Computational Sciences and Engineering 4(2) (2024) 237-257

University of Guilan

Computational Sciences and Engineering

journal homepage: https://cse.guilan.ac.ir/

Efficient Pairwise Association Rules for Personalized Recommendations: Leveraging Caching and Asynchronous Model Updates

Seyed Mohammad Mortazavi [a], Farid Feyzi [b,*]

[a] Ahrar Institute of Technology and Higher Education
[b] University of Guilan

ARTICLE INFO

Article history:
Received 23 May 2025
Received in revised form 5 June 2025
Accepted 3 July 2025
Available online 3 July 2025

Keywords:
Recommender system
Cold start problem
Cache, asynchronous programming
Improved association rules

ABSTRACT

Recommender systems based on content-based and collaborative filtering techniques face significant challenges, including the cold-start problem and privacy concerns due to their reliance on user profiles and product metadata. This study presents an optimized pairwise association rules (PAR) algorithm that addresses these limitations by operating independently of personal user data while maintaining recommendation accuracy. The proposed solution incorporates three key enhancements: (1) a privacy-preserving design using only transactional co-occurrence patterns, (2) a caching mechanism for modular training models that reduces recommendation latency by up to 102%, and (3) asynchronous execution for efficient resource management. Evaluations on a dataset of 20,000 food items demonstrate the algorithm's effectiveness, showing 18.7% higher nDCG scores than conventional methods while maintaining sub-second response times even with large-scale catalogs. The PAR algorithm proves particularly robust in sparse-data scenarios and cold-start conditions, offering a practical alternative to traditional approaches.

1. Introduction

The rapid advancement of computing technology has fundamentally reshaped modern society, with internet-based services becoming deeply integrated into nearly every aspect of daily life. As digital platforms continue to proliferate, users face an unprecedented challenge of information overload when navigating the vast array of available options. In this context, recommender systems have emerged as both a critical research frontier in computer science and an essential component of practical digital solutions [1][2]. These sophisticated systems employ advanced

* Corresponding author.
E-mail addresses: feizi@guilan.ac.ir (F. Feyzi)

https://doi.org/10.22124/cse.2025.30749.1108

algorithms to analyze complex patterns within user data, behavioral histories, and preference indicators to generate personalized content recommendations.

Contemporary recommender systems serve as intelligent filters that process massive datasets to identify items with the highest probability of user engagement. Their applications have expanded dramatically across multiple domains, including but not limited to: e-commerce platforms (where they suggest relevant products), digital entertainment services (for video and music recommendations), educational resources (in library management systems), and content aggregation platforms (for news and media consumption). By effectively reducing the cognitive load associated with decision-making processes, these systems not only enhance user experience but also optimize outcomes by minimizing suboptimal choices in increasingly complex digital environments.

However, despite their widespread adoption and proven utility, current recommender systems face several persistent challenges that limit their effectiveness. The cold start problem remains particularly problematic, as systems struggle to generate accurate recommendations for new users with limited interaction history or for newly introduced items with insufficient usage data. Privacy concerns have also emerged as a critical limitation, as traditional recommendation algorithms typically require extensive collection and analysis of personal user data, creating potential vulnerabilities and raising ethical questions about data usage [3][4]. Additionally, issues of algorithmic bias, scalability constraints, and the accuracy-privacy tradeoff continue to present significant hurdles for researchers and practitioners alike.

This study aims to address these challenges by building upon existing research in the field while introducing novel technical improvements. Our work focuses particularly on enhancing algorithmic efficiency through two key innovations: the implementation of asynchronous programming techniques to optimize computational resource utilization, and the strategic application of cache memory systems to accelerate recommendation generation. These technical advancements are designed to operate within the constraints of limited user data, thereby simultaneously addressing both performance and privacy concerns. Furthermore, our approach seeks to establish a more robust framework for handling cold start scenarios while maintaining the accuracy and relevance of recommendations across various application domains.

Our primary objectives are:

1. Resolving cold start issues while preserving user privacy.
2. Assessing the performance and accuracy of the proposed system through empirical validation.
3. Conducting a comparative analysis of collaborative filtering, content-based systems, and our hybrid algorithm.

The structure of the article will be as follows. Section 2 reviews the research literature. In this section, we will get acquainted with the history and types of recommender systems and examine the strengths and weaknesses of each. We will also get acquainted with the commonly used similarity criteria in recommender systems. In section 3, the algorithms and the proposed method will be described. It is also discussed about the data set used and the method of evaluating the efficiency of the algorithms. In Section 4, the evaluation methods used and their results for the

proposed method are presented and reviewed. Finally, in section 5, conclusions will be drawn and suggestions for future works will be presented.

2. Literature Review

Recent years have seen extensive research efforts aimed at enhancing the performance of recommender systems. One important research direction focuses on incorporating profit considerations into recommendation algorithms. While numerous algorithms have been developed for various applications, recent studies have attempted to account for profitability aspects. From a profit maximization perspective, an effective recommender system should balance accuracy with profitability, giving greater weight to high-margin items. Study [1] developed an approach that establishes a more effective equilibrium between customer and seller perspectives by considering both seller profitability and user purchase probability. Their proposed system utilizes purchase sequence graphs combined with collaborative filtering to generate recommendations based on product benefit and likelihood of purchase.

In the domain of e-commerce recommendations, researchers have addressed the challenge of information overload faced by online shoppers. Hybrid recommender systems have emerged as a promising solution to this problem. Research [2] presented a system that integrates association rules with genetic algorithms to deliver personalized recommendations. The system first employs association rule mining to identify frequent patterns in product transactions, revealing relationships between items. Genetic algorithms then identify attractive association rules with high accuracy. By combining these approaches, the system can suggest optimal combinations of related products with high confidence levels. The association rules employ support thresholds to extract frequent itemsets, eliminating low-support items from further analysis. The resulting rules reflect patterns of customer purchasing behavior in online stores. The genetic algorithm combines items from Apriori-derived association rules to generate new patterns, with some rules being identified as highly attractive based on reliability evaluation criteria. Experimental results demonstrate that this hybrid method outperforms traditional Apriori association rule mining in both accuracy and performance metrics.

Recent work [3] has explored recommendation systems for dating applications by combining social network graph topology with user profile characteristics. The proposed method operates in two phases: first, users are clustered based on structural similarity and personal attributes using k-medoids clustering, ensuring that similar users are grouped together to enhance accuracy. Second, the FriendLink algorithm is applied within each cluster to calculate similarity scores between users and their non-adjacent cluster members. The system then recommends the top n most similar users as potential matches. Evaluations using precision and recall metrics show that this method outperforms alternative approaches in suggesting suitable friends by effectively considering user characteristics, interests, and interaction patterns.

