Dynamic User Preferences Optimization in Time-aware Recommendations

Ainaz Ebrahimi[1], Zheying Zhang[1] and Kostas Stefanidis[1]

[1]Tampere University, Finland

Abstract

Recommender systems play a vital role in mitigating information overload by predicting user preferences. While traditional algorithms like collaborative filtering and content-based filtering have demonstrated their effectiveness, they often struggle to adapt to the dynamic nature of user preferences over time. This study addresses these limitations by enhancing the Time Correlation Coefficient (TCC) model with time-aware techniques, providing a more sophisticated understanding of the temporal shifts in user interests. We propose four advanced methodologies: Content-based Similarity, Time-based Decay, Cuckoo Search Optimization, and Decay Model Selection, each designed to improve recommendation accuracy by integrating dynamic, time-sensitive elements into the recommendation process. Our experiments reveal significant improvements in recommendation precision, demonstrating the advantages of these methodologies over the baseline TCC model in various performance metrics. The results emphasize the effectiveness of these dynamic strategies in personalizing user experiences, with a balanced approach to both accuracy and computational efficiency. This work lays a solid foundation for future research in recommendation technologies, offering practical insights and applications that can be extended across diverse domains. By enhancing recommender systems with a deeper understanding of temporal user behavior, we aim to improve the overall user experience in digital environments.

Keywords

Dynamic Attenuation Coefficient, Time-aware Recommendations

1. Introduction

Recommendation systems are an integral part of many online platforms, designed to deliver personalized suggestions to users across various sectors [1]. These systems analyze large volumes of user and item data to predict user preferences, aiming to recommend items that users are likely to enjoy, even without prior exposure to similar products or services [2]. Prominent examples such as Amazon and Netflix have led the way, utilizing recommendation systems to offer tailored suggestions based on user behavior and preferences.

There are two primary approaches used in recommendation systems: content-based filtering and collaborative filtering [3]. Content-based filtering recommends items by evaluating the attributes of items a user has previously engaged with, whereas collaborative filtering provides recommendations by leveraging the preferences of similar users. Both methods have their advantages and are frequently combined to enhance recommendation accuracy and reliability. The content-based filtering approach excels in recommending items like articles or news by focusing on the properties of the items themselves [4, 5]. It employs algorithms such as the Vector Space Model and probabilistic models, including the Naive Bayes Classifier, Decision Trees, and Neural Networks, to analyze item similarities and generate relevant recommendations [6, 7]. The collaborative filtering technique uses user-item interaction data to predict preferences based on the choices of similar users [8, 9, 10, 11]. Although both content-based and collaborative filtering have proven effective, they have inherent limitations. Content-based systems often lack diversity, as they tend to recommend items similar to those the user has already consumed [12]. Collaborative filtering, while addressing this issue, faces challenges with scalability, especially in large user bases with sparse data [13].

Hybrid recommendation systems have emerged to address these issues, integrating the strengths of both content-based and collaborative filtering to provide more accurate and diverse recommendations [14]. Leading companies such as Amazon and Netflix increasingly adopt these hybrid models, delivering more comprehensive and personalized recommendations by analyzing both content attributes and user behavior [14, 15]. However, these systems typically assume static user preferences, failing to capture the temporal evolution of user interests, which naturally shift over time due to personal growth, changing tastes, and external influences [12, 16]. To address these shortcomings, time-aware recommendation systems have been developed, incorporating temporal data to enhance the accuracy and relevance of predictions [17, 18, 19]. Recent advancements in this field have focused on integrating temporal factors into recommendation processes. One such study introduced the Time Correlation Coefficient (TCC) model, which combines a time correlation coefficient with optimized K-means clustering to improve recommendation accuracy [20]. Despite these advancements, existing time-aware models, particularly those using the TCC [20], often struggle to fully capture the evolving nature of user preferences. These models tend to overlook item similarity and the complex temporal dynamics of user behavior, leading to less precise recommendations.

To overcome these limitations, this paper focuses on enhancing the recently proposed TCC model, building upon its foundation to better address the evolving nature of user preferences over time. In this regard, we propose several techniques, all of which share the same overarching goal of improving the TCC model. Each technique offers a distinct approach, providing valuable advantages in different scenarios, but the ultimate objective remains to enhance the performance and adaptability of the TCC model. This leads to several important questions:

- How can the accuracy of time-aware recommendation systems, especially those using the TCC, be improved to better capture the evolving nature of user preferences?
- What modifications can be made to the TCC for-

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain

ainaz.ebrahimi@tuni.fi (A. Ebrahimi); zheying.zhang@tuni.fi (Z. Zhang); konstantinos.stefanidis@tuni.fi (K. Stefanidis)

0000-0002-6205-4210 (Z. Zhang); 0000-0003-1317-8062 (K. Stefanidis)

mula to more effectively integrate item similarity and temporal dynamics?
- What innovative techniques can be developed to incorporate temporal context and improve the accuracy and relevance of recommendations?

