions for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1754–1763. [CrossRef]
30. Robertson, S.; Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389. [CrossRef]
31. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
32. Sedhain, S.; Mehta, A.; Suthossat, S.; Piech, C. AutoRec: Autoencoders meet collaborative filtering. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 92–93. [CrossRef]
33. Moreno, A.; Mehmood, R.; Hästbacka, E.; Ericsson, D. Exploring the landscape of hybrid recommendation systems in e-commerce: A systematic literature review. *IEEE Access* **2024**, *12*, 34413–34433. [CrossRef]
34. Sharma, S.; Baishya, K.; Pandey, M.; Rautaray, S. Hybrid Product Recommendation System using Popularity Based and Content-Based Filtering. In Proceedings of the 2023 International Conference on Data Science, Agents & Artificial Intelligence, New York, NY, USA, 24–26 February 2023; pp. 642–649. [CrossRef]
35. Ratchford, B.T. Online Pricing: Review and Directions for Research. *J. Interact. Mark.* **2009**, *23*, 82–90. [CrossRef]

36. Luo, H. Research on the impact of online promotions on consumers' impulsive online shopping intentions. *J. Theor. Appl. Electron. Commer. Res.* **2021**, *16*, 2386–2404. [CrossRef]
37. Lin, C.C.; Chien, T.K.; Ma, H.Y. The effects of popularity: An online store perspective. *Int. J. Inf. Sci. Manag.* **2014**, *12*, 1–11.
38. Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Waterson, P. The YouTube Video Recommendation System. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 293–296. [CrossRef]
39. Park, Y.J.; Tuzhilin, A. The long tail of recommender systems and how to leverage it. In Proceedings of the 2008 ACM Conference on Recommender systems, Lausanne, Switzerland, 23–25 October 2008; pp. 11–18. [CrossRef]
40. Abdollahpouri, H.; Burke, R.; Mobasher, B. Controlling Popularity Bias in Learning-to-Rank Recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 42–46. [CrossRef]
41. Han, G.; Feng, Z.; Xu, Y. Bundle Recommendation for Budget-Conscious Consumers. *Tsinghua Sci. Technol.* **2024**. Available online: https://www.sciopen.com/article/10.26599/TST.2024.9010144?issn=1007-0214 (accessed on 30 September 2025).
42. Amatriain, X.; Basilico, J. Recommender Systems in Industrial Settings. In *Recommender Systems Handbook*, 2nd ed.; Ricci, F., Rokach, L., Shapira, B., Eds.; Springer: New York, NY, USA, 2016; pp. 385–419. [CrossRef]
43. Kohavi, R.; Deng, A.; Frasca, B.; Walker, T.; Xu, Y.; Pohlmann, N. Online Controlled Experiments at Large Scale. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 1168–1176. [CrossRef]