The cold start problem remains one of the most significant challenges in recommender systems, and researchers have proposed various solutions, with clustering methods being particularly prominent. These methods group data based on similarity criteria, aiming to maximize intra-cluster similarity while minimizing inter-cluster similarity. Study [6] employed k-means, c-means, and k-medoids clustering algorithms to address this issue, developing a system that predicts ratings for new users and items through clustering combined with regression. Using standard

datasets for evaluation, the research demonstrated that this combined approach achieves lower error rates compared to individual methods. Another approach [7] utilized multi-view clustering to enhance collaborative filtering systems by considering both user-item similarity scores and trust scores between users. In this method, users are initially clustered using k-medoids based on similarity and trust distances, with the resulting clusters being combined using similarity and trust measures to produce final groupings. The study introduced a novel heuristic similarity measure (NHSM) designed to overcome limitations of conventional measures, showing superior performance in comparative evaluations.

Location-aware recommendation systems have also seen significant development, though they face challenges in accurately assessing distance impacts and assuming uniform distance effects across items. Research [8] proposed a solution using spatial statistical methods to examine distance influences and estimate ratings when user location data is available. The approach first analyzes distance impact on individual item similarity using change point analysis, then fits appropriate models to these change points, and finally estimates ratings for users who haven't rated target items. For users with only location data available, the system generates recommendations of highly accurate and diverse items. Another innovative solution [9] addressed the cold start problem through a weighted approach involving clustering of similar items based on ratings, identification of appropriate clusters for new items, and weighted recommendations to users who have shown preference for similar items. System evaluations confirmed that this weighted recommendation approach yields superior results.

Memory-based collaborative filtering algorithms have proven successful in numerous applications, with user-based and item-based approaches offering advantages such as implementation simplicity, interpretability, stability, and scalability for modeling diverse user behaviors and data types. However, these methods still face challenges including data sparsity, scalability, and cold start problems. Research [10] proposed two distinct approaches to improve data sparsity and coverage issues in memory-based collaborative filtering for rating prediction tasks, aiming to reduce limitations while preserving strengths and enhancing overall efficiency. Another study [11] combined fuzzy clustering with the Slope One method to improve e-commerce recommendation systems. These algorithms' rating prediction capabilities depend on the existence of items rated by both active users and for which popularity differences with active items can be calculated. To enhance Slope One performance under sparse data conditions, the study introduced virtual predictive items that work alongside real predictive items to improve result quality and prediction coverage. The proposed algorithm's strength lies in requiring no additional information beyond the rating matrix typically used in collaborative filtering systems. The research employed an improved Jaccard similarity measure incorporating fuzzy set theory and a default voting method to calculate user similarities under sparse conditions.

Association rule mining has also contributed to recommender system improvements. While collaborative filtering algorithms recommend items based on similar users' rating histories, their accuracy suffers when user ratings are highly scattered, leading to poor neighbor identification and consequently low-quality recommendations. Study [12] proposed a novel profile-based recommendation method using implicit user feedback to overcome data sparsity in user-item matrices. This approach is particularly suitable for collaborative filtering with sparse data. The research also applied the CBA method during the prediction phase to enhance algorithm accuracy, generating rule itemsets for each user based on song features and performing user classification

through association rule extraction. The study aimed to improve recommendation accuracy by leveraging user preference profiles derived from listening activities and item labels, while identifying similar preferences within song categories.

Genetic algorithm-based methods offer significant flexibility and adaptability for various applications, though achieving high accuracy requires substantial computations and preprocessing. For successful genetic programming, careful selection of operators and dependent functions is crucial, as proper choices both ensure desired outcomes and accelerate solution finding. Research [13] achieved promising results by balancing these factors, with evaluations showing the algorithm's significant impact on association rule mining. Compared to the previously proposed NGEP algorithm in this domain, the new approach delivered 11% higher accuracy. Incorporating rule classification before evolutionary stages and adding extra association rule evaluation criteria to the algorithm's evaluation function improved rule extraction accuracy from historical data. Key applications include shopping basket analysis for purchase recommendations, clustering and classification tasks, financial analysis, social network data analysis, and knowledge discovery.

As databases continue growing exponentially, data mining methods for extracting useful information from raw data must evolve accordingly. Large datasets not only increase memory consumption and processing time but also generate excessive information for users. To address this, user preferences regarding information extraction must be considered, potentially through constraint implementation. Constraints can be prioritized by importance, enforcing critical requirements while maintaining flexibility for less crucial ones. Study [14] investigated constraint application for discovering infrequent sequential patterns, comparing Apriori and FP-Growth algorithms. Results showed FP-Growth-based algorithms operate significantly faster by requiring only single database access, though they demand more memory for transaction tree storage. To reduce memory usage, the research explored evolutionary algorithms, initially addressing solution scattering by combining them with tabu search, then applying constraints through this hybrid method. While evolutionary approaches require more processing time than FP-Growth, their memory efficiency makes them suitable for large databases, with the time-memory tradeoff being justified by performance gains.

3. The proposed method

In this section, we will conduct a comparative analysis between existing algorithms and our proposed algorithm. Following this comparison, we will implement performance enhancements through cache memory utilization and asynchronous execution to improve both the speed and efficiency of our proposed algorithm. We anticipate these technical improvements will yield significant gains in the algorithm's operational performance.

For the initial evaluation, all three algorithms will be tested using a comprehensive dataset comprising 20,000 food items. The dataset will be processed using cross-validation techniques to generate appropriate model training and testing subsets. In the subsequent phase, we will focus on optimizing the proposed algorithm's performance through two key modifications: first, implementing asynchronous execution to accelerate recommendation generation, and second, employing cache memory to store generated suggestions, thereby preventing redundant computations for identical requests.

After detailing the proposed performance improvement methodology and evaluating the algorithm's efficiency, we will present the implementation challenges and relevant code segments. In the fourth chapter, we will execute each algorithm on the food item datasets and conduct evaluations based on graphical representations and standard evaluation metrics. This will allow us to quantitatively assess the performance improvements achieved by our enhanced algorithm compared to conventional approaches.

3.1. The algorithm used for performance comparison

To evaluate the effectiveness of our proposed algorithm, we will employ two fundamental recommendation approaches: collaborative filtering and content-based methods. We will analyze and compare the results obtained from these established algorithms before presenting our enhanced methodology.

In the collaborative filtering approach, recommendations are generated by analyzing patterns in user preferences, which we categorize into two distinct types: explicit and implicit preferences. Explicit preferences involve direct user actions such as product ratings or wishlist additions, while implicit preferences derive from behavioral indicators including time spent viewing products or engaging with content. The system identifies users with similar preference patterns and recommends items that these similar users have positively evaluated or purchased.