This paper presents the following contributions to address these challenges:

- Enhancement of the TCC algorithm by incorporating item similarity scores, improving recommendation accuracy.
- Development of four innovative algorithms designed to adapt to the evolving nature of user preferences, enabling the detection of shifts in interests over time. These algorithms enhance the ability to deliver personalized and highly accurate recommendations.
- Extensive experiments conducted on three datasets (two from Amazon and one from MovieLens), demonstrating the effectiveness of the proposed model in improving the accuracy and relevance of recommendations.

The structure of the paper is as follows: In Section 2, we provide a review of recent developments in time-aware recommendation systems and collaborative filtering models. Section 3 presents the proposed algorithms based on the Time Correlation Coefficient and its improvements. Section 4 details the experimental setup, including the datasets and evaluation metrics employed, while Section 5 presents the results accompanied by a comprehensive analysis. Finally, Section 6 provides the concluding remarks of the paper.

2. Related Works

Over the past decades, research in recommendation systems has steadily evolved, progressing from traditional methods like Collaborative Filtering (CF) and Content-Based Filtering (CBF) to more sophisticated models that address dynamic changes in user preferences. Traditional recommendation systems, though effective, face limitations when it comes to accounting for temporal aspects of user-item interactions [1, 21].

A key challenge that traditional recommendation systems face is their inability to account for the evolution of user preferences over time. Time- aware recommendation systems (TARS) seek to remedy this by explicitly incorporating temporal factors into their models [22, 23]. TARS leverage the fact that user preferences are not static and change over time, improving the relevance of recommendations by modeling time-based patterns of user behavior. One of the earliest and most notable approaches in this field is Time-SVD++, introduced by Koren [22]. This method extends the matrix factorization technique by incorporating time-dependent factors for both users and items, allowing the model to account for the gradual changes in user preferences. The Time-SVD++ model proved its effectiveness during the Netflix Prize competition, where it outperformed many traditional collaborative filtering techniques by modeling user behavior over time.

With considering Temporal Dynamics in Matrix Factorization Building new matrix factorization models have emerged. Collaborative Evolution (CE) is one such model that captures temporal changes by introducing a time-dependent factor into matrix factorization [24]. Another significant advancement is Collaborative Topic Regression (CTR), which integrates content-based features with collaborative filtering through Latent Dirichlet Allocation (LDA), incorporating temporal dynamics to track changes in user preferences over time [25]. Matrix factorization techniques have also been combined with neural networks to capture temporal patterns more effectively. For instance, Collaborative Deep Learning (CDL), a hierarchical Bayesian model, merges deep representation learning for content with collaborative filtering for ratings. This model effectively manages sparse data while capturing the temporal evolution of user preferences [26].

Beyond matrix factorization, time-aware collaborative filtering approaches have been extensively explored. Time-Weighted Collaborative Filtering (T-UCF) applies an exponential decay formula to older data, giving more weight to recent interactions [27]. This approach ensures that more recent user behaviors have a greater impact on the recommendations, improving accuracy in scenarios where user preferences evolve rapidly. Additionally, temporal clustering models, such as the multiclass co-grouping (MCoC) model presented in [28], further enhance recommendation precision by segmenting users and items into subgroups based on temporal patterns. Another notable example is Bayesian Probabilistic Tensor Factorization (BPTF), which models the user- item interaction as a three-dimensional tensor (user, item, time). This allows the model to capture the evolution of both user preferences and item characteristics over time. The BPTF model, introduced by Xiong et al. [29], is particularly useful for handling large, sparse datasets, such as those found in movie recommendation scenarios [30].

In another study, Ahmadian et al. [31] proposed the Recommender System with Temporal Reliability and Confidence (RSTRC), which integrates temporal factors into reliability and confidence measurements. This system differs from previous work by incorporating time into both the confidence scores and reliability assessments of user profiles, thereby improving recommendation precision. Furthermore, FSTS, a novel search technique incorporating both time-sensitive parameters and stability variables, has been evaluated on the MovieLens dataset. The algorithm demonstrated improvements in coverage, popularity, recall, and precision, although it struggled with the dynamic changes in time-sensitive factors [32]. Cui et al. [20] developed a model specifically for Internet of Things (IoT) environments, combining a Time Correlation Coefficient with a refined K-means clustering algorithm. By leveraging temporal dynamics, their model demonstrated a 5.2 % improvement in recommendation accuracy on datasets such as MovieLens and Douban. This highlights the increasing importance of temporal factors in domains where user preferences are heavily time dependent. However, it encountered limitations in capturing time-dependent user preferences, which our study aims to address.

Our research integrates temporal dynamics to model the evolution of user preference behaviors more effectively. Our primary focus is on enhancing the Time Correlation Coefficient (TCC) by incorporating item similarity scores, allowing the model to account for both temporal variations and item-specific relationships. Furthermore, we introduce innovative algorithms designed to determine a personalized interest-shifting parameter for each user, enabling the system to dynamically adapt to changes in each user preferences over time. These advancements collectively aim to improve the precision and relevance of the recommenda-

tions provided by the TCC model.

3. Methodology

In this paper, we address key limitations in current time-aware recommendation system, particularly the Time Correlation Coefficient (TCC) model. The TCC model is one of the foundational approach for time-aware recommendations. It employs a coefficient (TCC) to adjust all ratings, incorporating the influence of time on user interests. , but it faces two critical limitations: (1) it applies a uniform, static attenuation coefficient to all user ratings without adapting to user behavior or context, and (2) it does not consider item similarity, which is essential for capturing relationships between items in a user's preference profile.

To overcome these limitations, we propose a set of four methodologies, each designed to supplement and improve the TCC model. While all these methods share the ultimate goal of enhancing the accuracy and performance of TCC, they address different aspects of its improvement. Three methodologies focus on determining the attenuation coefficient dynamically, making it more adaptive and personalized, while one of them addresses the lack of item similarity in TCC.

The Proposed Methods:

1. *Content-Based Similarity (Addressing Item Similarity)*: Incorporates item similarity into TCC using NLP techniques to analyze item content, ensuring older ratings for similar items remain relevant.
2. *Time-Based Decay (Addressing Dynamic Attenuation Coefficient)*: Introduces a time-sensitive coefficient to model the diminishing relevance of older ratings, adapting to temporal dynamics.
3. *Decay Model Selection (Addressing Dynamic Attenuation Coefficient)*: Dynamically selects the most suitable decay model based on user behavior and dataset characteristics.
4. *Cuckoo Search Optimization (Addressing Dynamic Attenuation Coefficient)*: Optimizes the attenuation coefficient using metaheuristic techniques to adapt to evolving user preferences.

To validate the proposed approaches, we conduct experiments on real-world datasets, including Amazon and MovieLens. The results demonstrate that these methods effectively improve the TCC model's accuracy and performance in diverse recommendation scenarios.

In the following sections, we first discuss the limitations of the existing TCC model in Section 3.1. From Sections 3.2 to 3.5, we provide detailed descriptions of each proposed method, highlighting their specific contributions to refining the TCC model.

3.1. Time Correlation Coefficient (TCC)

The Time Correlation Coefficient Collaborative Filtering (TCCF) is a recommendation approach that enhances accuracy by integrating temporal dynamics into the recommendation process. Traditional collaborative filtering methods often assume static user preferences, overlooking the phenomenon of "interest drift," where user preferences evolve over time. TCCF mitigates this issue by assigning greater weight to more recent interactions, as reflected in the Time Correlation Coefficient (TCC) formula [20]:

$$\mathrm{tcc}_i = 1 - \frac{1}{\sqrt{2\pi\sigma}}\left(1 - \exp\left(-\frac{\Delta t^2}{2\sigma^2}\right)\right) \tag{1}$$

where:

- $\Delta t = t_i - t_1$ is the time difference between the i-th and most recent interaction.
- σ is the attenuation coefficient.

Algorithm steps are in below:

1. **Time Differences Calculation:** Compute the time differences (Δt) between user interactions.
2. **Determine σ:** Use a single static attenuation coefficient determined through trial and error.
3. **Calculate TCC:** Apply the formula above to compute TCC values for all user ratings, weighting them accordingly.
4. **Generate Recommendations:** Utilize the adjusted ratings to produce personalized recommendations.

Limitations: While effective, this method has certain limitations:

1. **Static Attenuation Coefficient:** The use of a single σ value does not account for individual user behaviors or dynamically changing preferences.
2. **Lack of Item Similarity Consideration:** TCC does not incorporate item similarity, which is a critical factor in refining recommendations.

To overcome these challenges, this study proposes enhancements to the TCC model, which will be explained in the following.

3.2. Content-Based Similarity Model

This approach balances historical and recent user interactions to maintain the influence of past evaluations, thus enhancing personalization. The TCC cannot be calculated accurately without considering item similarity. Even older ratings can be valuable if the item is highly similar to the most recent item the user has rated. Proposed enhancements to the TCC involve using NLP techniques to calculate item similarity. Specifically, descriptive attributes of products are analyzed to determine similarity between the most recently rated item and others. Algorithm steps are in below:

1. **Descriptive Feature Analysis:** Analyze product features (e.g., brand, material) using *NLP techniques* such as *TF-IDF*.
2. **Similarity Score Calculation:** Compute the cosine similarity between the most recently rated product f_{recent} and previously rated products f_j:

$$s(f_{\text{recent}}, f_j) = \frac{\langle f_{\text{recent}}, f_j \rangle}{\|f_{\text{recent}}\| \|f_j\|} \tag{2}$$

3. **Threshold and Rating Check:**
   - If the similarity score $s(f_{\text{recent}}, f_j)$ is above a predefined threshold (based on domain knowledge and datasets) and the rating $r_j \geq 4$, set:
   $$\mathrm{TCC} = 1 \tag{3}$$
   - Otherwise, calculate the *TCC* based on its formula (Equation 1) and then multiply it by all ratings r_j for each user

4. **Generate Recommendations:**
    - Use the adjusted *TCC* values to generate recommendations.