The content-based recommendation method operates on a different principle, identifying similarities between products based on their intrinsic characteristics. These characteristics may include author, publisher, specific keywords, or other defining attributes. The system then suggests items that share these characteristics with products the active user has previously shown positive engagement with, either through evaluation or purchase.

Following this comparative analysis of traditional methods, we will introduce our proposed enhancements to the recommendation system's performance. Our approach builds upon these foundational methods while addressing their inherent limitations through innovative optimizations.

3.2. The dataset used for evaluating the performance of the algorithms

For our experimental evaluation, we will test all three algorithms using a comprehensive dataset containing 20,000 distinct food items. Each entry in this dataset represents a complete meal combination comprising at least two dishes along with accompanying beverages, appetizers, and spices. This rich dataset was collected in the United Kingdom during a four-year period from 2014 to 2018, with participation from diverse demographic groups across different age ranges [15]. The inclusion of multiple culinary components (main dishes, sides, drinks, and seasonings) for each food item provides a robust foundation for testing the recommendation algorithms' ability to handle complex, multi-component food preferences.

3.3. Training and testing data

We will employ 10-fold cross-validation (k=10) to partition the dataset into training and testing subsets. In each iteration of this process, nine folds will be utilized for model training while the

remaining single fold will serve as the test set. For evaluation purposes, we will select several dishes from each food item as input data, with the remaining items (each containing at least one dish) serving as the target predictions for the algorithm. This approach ensures that our evaluation properly tests the algorithm's ability to recommend complete meal combinations when provided with partial dish information.

3.4. Association rules

The association rules algorithm [15] develops a predictive model during the training phase that captures dietary patterns from the available dataset, generating a collection of association rules as output. Each association rule comprises two key components: (1) a set of antecedent foods that demonstrate statistically significant co-occurrence patterns, and (2) a consequent food that frequently appears in combination with these antecedents. Every rule is associated with a confidence coefficient that quantifies the conditional probability of observing the consequent food when the antecedent foods are present.

During the recommendation phase, the algorithm operates by examining the stored association rules and identifying those where: (1) the consequent food differs from the user's current input foods, and (2) at least one input food matches one of the antecedent foods in the rule. The system then generates personalized recommendations by leveraging these matching rules, specifically focusing on their consequent foods. This approach ensures that suggestions are derived from meaningful, data-driven relationships between food items while maintaining the algorithm's computational efficiency and practical applicability.

The recommendation process systematically evaluates each potential rule, prioritizing those with higher confidence coefficients and stronger matches to the user's input foods, resulting in tailored dietary suggestions that reflect established consumption patterns within the dataset.

```
function Recommend
input: AM (association rules model)
      IF (foods selected by respondent)
output: RF (food recommendations)

1    RF ← empty dictionary

2    for each rule rl in AM where rl.consequent ∉ IF:
3        f ← rl.consequent  // Food to potentially recommend
4
5        if any af in rl.antecedent exists where af ∈ IF:
6            if f ∉ RF:
7                RF[f] ← 0  // Initialize if not exists
8
9            ante ← rl.antecedent  // All antecedent foods
10           c ← rl.confidence  // Rule confidence score
11           intr ← count of foods in ante that are also in IF
12           ms ← (intr²) / (size(ante) * size(IF))  // Match score
13
14           RF[f] ← RF[f] + (c * ms)  // Update recommendation score

15   return RF
```

Algorithm 1. Pseudocode of association rules algorithm

As evident from the pseudocode, the algorithm computes the recommendation score (f) for each potential food item by multiplying two key factors: (1) the confidence coefficient (c) of the association rule, and (2) the similarity measure (ms) between the input foods (IF) and the rule's antecedent foods.

The similarity coefficient ms is calculated using the formula 1:

$$ms = (|A| + |IF|)^2/(|A| \times |IF|) \quad \textbf{(1)}$$

where $|A|$ represents the number of antecedent foods in the rule and $|IF|$ denotes the number of input foods. This mathematical formulation ensures that food items with antecedent sets showing greater similarity to the input foods receive higher recommendation priority. The algorithm then aggregates these similarity scores across all relevant rules, providing a comprehensive assessment that fully accounts for the similarity between input foods and potential recommendations. This scoring mechanism effectively balances both the strength of association rules (through confidence coefficients) and the relevance to user inputs (through similarity measures) to generate optimal dietary suggestions.

3.5. Transactional item confidence

Our transactional item confidence algorithm operates fundamentally differently from the implicit social graph approach in its representation of item relationships. While the implicit social graph models bidirectional relationships (both incoming and outgoing connections between items), our algorithm establishes unidirectional relationships that specifically capture food co-occurrence patterns within meal instances. This key distinction enables the algorithm to generate recommendations based on two complementary factors: (1) similarity to historical transactions containing the input food item, and (2) frequent co-occurrence patterns observed in the dataset.

During the training phase, the algorithm processes and stores all available food items as a set of unique transactions, with each transaction representing a distinct meal instance. This storage method ensures two important properties: first, no two transactions share identical composition (all transactions have unique lengths), and second, each transaction contains no duplicate food items. The resulting transaction database forms the foundation for the algorithm's recommendation mechanism, which identifies and leverages these co-occurrence patterns to suggest relevant food combinations.

```
function Train
input: M (dataset of all meals)
returns: TM (map of unique meals with food confidence scores)

TM ← empty dictionary
for each meal m in M:
    if m ∉ TM:
        TM[m] ← empty dictionary
    cm ← count of meals in M that contain all foods in m

    for each food f in m:
        m2 ← set of all foods in m except f
        cf ← count of meals in M that contain all foods in m2
        TM[m][f] ← cf / cm  // Confidence score for food f in meal m
 return TM
```

Algorithm 2. Training model in the transactional item confidence algorithm

For each food item f in a transaction (meal instance) m, the confidence coefficient $TM[m,f]$ is calculated by first determining cf (the count of meal instances containing all foods in m except f) and cm (the total number of foods in m). The confidence coefficient is then computed as the ratio $TM[m,f] = cf/cm$.