The proposed approach improves recommendation precision by considering item similarity and recent ratings, dynamically adjusting the TCC to reflect current user interests. It enhances efficiency through the use of NLP techniques for similarity calculations, delivering more precise and relevant recommendations.

3.3. Time-Based Decay Method

This method, along with the two subsequent approaches, dynamically calculates a personalized attenuation coefficient (σ) for each user, in contrast with the baseline model where σ was static and determined via trial and error. By tailoring σ to user behavior, this method captures temporal patterns more effectively, improving the personalization and accuracy of recommendations. The approach uses an exponential decay model to account for the diminishing influence of past interactions over time. Algorithm steps are in below:

1. **Time Differences Calculation**: Compute the time differences (time-diff$_i$) between consecutive interactions using normalized timestamps.
2. **Determine Decay Factor**: Set a constant decay factor (here is considered 0.5 as a balance) to control the rate at which past interactions lose relevance.
3. **Exponential Decay**: Use the formula:

$$\sigma_i = \exp(-\text{decay-factor} \cdot \text{time-diff}_i) \tag{4}$$

   This dynamically computes the decay coefficient (σ) for each interaction.
4. **Calculate TCC**: Integrate the computed σ into the TCC formula to adjust ratings, ensuring recommendations reflect the temporal relevance of user interactions.
5. **Generate Recommendations**: Use the adjusted TCC values to generate recommendations.

This method enhances temporal sensitivity by adapting to shifts in user preferences, using a smooth exponential decay function for realistic modeling.

3.4. Decay Model Selection Method

This approach employs both exponential and Gaussian decay functions to analyze how user ratings evolve over time. By utilizing a dual-model technique, it identifies whether ratings decline rapidly (exponential) or steadily (Gaussian), offering insights into user engagement and satisfaction. The method aims to dynamically adapt to temporal changes in user preferences, enhancing the accuracy of time-aware recommendations. Algorithm steps are in below:

1. **Check for Sufficient Data:** Ensure at least two data points are available for model fitting.
2. **Fit Exponential Decay Model:** Use the formula provided in Equation 5 to fit the data and extract the decay parameter b_{exp}.
3. **Fit Gaussian Decay Model:** Use the formula provided in Equation 6 to fit the data and extract the decay parameter b_{gauss}.
4. **Return Decay Parameters:** Provide both b_{exp} and b_{gauss} for further analysis.
5. **Calculate TCC:** Integrate the decay parameters into the Time Correlation Coefficient (TCC) formula to compute the final coefficient.
6. **Generate Recommendations:** Use the decay parameters to refine user ratings and improve engagement strategies.

The decay functions are defined as follows:

$$y_{\text{exp}} = a \cdot e^{-b_{\text{exp}} x} \tag{5}$$

$$y_{\text{gauss}} = a \cdot e^{-\left(\frac{x - b_{\text{gauss}}}{c}\right)^2} \tag{6}$$

where:

- a is the initial value or amplitude, determining the starting point of the decay curve.
- b_{exp} and b_{gauss} are the decay rate parameters, indicating how quickly the decay occurs over time.
- c is the standard deviation in the Gaussian model, controlling the width of the decay curve and indicating how spread out the decay is around the mean.

Decay functions provide valuable insights into user behavior by revealing how quickly user satisfaction diminishes over time. For instance, a rapid decline (exponential) may indicate the need for immediate follow-ups, while a gradual decline (Gaussian) suggests sustained engagement efforts. Additionally, decay parameters enable forecasting trends, allowing organizations to proactively address potential declines in user satisfaction.

3.5. The Cuckoo Search Optimization Technique

In this section, we explain the Cuckoo Search optimization, a metaheuristic algorithm inspired by the brood parasitism behavior of cuckoo birds. We then describe the steps involved in using Cuckoo Search to determine the optimal sigma (σ) value for each user, which is critical for improving recommendation accuracy.

3.5.1. Cuckoo Search Algorithm

The Cuckoo Search algorithm is a metaheuristic optimization technique that excels in solving complex optimization problems by balancing exploration and exploitation. It uses random nest selection and Lévy flights for exploration, inspired by the cuckoo bird's brood parasitism behavior. The algorithm operates under the following key rules [20]:

- Random egg-laying in nests.
- Retaining nests with the best eggs (solutions).
- Generating new nests via Lévy flights if a nest is discovered with a probability p.