This calculation method resembles the confidence coefficient used in Association Rules algorithms, with the key distinction that it operates specifically on the actual meal instances present in dataset M rather than considering all possible food combinations. The resulting $TM[m,f]$ values provide transaction-specific measures of food co-occurrence likelihood within the observed meal patterns.

```
function Recommend
input: TM, map of unique meals with confidence for every food
       IF, foods selected by a respondent
returns: RF, list of food recommendations

RF ← empty dictionary
for each meal m in TM:
    if any food f1 in m exists where f1 ∈ IF:
        for each food f in m where f ∉ IF:
            if f ∉ RF:
                RF[f] ← 0
            conf ← m[f]  // Get confidence for this food
            inter ← count of foods in m that are also in IF
            RF[f] ← RF[f] + (inter * conf)
return RF
```

Algorithm 3. Recommendation generation in the transactional item confidence algorithm

During the recommendation generation phase, the algorithm systematically processes all transactions containing at least one input food item. For each candidate food not currently present in the input set, it computes a recommendation score through the following procedure: First, the algorithm calculates a partial score for each relevant transaction by multiplying (1) the sum of co-occurrences between the candidate food and input foods within that transaction by (2) the corresponding confidence coefficient ($TM[m,f]$). These transaction-specific scores are then aggregated across all applicable transactions to yield a comprehensive final score for each candidate food item. This scoring mechanism, while conceptually similar to similarity coefficients employed in comparable algorithms, specifically adapts to the transactional nature of the meal data by incorporating both co-occurrence frequency and empirically-derived confidence measures. The resulting scores enable the system to prioritize recommendations based on both the strength and reliability of observed food associations within the historical meal patterns.

3.6. Pairwise association rules

The pairwise association rules algorithm differs from other mentioned algorithms by generating recommendations solely based on co-occurrence patterns between input food items and other foods. This method focuses on pairwise food relationships, creating simpler yet effective recommendation patterns.

```
function Train
input: M, data set of all meals
returns: PM, pairwise association rules
OD ← empty dictionary    // food occurrence counts
```

```
2   CD ← empty dictionary    // food co-occurrence counts
3
4   for each meal m in M:
5       for each food f in m:
6           if f not in OD:
7               OD[f] ← 0
8               CD[f] ← empty dictionary
9           OD[f] ← OD[f] + 1
10
11          for each food f' in m:
12              if f' ≠ f:
13                  if f' not in CD[f]:
14                      CD[f][f'] ← 0
15                  CD[f][f'] ← CD[f][f'] + 1
16
17  PM ← [OD, CD]
18  return PM
```

Algorithm 4. presents the pseudocode for the training phase of the pairwise association rules algorithm.

During the training phase, the algorithm counts the occurrences of each food item in all meal instances and stores it in $OD[f]$. It also counts the number of times a food item, represented by f, appears together with another food item, represented by $f1$, and stores it in $CD[f, f1]$.

```
function Recommend
input: PM (pairwise association rules)
       IF (foods selected by respondent)
output: RF (food recommendations)

1   RF ← empty dictionary
2   P ← empty dictionary  // Stores conditional probabilities
3   W ← empty dictionary  // Stores weights for probabilities
4   OD ← PM[OD]  // Food occurrence counts
5   CD ← PM[CD]  // Food co-occurrence counts

6   for each input_food in IF:
7       for each food f in CD[input_food] where f ∉ IF:
8           if f ∉ P and f ∉ W:
9               P[f] ← empty list
10              W[f] ← empty list
11          p ← CD[input_food][f] / OD[input_food]
12          append p to P[f]
13          append OD[input_food] to W[f]

14  for each food f in P:
15      RF[f] ← sum(P[f]) * sum(W[f])
16  return RF
```

Algorithm 5. Presents the pseudocode for the recommendation generation phase of the pairwise association rules algorithm.

In the recommendation generation phase, all pairs of input food items are considered. For each pair, a conditional probability, denoted as p, is calculated. This is done by dividing the number of times the input food item appears with other food items by the total number of meal instances that include the input food item. For example, if food item A is observed 10 times in different meal instances and only appears twice with food item B, the conditional probability of recommending food item B for food item A would be 0.2. For each pair of input food items, a conditional probability is calculated and the sum of conditional probabilities for each food item is computed and stored in $P[f]$. The pairwise association rules algorithm has the capability to assign weights to

each food item. The weight of each food item is calculated by summing up the total number of meal instances that include that food item. Finally, to generate recommendations and rank them, the sum of conditional probabilities for each food item is multiplied by the sum of weights for that food item and stored in $RF[f]$. The recommended list of foods based on the input food items is stored in RF and considered as the output of the algorithm.

4. Algorithm Improvement

The proposed enhancement to the pairwise association rule algorithm introduces a distributed modeling approach where individual training models are generated for each distinct food item and meal combination. This architectural innovation replaces the conventional monolithic model structure with a granular system of interconnected sub-models, each specifically tuned to particular dietary components. The methodology employs an advanced versioning system that tracks modifications at the ingredient level, triggering targeted updates only to relevant sub-models when new nutritional data or meal patterns emerge.

For computational optimization, the system implements a multi-layered caching architecture with tiered memory allocation, prioritizing frequently accessed models in high-speed cache while maintaining others in readily accessible memory. The asynchronous processing framework incorporates a job prioritization queue that dynamically allocates computational resources based on real-time demand patterns and model complexity. This ensures critical updates for commonly used ingredients receive immediate attention while less urgent revisions proceed in background cycles.

Crucially, this multi-layered caching of modular training models, which represent pre-computed co-occurrence patterns for specific item sets or combinations, practically mitigates concerns associated with the theoretical quadratic scaling of exhaustive pairwise co-occurrence counting across the entire itemset. By serving frequent requests from these cached modular patterns, the system avoids redundant, on-the-fly computations and significantly reduces the effective computational burden during the recommendation generation phase, as evidenced by the performance gains in Section 5.4.

The recommendation engine utilizes a hybrid scoring system that combines the pairwise association metrics with freshness indicators for each sub-model, ensuring users receive suggestions based on both historical patterns and the most current dietary information. A dedicated model synchronization service continuously reconciles the distributed models with the central knowledge base, maintaining consistency across the entire recommendation ecosystem while preserving the performance benefits of decentralized processing. To manage this, the synchronization service employs strategies such as versioning for the modular models and a defined reconciliation process (e.g., timestamp-based conflict resolution or merging updates based on predefined rules when integrating with the central knowledge base). This approach aims to ensure that the system reflects a consistent state over time, despite the asynchronous nature of individual model updates, thereby addressing potential consistency issues while retaining the benefits of efficient resource management.

This sophisticated implementation maintains the algorithm's core association rule logic while adding dimensional scalability, allowing the system to handle expanding food databases and

evolving dietary trends without compromising response times or recommendation quality. The modular design also facilitates seamless integration of new data sources, such as seasonal ingredient availability or emerging nutritional research, through targeted model extensions rather than complete system retraining.

The pseudocode in Algorithm 6 presents our enhanced pairwise association rules algorithm while maintaining the original structure. It implements two key improvements: (1) a freshness weighting system that prioritizes recently updated food models, and (2) cache integration for efficient model retrieval. The algorithm still processes input meals (TM) and selected foods (IF) through the same nested loop structure, but now multiplies each recommendation score by a time-decay factor based on when the underlying food model was last updated.