The position update for a cuckoo is defined as:

$$\mathbf{y}_i^{(t')} = \mathbf{y}_i^{(t)} + \beta \cdot 0.01 \cdot M \cdot (\mathbf{y}_i^{(t)} - \mathbf{q}_g^{(t)}) \cdot s \tag{7}$$

where:

- $\mathbf{y}_i^{(t')}$ and $\mathbf{y}_i^{(t)}$ represent the positions of the i-th cuckoo at times t' and t, respectively.
- $\beta > 0$ is the scaling factor for the step size.

- M is a randomized value from the Lévy distribution.
- s is a random variable generated using a normal (Gaussian) distribution, $N(0, 1)$.

For local search (when a cuckoo's nest is observed), the position update is defined as:

$$\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t')} + \text{rand} \cdot (\mathbf{y}_k^{(t')} - \mathbf{y}_j^{(t')}) \tag{8}$$

where:

- $\mathbf{y}_k^{(t')}$ and $\mathbf{y}_j^{(t')}$ represent randomly selected positions of other cuckoos at time t'.
- rand is a uniformly generated random variable that guides the exploration process.

3.5.2. Optimization Steps Using Cuckoo Search

The Cuckoo Search optimization technique is used to determine the optimal sigma (σ) value for each user, which is critical for improving the precision and recall of recommendations. Algorithm steps are in below:

1. **Data Preprocessing:** Load and preprocess the dataset to extract user ID, item ID, rating, and timestamp.
2. **Population Initialization:** Create a diverse set of initial sigma (σ) values within a specified range.
3. **Lévy Flight:** Update sigma values using Lévy flight steps (Equation 7) to balance exploration and exploitation.
4. **Fitness Function:** Evaluate the performance of each sigma value using a fitness function based on precision and recall metrics:

$$f(\sigma) = \text{precision}(\sigma) + \text{recall}(\sigma) \tag{9}$$

5. **Calculate TCC:** Integrate the optimized sigma values into the Time Correlation Coefficient (TCC) formula to compute the final coefficient.
6. **Generate Recommendations:** Use the optimized TCC values to generate personalized recommendations for each user.

This optimization method fine-tunes the sigma (σ) value for each user, enhancing precision and recall, and thereby boosting overall recommendation accuracy. By adapting to evolving user interests, the system remains resilient to changes in behavior and preferences. The personalized sigma values ensure that recommendations are tailored to individual users.

4. Experimental Evaluation

4.1. Datasets

In our evaluation, we utilize three datasets to assess the proposed methods: the Amazon Phones and Accessories dataset with 20.8M ratings, 1.3M items, and 11.6M users, the Amazon Video Games dataset containing 4.6M ratings, 137.2K items, and 2.8M users which the timespan for these datasets is from 1996 to 2023[33], and the MovieLens dataset [34], which includes 20000263 ratings across 27278 movies which collect from 1995 to 2015. These diverse datasets offer a robust evaluation platform for the proposed techniques. They contain realworld user interactions, including ratings, users, and timestamps, providing authentic data for training recommendation algorithms. These datasets cover a wide range of products and media, enabling the evaluation of methods across diverse contexts. Both Amazon and MovieLens datasets support collaborative filtering techniques, leveraging user-item interactions to identify patterns. However, the results can vary across different datasets due to the temporal dynamics and data characteristics. Additionally, the diversity in rating patterns, time intervals, and the inclusion of temporal features like time differences or timestamps can influence how effectively the model adapts to the dataset's structure. Thus, the alignment between the dataset's temporal properties and the method's design significantly impacts performance.

4.2. Evaluation Metrics

To validate the proposed approaches, we employed several comparison metrics, including Recall, Precision, F-measure, and Mean Absolute Error (MAE), alongside time complexity analysis and memory usage. These metrics are crucial for assessing the performance and effectiveness of recommendation systems. They introduced shortly in below:

1. **Recall (R):** Recall measures the ability of the recommendation system to identify relevant items. It is calculated as the ratio of the number of recommended items that are also favorite items for the user to the total number of favorite items. The formula for recall is given by:

$$R = \frac{|A(i) \cap B(i)|}{|B(i)|} \tag{10}$$

   Here, $A(i)$ is the number of recommended items to the target user u, and $B(i)$ is the number of favorite items for user u.
2. **Precision (P):** Precision evaluates the relevance of the recommended items. It is defined as the ratio of the number of recommended items that are actually relevant to the total number of recommended items. The formula for precision is given by:

$$P = \frac{|A(i) \cap B(i)|}{|A(i)|} \tag{11}$$

3. **F-measure:** The F-measure combines precision and recall into a single score, providing a balanced view of the model's performance. It is particularly useful in imbalanced class scenarios. The formula for F-measure is:

$$\textit{F-measure} = \frac{2 \cdot P \cdot R}{P + R} \tag{12}$$

   where P is precision and R is recall.
4. **Mean Absolute Error (MAE):** MAE measures the average absolute difference between predicted and actual ratings, providing a straightforward assessment of recommendation quality. A smaller MAE indicates better recommendation quality. The MAE is calculated as:

$$MAE = \frac{\sum_{i=1}^{N} |x_i - y_i|}{A} \tag{13}$$

   Here x_i is the predicted user's score, y_i is the actual user's score, and A is the recommended items to the intended user.

Table 1
MAE Comparison Across Datasets

Method	MovieLens MAE	Cellphone MAE	Videogame MAE	Average MAE
TCC	0.8258	1.2402	1.3472	1.1377
Content-Based	0.6864	1.1744	1.2968	1.0525
Time-Based	0.8236	1.1255	1.3161	1.0884
Cuckoo Search	0.8964	1.0925	1.2952	1.0947
Decay Model	0.8204	1.2222	1.3666	1.1364

Table 2
Precision Comparison Across Datasets

Method	MovieLens Precision	Cellphone Precision	Videogame Precision	Average Precision
TCC	0.7604	0.7473	0.7941	0.7673
Content-Based	0.8729	0.7859	0.8283	0.8290
Time-Based	0.7622	0.7764	0.8016	0.7801
Cuckoo Search	0.7778	0.8502	0.8252	0.8177
Decay Model	0.7613	0.7567	0.8016	0.7732

5. **Time Complexity:** Time complexity measures the efficiency of an algorithm by analyzing how its runtime grows as the input size increases. It is typically expressed using Big-O notation to represent the upper bound of an algorithm's growth rate. The general formula for time complexity is:

$$T(n) = O(f(n)) \tag{14}$$

Here $T(n)$ represents the time complexity as a function of the input size n. $O(f(n))$ describes the growth rate of the algorithm (e.g., $O(1)$, $O(n)$, $O(n^2)$, etc.).

6. **Memory Usage (Space Complexity):** Memory usage refers to the amount of storage an algorithm requires during its execution. This includes both fixed storage (e.g., constants, program code) and variable storage that depends on the input size. The general formula for memory usage is:

$$S(p) = A + S(I) \tag{15}$$

Here A is the fixed part, such as constants or program code. $S(I)$ is the variable part, which depends on the input size I (e.g., recursion stack, dynamic memory allocation).

5. Experimental Result

5.1. comparison based on MAE, Precission, Recall,and F-measure

The results of the comparison between the basic variant of the TCC model and the proposed models, based on ten recommended items, are presented for different metrics. Based on given result in Table 1, 2, 3, and 4:

- Content-Based achieves the lowest MAE on average (1.0525), outperforming TCC by 7.48% on average.
- Time-Based also slightly improves over TCC with a 4.33% reduction in MAE.
- Cuckoo Search does not outperform TCC for MAE consistently but is comparable.
- Cuckoo Search achieves the best F-measure (0.6460), improving over TCC by 22.25%.
- Content-Based improves F-measure by 8.42%, while Time-Based shows an improvement of 3.66%.
- Cuckoo Search achieves the best recall (0.7402), improving over TCC by 10.13%.
- Content-Based and Time-Based also slightly outperform TCCF with 2.99% and 1.49% increases in Recall, respectively.
- Content-Based achieves the highest average precision (0.8290), improving over TCC by 8.04%.
- Cuckoo Search also significantly improves, showing a 6.58% increase in precision over TCC.

5.2. Comparison based on Time Complexity and memory usage

As you can see in Table 5:

- Content-Based: More computationally efficient than TCC due to its focus on vector dimensions (d) rather than trial steps (j). In terms of memory usage is slightly higher than TCC due to the inclusion of vector dimensions, but still manageable for smaller d.
- Time-Based: *S*impler and faster than TCC, focusing only on user-record pairs without trial steps. More efficient than TCC, in memory usage, with linear scaling based on the number of records (m).
- Cuckoo Search: Provides optimization capabilities for σ, though computationally expensive in terms of iterations (i) and population size (p).Extremely low in memory usage, making it suitable for memory-constrained environments.
- Decay Model: Faster than TCC due to its focus on decay fitting (f), which is typically small.its memory usage is moderate, scaling linearly with records (m) and filtered data (k).