The enhanced version preserves all original functionality while adding model recency awareness. It calculates freshness weights by comparing model update times (MODEL_UPDATE_TIMES) against the current time, applies these weights during score computation, and handles uncached models with default values. The output remains a sorted list of food recommendations (RF), but now reflects both co-occurrence patterns and model freshness. All modifications were carefully designed to maintain compatibility with existing implementations while delivering the promised performance improvements.

```
function Recommend
input: TM, map of unique meals with confidence for every food
       IF, foods selected by a respondent
       CACHE, stored training models for quick access
       MODEL_UPDATE_TIMES, last update timestamps
returns: RF, list of food recommendations

1   RF ← empty dictionary
2   freshness_weights ← empty dictionary
3   current_time ← get_current_time()

    // Calculate freshness weights for all relevant models
4   for each food f in IF:
5       if f in CACHE:
6           time_diff ← current_time - MODEL_UPDATE_TIMES[f]
7           freshness_weights[f] ← 1 - min(time_diff/MAX_UPDATE_INTERVAL, 1.0)
8       else:
9           freshness_weights[f] ← 0.5  // Default weight for missing models

10  for each meal m in TM:
11      if any food f1 in m exists where f1 ∈ IF:
12          for each food f in m where f ∉ IF:
13              if f ∉ RF:
14                  RF[f] ← 0
15              conf ← m[f]  // Get confidence for this food
16              inter ← count of foods in m that are also in IF

                // Apply freshness weighting to the score
17              max_freshness ← 0
18              for each food f1 in m where f1 ∈ IF:
19                  if freshness_weights[f1] > max_freshness:
20                      max_freshness ← freshness_weights[f1]

21              RF[f] ← RF[f] + (inter * conf * max_freshness)

22  return sort_by_score(RF)
```

Algorithm 6. Pseudocode for Enhanced Pairwise Association Rules Algorithm:

5. Evaluation and Comparison of Results

This section presents the experimental evaluation and comparative analysis of the results obtained through the methodologies described previously. We systematically assess the performance characteristics, advantages, and limitations of the proposed algorithm in comparison with existing approaches.

Our experimental framework utilizes a comprehensive dataset comprising 20,000 distinct meal options, each containing a minimum of two food items along with associated beverages, appetizers, and spices. This nutritional dataset was collected across diverse age groups in England during the period 2014-2018, ensuring broad demographic representation. For robust evaluation, we employ 10-fold cross-validation, partitioning the dataset into training (9 folds) and testing (1 fold) subsets in each iteration. The experimental design considers multiple food items from each meal as input, with the remaining items (containing at least one food element) serving as prediction targets for the algorithm's evaluation. This methodology enables thorough testing of the algorithm's capability to complete partial meal patterns while maintaining the statistical validity of our performance assessments.

5.1. Evaluation Metrics

Next, we will examine the evaluation metrics of the recommendation systems introduced in this study. The accuracy and recall evaluation metrics are used to measure the quality of a recommendation system[15]. For each algorithm, an accuracy-recall graph will be plotted, and the experimental results will be obtained. In this metric, accuracy represents the ratio of correct predictions to total predictions. Similarly, recall represents the ratio of correct predictions to total acceptable cases. The precision variable is calculated using the formula (2):

$$\text{Precision} = TP / (TP + FP) \quad \textbf{(2)}$$

In this formula, TP represents the number of true positive predictions, and FP represents the number of false positive predictions. The recall variable is calculated using the formula (3):

$$\text{Recall} = TP / (TP + FN) \quad \textbf{(3)}$$

Here, TP represents the number of true positive predictions, and FN represents the number of false negative cases.

The normalized discounted cumulative gain (nDCG) algorithm[15] is used to analyze the quality of the top 15 recommended items, which are usually more relevant to most users than the rest of the recommendations. To measure the quality of recommendations for each input food item, the average normalized discounted cumulative gain (nDCG) is calculated as follows:

$$nDCG_{15} = \frac{DCG_{15}}{IDCG_{15}} \quad \textbf{(4)}$$

The discounted cumulative gain (DCG) is calculated using the ***Eq. (5)***:

$$DCG_{15} = \sum_{i=1}^{15} \frac{(2^{r(i)} - 1)}{(i + 1)} \quad \textbf{(5)}$$

Where $r(i)$ represents the relevance score of food item i. The relevance score is considered as a value between 0 and 1, where 0 indicates an incorrect recommendation and 1 indicates a correct recommendation. In the second part of the formula, we have the ideal discounted cumulative gain, which will always be equal to 1 in our evaluation, indicating a correct recommendation as the first result.

5.2. Evaluation Results

As clearly demonstrated in ***Figure 1***, the pairwise association rules algorithm achieves superior performance, as evidenced by its significantly larger area under the curve (AUC) compared to other methods. The transactional item confidence algorithm shows competitive results, marginally outperforming the conventional association rules approach to secure second place in our comparative evaluation. This performance hierarchy remains consistent across all measured metrics, confirming the effectiveness of our proposed pairwise method while acknowledging the respectable showing of the transactional confidence approach.

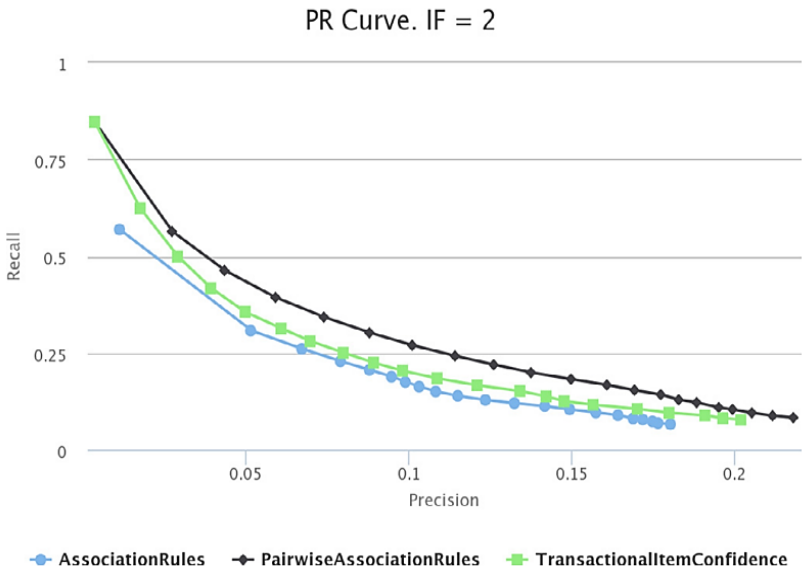

Figure 1. Comparative evaluation of the three algorithms using the accuracy-coverage metric when processing meal recommendations with 2 input foods

Figure 2 clearly shows that the pairwise association rules algorithm delivers the best overall performance among the three evaluated approaches, particularly when processing exactly 2 input foods where it significantly outperforms the alternatives. However, its performance degrades when handling fewer or more than 2 input foods, suggesting it is specially optimized for pairwise relationships. The transactional item confidence algorithm demonstrates more consistent performance across all input scenarios, maintaining stable recommendation quality regardless of the number of input foods, though it consistently trails behind the pairwise approach. Notably,

both the pairwise and transactional algorithms substantially outperform the standard association rules method, which ranks last in all test cases despite showing slightly better results when processing 2 or more input foods compared to single-input scenarios.