Table 3
Recall Comparison Across Datasets

Method	MovieLens Recall	Cellphone Recall	Videogame Recall	Average Recall
TCC	0.3119	0.9009	0.8038	0.6722
Content-Based	0.3329	0.9148	0.8291	0.6923
Time-Based	0.3273	0.9060	0.8133	0.6822
Cuckoo Search	0.4963	0.9018	0.8226	0.7402
Decay Model	0.3242	0.9066	0.8097	0.6802

Table 4
F-measure Comparison Across Datasets

Method	MovieLens F-Measure	Cellphone F-Measure	Videogame F-Measure	Average F-Measure
TCC	0.2865	0.6708	0.6277	0.5283
Content-Based	0.3122	0.7188	0.6870	0.5727
Time-Based	0.3007	0.6988	0.6434	0.5476
Cuckoo Search	0.4703	0.7694	0.6983	0.6460
Decay Model	0.2984	0.6811	0.6426	0.5407

5.3. Overall Comparison and Key insight

- Content-Based:
  Performance: Achieves the lowest MAE (7.48% lower than TCC) and the highest Precision (8.04% higher), making it the most accurate method for prediction tasks.
  Time Complexity $O(n \cdot m \cdot d)$, efficient for large-scale systems.
  Memory Usage: Slightly higher than TCC ($O(16d + 40m)$), manageable for smaller d.
  Best Use Case:Accuracy-driven tasks with balanced computational and memory requirements.

- Cuckoo Search:
  Performance: Achieves the highest F-Measure (22.25% higher) and Recall (10.13% higher) than TCC, making it ideal for applications where both accuracy and recall are critical.
  Time Complexity: $O(i \cdot p \cdot n \cdot m)$, computationally expensive.
  Memory Usage: Extremely low ($O(8 \cdot (p + 5))$), ideal for memory-constrained environments.
  Best Use Case: Performance-critical applications with sufficient computational resources.

- Time-Based:
  Performance: Modest improvements in F-Measure (3.66%) and MAE (4.33%) over TCC.
  Time Complexity: $O(n \cdot m)$, the simplest and most computationally efficient method.
  Memory Usage: Linear ($O(24m)$), highly scalable.
  Best Use Case: Real-time or resource-constrained environments.

- Final selection:
  Best Overall: Content-Based for its balance of performance, computational efficiency, and memory usage.
  Best for Performance: Cuckoo Search for superior recall and F-Measure.
  Best for Simplicity: Time-Based for scalability and low resource requirements.

6. Conclusion

In conclusion, this study has advanced the understanding of time-aware recommendation systems by addressing the limitations of existing algorithm, Time Correlation Coefficient (TCC). The proposed methodologies—Content-based, Cuckoo Search, Decay Model, and Time-Based—highlight the need to balance performance with computational efficiency. Future research should focus on the generalizability of these methods across diverse datasets and the integration of real-world user feedback, ensuring that recommendation systems continue to evolve and effectively meet users' dynamic preferences. Ultimately, these advancements aim to enhance user experience and foster greater satisfaction and loyalty.

References

[1] A. Y.-A. Chen, D. McLeod, Collaborative filtering for information recommendation systems, in: Encyclopedia of E-Commerce, E-Government, and Mobile Commerce, IGI Global, 2006, pp. 118–123.
[2] Y. Deldjoo, Z. He, J. McAuley, A. Korikov, S. Sanner,

Table 5
Complexity and Memory Usage Comparison Across Datasets

Method	Time Complexity	Memory Usage
TCC (Baseline)	$O(k \cdot m \cdot j)$, n = users, m = user records, j = trial steps	$O(k \cdot m)$, n = users, m = user records
Content-Based	$O(k \cdot m \cdot d)$, d = vector dimensions, m = user records	$O(k \cdot d + m)$, d = vector dimensions, m = user records
Time-Based	$O(n \cdot m)$, m = user records	$O(2m)$, m = user records
Cuckoo Search	$O(k \cdot m \cdot i)$, i = iterations, p = population size	$O(k \cdot (p + f))$, p = population size
Decay Model	$O(k \cdot f \cdot m)$, f = decay fitting, m = user records	$O(k \cdot 1 + k)$, m = user records, k = filtered data

A. Ramisa, R. Vidal, M. Sathiamoorthy, A. Kasirzadeh, S. Milano, A review of modern recommender systems using generative models (gen-recsys), in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 6448–6458.

[3] F. Sayyed, R. Argiddi, S. Apte, Generating recommendations for stock market using collaborative filtering, Int. J. Comput. Eng. Sci 3 (2013) 46–49.

[4] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-based systems 46 (2013) 109–132.

[5] R. Burke, Hybrid recommender systems: Survey and experiments, User modeling and user-adapted interaction 12 (2002) 331–370.

[6] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine learning 29 (1997) 131–163.

[7] E. Bartocci, E. A. Gol, I. Haghighi, C. Belta, A formal methods approach to pattern recognition and synthesis in reaction diffusion networks, IEEE Transactions on Control of Network Systems 5 (2016) 308–320.