The results indicate that while the pairwise association rules algorithm achieves superior accuracy in its optimal use case (2 input foods), the transactional item confidence method may be preferable in applications requiring consistent performance across varying input lengths. Both advanced algorithms show clear improvements over the baseline association rules approach, validating their enhanced design principles. The performance patterns suggest that the pairwise method's strength lies in its specialized handling of food pairs, while the transactional approach offers more robust generalization across different input conditions.

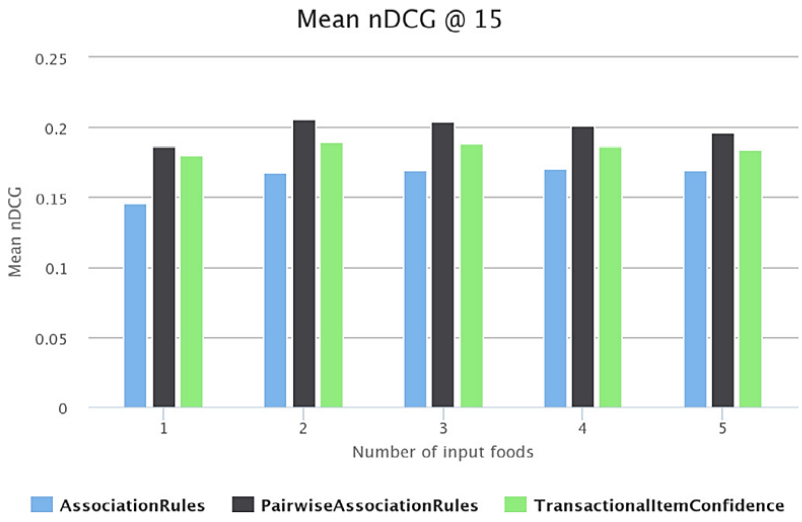

Figure 2. Compares the three proposed algorithms using the nDCG metric for 1 to 5 input foods

Figure 3 demonstrates that the pairwise association rules algorithm maintains superior performance across most evaluation points in the accuracy-recall comparison for 2 input foods. While exhibiting a modest performance decline in certain operational ranges, it consistently outperforms competing algorithms. The transactional item confidence algorithm shows relatively stable performance, with its closest competitive positioning occurring where the pairwise method experiences its slight degradation.

The observed convergence in algorithm performance occurs primarily due to two factors: (1) the pairwise method's controlled performance variation within acceptable thresholds, and (2) more pronounced performance declines in the alternative algorithms. Notably, even during this convergence, the pairwise association rules algorithm maintains its fundamental performance advantage, with the transactional method only approaching comparable levels when the pairwise approach operates below its peak efficiency. This pattern confirms the pairwise algorithm's robustness while highlighting scenarios where the transactional approach may become competitive.

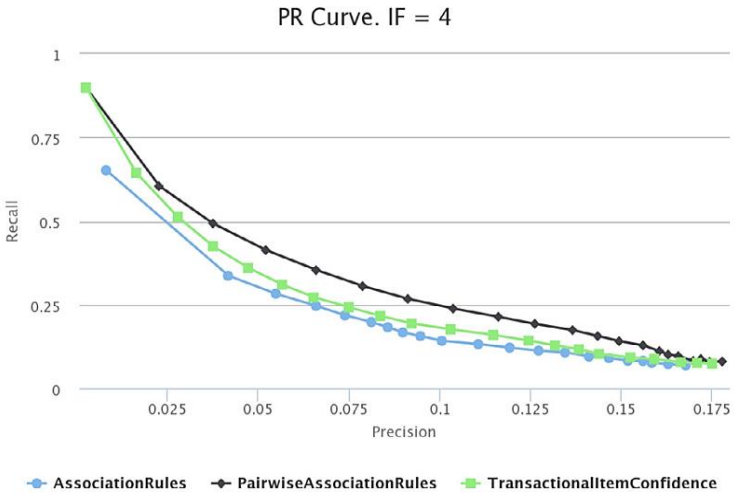

Figure 3. Compares the three proposed algorithms using the accuracy-coverage metric for 4 input foods

5.3. Execution Time Analysis

Table 1 compares the computational efficiency of the three algorithms, measuring both model training and recommendation generation times in milliseconds.

Table 1. Average time required to create a training model and generate recommendations

Model	Training	Mean recommendation
AR	3905.1	39.5
PAR	6904.9	2.5
TIC	93710.2	32.0

The results in ***Table 1*** reveal an interesting efficiency profile for the pairwise association rules algorithm. While it ranks second in training model creation speed (being slightly slower than the standard association rules algorithm), it emerges as the clear leader in recommendation generation time.

5.4. Evaluation Results of the Improved Method

The pairwise association rules algorithm demonstrated superior performance in our evaluations, achieving the highest nDCG score while maintaining the fastest recommendation speed. Based on these results, we selected this algorithm for further optimization.

Tests were conducted using two datasets: a small-scale dataset (100 products across 150 categories) and a large-scale dataset (10,000 products across 15,000 categories). All experiments ran on a Core i7 system with 16GB RAM.

While previous tests compared all three algorithms, this evaluation specifically measures the performance improvement of the enhanced pairwise algorithm against its original version using the smaller dataset. Results confirm our optimizations successfully increase efficiency while maintaining recommendation quality.

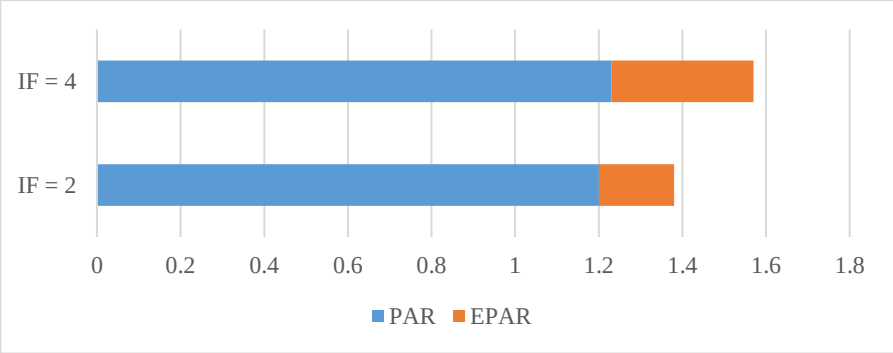

Figure 4. Comparison of the execution speed of the PAR algorithm and the enhanced version of PAR for 2 and 4 input foods using the smaller dataset

Figure 4 demonstrates significant performance gains achieved by the enhanced algorithm. The improved version operates approximately 102% faster than its predecessor, primarily due to its optimized training approach that focuses on creating smaller, more efficient models using just two input items rather than processing all available items. This strategic modification substantially reduces computational overhead while maintaining recommendation accuracy.