[8] P. Lops, M. De Gemmis, G. Semeraro, Content-based recommender systems: State of the art and trends, Recommender systems handbook (2011) 73–105.

[9] R. Borges, K. Stefanidis, On measuring popularity bias in collaborative filtering data, in: Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference, Copenhagen, Denmark, March 30, 2020, volume 2578 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020.

[10] R. Borges, K. Stefanidis, Feature-blind fairness in collaborative filtering recommender systems, Knowl. Inf. Syst. 64 (2022) 943–962.

[11] K. Stefanidis, E. Ntoutsi, H. Kondylakis, Y. Velegrakis, Social-based collaborative filtering, in: R. Alhajj, J. G. Rokne (Eds.), Encyclopedia of Social Network Analysis and Mining, 2nd Edition, Springer, 2018.

[12] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE transactions on knowledge and data engineering 17 (2005) 734–749.

[13] D. H. Stern, R. Herbrich, T. Graepel, Matchbox: large scale online bayesian recommendations, in: Proceedings of the 18th international conference on World wide web, 2009, pp. 111–120.

[14] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The adaptive web: methods and strategies of web personalization, Springer, 2007, pp. 291–324.

[15] C.-N. Ziegler, S. M. McNee, J. A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th international conference on World Wide Web, 2005, pp. 22–32.

[16] C. C. Aggarwal, C. C. Aggarwal, Time-and location-sensitive recommender systems, Recommender Systems: The Textbook (2016) 283–308.

[17] K. Stefanidis, I. Ntoutsi, K. Nørvåg, H. Kriegel, A framework for time-aware recommendations, in: DEXA, volume 7447 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 329–344.

[18] K. Stefanidis, E. Ntoutsi, M. Petropoulos, K. Nørvåg, H. Kriegel, A framework for modeling, computing and presenting time-aware recommendations, Trans. Large Scale Data Knowl. Centered Syst. 10 (2013) 146–172.

[19] F. O. Isinkaye, Y. O. Folajimi, B. A. Ojokoh, Recommendation systems: Principles, methods and evaluation, Egyptian informatics journal 16 (2015) 261–273.

[20] Z. Cui, X. Xu, X. Fei, X. Cai, Y. Cao, W. Zhang, J. Chen, Personalized recommendation system based on collaborative filtering for iot scenarios, IEEE Transactions on Services Computing 13 (2020) 685–695.

[21] R. Cai, R. Lu, W. Chen, Z. Hao, Counterfactual contextual bandit for recommendation under delayed feedback, Neural Computing and Applications (2024) 1–15.

[22] Y. Koren, Collaborative filtering with temporal dynamics, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 447–456.

[23] I. Al-Hadi, N. M. Sharef, M. N. Sulaiman, N. Mustapha, Review of the temporal recommendation system with matrix factorization, Int. J. Innov. Comput. Inf. Control 13 (2017) 1579–1594.

[24] Y. Wan, Y. Chen, C. Yan, An integrated time-aware collaborative filtering algorithm, in: Knowledge Management in Organizations: 15th International Conference, KMO 2021, Kaohsiung, Taiwan, July 20-22, 2021, Proceedings 15, Springer, 2021, pp. 369–379.

[25] A. Hamzehei, R. K. Wong, D. Koutra, F. Chen, Collaborative topic regression for predicting topic-based social influence, Machine Learning 108 (2019) 1831–1850.

[26] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1235–1244.

[27] H. Su, X. Lin, B. Yan, H. Zheng, The collaborative filtering algorithm with time weight based on mapreduce, in: Big Data Computing and Communications: First International Conference, BigCom 2015, Taiyuan, China, August 1-3, 2015, Proceedings 1, Springer, 2015, pp. 386–395.

[28] J. Bu, X. Shen, B. Xu, C. Chen, X. He, D. Cai, Improving collaborative recommendation via user-item subgroups, IEEE Transactions on Knowledge and Data Engineering 28 (2016) 2363–2375.

[29] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, J. G. Carbonell, Temporal collaborative filtering with bayesian probabilistic tensor factorization, in: Proceedings of the 2010 SIAM international conference on data mining, SIAM, 2010, pp. 211–222.

[30] T. Xue, B. Jin, B. Li, W. Wang, Q. Zhang, S. Tian, A spatio-temporal recommender system for on-demand cinemas, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1553–1562.

[31] S. Ahmadian, N. Joorabloo, M. Jalili, M. Ahmadian, Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach, Expert Systems with Applications 187 (2022) 115849.

[32] S. Pang, S. Yu, G. Li, S. Qiao, M. Wang, Time-sensitive collaborative filtering algorithm with feature stability, Computing and Informatics 39 (2020) 141–155.

[33] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, J. McAuley, Bridging language and items for retrieval and recommendation, arXiv preprint arXiv:2403.03952 (2024).

[34] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) 5 (2015) 1–19.