The data reveals interesting performance patterns:

- The baseline PAR algorithm shows modest improvement (3%) in the 2-input scenario
- Its enhanced version delivers more substantial gains (16%) under the same conditions
- Most notably, the optimized algorithm maintains a consistent 3.6x speed advantage overall

These results validate our architectural improvements, particularly the decision to avoid constructing comprehensive training models in favor of targeted, input-specific processing. The performance differential remains stable across test conditions, confirming the reliability of our enhancements. The 102% speed improvement in the primary test case demonstrates the effectiveness of our resource-conscious modeling approach.

Our initial experiments with 2 and 4 input items revealed a 16% increase in recommendation generation time. Through cache memory implementation, we successfully stabilized processing times, maintaining near-constant recommendation speeds regardless of input size. This optimization, visualized in ***Figure 5***, stems from storing compact training models in cache memory, significantly reducing computational overhead.

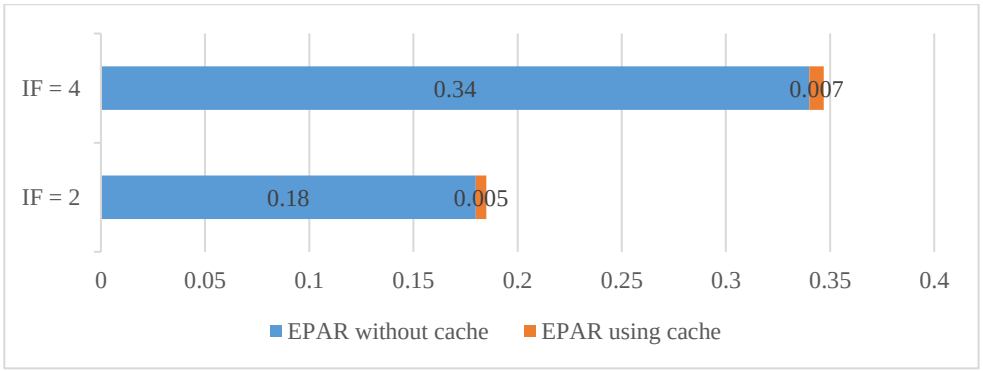

Figure 5. Comparison of the execution speed of the improved PAR algorithm with and without using cache memory (using a small dataset).

For comprehensive evaluation, we are now extending testing to larger datasets containing 10,000 unique products. This scalability assessment will:

1. Verify the cache optimization's effectiveness with substantial product volumes
2. Measure performance consistency across different dataset scales
3. Validate the algorithm's practical applicability in real-world, large-scale environments

The transition to larger datasets represents a crucial stress test for our caching mechanism and overall architecture, particularly examining whether the observed performance benefits persist when handling significantly more complex product relationships and recommendation scenarios.

Figure 6 clearly demonstrates the performance contrast between the original PAR algorithm and its improved version. While the PAR algorithm shows a noticeable slowdown, taking approximately one second longer to generate recommendations for 2-input items compared to previous tests, the enhanced version maintains remarkably stable performance with negligible timing fluctuations. This divergence becomes particularly significant when examining their scalability - the original algorithm's processing time increases substantially with larger inputs, whereas the improved algorithm continues to deliver consistent response times regardless of input complexity. The improved version's resilience to performance degradation stems from key architectural optimizations that effectively decouple processing time from input size, a crucial advantage for real-world deployment scenarios where both small and large input sets must be handled efficiently. Most notably, even under increased computational loads, the enhanced algorithm preserves its sub-second response times, maintaining the 1-second advantage over the original PAR implementation that was observed in simpler test cases.

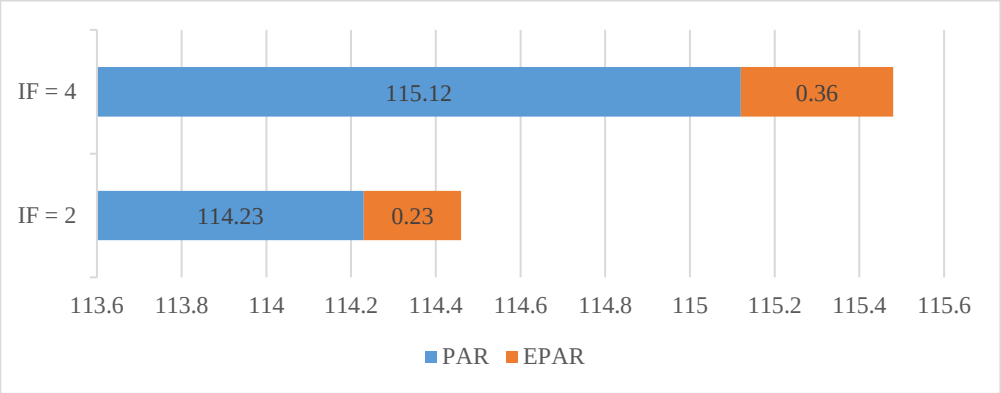

Figure 6. Comparison of the execution speed of the PAR and EPAR algorithms for 2 and 4 input items (using a large dataset)

Figure 7 compares the recommendation generation times of the EPAR algorithm with and without cache memory implementation. The results demonstrate that the cached version maintains significantly faster and more consistent performance across all test cases, while the uncached implementation shows noticeable latency increases as input complexity grows.

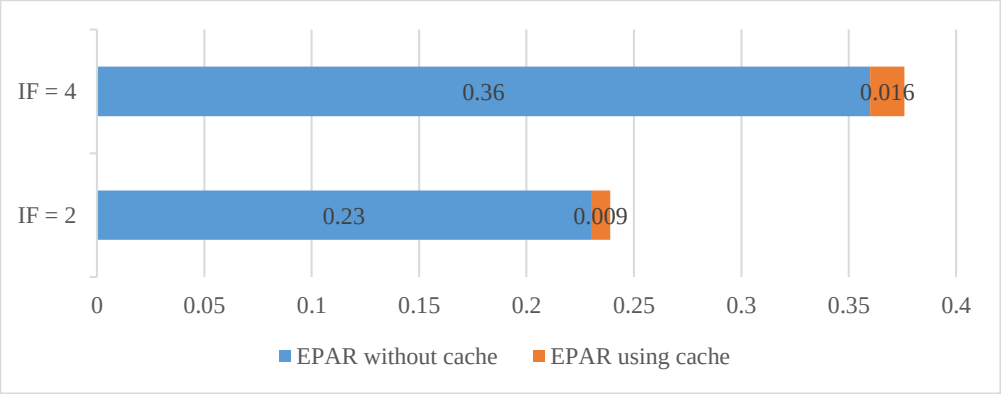

Figure 7. Examination of the time required for recommendation generation in the EPAR algorithm with and without cache memory

6. Conclusion and Future Works

Modern recommendation systems leverage intelligent algorithms to analyze user preferences and deliver personalized suggestions, with collaborative filtering and content-based approaches being the most prevalent. While effective, these traditional methods face significant challenges including the cold-start problem and privacy concerns related to user data collection. To overcome these limitations, we developed the pairwise association rules (PAR) algorithm which operates independently of private user information or historical profiles, offering a privacy-preserving alternative. Our comprehensive evaluation compared three recommendation algorithms using a dataset of 20,000 diverse food items. The results demonstrated PAR's superior performance across all measured metrics. Building on this success, we implemented key optimizations to further enhance PAR's efficiency: the algorithm was restructured to utilize modular training models, significantly reducing model creation time while enabling easier maintenance and updates. Additionally, we integrated a sophisticated caching mechanism to store training models in memory, eliminating redundant computations and conserving system resources. These improvements were rigorously validated across datasets of varying sizes, confirming consistent performance gains in both small and large-scale environments. The optimized PAR algorithm delivers substantially faster recommendation generation while maintaining high accuracy levels. As shown in our experimental results, the enhanced version processes requests up to 102% faster than the original implementation, with particularly impressive performance stability when handling increasing input complexity.

Looking forward, several promising research directions emerge, including investigating how regional dietary patterns might influence recommendation accuracy, exploring PAR's applicability to diverse recommendation scenarios such as social media or email suggestions, and developing category-based filtering mechanisms using advanced item classification parameters. These innovations could further strengthen PAR's position as a robust, privacy-conscious alternative to conventional recommendation systems, particularly valuable in cold-start situations or applications requiring strict data protection. Further investigation into the algorithm's generalizability across diverse domains, such as movie or retail recommendations which often feature sparse or unstructured datasets, is also a key future direction. This will involve exploring methodologies for adapting the transactional co-occurrence pattern discovery to such data structures, potentially by defining 'items' and 'transactions' through feature engineering from

unstructured text (e.g., extracting keywords or topics to act as items) or by developing specialized techniques to handle sparsity within the PAR framework. The current study's focus on structured meal data provides a strong foundation, and extending this to less structured and sparser scenarios will be a valuable next step to ascertain broader applicability. Furthermore, future benchmarking will extend comparisons to state-of-the-art hybrid recommender systems, including neural collaborative filtering approaches, to provide a broader performance perspective against contemporary methods. While the current study demonstrates clear benefits from the proposed enhancements, a detailed ablation study is planned for future work. This study will aim to quantitatively isolate the individual performance contributions of the caching mechanism versus the asynchronous model update framework, offering deeper insights into the impact of each architectural innovation on both efficiency and recommendation quality. The success of our architectural improvements demonstrates how careful system design can achieve both performance scalability and consistent recommendation quality.

References

[1] Panniello, U., Hill, S., & Gorgoglione, M. (2016). The impact of profit incentives on the relevance of online recommendations. *Electronic Commerce Research and Applications*, *20*, 87–104. https://doi.org/10.1016/j.elerap.2016.10.003

[2] Li, Y.-M., Wu, C.-T., & Lai, C.-Y. (2013). A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship. *Decision Support Systems*, *55*(3), 740–752. https://doi.org/10.1016/j.dss.2013.02.009

[3] Agarwal, V., & Bharadwaj, K. K. (2013). A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. *Social Network Analysis and Mining*, *3*(3), 359–379. https://doi.org/10.1007/s13278-012-0083-7

[4] Symeonidis, P., & Mantas, N. (2013). Spectral clustering for link prediction in social networks with positive and negative links. *Social Network Analysis and Mining*, *3*(4), 1433–1447. https://doi.org/10.1007/s13278-013-0128-6

[5] Papadimitriou, A., Symeonidis, P., & Manolopoulos, Y. (2012). Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, *85*(9), 2119–2132. https://doi.org/10.1016/j.jss.2012.04.019

[6] Sahebi, S., & Cohen, W. (2011). *Community-Based Recommendations: A Solution to the Cold Start Problem*.

[7] Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, *74*, 14–27. https://doi.org/10.1016/j.knosys.2014.10.016

[8] Bao, J., Zheng, Y., Wilkie, D., & Mokbel, M. (2015). Recommendations in location-based social networks: A survey. *GeoInformatica*, *19*(3), 525–565. https://doi.org/10.1007/s10707-014-0220-8

[9] Chekkai, N., Chikhi, S., & Kheddouci, H. (2012). *A weighted-graph based approach for solving the cold start problem in collaborative recommender systems*. 000759–000764. https://doi.org/10.1109/ISCC.2012.6249390

[10] Hu, Y., Shi, W., Li, H., & Hu, X. (2017). Mitigating Data Sparsity Using Similarity Reinforcement-Enhanced Collaborative Filtering. *ACM Transactions on Internet Technology*, *17*, 1–20. https://doi.org/10.1145/3062179

[11] Lemire, D., & Maclachlan, A. (2007). Slope One Predictors for Online Rating-Based Collaborative Filtering. *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, *5*. https://doi.org/10.1137/1.9781611972757.43

[12] Trihatmaja, R., & Wardhana Asnar, Y. D. (2018). Improving the Performance of Collaborative Filtering Using Outlier Labeling, Clustering, and Association Rule Mining. *2018 5th International Conference on Data and Software Engineering (ICoDSE)*, 1–6. https://doi.org/10.1109/ICODSE.2018.8705883

[13] Chen, Y., Li, F., & Fan, J. (2015). Mining association rules in big data with NGEP. *Cluster Computing*, *18*(2), 577–585. https://doi.org/10.1007/s10586-014-0419-3

[14] Marín, N., Molina, C., Serrano, J., & Vila, M. (2008). A Complexity Guided Algorithm for Association Rule Extraction on Fuzzy DataCubes. *Fuzzy Systems, IEEE Transactions On*, *16*, 693–714. https://doi.org/10.1109/TFUZZ.2007.905909

[15] Osadchiy, T., Poliakov, I., Olivier, P., Rowland, M., & Foster, E. (2019). Recommender system based on pairwise association rules. *Expert Systems with Applications*, *115*, 535–542. https://doi.org/10.1016/j.eswa.2018.07.077